UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

FACULDADE DE ARQUITETURA

PROGRAMA DE PÓS-GRADUAÇÃO EM DESIGN

Bruno Spanevello Pergher

SISTEMATIZAÇÃO DE CRIAÇÃO DE CONTEÚDO INTERATIVO 3D (WEBGL) APLICADO AO DESIGN VIRTUAL

Dissertação de Mestrado

Porto Alegre

2017

BRUNO SPANEVELLO PERGHER

Sistematização de criação de conteúdo interativo 3D (WebGL) aplicado ao Design Virtual

Dissertação apresentada ao Programa de Pós-Graduação em Design da Universidade Federal do Rio Grande do Sul, como requisito parcial à obtenção do título de Mestre em Design.

Orientador: Prof. Dr. José Luís Farinatti Aymone

Porto Alegre

2017

CIP - Catalogação na Publicação

Pergher, Bruno
Sistematização de criação de conteúdo interativo 3D (WebGL) aplicado ao Design Virtual / Bruno Pergher. -- 2017.
229 f.

Orientador: José Luís Farinatti Aymone.

Dissertação (Mestrado) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Programa de Pós-Graduação em Design, Porto Alegre, BR-RS, 2017.

1. Design Virtual. 2. Ambientes Tridimensionais. 3. WebGL. 4. Computação Gráfica. I. Farinatti Aymone, José Luís, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

PGDESIGN | Programa de Pós-Graduação Mestrado | Doutorado

UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Bruno Spanevello Pergher

SISTEMATIZAÇÃO DE CRIAÇÃO DE CONTEÚDO INTERATIVO 3D (WEBGL) APLICADO AO DESIGN VIRTUAL

Esta Dissertação foi julgada adequada para a obtenção do Título de Mestre em Design, e aprovada em sua forma final pelo Programa de Pós-Graduação em Design da UFRGS.

Porto Alegre, 25 de Agosto de 2017.

_____

Prof. Dr. Régio Pierre da Silva

Coordenador do Programa de Pós-Graduação em Design da UFRGS

Banca Examinadora:

_____

Orientador: **Prof. Dr. José Luís Farinatti Aymone**

Departamento de Design e Expressão Gráfica (DEG) - UFRGS

_____

Prof. Dr. Vinicius Gadis Ribeiro

Centro Universitário Ritter dos Reis – UNIRITTER

_____

Prof.ª Dr.ª Jocelise Jacques de Jacques

Departamento de Design e Expressão Gráfica (DEG) – UFRGS

_____

Prof. Dr. Rodrigo Antonio Marques Braga

Departamento de Design e Expressão Gráfica (DEG) – UFRGS

AGRADECIMENTOS

Agradeço aos meus pais, Antônia e Antônio, pelo incentivo e por acreditarem na importância do ensino e do aperfeiçoamento acadêmico.

Agradeço ao competente professor e orientador José Luís Farinatti Aymone pelos conselhos e ajuda na condução deste trabalho. Obrigado pela acessibilidade, sinceridade e responsabilidade de sempre.

Por fim, um agradecimento aos professores e servidores do PGDesign da UFRGS, que me indicaram os caminhos para um melhor aproveitamento da minha trajetória dentro desta pós-graduação. E a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro durante o mestrado.

Muito Obrigado

In virtual reality, we're placing the viewer inside a moment or a story... made possible by sound and visual technology that's actually tricking the brain into believing it's somewhere else.

Na realidade virtual, estamos colocando o espectador dentro de um momento ou de uma história... tornada possível pela tecnologia sonora e visual que realmente está iludindo o cérebro para acreditar que está em outro lugar.

Chris Milk

RESUMO

PERGHER, Bruno. S. **Sistematização de criação de conteúdo interativo 3D (WebGL) aplicado ao Design Virtual**. 2017. 229 f. Dissertação (Mestrado em Design) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

Na medida em que surgem novas tecnologias de criação de ambientes tridimensionais virtuais, como o WebGL, ampliam-se as possibilidades de uso destes e possibilita-se um aprimoramento da qualidade gráfica e das formas de interação com os mesmos. A presente pesquisa objetiva aproximar designers e a promissora tecnologia WebGL, ainda muito ligada a conhecedores da programação computacional escrita e distante de grande parte dos designers. Com isso, visa-se melhorar a qualidade de criação e distribuição dos trabalhos de Design Virtual. Para tal, propõe-se uma sistemática de criação de conteúdos interativos tridimensionais utilizando o padrão WebGL, balizada por requisitos estipulados com base na revisão bibliográfica do assunto. Dentre os principais requisitos estão a utilização de *softwares* e recursos que não exijam conhecimentos prévios de linguagem escrita de programação, a compreensão dos três principais pilares dos ambientes virtuais 3D (modelagem, animação e interação) e a exportação facilitada do conteúdo criado em um arquivo (HTML) executável nos principais navegadores web, empregando o padrão WebGL, sem o uso de qualquer complemento (*plug-in*). Após a análise de 34 ferramentas, escolheu-se o conjunto de programas Blender + Blend4Web para ser utilizado na sistemática. Como resultado, obteve-se uma sistemática de criação de ambientes virtuais tridimensionais com recursos bastante diversificados, que se mostrou útil na atualização dos conhecimentos de estudantes e profissionais de Design Virtual.

Palavras-chave: Design Virtual; Ambientes Tridimensionais; WebGL.

ABSTRACT

PERGHER, Bruno. S. **Sistematização de criação de conteúdo interativo 3D (WebGL) aplicado ao Design Virtual**. 2017. 229 f. Dissertação (Mestrado em Design) – Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

As new technologies for the creation of three-dimensional virtual environments, such as WebGL, the possibilities of their use are expanded and it is possible to improve the graphic quality and the ways of interacting with them. The present research aims to approach designers and the promising WebGL technology, still very much related to the knowledge of the written and distant programming of most of the designers. With this, it aims to improve the quality of creation and distribution of the Virtual Design works. For this, a systematic creation of three-dimensional interactive contents using the WebGL standard is proposed, based on requirements stipulated based on the bibliographic review of the subject. Among the main requirements are the use of software and resources that do not require prior knowledge of written programming language, understanding the three main pillars of 3D virtual environments (modeling, animation and interaction) and facilitating the export of content created in a file (HTML) executable in major web browsers, employing the WebGL standard, without the use of any plug-in. After the analysis of 34 tools, the set of Blender + Blend4Web programs was chosen to be used in the systematics. As a result, a systematic creation of three-dimensional virtual environments with very diversified resources was obtained, which proved useful in updating the knowledge of students and professionals of Virtual Design.

***Keywords**: Virtual Design; Three-dimensional Environments; WebGL.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2D	Duas Dimensões / Bidimensional
3D	Três Dimensões / Tridimensional
API	*Application Programming Interface*
AV3D	Ambiente Virtual Tridimensional
CEO	*Chief Executive Officer*
DOS	*Disk Operating System*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
HTML	*Hypertext Markup Language*
IHC	Interface Humano-Computador
ISO	*Internacional Standards Organization*
MIT	*Massachussets Institute of Technology*
OO	Orientação a Objetos
OPENGL	*Open Graphics Library*
POO	Programação Orientada a Objetos
RA	Realidade Aumentada
RV	Realidade Virtual
UFRGS	Universidade Federal do Rio Grande do Sul
VRML	*Virtual Reality Modeling Language*
WEBGL	*Web Graphics Library*
X3D	*Extensible 3D Graphics*
XML	*Extensible Markup Language*
HD	*Hard Disk*
SSD	*Solid-state Drive*

LISTA DE APÊNDICES

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Os ambientes virtuais tridimensionais têm seu campo de aplicações cada vez mais estendido à medida que o avanço da tecnologia computacional aprimora as capacidades tecnológicas para criação e utilização deles. Mesmo tendo surgido décadas atrás, a Realidade Virtual (RV) é hoje concreta e viável. Tornou-se uma grande aposta tecnológica para os próximos anos, convertendo-se já em 2016 num nicho de mercado bilionário. O qual faturou, segundo a empresa de auditoria, assessoria e consultoria empresarial Deloitte, aproximadamente 700 bilhões dólares em vendas de hardware e 300 bilhões em vendas de conteúdo para computadores, videogames e plataformas móveis (LEE; STEWART, 2016).

Espera-se ainda que a venda de aparelhos de realidade virtual cresça em um ritmo anual de 84,5% até 2020 (GUERRA, 2016). E por de trás dos números de faturamento está toda uma cadeia de desenvolvedores de conteúdo virtual que necessita aliar conhecimentos de diferentes áreas para garantir o crescimento do mercado de RV, principalmente no Brasil.

> Sem conteúdo não há crescimento, porque não há o que ser consumido. Neste processo, os estúdios de produção imersiva têm um papel fundamental e ao mesmo tempo desafiador: criar em quantidade e com qualidade para atender a demanda, diz Antônio Rabello, diretor do VR Studios (REVISTA EXAME, 2016).

Para melhorar progressivamente a qualidade dos conteúdos de realidade virtual são necessários conhecimentos de Design e sem dúvida os designers se apresentam como importantes profissionais no desenvolvimento de ambientes virtuais, em especial os tridimensionais. No entanto, esta é uma indústria multidisciplinar, destacando-se os que possuem um conhecimento mais abrangente,

não só de Design, mas também das tecnologias de criação que hoje estão muito mais próximas dos programadores.

Algumas das principais tecnologias de descrição de cenas para web foram desenvolvidas até o ponto que usuários sem conhecimento algum de programação também pudessem gerar conteúdo. Isso aconteceu com o VRML por exemplo e foi essencial para sua popularização e crescimento das pesquisas entorno do assunto. Hoje estamos diante do WebGL, um padrão de processamento de imagens e efeitos gráficos acelerados pela Unidade de Processamento Gráfico (GPU, sigla em inglês) de computadores e dispositivos móveis, tudo isso interagindo com os elementos padrão de páginas web (HTML) (SUZUKI, 2015).

Muito mais poderoso, o WebGL traz consigo um salto de qualidade gráfica e de processamento, possibilitando a distribuição de jogos, animações, simulações científicas, apresentações e tantas outras aplicações interativas pela internet para muitas plataformas. Como o esperado desde o início de seu desenvolvimento, todos os principais dispositivos móveis (*smartphones* e *tablets*) ou "de mesa" (computadores e videogames) e sistemas operacionais (Windows, Mac OS, Linux) já possuem a capacidade de exibir conteúdo 3D *online* e *off-line* por meio desta tecnologia.

A inclusão de designers nesse meio parece natural e necessária para alcançar o aprimoramento constante da qualidade dos produtos digitais. Resta apenas propor caminhos para que o WebGL inicie a sua aproximação destes profissionais, proporcionando-lhes mais conhecimento e aperfeiçoamento de suas capacidades técnicas e inventivas.

1.1. PROBLEMA DE PESQUISA

Como aliar a tecnologia WebGL aos conhecimentos de criação de objetos e ambientes tridimensionais virtuais dos designers, a fim de melhorar a apresentação, a distribuição e o alcance de seus trabalhos?

1.2. OBJETIVOS

O objetivo geral desta pesquisa é propor para uso no Design, uma sistemática[1] de criação de ambientes tridimensionais interativos que possam ser apresentados e distribuídos, a partir da internet ou não, utilizando a tecnologia de representação virtual WebGL, beneficiando-se de suas vantagens.

Os objetivos específicos são:

a. Compreender em que consiste e como funciona o WebGL, bem como as suas vantagens e desvantagens em relação a tecnologias anteriores;
b. Detectar em quais pontos o WebGL pode ser útil aos projetos de Design Virtual (animação, jogos, simulações...);

1.3. DELIMITAÇÃO

Apesar da possibilidade de desenvolvimento de novas ferramentas e recursos utilizando a tecnologia WebGL com o objetivo de gerar ambientes virtuais 3D, esta pesquisa delimita-se a trabalhar com *softwares* já existentes de criação e exportação de conteúdos tridimensionais virtuais para compor a sistemática relacionando o WebGL aos projetos de Design. Ainda, o processo não tem a intenção de abranger ensinamentos, conceitos e técnicas a respeito da modelagem tridimensional nos programas utilizados ou em quaisquer outros. Assim, trabalha-se aqui com objetos 3D modelados previamente pelo usuário ou obtidos já prontos por outros meios.

[1] Sistemática é o resultado de um processo de sistematização, que por sua vez é uma "interpretação crítica de uma ou várias experiências que, a partir de seu ordenamento e reconstrução, descobre ou explicita a lógica do processo vivido, os fatores que intervieram no dito processo, como se relacionaram entre si e porque o fizeram desse modo" (HOLLIDAY, 2006, p. 24).

1.4. JUSTIFICATIVA

É fato que a criação e manipulação de objetos tridimensionais virtuais são habilidades de muitos designers e projetistas em geral, especialmente animadores, desenvolvedores de jogos digitais ou ainda de produtos. Ainda que inerente a estes profissionais, quando acompanhada de novas tecnologias esta aptidão pode ser aprimorada em diversos sentidos.

São os benefícios proporcionados pela tecnologia WebGL aos trabalhos de Design que motivam esta pesquisa. Pode-se por exemplo, elevar a qualidade visual de apresentação dos projetos de Design Virtual, com visualizações interativas inseridas em contextos em plena evolução como a Realidade Virtual. Ou ainda, facilitar a distribuição de um conteúdo digital em formato HTML, devido a sua capacidade de execução multiplataforma, inclusive em dispositivos móveis (*smartphones e tablets*), sem a necessidade de complementos específicos e por meio de simples navegadores web (ZHANG; GRAČANIN, 2013, p. 195).

Uma das grandes chaves para que esta mistura de tecnologias funcione é a existência de *softwares* que trabalham de maneira muito semelhante aos já utilizados para modelagem tridimensional no Design, como por exemplo 3ds Max, Blender, SolidWorks e tantos outros, porém possuem a funcionalidade de acréscimo de interações e animações aliado à exportação deste conteúdo para esta Interface de Programação de Aplicativos (em inglês, *Application Programming Interface - API*) denominada WebGL (ver item 2.7).

Acredita-se que a aproximação destes *softwares*, utilizados no Design, que trabalham com criação de objetos tridimensionais virtuais e essa API, por meio de uma sistemática destinada à aplicação na área criativa por designers, pode facilitar o desenvolvimento e divulgação dos ambientes virtuais com maior qualidade gráfica e com formas de interação mais intuitivas, proporcionando uma ampliação de suas capacidades e resultando em um maior alcance na visibilidade e no uso dos mesmos. A aproximação da tecnologia WebGL dos designers, além de popularizá-la e trazer benefícios aos profissionais da área, pode também ajudar na sustentação da própria tecnologia como um duradouro padrão de descrição de cenas tridimensionais se considerar que segundo Danchilla (2012, p. 301), um dos pilares para alcançar este patamar é a adoção maciça da comunidade de desenvolvimento, onde se incluem os designers.

Corroborando com esta ideia, é notório que diversas tecnologias amplamente difundidas hoje começaram a se popularizar a partir do momento que tiveram seu entendimento e utilização facilitados para o usuário. Como exemplo, a plataforma de streaming de vídeos Youtube, criada em 2005, tornou acessível a qualquer pessoa a geração, distribuição e promoção de vídeos pela internet, tarefa antes restrita a profissionais especializados do áudio visual (FERREIRA, 2015, p. 01 e 02). Outro bom exemplo são os óculos de Realidade Virtual (RV) que se democratizaram por meio do Cardboard da Google, um produto composto basicamente por papelão e que por seu baixo custo e complexidade levou a experiência da RV onde os modelos mais complexos e caros ainda não chegaram, abrindo um novo nicho de mercado em crescente expansão (MALOSSI, 2015; MORAES, 2015; RIZZO, 2015).

Os benefícios deste trabalho podem atingir ainda o campo da educação. Igualmente ao realizado pelo Prof. Dr. José Luís Farinatti Aymone na disciplina de Design Virtual da pós-graduação em Design da UFRGS com relação a tecnologias como o VRML, futuramente o desenvolvimento de ambientes virtuais utilizando WebGL poderá ser ensinado a acadêmicos e interessados em Design Virtual, tendo como base uma sistemática especialmente desenvolvida para os profissionais da área. Por sua vez, estes aprendizes poderão dar sequência à pesquisa nesta área.

1.5. ESTRUTURA DO TRABALHO

Esta dissertação está dividida em oito capítulos, sendo o **primeiro** a Introdução, onde o problema de pesquisa é esclarecido, a hipótese inicial é construída e os objetivos, justificativa e delimitações são especificados.

O **segundo capítulo** (Design + WebGL) é composto por sete seções. Elas abordam os temas considerados relevantes para a compreensão do universo da pesquisa e buscam um entendimento, por meio de fundamentação teórica e prática, sobre assuntos como a representação tridimensional virtual no Design, a realidade virtual e aumentada, os ambientes tridimensionais virtuais, a programação orientada a objetos, as tecnologias de representação e disseminação de conteúdo 3D virtual e um aprofundamento sobre aspectos mais profundos do WebGL.

O **capítulo número três** (Materiais e Métodos) expõe em detalhes a metodologia utilizada para a realização da presente pesquisa. Neste capítulo a

pesquisa é classificada segundo sua abordagem, objetivos e procedimentos e são apresentados os métodos e materiais utilizados.

No **quarto capítulo** (Análise e Seleção de *Softwares*) estão descritas as etapas de levantamento de ferramentas com potencial de uso no trabalho; elaboração dos critérios para avaliação das ferramentas levantadas; descoberta dos requisitos da sistemática em desenvolvimento; e, baseado nestes critérios e requisitos, seleção dos *softwares* a serem utilizados diretamente na construção da sistemática.

O **quinto capítulo** (Sistemática) traz o resultado maior da pesquisa, a sistemática de criação de conteúdo virtual tridimensional interativo, utilizando a tecnologia WebGL através do programa Blender e seu complemento Blend4Web. O capítulo está dividido conforme as etapas da sistemática, desde a obtenção e instalação dos softwares utilizados até o processo de exportação do conteúdo pronto para ser distribuído e visualizado.

O **capítulo número seis** (Resultados e Testes) apresenta os resultados obtidos a partir da construção de ambientes-teste utilizando todos os recursos presentes na sistemática. Ao fim do capítulo realiza-se um comparativo entre as tecnologias VRML e WebGL.

O **sétimo capítulo** (Considerações Finais) relata as considerações finais acerca deste trabalho como um todo. Discutem-se as etapas realizadas, os resultados alcançados e as dificuldades enfrentadas ao longo de seu desenvolvimento. Também, neste capítulo apresentam-se os trabalhos mais relevantes produzidos em paralelo com esta pesquisa, no mestrado em Design na Universidade Federal do Rio Grande do Sul. E ao final, sugerem-se algumas possiblidades de pesquisa para futuros trabalhos na mesma área.

No **oitavo e último capítulo** (Referências Bibliográficas) estão todas as referências das citações utilizadas nesta dissertação. E por fim, logo após as referências, está o Apêndice A (Tabela de *Softwares* Analisados), que contém a listagem dos programas levantados e analisados durante o trabalho.

Capítulo 2

DESIGN + WEBGL

> "O mundo dos gráficos tridimensionais não tem fronteiras nem limitações [...] - podemos melhorá-lo colocando ainda uma quarta dimensão: a dimensão da nossa imaginação". (MAZURYK; GERVAUTZ, 1996, p. 01)

Para compreender como o Design ao longo dos anos agregou a representação virtual, em especial aquela que trabalha com a tridimensionalidade, às suas capacidades, buscou-se as bases conceituais e históricas e as tecnologias que influenciaram neste contexto e continuam aprimorando suas habilidades.

O presente capítulo apresenta as definições e exemplos, obtidos por revisão bibliográfica, referentes a temas de relevância para a construção do entendimento da inserção e utilização de tecnologias digitais na criação de ambientes tridimensionais virtuais por designers.

2.1. A REPRESENTAÇÃO TRIDIMENSIONAL VIRTUAL NO DESIGN

A representação pressupõe a duplicidade de qualquer coisa, material ou imaterial, física ou virtual (PANISSON, 2007, p. 45). No Design a representação tem por um de seus objetivos replicar conceitos em figuras e objetos que sirvam para visualizar ou comunicar algo. Em outras palavras, as representações são utilizadas para materializar ideias a serem comunicadas ou ainda possibilitar simulações durante um projeto.

Dentre os tipos de representações, destaca-se aqui as que apresentam tridimensionalidade. Um objeto tridimensional (3D) é aquele que apresenta volume e pode ser definido nas dimensões altura, largura e profundidade e que por sua vez está inserido em um ambiente também tridimensional, seja ele físico ou virtual.

A confecção de modelos tridimensionais físicos sempre foi de grande importância no Design, ganhando visibilidade até mesmo pela popularização da prototipagem rápida e das impressoras 3D. Seja por meio de modelos ou protótipos, os projetistas podem visualizar de uma melhor maneira sua criação e eventualmente testá-la, se necessário.

Paralelamente a eles, com avanço da computação gráfica e advento dos computadores pessoais, se tornaram comuns os modelos virtuais construídos dentro do âmbito computacional, para os mais diversos fins, a partir de ferramentas específicas como os *softwares* CAD (*Computer Aided Design*) (LIMA, 2010, p. 117).

O conceito moderno de CAD, em português Projeto Assistido por Computador, foi concebido em 1963 por Ivan Sutherland (Figura 1) e seu orientador no *Massachussets Institute of Technology* (MIT) quando desenvolveram um modo de integrar o projeto a um programa de análise criando também as primeiras ferramentas para implementar esse sistema. Com relação à utilização destas ferramentas no *Design*, bem como na Arquitetura, desde o fim da década de 90, houve um crescimento exponencial do uso da representação gráfica virtual, tanto no meio acadêmico como profissional (JUNG, 2014, p. 53 e 56).

Figura 1 - Ivan Sutherland, o precursor da tecnologia CAD.

Fonte: MIT MUSEUM (1963).

Tal modo de representação trouxe grandes vantagens se comparado ao que antes era feito à mão. São menores seus custos e o tempo de desenvolvimento, oferecendo maior flexibilidade com relação a alterações. Por estas razões, o uso da modelagem virtual é altamente interessante para as áreas de engenharia e Design industrial (BJOERKLI, 2014, p. 02).

Devido ao crescente avanço tecnológico computacional dos últimos anos, tudo aquilo que antes era feito com protótipos físicos, foi sendo possível de ser realizado virtualmente. Os objetos tridimensionais hoje podem ser simulados e apresentados de maneira fiel à realidade utilizando ferramentas digitais.

Atualmente, no Design de produtos, é praticamente indispensável o conhecimento de modelagem virtual, seja para a realização de testes, prototipação ou para a apresentação e divulgação do projeto a clientes e usuários. São muitos os usos da representação tridimensional virtual no Design e as ferramentas disponíveis de criação de objetos e ambientes 3D continuam evoluindo e oferecendo cada vez mais realismo e capacidades de animação e interação para o conteúdo desenvolvido.

2.2. A REALIDADE VIRTUAL

Sabe-se que o mundo é compreendido pelos seres humanos através de seus sentidos e sistemas de percepção. Tudo o que se sabe sobre a nossa realidade vem por meio dos sentidos. Em outras palavras, toda nossa experiência da realidade é simplesmente uma combinação de informações sensoriais e nossos cérebros, mecanismos que determina sentido à essas informações. Parece lógico então, que se forem apresentados aos sentidos informações mentirosas ou inventadas, a percepção da realidade também mudará em resposta a isso. A pessoa será apresentada à uma versão da realidade que não está realmente lá, mas a partir de sua perspectiva, seria percebido como real, gerando assim a chamada Realidade Virtual (VIRTUAL REALITY SOCIETY, 2016a).

Sobre a história por trás da Realidade Virtual (RV), Kirner e Siscoutto (2007, p. 04) explicam, que as primeiras interfaces computacionais, usadas nas décadas de 40 e 50, somente permitiam uma comunicação com o computador baseada em linguagem de máquina, algo muito distante dos conhecimentos dos leigos no

assunto. Anos depois, surgiram os consoles com vídeo, dando início às interfaces gráficas rudimentares, mas ainda utilizadas apenas por especialistas. A partir do avanço tecnológico na área e a popularização dos computadores pessoais isso foi mudando.

> Com a utilização de microprocessadores, nas décadas de 70 e 80, os microcomputadores se popularizaram, usando interface baseada em comando, como o DOS. A evolução desta interface resultou no Windows, que, explorando técnicas de multimídia, persiste até hoje. Apesar de interessante e de ter bom potencial de uso, a interface Windows fica restrita à limitação da tela do monitor e ao uso de representações como menus e ícones (KIRNER; SISCOUTTO, 2007, p. 04).

Como parte desta busca por melhores interações e interfaces homem/máquina, surgiu então a Realidade Virtual (RV), que apesar de ter ganhado força apenas nos anos 90 e estar em um estágio de popularização atualmente, não é uma área de pesquisa nova. Investigações no campo da RV vem sendo realizadas há décadas, tendo sido reconhecida como uma poderosa ferramenta para a criação de interfaces mais naturais e intuitivas entre computadores e humanos.

A Virtual Reality Society (2016) explica que realidade virtual é o termo utilizado para descrever um ambiente tridimensional, gerado por computador, que pode ser explorado e interagido por uma pessoa. Essa pessoa torna-se parte deste mundo virtual ou está imerso nesse ambiente e lá é capaz de manipular objetos ou executar uma série de ações.

A RV, conforme Pan et al. (2006, p. 20), consiste na utilização de sistemas de computação gráfica em combinação com dispositivos de visualização e de interface para proporcionar um efeito de imersão no ambiente interativo tridimensional gerado por computador. Na definição de Tori e Kirner (2006, p.06), em outras palavras, a Realidade Virtual é uma “interface avançada do usuário” para acessar aplicações executadas no computador, tendo como características a visualização de ambientes tridimensionais em tempo real e a interação com elementos desse ambiente. A experiência ainda pode ser aprimorada com estímulos a outros sentidos humanos como o tato e a audição.

Assim como qualquer nova tecnologia, a interação com os dispositivos de RV necessita de um tempo de aprendizado, ainda que este seja curto. Mas a grande vantagem das interfaces de RV é o fato de as habilidades e conhecimentos intuitivos do usuário poderem ser utilizados para a manipulação dos objetos virtuais, como o movimento natural da cabeça e mãos por exemplo.

> O importante é que haja por parte do usuário a impressão de estar atuando dentro do ambiente virtual, apontando, pegando, manipulando e executando outras ações sobre os objetos virtuais, em tempo real, ou seja, dentro de limites de tempo bem definidos, ou com atrasos que não lhe causem desconforto (TORI; KIRNER, 2006, p. 06).

As primeiras ideias de realidade virtual surgiram dos cineastas na década de 50, mas foi nos anos 60 com os estudos de Ivan Sutherland, que se introduziu os primeiros conceitos e ferramentas do Projeto Assistido por Computador. Em 1965 Sutherland disse que queria fazer com que esse mundo (virtual) representasse o olhar real, o som real, parecesse real e respondesse de forma realista às ações do espectador (MAZURYK; GERVAUTZ, 1996, p. 02).

Embora fossem grandes as ambições de Ivan Sutherland já em 1965, no princípio eram necessários diversos equipamentos especiais como luvas, capacetes, óculos estereoscópicos, mouses 3D e outros, para que o usuário conseguisse interagir com o ambiente virtual (KIRNER; SISCOUTTO, 2007, p. 05). Desde então tais limitações vem sendo reduzidas ou eliminadas à medida que evoluem as tecnologias de RV.

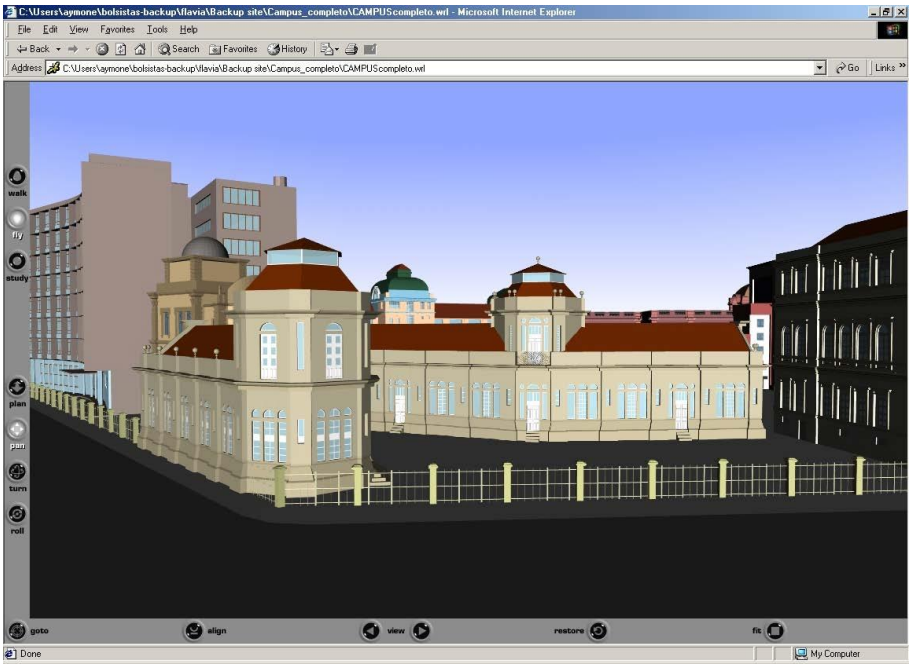
Nos anos 90 se tornou popular o VRML (*Virtual Reality Modeling Language*), uma linguagem de representação de objetos e ambientes tridimensionais que, na época, rapidamente se tornou o formato de arquivo padrão para a transmissão de "mundos virtuais 3D" através da internet (TAUBIN et al., 1998, p. 1228). Na literatura identifica-se inclusive a forte importância do VRML como uma linguagem que permitem a preparação e a interação com objetos virtuais para a Realidade Virtual como um todo. Quando em seu auge, o VRML foi identificado como um dos recursos mais populares nesta área (KIRNER; TORI, 2006, p. 29).

Mantendo sua importância para a área de Realidade Virtual nos anos 2000, as diversas possibilidades trazidas na época pela tecnologia VRML foram colocadas a

prova em uma pesquisa realizada pelo Prof. Dr. José Luís Farinatti Aymone da Universidade Federal do Rio Grande do Sul (UFRGS), que desenvolveu um modelo virtual do Campus Central da UFRGS em Porto Alegre (AYMONE, 2003).

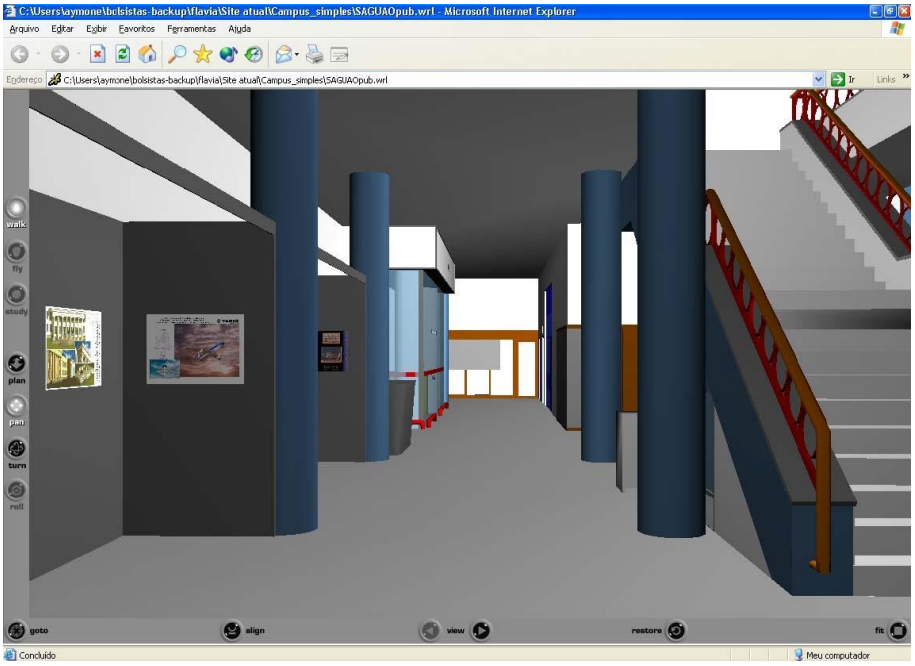
Foram utilizados modelos tridimensionais de prédios da UFRGS criados em trabalhos anteriores para, juntamente com novos elementos, compor um cenário interativo que possibilitou a realização de um *tour* virtual por entre as edificações do Campus (Figura 2) e áreas internas da Faculdade de Arquitetura da UFRGS (Figura 3). Diante das limitações de processamento computacional e de internet da época, a pesquisa obteve ótimos resultados com a otimização dos modelos tridimensionais utilizados, propiciando uma experiência de realidade virtual fluída e agradável que pôde ser utilizada para fins de ensino posteriormente.

Figura 2 - Visualização externa dos prédios do Campus Central da UFRGS em realidade virtual.



Fonte: AYMONE (2003, p. 02).

Figura 3 - Saguão do prédio da Faculdade de Arquitetura da UFRGS em realidade virtual.



Fonte: AYMONE (2003, p. 02).

Com o passar dos anos a realidade virtual (RV) amadureceu e se tornou uma tecnologia útil para muitas áreas que não só o ensino, como a medicina, a construção civil, o entretenimento, etc. No Design de produto tecnologias de RV podem ser ferramentas de projeto eficientes e eficazes desde a fase inicial do processo de concepção até atividades específicas do projeto, como por exemplo, a gestão do ciclo de vida do produto (YE et al., 2007, p. 1193). Um ambiente de desenvolvimento de produto baseado em RV pode fornecer uma melhor visualização, interação e manipulação do produto antes mesmo de ser posto na linha de produção.

Seja para estes ou outros fins, estas tecnologias estão remodelando as interfaces e interações das pessoas com os ambientes virtuais oferecendo novas abordagens para a comunicação de informações, a visualização de processos, expressão de ideias, o entretenimento e muito mais.

Em se tratando de entretenimento, é por ele que estamos vendo a realidade virtual se popularizar, se tornando mais presente na vida de todos e com promessas de revolucionar nossas vidas. Recentemente algumas iniciativas de empresas de tecnologia trouxeram ao mercado produtos que, semelhantes aos primeiros dispositivos especiais de RV, possibilitam tanto a visualização como a interação do usuário com os mais diversos ambientes 3D. Em sua maior parte, tais iniciativas estão

voltadas ao mundo dos jogos digitais e dos filmes, oferecendo um apelo mais do que suficiente para que todos queiram usufruir da tecnologia. Algumas das iniciativas recentes mais relevantes neste universo da realidade virtual e entretenimento são o Oculus Rift e o Cardboard.

O Oculus Rift é um *display* de realidade virtual com áudio integrado e com formato que lembra em parte uns óculos. Criado pela empresa americana Oculus e anunciado pela primeira vez no ano de 2012 ainda em fase de desenvolvimento, teve sua primeira versão voltada ao consumidor final (Figura 4) somente em 2016. Visto que anteriormente algumas empresas já tentaram criar um produto parecido, o Rift é talvez o mais funcional e o primeiro a ter chances reais de fixação no mercado.

Figura 4 - Versão final do Oculus Rift, destinada ao consumidor final.

Fonte: PAGET (2016).

Este dispositivo deu o início à promissora era da RV, inicialmente tendo como apelo os jogos digitais e posteriormente sendo abastecido de aplicações para as mais diferentes finalidades. A empolgação e as promessas em torno do Oculus Rift foram tão grandes, que em março de 2014 o CEO do Facebook, Mark Zuckerberg, anunciou a compra da empresa Oculus por cerca de dois bilhões de dólares e desde então é a responsável pelo ciclo de desenvolvimento contínuo e promoção deste produto.

Frente ao cenário criado pelo Oculus Rift, em 2014 foi a vez da Google apresentar a sua proposta de visualizador de realidade virtual. Buscando uma

maneira de levar o maior número de pessoas possível a experimentar a RV. E considerando que grande parte da população mundial tem hoje um *smartphone*, a Google criou então o Cardboard (Figura 5). Um produto essencialmente composto por papelão, com custo de produção extremamente baixo e especificações disponíveis para qualquer um montá-lo facilmente e que em conjunto com um *smartphone* pode trazer a experiência da RV utilizando os princípios básicos do próprio Oculus Rift.

Figura 5 - Primeira versão do Google Cardboard lançada em 2014.

Fonte: CAMPO (2016).

Sendo livre a sua reprodução, rapidamente surgiram diversos outros modelos, comerciais ou não, que utilizavam a lógica de funcionamento do produto da Google. Alguns destes modelos são confeccionados em diferentes materiais, oferecendo até mesmo maior conforto e funcionalidades em comparação a ideia “popular” da gigante de tecnologia. Em 2015 a Google ainda lançou uma nova versão aprimorada do Cardboard, mas que em essência pouco mudou.

Foi então que no embalo destas duas bem-sucedidas iniciativas e no promissor mercado criado, outras grandes empresas de tecnologia resolveram apostar na RV e trouxeram novos dispositivos que estão rapidamente ganhando força, esse é o caso do Vive da HTC, do Playstation VR da Sony, do Gear VR da Samsung e do Daydream View, outro produto de RV da Google. Para complementar

ainda mais essas novas experiência de RV com um maior número de respostas sensoriais aos usuários, alguns outros produtos estão sendo desenvolvidos como luvas, joysticks, esteiras omnidirecionais e uma série de sensores que mapeiam os movimentos e o ambiente no qual ele está inserido.

Por sua vez, toda a parte de *software* para trabalhar em conjunto com estes dispositivos está avançando também a largos passos. Para suprir a necessidade de aplicativos e conteúdos interativos para estas novas plataformas, desenvolvedores aliam a tradicional forma de criação de ambientes tridimensionais, muita utilizada nos jogos digitais, com os requisitos de interação e sensações próprias da realidade virtual.

E para acompanhar o desenvolvimento de conteúdos para estes dispositivos, os *softwares* de criação de conteúdo tridimensional virtual também estão se adaptando aos novos requisitos dos projetos destinados à realidade virtual. Dentre os *softwares* de criação de ambientes interativos 3D, a Unreal Engine, a partir de sua versão 4.12, se destaca por oferecer um modo de edição (Figura 6) no qual é possível utilizar como manipulador os próprios dispositivos para os quais o conteúdo está sendo criado, levando a interação intuitiva e natural, própria da RV, para as etapas de criação das aplicações.

Figura 6 - Modo *VR Editor* da Unreal Engine que utiliza um editor em realidade virtual para construir ambiente para a própria RV.

Fonte: PASCHALL (2016).

Surge então uma nova forma de desenvolver ambientes tridimensionais virtuais destinados à realidade virtual, que sejam reproduzidos tanto em computadores pessoais como em dispositivos móveis. Esse e outros requisitos características dos conteúdos para RV se encontram com as possibilidades da tecnologia WebGL.

Visualizando as vantagens da utilização do WebGL e seu futuro promissor para a criação deste tipo de conteúdo, grandes empresas, como a Mozilla e a Google, estão investindo na construção de ferramentas para esta finalidade. A Mozilla criou uma biblioteca de desenvolvimento para WebGL (ver item 2.7.1), chamada A-Frame, com suporte total a reprodução em Realidade Virtual e de fácil aprendizado. Já a Google, utilizando o WebGL, oferece na página do *Chrome Experiments for Virtual Reality* arquivos base para a inserção de cenários e objetos 3D em ambientes preparados para a RV.

2.3. A REALIDADE AUMENTADA

Considerada por diversos autores parte da Realidade Virtual e por tantos outros uma tecnologia paralela à RV, a Realidade Aumentada (RA) ou Realidade Mista como explica Pan et al. (2006, p. 20), refere-se à incorporação de objetos gráficos virtuais em uma cena real tridimensional, ou, alternativamente, a inclusão de elementos do mundo real em um ambiente virtual. Enquanto a Realidade Virtual "mergulha" os sentidos em um mundo totalmente digital, a Realidade Aumentada projeta objetos digitais, como modelos 3D, vídeos, imagens e sons em nossa visão da realidade, como se realmente estivessem lá (VIRTUAL REALITY SOCIETY, 2016b).

Em outras palavras, a RA mantém a presença do mundo real, enquanto que, na RV "pura" o ambiente é todo virtual e controlado por computador. A colocação de conteúdos virtuais, de duas ou três dimensões, no ambiente real e com uma interação processada em tempo real, facilita a visualização e possíveis análises que correlacionem o objeto virtual e o ambiente físico. Um exemplo são os aplicativos de RA onde é possível visualizar como certo mobiliário ou peça de decoração ficará em um cômodo de uma casa, antes mesmo de comprá-lo, reduzindo incertezas (Figura

7). Como ferramenta é utilizada por diferentes profissionais e fins na área médica, engenharias, publicidade e muitos outros.

> O fato dos objetos virtuais serem trazidos para o espaço físico do usuário (por sobreposição) permitiu interações tangíveis mais fáceis e naturais, sem o uso de equipamentos especiais (KIRNER; SISCOUTTO, 2007b, p.05).

Figura 7 - Exemplo de visualização de mobiliários sobrepostos a um ambiente físico a partir da Realidade Aumentada.

Fonte: GUÉNO (2014).

A vantagem de realidade aumentada sobre livros, por exemplo, ou outras fontes de dados *off-line* é que a informação pode ser apresentada no mesmo local que o objeto que se relaciona. Isto fornece o contexto para a informação, muitas vezes tornando-o mais atraente e fácil de entender. A capacidade de contextualizar e localizar informação virtual é uma das maiores forças da tecnologia de realidade aumentada (WITHER; DIVERDI; HOLLERER, 2009, p. 679).

Historicamente a tecnologia foi propiciada pela evolução tecnológica na década de 90, permitindo a sobreposição de objetos virtuais com o ambiente físico, através de algum dispositivo tecnológico. Entretanto, essas aplicações somente se tornaram mais acessíveis a partir do início dos anos 2000, com a convergência de técnicas de visão computacional, *software* e dispositivos com melhor custo-benefício, possibilitando uma melhor qualidade da experiência em RA e ampliando suas possibilidades de aplicação (KIRNER; SISCOUTTO, 2007b, p. 05).

O momento favorável dos últimos anos, em termos de avanços tecnológicos, criou uma série de expectativas para uma nova geração de interfaces baseadas em RA. Mais uma vez, como o ocorrido na Realidade Virtual, a área do entretenimento é uma das que mais contribui para a popularização da tecnologia pois por meio dela se atinge escalas de consumo de conteúdo bastante altas. Praticamente ao mesmo tempo em que ocorreu a recente explosão de entusiasmo e avanços com a realidade virtual, a realidade aumentada também despertou interesses das gigantes da tecnologia que trouxeram produtos promissores e com abrangência.

Novamente a Google deu os primeiros passos da "era moderna" da realidade aumentada com o Google Glass e o Project Tango. Mesmo a RA sendo uma tecnologia utilizada há alguns anos e de relativa facilidade de acesso, a Google trouxe a ela novos patamares e aplicações mais robustas.

O Google Glass (Figura 8), uma espécie de óculos que utilizando a RA e uma Inteligência Artificial, projeta em uma tela transparente em frente aos olhos do utilizador, informações relacionadas a ações do próprio usuário ou do ambiente ao seu entorno (Figura 9). O produto foi um sucesso em termos de expectativas e promessas e por alguns anos contou com a ajuda de muito desenvolvedores e entusiastas em seu desenvolvimento. O produto da Google não chegou ao público em geral com um bom custo/benefício e perdeu um pouco do seu encantamento. Atualmente existem alguns rumores de que o projeto foi extinto pela própria Google que talvez novamente surpreenda o mundo com uma evolução do produto em questão.

Figura 8 - Google Glass, uma das apostas da Google com dispositivos inteligentes vestíveis e que utiliza a RA.

Fonte: KEATS (2014).

Figura 9 - A ideia do Google Glass é sobrepor a visão do mundo real do utilizador com informações úteis e respostas às suas ações.

Fonte: AXIOM (2013).

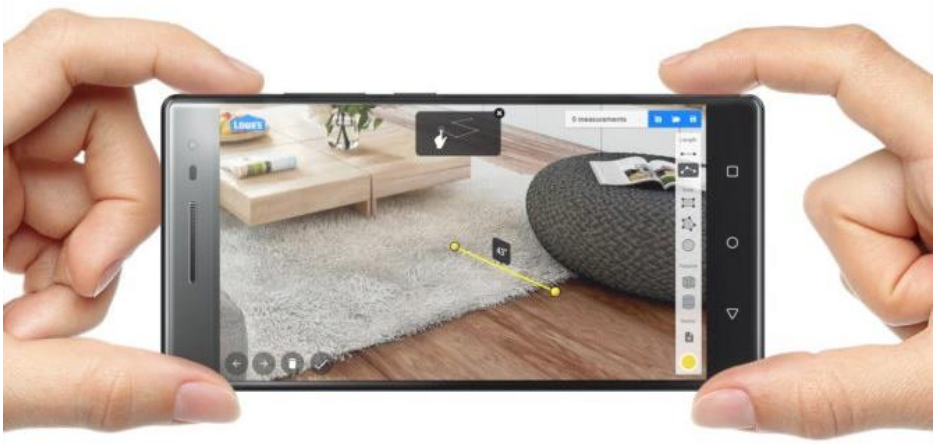
A nova aposta da Google foi então o Project Tango. Uma tecnologia a ser incorporada em dispositivos móveis (*smartphones* e *tablets*), que através de uma série de sensores permite que os aparelhos detectem a sua posição e mapeiem em 3D o mundo em torno deles sem utilizar GPS ou quaisquer outros sinais externos. Usando um projetor de infravermelhos e um par de câmeras extras, ele pode detectar coisas como paredes e pisos com certa precisão ou capturar uma varredura 3D detalhada de um espaço e isso representa um grande potencial de expansão para a realidade aumentada.

O Project Tango permite que aplicativos rastreiem posição e orientação de objetos dentro de um ambiente 3D detalhado. Isso torna possível aplicações como, medição visual, utilitários de mapeamento, apresentação e ferramentas de Design, bem como uma grande variedade de jogos imersivos. Tais aplicações funcionam a partir de princípios da Realidade Aumentada, porém esta nova tecnologia da Google aprimora, e muito, as possibilidades.

O primeiro aparelho comercial a embarcar o Project Tango é o Lenovo Phab2 Pro (Figura 10), um *smartphone* que terá a chance de provar se a tecnologia terá sucesso pelo seu potencial ou será deixada de lado em pouco tempo como o ocorrido com o Google Glass. De uma maneira ou outra, é um marco diante dos esforços de

aperfeiçoar a realidade virtual como um todo e mais especificamente a realidade aumenta para fornecer ferramentas realmente úteis ao ser humano.

Figura 10 - Lenovo Phab2 Pro, primeiro smartphone com a tecnologia Project Tango, realizando a medição de objetos a distância.



Fonte: URBANO (2016).

Outro aparelho de RA e de funcionalidades promissoras é o HoloLens da Microsoft (Figura 11). Mesmo ainda não sendo possível encontra-lo nas lojas o HoloLens já encheu de expectativas aqueles que sonham unir o “mundo real” ao encantamento e possiblidades dos ambientes virtuais. Funciona assim, basicamente coloca-se o dispositivo, que se parece com uns óculos, em sua cabeça e o visor “projeta” conteúdo sobre o ambiente que você estiver vendo.

Figura 11 - O HoloLens pode ser um instrumento profissional importante.

Fonte: MICROSOFT (2016).

Ele possibilita uma interação em tempo real, com gestos naturais e com processamento interno, sem a necessidade de estar conectado a um computador como no caso dos óculos de realidade virtual Oculus Rift e HTC Vive. Esta interação também pode ser compartilhada com outras pessoas que estejam utilizando o produto, ampliando ainda mais o seu potencial. Assim, sendo para jogar ou trabalhar, a Microsoft aposta que o HoloLens será a grande ferramenta de mistura dos mundos físico e virtual dos últimos anos.

2.3.1. Elaboração de conteúdo para a Realidade Aumentada

Pode-se dizer que a realidade aumentada já é popular entre pesquisadores e entusiastas há alguns anos. Prova disso é um dos estudos do Prof. Claudio Kirner de 2004 onde ele utilizou uma ferramenta de criação de realidade aumentada para aprimorar a visualização de dados na forma de gráficos tridimensionais (Figura 12).

Figura 12 - Gráficos tridimensionais sendo visualizados por meio da realidade aumentada.

Fonte: KIRNER et al. (2004, p. 08).

Desde os anos 90 surgiram diversos *softwares* que auxiliavam na criação de experimentos simplificados com RA. Para usá-los era necessário geralmente dispor

de um computador com webcam instalada, que lia informações de um código visual posicionado no ambiente físico. Com o tempo apareceram os *smartphones* e *tablets*, e então foram criados novos aplicativos que utilizam os componentes presentes neles próprios, como o seu processamento, tela e câmera.

Em se tratando da tecnologia WebGL como suporte para a criação de Realidade Aumentada, assim como na Realidade Virtual, recentemente foram criadas algumas bibliotecas de desenvolvimento que trabalham também com a RA. A que se destaca é a *awe.js*, que tem como foco a RA e oferece uma documentação facilitada para que iniciantes possam trabalhar com ela. A grande vantagem de aliar o WebGL a RA é poder distribuir os conteúdos gratuitamente para diversas plataformas sem a necessidade de se ter um aplicativo específico para visualizá-lo a não ser um navegador de internet qualquer, já presente em qualquer computador, *smartphone* ou *tablet*. Outro benefício é a ampliação da gama de interações possíveis de serem realizadas, podendo até mesmo correlacionar interferências realizadas em objetos virtuais projetados através da RA com ações no mundo físico, como por exemplo, enviar comandos para alterar a cor da iluminação de um ambiente.

2.4. OS AMBIENTES TRIDIMENSIONAIS VIRTUAIS

Após a compreensão de como se insere a representação tridimensional virtual no Design e a importância, influência e aplicação das realidades virtual e aumentada desde o princípio até os dias de hoje, é necessário entender como são definidos e construídos os ambientes em três dimensões, que são o centro do assunto e fundamentais para o seu desenvolvimento.

> Um ambiente envolve um determinado espaço e uma situação delimitada, incluindo todos os componentes neles inseridos, como o conjunto de objetos e de condições passíveis de serem percebidos e com os quais é possível interagir. Um ambiente virtual é um ambiente interativo, gerado por um computador e disponibilizado através de um sistema de realidade virtual (Stuart, 1996 apud KIRNER; SALVADOR, 2007, p. 91).

Definido resumidamente por CUNHA e MAINENTE (2011, p. 05), um ambiente virtual é um cenário tridimensional, gráfico e interativo gerado por computador, com o objetivo de possibilitar uma representação cibernética similar ao mundo físico, onde o usuário pode mover-se através dele e com recursos que propiciam uma sensação de profundidade e imersão.

Em vista disto, um ambiente virtual tridimensional (AV3D) possui ao menos quatro características essenciais que o identificam, como a evidente representação em três dimensões; a interatividade, no mínimo a nível de navegação; a imersão, em mundos que imitam a realidade ou cenários inteiramente inventados e a inserção em um espaço também tridimensional. Tal espaço, que também é definido pelas dimensões altura, largura e profundidade, é aquele onde estão inseridos os objetos, animações e interações que em conjunto caracterizam um ambiente 3D por completo.

Como explicita NASCIMENTO (2010, p. 09), juntamente com a animação e a interação (estudos a parte nos itens 2.4.1 e 2.4.2), os objetos 3D são elementos essenciais de um cenário tridimensional (Figura 13), uma vez que representam as entidades materiais do mundo real emulado pelo ambiente virtual. De acordo com o uso ao qual o objeto se destina, diferentes técnicas de criação, representação e realismo podem ser adotadas.

Figura 13 - Exemplo de ambiente virtual (cidade) composto por objetos 3D (edifícios) e inserido em um espaço tridimensional delimitado.

Fonte: GHALL (2011).

Os AV3D servem tanto para simular lugares físicos no mundo real, bem como mundos imaginários. Estão presentes em uma ampla gama de aplicações da educação e formação, à exploração científica e visualização, Design e prototipagem, diagnóstico médico, entretenimento digital e muitos outros. Existe uma busca constante para se criar experiências virtuais com cada vez mais fidelidade, o que na maioria das vezes está ligado diretamente ao estágio atual de desenvolvimento do poder de processamento da computação gráfica (ZHANG; WU, 2012, p. A13).

Hoje, em diversas áreas os ambientes 3D virtuais são de grande importância e talvez seja no Design onde eles encontram a maior pluralidade de aplicações. Equilibrando técnica e criatividade, designers fazem uso dos AV3D para criar jogos digitais, visualizações realistas de produtos, cenários cinematográficos, representações técnicas, simulações computacionais, ambientes educacionais, material publicitário e o que mais a imaginação permitir.

Quanto a visualização e distribuição destes ambientes virtuais, em particular por meio da internet, houve uma constante evolução desde o VRML até o WebGL. Projetado para fornecer uma maneira padronizada para descrever informações 3D interativas para aplicativos baseados na Web, o VRML se estabeleceu nos anos 90 como a forma mais eficiente de se construir e disseminar conteúdos tridimensionais, podendo ser gerado por *softwares* de modelagem 3D facilmente (KENT; WILLIAMS, 2002, p. 180).

Apesar das vantagens trazidas pelo VRML e da significativa quantidade de trabalhados realizados com ele na época, aos poucos ele foi sendo substituído por tecnologias com menos limitações e que aproveitavam melhor os novos recursos advindos da evolução da computação. Basicamente como um aperfeiçoamento do VRML, mais maduro e refinado, o X3D, uma arquitetura de formato de arquivo padronizada, também tinha por objetivo representar e comunicar cenas 3D e objetos (WEB3D CONSORTIUM, 2016). Ainda que prometendo diversos avanços com relação ao VRML, o X3D não se consolidou da mesma maneira que seu antecessor, que continuou sendo utilizado mesmo que perdendo no decorrer dos anos.

Algumas iniciativas no âmbito educacional ainda mantêm vivo o padrão de linguagem para modelagem de realidade virtual, popular nos últimos anos do século XX. Uma delas é a disciplina de Design Virtual da pós-graduação em Design da Universidade Federal do Rio Grande do Sul, ministrada pelo professor doutor José

Luís Farinatti Aymone, que apresenta aos alunos os principais conceitos e estruturas do VRML como forma de introduzi-los neste universo.

No entanto, hoje, o promissor WebGL busca uniformizar a programação de gráficos 3D para distribuição *online* ou *off-line*. Devido às suas vantagens em relação às tecnologias anteriores (ver item 2.7), está sendo rapidamente adotado como o novo e promissor padrão, principalmente para a Web. O sucesso do WebGL gera interesse em grandes e pequenos do mundo da tecnologia e diariamente suas aplicações vêm sendo aprimoradas.

Vale ressaltar que as interfaces e padrões voltados a visualização e compartilhamento de conteúdo 3D não se resumem somente a estas três tecnologias, todavia estas são as que tiveram maior representatividade e impacto ao longo dos últimos anos e serão as estudadas neste trabalho.

2.4.1. A animação em ambientes virtuais 3D

Característica fundamental em um ambiente tridimensional interativo, a animação virtual, de um objeto, câmera ou cenário como um todo, existe quando ocorre mudança na forma dos componentes ou cena com o tempo, em diferentes níveis, através de ativadores ou de comportamentos pré-determinados (PAL; SARKAR, 2002, p. 169). Essa alteração de forma dos objetos compreende a transformação de determinados parâmetros, como por exemplo posição, tamanho, cor, textura e outros.

“A palavra animação provém do latim Animus/Anima, que significa ar, respirar, vida, alma e mente. Animar é então dar a ilusão de vida no que está inanimado” (LUZ, 2009, p. 921). Fundamentalmente, na animação virtual existe um comportamento de mudança de estado atribuído à um objeto também virtual.

A animação tridimensional apresenta algumas características que a diferem da animação em duas dimensões (bidimensional). Ela procura simular filmagens ou perspectivas do mundo real, possibilitando ver uma cena criada em computador, de vários ângulos ou pontos de vista, pois se trata da criação de modelos num espaço tridimensional, possuindo comprimento, largura e profundidade (DUARTE, 2016).

Apesar dos outros tantos parâmetros possíveis de serem trabalhados, o movimento foi sempre o objetivo central da animação. Simplificando seu conceito, até pode-se dizer que animar é o processo de dar movimentos a câmeras, luzes ou objetos que fazem parte da cena. Para tal, existem diversas técnicas de criação de movimentos totalmente gerados por computador e captura de movimento a partir de objetos reais, para serem inseridos em ambientes virtuais.

Naturalmente as técnicas utilizadas para desenvolver uma animação variam de acordo com o objetivo proposto e o grau de experiência do animador, podendo ir de um simples deslocamento ou rotação de um objeto 3D rígido em uma cena até mesmo um complexo sistema de animação que simula as colisões tal qual aconteceriam com os objetos físicos, como já é utilizada em grandes estúdios de animação como a Pixar (Figura 14).

Figura 14 - Dedo do personagem se deforma de forma realista com o sistema de colisões utilizado pelos Estúdios Pixar.

Fonte: MURATTAHAN (2012).

Independentemente do tipo de animação, 2D, 3D, *stop-motion* ou de técnica mista, programada ou executada em tempo real, pode-se dizer que ela é formada substancialmente por uma sequência de imagens estáticas, chamadas quadros, cada

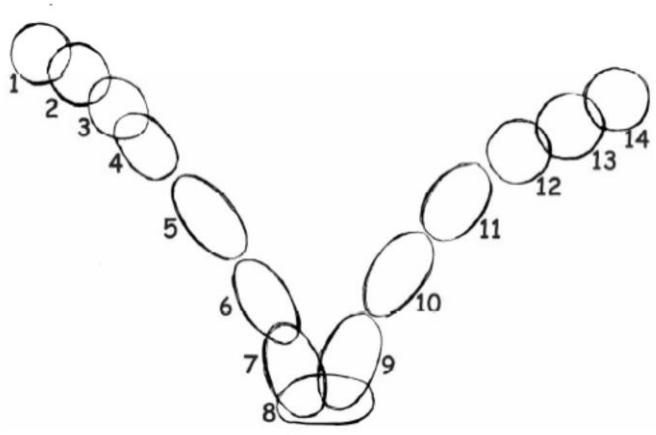
qual com um estado da forma diferente do anterior, que em conjunto representam um movimento ou outra alteração visual no objeto animado. Quanto maior o número de quadros por segundo de animação, maior a fluidez da mesma e menor a sensação visual de cortes durante uma movimentação, por exemplo.

Cenas de animação tridimensional geralmente contêm alguns objetos estáticos agrupados com a função de ambientação e objetos animados que mudam seu estado ao longo do tempo. Além disso, as cenas são vistas usando câmeras virtuais e podem ser iluminadas por fontes de luz sintéticas. Estas câmeras e luzes também podem sofrer alteração ao longo do tempo, como se manipulado por operadores de câmera (THALMANN, 1993, p. 05). O problema a se resolver é saber utilizar o método mais adequado para tornar algo animado.

Os métodos de animação 3D virtual não diferem muito daqueles utilizados para criar os antigos desenhos animados desenhados originalmente a mão. De acordo com o Prof. Eliseu Lopes (2005, p. 41, 42 e 43) existem basicamente dois métodos para se animar:

- **ANIMAÇÃO DIRETA (*STRAIGHT-AHEAD*)** – que consiste em ir desenhando quadro a quadro (Figura 15), um após o outro em sequência, desde a condição inicial do objeto, personagem, câmera, etc., até último ponto do curso da animação.

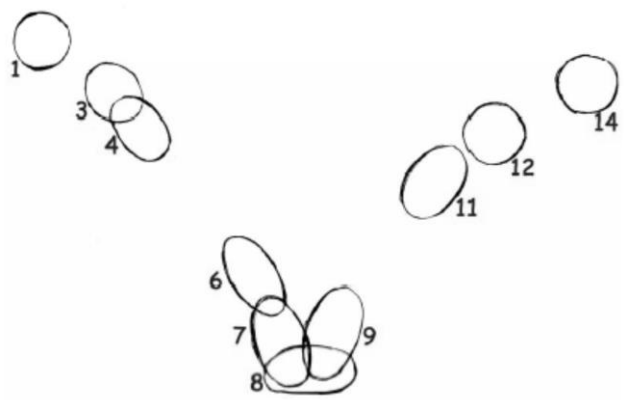
Figura 15 - No caso desta animação os 14 desenhos (quadros) que compõem o movimento foram desenhados um seguido do outro em animação direta.



Fonte: (LOPES, 2005, p. 42).

- **ANIMAÇÃO DE EXTREMOS (*POSE-PLANING*)** – neste se usa as pontas de animação ou posições chave (Figura 16). Consiste em desenhar primeiro as posições extremas ou principais de uma ação e posteriormente utilizar alguma técnica ou ferramenta que auxilie na interpolação deste, criando assim os quadros intermediários.

Figura 16 - Neste caso, os desenhos chaves ou pontas precisam ser interpolados.



Fonte: (LOPES, 2005, p. 42).

Geralmente animações criadas com o método de Animação Direta possuem baixas taxas de quadros por segundo por ser um processo mais trabalhoso. Porém, ainda que penoso, é bastante utilizado em animações confeccionadas através da sucessão de fotografias feitas uma a uma, o chamado *stop-motion*. Já o método de Animação de Extremos é o mais eficiente quando se trata de animação 3D, devido à facilidade com que o os programas de animação intercalam estas pontas.

Wilson Paula (2000, p.24) destaca que a qualidade gráfica ou quantidade de quadros por segundo da animação criada para ser executada em tempo real, como em dispositivos de Realidade Virtual ou Aumentada, é limitada pela velocidade de processamento do dispositivo utilizado. Este se configura como um dos principais desafios da RV e RA para proporcionar uma imersão cada vez mais próxima ao mundo físico. Qualidade gráfica e desempenho de visualização devem ser equilibrados durante o desenvolvimento de uma aplicação para RV.

A animação para realidade virtual tende a ser cada vez mais baseada na física e em métodos de simulação dinâmica. Como visto, já se utiliza hoje dispositivos de RV e métodos de simulação para a criação de complexas animações e ambientes

virtuais por inteiro, construídos para serem executados em tempo real e com interações progressivamente mais naturais.

Independentemente do método utilizado, a animação 3D é utilizada em muitas indústrias, como a do entretenimento, publicidade e técnica científica. Conforme BEANE (2012), cada indústria usa a animação em 3D de maneiras completamente diferentes e para saídas distintas, incluindo filmes, jogos, prototipagem rápida, e muitos outros. Sem dúvida o Design se faz presente em todas estas áreas e seus produtos e por isso a animação é considerada uma ferramenta importante para designers que desejem trabalhar nestes setores.

2.4.2. A interação em ambientes virtuais 3D

Também essencial nos ambientes virtuais, a interação segundo ZORZAL et al. (2007, p. 262) desperta o interesse dos usuários e é capaz de potencializar o entendimento de determinada visualização. Interagindo com um sistema, o usuário pode alterar a visualização do mesmo a fim de alcançar objetivos.

"No contexto de interface homem-máquina, interação é a maneira com que o usuário se comunica com a aplicação, podendo esta comunicação ocorrer através de dispositivos ou de forma simbólica" (SCHNEIDERMAN; PLAISANT, 2004 apud KELNER; TEICHRIEB, 2007, p. 53).

Outra forma de conceitualizar a interação nos ambientes virtuais 3D é identificando seus principais tipos, que segundo ROGERS, SHARP, PREECE (2013, p. 47) são quatro:

- **Instrução** – quando os usuários emitem instruções a um sistema. Isso pode ser feito de inúmeras maneiras, incluindo: digitar comandos, selecionar opções de menus em um ambiente de janelas ou em uma tela multitoque, comandos sonoros, gesticular, pressionar botões ou usar uma combinação de teclas de função;
- **Conversação** – quando os usuários têm um diálogo com um sistema. Eles podem falar por meio de uma interface ou escrever perguntas que o sistema responde via texto ou saída de voz;

- **Manipulação** – quando os usuários interagem com os objetos em um espaço virtual ou físico, manipulando-os (por exemplo, abrir, segurar, fechar, colocar);
- **Exploração** – quando os usuários se movem por um ambiente virtual ou um espaço físico. Ambientes virtuais incluem mundos 3D e sistemas de realidade virtual e aumentada.

A interação do usuário também pode ocorrer de diferentes formas de acordo com a natureza do ambiente 3D. Ambientes podem ser classificados como fisicamente imersivos, quando utilizam dispositivos especiais para esse tipo de interação, como os óculos de Realidade Virtual, volantes controladores para simuladores de corrida com resposta háptica (ex.: vibração) (Figura 17) ou ainda luvas com sensores que captam os movimentos do usuário e provocam nele sensações físicas como força, pressão, peso e outros. Ou ainda, mentalmente imersivos, quando o usuário tem uma profunda sensação de participação e envolvimento no sistema, mesmo sem estímulos corporais (NASCIMENTO, 2010, p. 06 e 07).

Figura 17 - Conjunto de volante, transmissão e pedais para simuladores virtuais de corrida.

Fonte: RALLYE DESIGN (2011).

Cenários virtuais tridimensionais interativos, onde sua manipulação é feita através de dispositivos de uso geral (como mouse e teclado) (Figura 18), podem ser classificados como sistemas com imersão mental, uma vez que ocorra uma forte sensação de envolvimento e interação por parte do usuário.

Figura 18 - Interação em ambientes imersivos através de teclado e mouse convencionais.

Fonte: UPLOA (2013).

Resta entender para qual propriedade os designers devem dar maior atenção com relação a qualidade da interação ao desenvolver um ambiente tridimensional virtual, seja para realidade virtual, realidade aumentada ou qualquer outra finalidade. Esta propriedade parece ser a usabilidade, utilizada para "descrever a qualidade de interação de uma interface diante de seus usuários" (HIX; HARTSON, 1993 apud FERREIRA, 2002, p. 09).

A eficiência e facilidade de interação passa pela usabilidade das interfaces de um sistema virtual, no caso, um ambiente 3D. De acordo com PRATES e BARBOSA (2007, p. 266) a usabilidade compreende os seguintes fatores:

- Facilidade de aprendizado;
- Facilidade de uso;
- Eficiência de uso e produtividade;
- Satisfação do usuário;
- Flexibilidade;

- Utilidade;
- Segurança no uso.

Compreendendo os tipos de interação e os parâmetros associados a ela é possível identificar requisitos para a análise das ferramentas a serem integradas à sistematização de criação de conteúdo interativo 3D proposta ao fim desta pesquisa.

2.4.3. A criação de conteúdos tridimensionais por designers

De acordo com Martins (2000 apud KIRNER; SALVADOR, 2007, p. 93) o desenvolvimento de ambientes virtuais engloba o estudo da modelagem gráfica 3D e da interface humano-computador (IHC), sendo que está diretamente ligado ao realismo visual, percebido através de nossos sentidos, e à interação.

Os objetos geométricos virtuais, também chamados de modelos 3D, são criados por meio de *softwares* de modelagem tridimensional. São compostos por um conjunto de malhas que arranjadas de certas maneiras, permitem o reconhecimento do objeto no espaço tridimensional virtual. Logo, segundo Lima (2015, p. 64), são "entidades em três dimensões construídas com ferramentas de modificação de malha, que ocupam espaço em um cenário digital". Em conjunto com luzes, câmeras, animações, ações interativas e outros elementos, formam, propriamente, o ambiente virtual.

Atualmente existem diversos programas de modelagem 3D. Alguns possuem um foco mais restrito, com funções mais refinadas e específicas para fins determinados, como animação, simulação de esforços e escultura digital, por exemplo. Outros, apresentam utilidades mais generalistas e funções extremamente diversificadas, possibilitando tanto a criação de objetos 3D únicos, como ambientes inteiros com animações e interações das mais básicas às mais complexas.

A sequência de criação de modelos tridimensionais para jogos digitais elucidada por Lima e Meurer (2011 apud LIMA, 2015, p. 65) também é mesma utilizada na concepção da maioria dos objetos 3D virtuais no Design, sendo constituída de três níveis:

- **Nível Técnico** – é a fase de entendimento e definição de onde o modelo será usado, para assim produzir sua malha tridimensional de maneira adequada. Há fins que exigem modelos com uma malha de altíssima definição (grande quantidade de polígonos - *highpoly*), como o cinema por exemplo, e outros que necessitam de objetos 3D de média ou baixa resolução (pequena quantidade de polígonos - *lowpoly*), como os jogos digitais para dispositivo móveis.

- **Nível Funcional** – é a fase onde define-se como será a construção das malhas dos modelos de acordo com seus requisitos técnicos. Dependendo da destinação, o objeto por ser desenvolvido com partes possíveis de serem movimentadas separadamente, como é o caso dos personagens de uma animação, ou sem elas. Isso influencia diretamente em como seja construído o objeto no *software*.

- **Nível Estético** – resolvidos os aspectos técnicos e funcionais, pode-se então trabalhar a estética do modelo, por meio da modelagem 3D propriamente dita. Este é o momento de se trabalhar também as questões visuais mais refinadas do modelo, como cores, materiais, texturas e propriedades físicas.

Toda esta sequência de passos para a criação de modelos 3D constitui apenas uma das etapas do processo de desenvolvimento de um ambiente tridimensional, a modelagem. Tal processo é formado por pelo menos três macro fases compostas de etapas mais específicas (JOAQUIM, 2012):

1. **Modelagem**
   Concepção da geometria
   Aplicação de materiais e texturas

2. **Configuração do layout da cena**
   Mapeamento
   Iluminação

Geração de Câmeras

3. **Geração de cena**

   Renderização (Imagens estáticas)

   Animação

   Interação

Destas, talvez a fase de modelagem seja a mais conhecida e dominada por designers, especialmente os de produto. A fase de configuração do layout da cena também não apresenta grandes dificuldades àqueles que já trabalham com renderização digital, também bastante comum na área. A maior dificuldade encontra-se na terceira fase, no momento de gerar as animações e interações da cena em questão (um jogo, filme, museu virtual...), que exige conhecimentos um pouco mais avançados e muitas vezes a utilização de *softwares* específicos voltados à estas funções. Por exemplo, dependendo da complexidade das interações pretendidas ou do *software* utilizado, é necessário um entendimento considerável de programação ou o auxílio de um programador, para criar e atribuir os comportamentos interativos aos objetos tridimensionais.

Atualmente temos diversos programas destinados à criação e edição de conteúdo tridimensional virtual, que mantêm ciclos de atualizações contínuas onde constantemente adicionam e aprimoram funções. Observa-se também que é comum a utilização de mais de um *software* em conjunto durante um projeto, funcionando como programas auxiliares em funções específicas (modelagem, animação, renderização...), proporcionando uma melhor qualidade ou fluxo de trabalho que programas mais generalistas. Destaca-se a seguir uma amostra de quatro programas com características relevantes ao assunto do trabalho.

2.4.3.1. 3ds Max

O primeiro *software* é o 3ds Max (Tabela 1), desenvolvido pela empresa Autodesk, líder mundial no ramo de computação gráfica tridimensional. Possui uma versão educacional gratuita e é um dos programas mais completos, trabalhando com

modelagem, animações, renderizações, escultura digital e efeitos especiais de qualidade profissional. Apesar do grande número de ferramentas disponíveis para as mais diversas aplicações e formato de arquivos suportados, ainda existe uma grande quantidade de complementos, os chamados *plug-ins*, que adicionam novas funções ao *software*.

Vale lembrar que o 3ds Max é um dos poucos que mantém o suporte ao VRML, sendo possível exportar ambientes neste formato de forma nativa ao menos até a versão 2017 do *software*.

Tabela 1 - Características do *software* 3ds Max, versão 2017.

	Aplicações	**Existência de Plug-ins**	**Tipos de Licença**	**Sistemas Operacionais Suportados**
3ds Max 2017	Modelagem, Animação, Renderização, Escultura	Sim	Versão Paga e Versão Educacional Gratuita	Windows 7 (ou superior)

2.4.3.2. Blender

Outro *software* é o Blender (Tabela 2), um programa de código-aberto da Blender Foundation, que está em crescente popularidade devido a ser totalmente gratuito e trazer muitos dos recursos dos grandes programas pagos em conjunto com recursos inéditos. Por ser um *software* de código-aberto, recebe constantes atualizações e melhorias provindas de sua enorme comunidade de usuários.

O Blender impressiona pelos vários recursos presentes em sua interface, para modelagem, renderização, animação, simulação de partículas, criação de jogos e até edição de vídeos.

Tabela 2 - Características do *software* Blender, versão 2.78c.

	Aplicações	**Existência de Plug-ins**	**Tipos de Licença**	**Sistemas Operacionais Suportados**
Blender 2.78c	Modelagem, Animação, Renderização, Escultura, Simulação de Partículas, Desenvolvimento de Jogos, Edição de Vídeos	Sim	Gratuito	Windows Vista 32bit ou 64bit (ou superior), Mac OS X 10.6 64bit (ou superior) e Linux 32bit ou 64bit

2.4.3.3. Unity

Mais um *software* popular e com versão gratuita é a Unity (Tabela 3). Criado pela *Unity Technologies*, a Unity se difere dos *softwares* de modelagem 3D, sendo propriamente um Motor Gráfico para Jogos (em inglês, *Game Engine*), que tem por objetivo principal proporcionar ferramentas de criação, edição e publicação de jogos digitais para as mais diversas plataformas de mesa e *mobile*. Sendo um dos mais utilizados e reconhecidos nesta área.

Apesar de não ter foco na modelagem em si, a Unity fornece inúmeros recursos de criação de ambientes virtuais tridimensionais compostos de animações e interatividade. Sendo possível então, utilizá-la para desenvolver conteúdos deste tipo para fins distintos dos jogos. Por ser um programa de uso profissional, seus recursos de animação e interação exigem conhecimentos avançados de programação, assim afastando um pouco os leigos e iniciantes.

Além da diversidade de ferramentas do programa, uma de suas grandes vantagens é a possibilidade de exportar as criações para computadores, *smartphones*, *tablets*, videogames e *smart* tvs. Recentemente, também foi adicionado o suporte nativo ao WebGL, proporcionando a exportação direta de conteúdo para ser visualizado e manipulado em qualquer dispositivo com navegador compatível.

Outro fator importante da Unity é que ela permite a produção de conteúdo para Realidade Virtual e Aumentada, dando cada vez mais valor para este segmento.

Tabela 3 - Características do *software* Unity, versão 5.4.1.

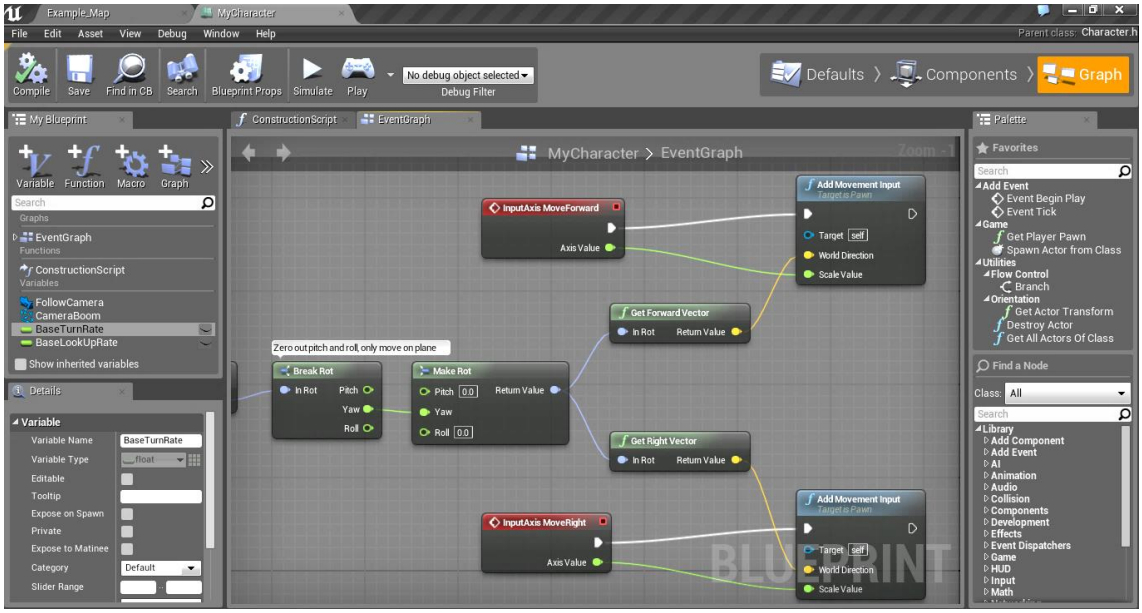
	Aplicações	**Existência de Plug-ins**	**Tipos de Licença**	**Sistemas Operacionais Suportados**
Unity 5.4.1	Desenvolvimento de Jogos, Animação, Renderização	Sim	Versão Gratuita e Versões Pagas	Windows 7 (ou superior) e Mac OS X 10.8 (ou superior)

2.4.3.4. Unreal Engine

Por último, a Unreal Engine (Tabela 4), também um famoso Motor Gráfico para Jogos, desenvolvido pela Epic Games. Inteiramente gratuito, tem por função principal a criação de jogos digitais, mas, assim como o Unity, possui ferramentas para criar ambientes interativos para quaisquer fins.

Apesar de possuir uma interface organizada e grande parte em português, o programa ainda exige conhecimentos específicos de programação para a criação de interações complexas. A grande vantagem deste *software* para os menos familiarizados com linguagens de programação escrita, ele também possibilita a criação de ações interativas por meio de uma linguagem visual chamada *Blueprint*, que foi criada pela própria Epic Games. Mesmo necessitando de um período de aprendizado e de certo conhecimento prévio de programação, o *Blueprint* (Figura 19) facilita bastante o processo de construção e adição de interatividade a objetos e ambientes virtuais, oferecendo vantagens até mesmo para programadores experientes.

Figura 19 – Exemplo da estrutura de uma interação construída com a Blueprint, linguagem visual de programação presente na Unreal Engine.



Fonte: ROMERO (2014).

Outro ponto muito interessante da Unreal Engine são os projetos prontos embarcados no programa. Alguns pequenos jogos e ambientes foram incluídos como exemplos dos recursos do *software* e também são ótimas ferramentas de aprendizagem da linguagem *Blueprint*, já que todos eles são editáveis e construídos de maneira a facilitar o seu entendimento. É possível ainda, utilizar pedaços prontos destes códigos para inclusão em outro projeto que esteja necessitando de funções semelhantes, bem como comprar códigos prontos na loja virtual do programa.

Em se tratando de Realidade Virtual, a Unreal está sempre buscando aprimorar suas ferramentas. Um exemplo é o modo VR Editor, onde a edição do conteúdo é realizada utilizando os próprios dispositivos de RV que será plataformas de execução dos jogos criados.

Tabela 4 - Características do *software* Unreal Engine, versão 4.13.

	Aplicações	**Existência de Plug-ins**	**Tipos de Licença**	**Sistemas Operacionais Suportados**
Unreal Engine 4.13	Desenvolvimento de Jogos, Animação, Renderização	Sim	Gratuito	Windows 7 64bit (ou superior), Mac OS X 10.9.2 (ou superior), Ubuntu 15.04 (ou superior)

2.5. A PROGRAMAÇÃO ORIENTADA A OBJETOS (POO)

Em se tratando de como são criados os ambientes tridimensionais virtuais, mesmo no Design, vale entender sobre a Programação Orientada a Objetos (POO) e porque ela é importante quando falamos sobre objetos virtuais com propriedades e funções, sejam eles modelos 3D ou entidades (instâncias) abstratas.

Segundo David Hemmendinger (2008), a programação orientada a objetos é o uso de unidades modulares de programação (objetos[2], classes[3], subclasses[4], e assim por diante), a fim de tornar a programação mais rápida e de fácil manutenção, se for o caso. Linguagens orientadas a objetos ajudam a gerenciar a complexidade em grandes programas, permitindo a um programador pensar em cada parte do programa de forma isolada.

Em outra definição, a POO é a prática de criar uma arquitetura de *software* que permite flexibilidade através de um Design modular. Ela não é uma linguagem e sim uma prática de arquitetura e de processo de pensamento por trás da estruturação da programação (SMITH, 2014, p. 01).

A orientação a objetos (OO), como explica Marcio David (2007), foi criada como uma tentativa de aproximar o mundo físico do mundo virtual, dentro do computador. Para isso, o mais natural foi utilizar objetos, pois afinal, o mundo real é composto por objetos. Assim, começa a ser respondido o porquê da importância de

[2] Na POO, objeto é um tipo de instância que possui os atributos e comportamentos da classe à qual pertence (SMITH, 2014, p. 21).

[3] Classes são classificações de propriedades definidas, estados e comportamentos que podem ser compartilhados ou modificados entre as instâncias (SMITH, 2014, p. 21).

[4] Subclasses são classes pertencentes a outra classe maior, chamada Superclasse (SMITH, 2014, p. 21).

se compreender as bases da POO para o desenvolvimento de ambiente virtuais interativos formatos por modelos (objetos) tridimensionais.

Abstraindo o conceito em um primeiro momento, pode-se comparar os objetos da POO com os objetos do mundo físico, onde ambos possuem características, comportamentos e estados. A fim de facilitar o entendimento do que constitui um objeto, faz-se uma relação com o que já conhecemos. As características de um objeto (como sua cor, peso, tamanho, identidade...) dizem o que ele é. O comportamento de um objeto (correr, andar, ligar...) detalha o que ele faz. E o estado mostra como o objeto se encontra em um determinado instante. Como exemplo, uma porta ora está aberta (estado: aberta), ora fechada (estado: fechada) (LACERDA, L. C. De; RAMOS, 2015).

Seguindo o raciocínio, objetos com características e comportamentos iguais podem ser considerados de uma mesma classe. E assim, classes podem ser entendidas como algo abstrato que engloba indivíduos (objetos) combinados por seus estados, atributos e comportamentos iguais. Isso facilita a organização estrutural do *software* desenvolvido conforme a OO.

Esta separação por classes, a fim de facilitar o entendimento do problema, foi a base da POO, que teve início nos anos 60. Explica Exforsys (2006) que, na Noruega, Kristen Nygaard e Ole-Johan Dahl estavam trabalhando em simulações de navios explodindo e perceberam que poderiam agrupar os navios em diferentes categorias. Cada tipo de navio teria a sua própria classe com comportamentos e dados únicos. Assim, Kristen e Ole-Johan foram responsáveis pela introdução do conceito de "classe", juntamente com o conceito de instância de uma classe.

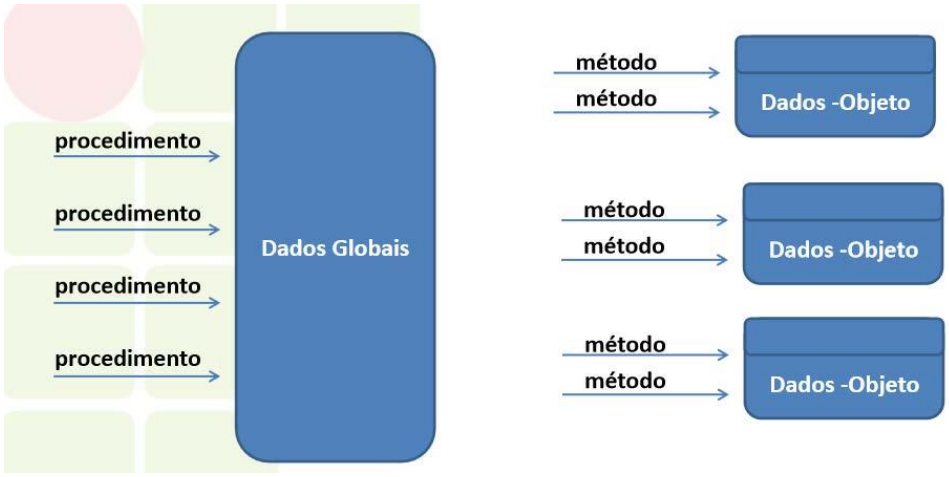
Com o tempo esta prática foi aprimorada, se tornando uma alternativa à tradicional forma de programação estruturada ou procedural. Apesar de não ser adequada para todo e qualquer propósito, existem vantagens claras da POO em comparação com a Programação Procedural, como a sua capacidade de prover uma melhor organização do código e de contribuir para o seu reaproveitamento sem ter que replicá-lo diversas vezes (MÜLLER, 2016).

Estas e outras vantagens da Programação Orientada a Objetos provêm de suas principais qualidades, como a capacidade de abstração, encapsulamento, modularidade, herança e polimorfismo. Destas, duas são de grande importância e

relacionam bem a POO ao desenvolvimento de ambientes virtuais tridimensionais, são elas, a modularidade e a herança.

Aqui a modularidade consiste na divisão de um programa em partes (Figura 20), que se tornam mais fáceis de serem construídas, examinadas ou reparadas (ALVES, 2013, p. 18). Em *softwares* complexos a modularidade facilita o entendimento do código, simplificando sua manutenção e aperfeiçoamento.

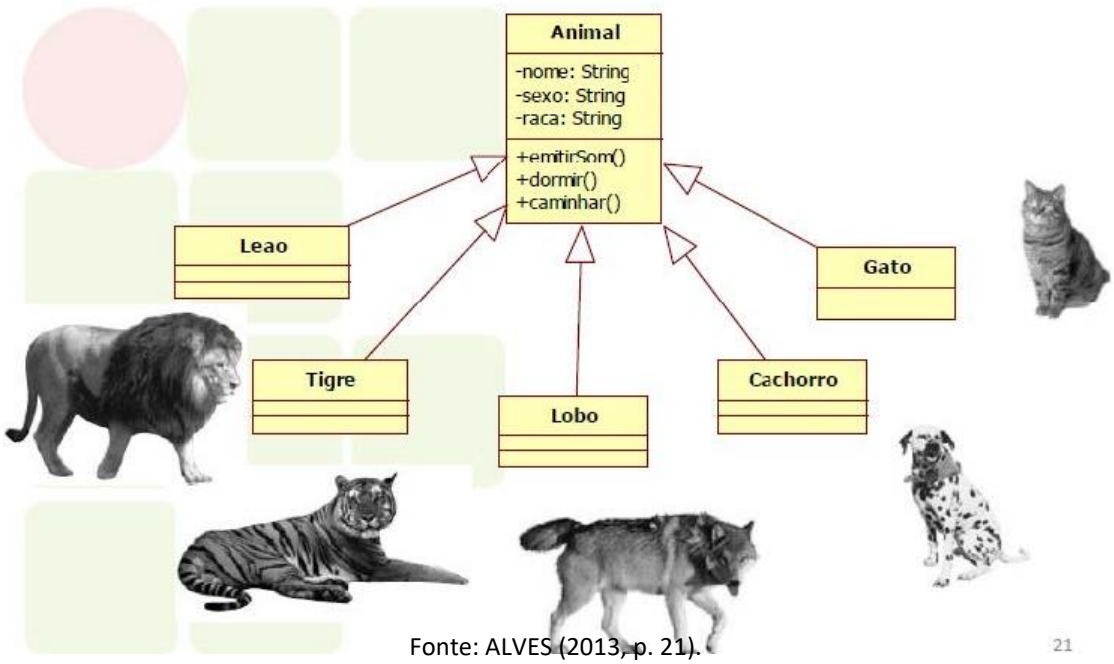
Figura 20 - A esquerda, a representação do pensamento procedural, basicamente uma lista de procedimentos sequenciais. A direita, a representação da modularidade da orientação a objetos, onde se divide o todo em partes para facilitar o seu entendimento e construção.



Fonte: ALVES (2013, p. 05).

Já a herança, basicamente está relacionada a hierarquia de classes, onde as subclasses herdam todas as propriedades e componentes da classe pai. Essa característica torna mais enxuta a programação, já que referenciando subclasses ou objetos à uma classe pai (Figura 21), naturalmente elas recebem seus atributos sem a necessidade de duplicação do código (MANSSOUR, 2002, p. 03).

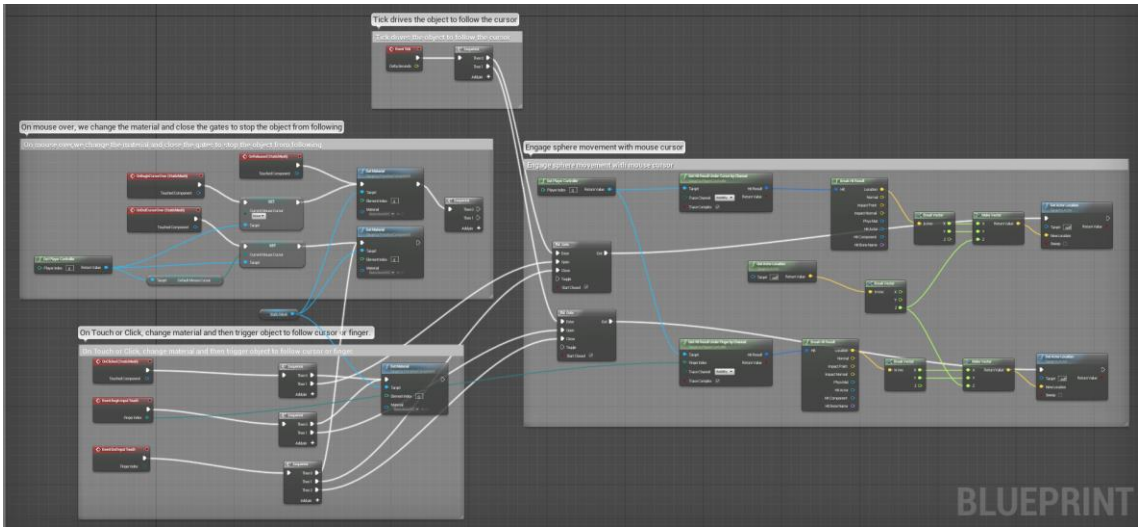
Figura 21 - Leão, tigre, lobo, cachorro e gato pertencem a classe Animal e naturalmente possuem as mesmas propriedades e comportamentos básicos de sua classe pai.



Fonte: ALVES (2013, p. 21).

Ilustrando esta relação da POO com a criação de ambientes virtuais, a linguagem visual de programação *Blueprint*, presente na Unreal Engine (ver item 2.4.3.4), utiliza os princípios da orientação a objetos para a definição de classes e objetos através de um diagrama de blocos interligados. A *Blueprint* é um bom exemplo de como a OO torna mais fácil a elaboração e visualização da estrutura de programação em módulos e o seu reaproveitamento. Também exemplifica como podem ser representadas características e comportamentos de objetos do mundo físico na programação virtual de um ambiente tridimensional, bem como suas inter-relações, com blocos que podem receber e enviar informações a outros objetos (Figura 22).

Figura 22 - Exemplo de estrutura de blocos (Blueprint) que compartilham dados entre si para formar a lógica.



Fonte: SAMD0G (2014).

2.5.1. Programação Escrita e Programação Visual

Como visto, a *Blueprint* se configura como uma Linguagem Visual de Programação, apresentando objetos e funções na forma de blocos, relacionando-os por meio de “fios” que representam o envio ou recebimento de um dado, de um ponto a outro. Nela, as propriedades físicas e mecânicas, bem como as animações e comportamentos atribuídos aos objetos tridimensionais virtuais estão estruturados e descritos de maneira mais clara em comparação com linguagens textuais. Isso facilita tanto o entendimento como a edição, tendo potencial de se tornar uma poderosa ferramenta para designers que estão se interessando cada vez mais projetos que envolvem programação computacional.

Basicamente, uma linguagem de programação é um método padronizado que permite a um programador especificar precisamente os dados e instruções a serem transmitidas à um computador (DIGITALDEV, 2014). Existem hoje diversas linguagens com capacidades diferentes e que são utilizadas para diferentes fins. O detalhamento delas não faz parte do escopo desta pesquisa.

Portanto, na programação escrita, todas as instruções são descritas de forma textual (Figura 23). As linguagens que trabalham desta maneira, utilizam “regras

especificadas que regem como um código (escrito) deve se comportar para produzir programas de computador" (BIGOWN, 2015).

Figura 23 - Parte de um código escrito, representando a criação de uma cena virtual para futura adição de elementos tridimensionais.

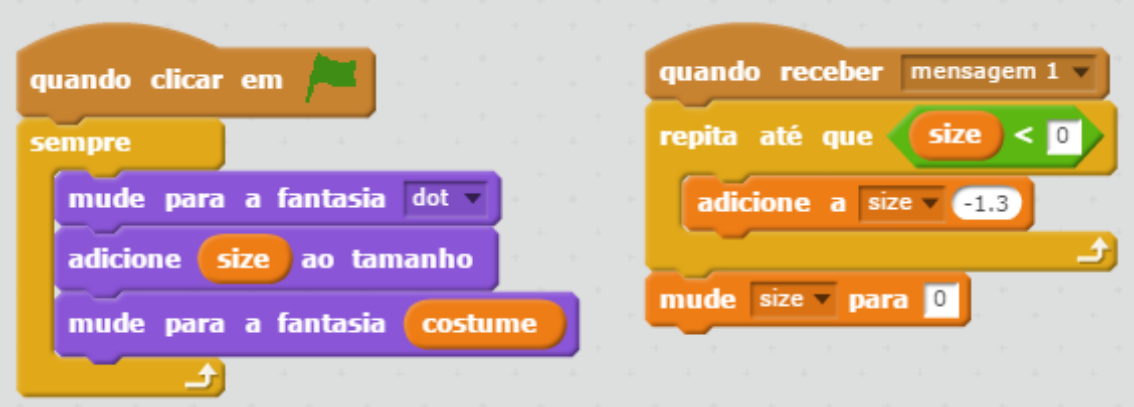
```
// set the scene size
var WIDTH = 400,
  HEIGHT = 300;

// set some camera attributes
var VIEW_ANGLE = 45,
  ASPECT = WIDTH / HEIGHT,
  NEAR = 0.1,
  FAR = 10000;

// get the DOM element to attach to
// - assume we've got jQuery to hand
var $container = $('#container');

// create a WebGL renderer, camera
// and a scene
var renderer = new THREE.WebGLRenderer();
var camera =
  new THREE.PerspectiveCamera(
    VIEW_ANGLE,
    ASPECT,
    NEAR,
    FAR);

var scene = new THREE.Scene();

// add the camera to the scene
scene.add(camera);

// the camera starts at 0,0,0
// so pull it back
camera.position.z = 300;

// start the renderer
renderer.setSize(WIDTH, HEIGHT);

// attach the render-supplied DOM element
$container.append(renderer.domElement);
```

Fonte: LEWIS (2016).

Já na programação visual, existe um ambiente de desenvolvimento e sintaxe compostos por elementos majoritariamente visuais. De acordo com Renato Santos (2013, p. 25), as linguagens deste tipo procuram em geral facilitar a resolução de problemas de uma forma mais intuitiva, atrativa e natural, voltada especialmente para a autoaprendizagem (Figura 24). Tal consideração vai ao encontro do que a empresa Unreal espera com a *Blueprint*, oferecendo gratuitamente uma grande

gama de conteúdos desenvolvidos com ela, para que o usuário aprenda a partir de exemplos.

Figura 24 - Parte de um *software* construído em blocos coloridos com a ferramenta online Scratch, um bom exemplo de ambiente visual de programação voltado ao aprendizado.



Fonte: FUPICAT (2016).

Segundo Este tipo de programação permite a construção de algoritmos por meio da interação com elementos visuais. Normalmente, estes elementos representam blocos de controle similares a estruturas lógicas como FOR, WHILE, IF-THEN-ELSE, (respectivamente PARA, ENQUANTO, SE-ENTÃO-SENÃO em português) e variáveis (RIBEIRO et al., 2012, p. 02 e 03). Assim, pretendem melhorar a compreensão da resolução de problemas computacionais ao eliminar a preocupação com a sintaxe de uma linguagem procedimental tradicional. Ainda, RIBEIRO et al. (2012, p. 04) revela um impacto positivo na motivação e no aprendizado dos alunos que aprendem através de linguagens visuais ao invés de uma linguagem de programação escrita tradicional.

Apesar de as linguagens visuais de programação demonstrarem ser, mais intuitivas, ainda é a programação textual, com códigos escritos, que domina a produção de *softwares*. Por isso, nesta pesquisa, busca-se identificar as linguagens e ferramentas de programação predominantemente visual, com capacidade de representar cenários tridimensionais virtuais onde seja possível a adição e criação de animações e interações. Além disso, a facilidade de aprendizado e uso é um fator de grande importância para a escolha das ferramentas que farão parte da sistematização proposta, visto que ela é direcionada a designers com pouco ou até mesmo nenhum conhecimento prévio do campo da programação.

2.6. TECNOLOGIAS DE REPRESENTAÇÃO E DISSEMINAÇÃO DE CONTEÚDO 3D VIRTUAL

Desde a popularização da internet no fim dos anos 80 e início dos 90, percebeu-se a necessidade de se aproveitar o seu potencial para o desenvolvimento e distribuição de conteúdo interativo, inclusive com gráficos tridimensionais. Na época o problema era criar uma linguagem padrão para se descrever cenas 3D especialmente para distribuição via internet.

A primeira tecnologia desenvolvida com este propósito foi o chamado VRML (*Virtual Reality Modeling Language*) em 1995, que conquistou o apreço de muitos, tendo grande importância até o início dos anos 2000 e ainda utilizado em algumas iniciativas acadêmicas, como na disciplina de Design Virtual na pós-graduação em Design da UFRGS. A partir dela já era possível executar os cenários 3D através da internet ou *off-line*, o que seguiu nas próximas tecnologias estudadas. Na sequência, em 2003, foi a vez do X3D, uma nova geração de padrão de descrição de ambientes tridimensionais para Web, que surgiu como uma revisão do VRML com o intuito de substituí-lo, no entanto, não teve a mesma abrangência do seu antecessor.

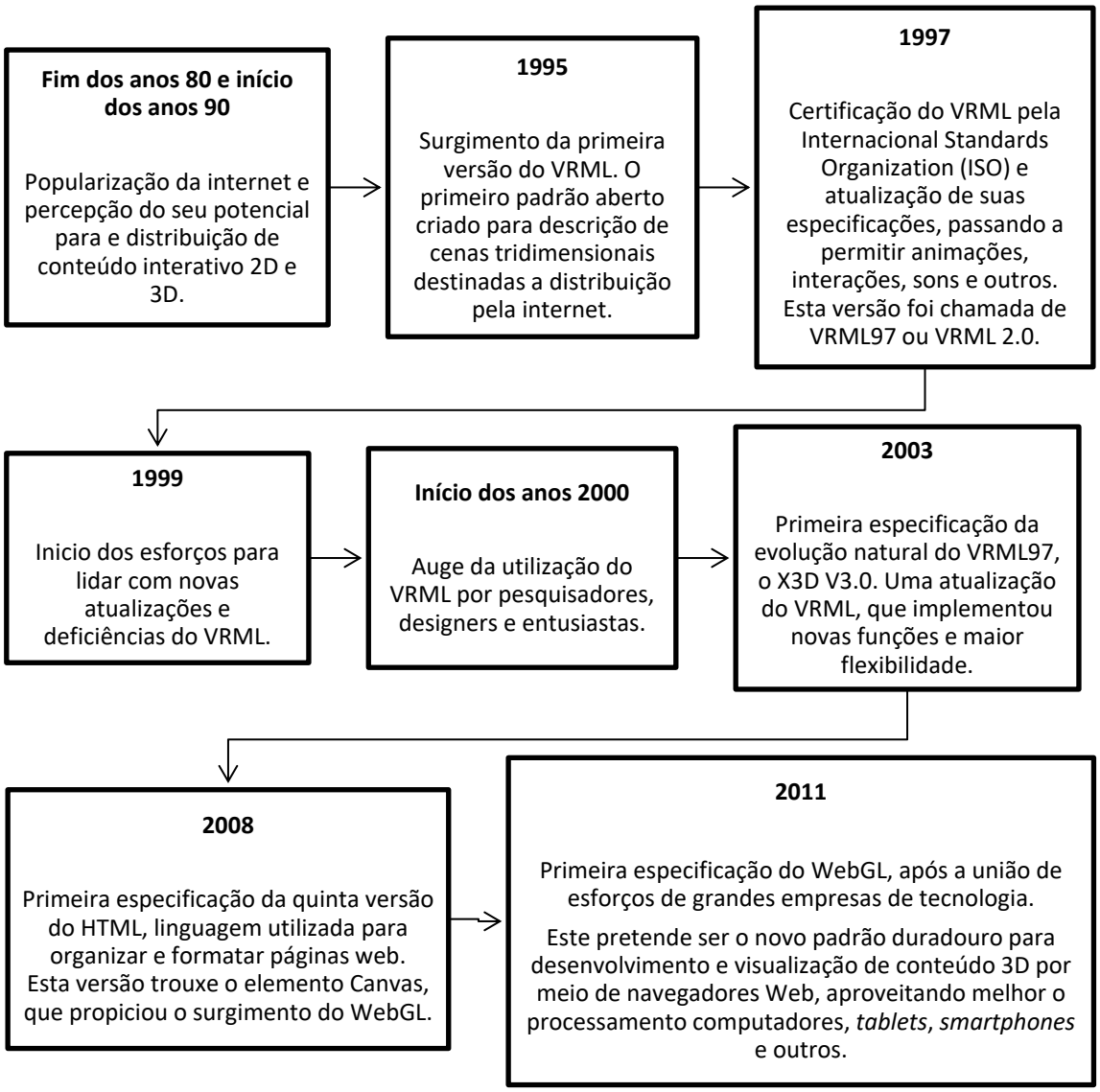
Após não mais suprirem as necessidades da internet moderna, o VRML e o X3D foram perdendo o suporte e o espaço que conquistaram. Novas tecnologias tentaram se estabelecer como o novo padrão de distribuição de conteúdo 3D e algumas tiveram relativo sucesso. Foi então que, através do aprimoramento das capacidades da Web, bem como dos computadores e dispositivos móveis, lançou-se em 2011 o WebGL. Tecnologia que rapidamente está substituindo as demais e trazendo um novo nível de qualidade para a interatividade *online* e *off-line*.

É importante ressaltar que o VRML, o X3D e o WebGL são tecnologias de diferentes tipos e diferentes épocas (Figura 25). Enquanto as duas primeiras são linguagens textuais descritivas, a terceira se configura como uma Interface de Programação de Aplicativos (API[5], sigla em inglês) que se utiliza de outras linguagens de programação para o desenvolvimento de aplicações. As três estão ligadas

[5] API é a sigla do termo em inglês *Application Programming Interface*, que traduzido para o português significa Interface de Programação de Aplicativos. Consiste em um conjunto de instruções e padrões de programação que permite a construção de aplicativos, inclusive os baseados na internet, e a comunicação de recursos entre aplicações (CANALTECH, 2015; VENTURA, 2015).

principalmente pela semelhança de seus propósitos, ou seja, a representação de conteúdos 3D interativos por meio de navegadores web.

Figura 25 – Linha do tempo das tecnologias mais importantes de representação de conteúdo 3D voltado para web.



A seguir, as três importantes tecnologias terão o porquê de sua relevância explicado, desde o surgimento até a substituição por novos padrões mais promissores.

2.6.1. VRML (*Virtual Reality Modeling Language*)

O VRML ou *Virtual Reality Modeling Language* é esclarecido por Marins et al. (2007, p. 03), como uma linguagem textual que descreve objetos e suas inter-relações em um espaço virtual tridimensional, "sem que exista procedimentos de lógica característicos das linguagens de programação convencionais" (PRADO *et al.*, 1999).

Projetada para a internet, permite construir mundos virtuais incorporando formas 3D, iluminação, nevoeiro, animação, efeitos sonoros, interações simples e *hyperlinks*. Cada mundo ou ambiente pode ser composto por um ou mais arquivos nomeados com a extensão ".wrl" (abreviação de mundo em inglês) e tem como destinação principal a distribuição e visualização na Web por meio de um navegador, embora possa ser executado também *off-line* (NADEAU, 1999, p. 18).

Com sua primeira versão (1.0) lançada em 1995, o VRML foi a primeira linguagem padronizada e especificada de descrição escrita de cenas tridimensionais para compartilhamento pela internet. Em sua versão 1.0, a linguagem permitia somente a criação de cenas estáticas. Porém, a partir do esforço de algumas empresas interessadas no avanço desta tecnologia, em 1997 a *Internacional Standards Organization* (ISO) certificou a proposta destas e assim a especificação foi reescrita e chamada de VRML97 ou VRML 2.0, quando se introduziu diversas modificações que permitiram animação, interação, som e outros (PRADO *et al.*, 1999).

Como explica Isabel Manssour (2000), para visualizar documentos VRML, é necessário uma aplicação específica ou um *plug-in* (complemento) para o navegador de internet. No auge de sua utilização (fim dos anos 90 e início dos anos 2000) a lista destes *plug-ins* era grande, contudo, o surgimento de novas e mais vantajosas tecnologias fez com que eles quase sumissem, restando hoje o suporte de pouquíssimas empresas que mantêm a possibilidade de se visualizar conteúdos em VRML em navegadores e sistemas operacionais modernos. Alguns exemplos de *softwares* que continuam sustentando vivo o VRML são:

- **3ds Max**: *Software* de modelagem tridimensional e animação da empresa Autodesk. Com uma nova versão recém lançada (v. 2017),

continua trazendo os mais completos recursos de criação de ambientes interativos em VRML (Figura 26), que podem ser exportados para o formato (.wrl) diretamente pela interface principal do programa.

Figura 26 - Conjunto de ferramentas presentes no *software* 3ds Max 2017, destinadas especificamente a criação de cenas em VRML.

- **Cortona3D Viewer**: Mantido pela empresa Parallel Graphics, o Cortona é talvez o mais famoso *plug-in* remanescente para visualização de arquivos VRML nos navegadores atuais (Internet Explorer, Mozilla Firefox, Google Chrome, Opera) e outras aplicações (Microsoft PowerPoint, Microsoft Word). Possui completo suporte ao VRML97 e utiliza recursos modernos de aceleração gráfica para proporcionar uma melhor experiência.
- **VrmlPad**: Também da empresa Parallel Graphics, o VrmlPad é um ótimo editor de código voltado para VRML. Apesar da possibilidade de edição de um código VRML (Figura 27) em qualquer editor de texto, o VrmlPad traz vantagens como, o realce de termos reservados da sintaxe da linguagem e a detecção de erros ao compilar o código.

Figura 27 - Parte de um código VRML com especificações de um modelo 3D inserido em um ambiente virtual.

```
DEF marty-4473 Transform {
translation -0.000701904 0.301058 -0.0749664
rotation 0 -1 0 -3.14159
scale 0.00136749 0.00136749 0.00136749
scaleOrientation -0.235245 0.15295 0.959826 -0.207148
children [
```

Orientação e escala do modelo no espaço

```
Shape {
appearance Appearance {
material Material {
diffuseColor 0.988235 0.988235 0.988235
ambientIntensity 1.0
specularColor 0 0 0
shininess 0.145
transparency 0
}
}
```

Características da superfície do modelo 3D

```
geometry DEF marty-4473-FACES IndexedFaceSet {
ccw TRUE
creaseAngle 0.785
solid TRUE
```

Características, descrição e posicionamento da geometria do modelo tridimensional

```
coord DEF marty-4473-COORD Coordinate { point [
-79.4214 -1036.17 -2.09863, -72.9663 -1029.26 -30.8319, -79.7761 -1022.54 -1.92267,
-79.9889 -1034.02 29.9048, -80.0474 -1019.52 32.8574, -103.198 -1031.94 92.4825,
-102.705 -1012.59 91.0229, -133.6 -1006.8 109.138, -131.643 -1007.34 104.027,
-170.931 -1008.8 89.6334, -173.024 -1008.18 93.9798, -191.232 -1016.68 55.4833,
-191.423 -1033.98 57.279, -185.144 -1035.56 4.07975, -184.654 -1021.14 3.8647,
-169.913 -1023.34 -29.6016, -179.909 -1012.22 4.84174, -166.648 -1015 -28.7467,
-158.454 -994.944 -25.2548, -153.327 -1018.34 -62.0728, -147.114 -995.51 -67.0262,
-140.374 -962.778 -46.7864, -143.487 -977.592 -21.8425, -105.239 -976.847 -8.49491,
-148.52 -980.447 12.1455, -119.71 -979.254 20.1573, -155.861 -973.673 42.437,
-128.01 -972.062 50.5084, -162.417 -975.471 60.7228, -130.945 -973.913 71.2413,
-171.505 -995.791 83.6901, -130.993 -995.612 98.6175, -104.586 -1001.91 78.73,
-84.0663 -1010.27 29.0231, -85.2473 -1015.23 -3.06565, -71.5539 -1016.76 -30.0542,
-75.4807 -1010.76 -31.1928, -48.155 -1016.88 -93.0134, -52.9646 -998.834 -87.317,
-80.6685 -991.726 -31.5571, -91.1639 -989.147 -5.11795, -98.673 -989.097 24.9223,
-107.036 -982.527 61.5539, -91.1939 -961.862 -34.2837, -183.748 -1002.01 50.558,
-172.09 -983.282 41.0623, -166.67 -991.142 7.15059, -45.6639 -1038.04 -97.355,
-118.329 -1039.23 -136.672, -156.721 -1037.32 -62.4977, -171.243 -1036.36 -29.8367,
-175.187 -1028.26 102.321, -135.565 -1025.76 117.1, -157.031 -1025.1 -62.9154,
-45.4978 -1023.1 -93.3247, -148.075 173.129 -147.443, -126.762 162.746 -86.8276,
-170.645 105.481 -93.079, -37.9275 -218.198 26.8717, -33.529 -162.087 22.8005,
-74.9498 -1015.13 -160.17, -115.75 -1004.88 -137.949, -78.0389 -1003.92 -153.885,
-43.9853 -1001.59 -130.62, -58.8419 -986.428 -78.3973, -188.897 -16.8167 -40.3394,
-176.216 36.9026 -40.7609, -159.909 29.1047 -83.3763, -149.719 63.0571 -85.6317,
-165.878 76.7969 -37.6223, -131.895 116.643 -81.5095, -152.985 127.175 -31.5212,
-118.681 144.579 -79.7561, -151.207 160.177 -28.0613, -119.01 165.845 -82.4917,
-153.725 170.597 -23.9991, -194.226 252.281 -29.1966, -167.801 57.921 -129.05,
-176.895 29.3115 -128.502, -172.509 -24.1444 -82.4045, -188.781 -25.9791 -129.187,
-176.396 -40.6053 -80.2268, -192.919 -44.1534 -131.267, -179.645 -56.6072 -78.8359,
-136.946 -982.366 -78.9018, -140.895 -957.446 -54.2813, -80.8747 -953.942 -36.8071,
-186.18 193.197 -155.526, -150.842 112.33 -126.21, -135.719 156.882 -131.47,
-142.738 173.944 -142.312, -180.832 214.927 -164, -116.569 -1020.26 -138.962,
-118.859 -1025.21 -140.01, -78.9741 -1018.35 -155.043, -79.1374 -1023.74 -156.475,
-78.098 -1040.09 -157.525, -41.6436 -1037.27 -134.056, -42.4941 -1022.35 -133.09,
```

Estes e outros poucos programas de criação, edição e visualização de arquivos VRML ainda disponíveis, possibilitam que pessoas de distintas áreas continuem fazendo uso deste antigo padrão, muito por seu potencial de criação de conteúdos interativos tridimensionais de forma simples e com código de fácil compreensão. Uma das áreas que já utilizou muito e segue hoje em ações isoladas explorando este padrão é a academia, na aprendizagem das próprias ferramentas de criação ou no

desenvolvimento de aplicações interativas para a visualização de dados, ensino e outros.

A simplicidade de manipulação da linguagem VRML segue atraindo pesquisadores e profissionais de diferentes áreas, como os designers, mesmo que o impedimento da utilização de modelos 3D muito detalhados, que não são processados adequadamente, e a necessidade de programas complementares para a visualização, sejam vistos como desvantagens significativas na atualidade.

2.6.2. X3D (*Extensible 3D Graphics*)

Basicamente com o mesmo propósito do VRML, o X3D também é um padrão aberto para representação, comunicação e distribuição de conteúdos tridimensionais e interativos, principalmente através da internet. Os esforços iniciais para a sua criação começaram em 1999 pela *Web3D Consortium*, organização criada para definir e desenvolver o X3D (DALY; BRUTZMAN, 2007, p. 130). Surgiu como uma revisão, mais madura, refinada e completa, de seu predecessor, constituindo uma nova arquitetura de formato de arquivo livre de *royalties*, composta por um conjunto de especificações e interface de programação (SOARES, 1999; WEB3D CONSORTIUM, 2016).

Eduardo Falcão (2011, p. 43) explica que o X3D aproveita as ideias básicas do VRML e as amplia somando novas funcionalidades. Uma importante mudança é adoção de um novo formato de codificação para o padrão, o XML[6], que permite a integração das cenas 3D com a Web mais facilmente.

Assim como acontece com os arquivos VRML, as cenas criadas com o padrão X3D ainda não dispensam a utilização de complementos para navegadores (*plug-ins*) ou programas específicos para a sua visualização. Ainda assim, suas capacidades são significativas e indicadas por Hopf et al. (2007, p. 06).

---

[6] Essencialmente o XML (Extensible Markup Language) é uma linguagem de marcação, que por sua vez são códigos "que podem ser aplicados a dados ou textos para serem lidos por computadores ou pessoas". Outro exemplo de linguagem de marcação é o HTML que serve para organizar e formatar uma página web. Com o XML é parecido, porém ele é utilizado para "padronizar uma sequência de dados com o objetivo de organizar, separar o conteúdo e integrá-lo com outras linguagens" (PEREIRA, 2009).

> A tecnologia X3D dá suporte para gráficos 3D, transformações de geometria, iluminação, materiais, texturas, mapeamento, pixels, vértices e aceleração de hardware. Permite animação com temporizadores e interpoladores de condução contínua. Permite também interação com o mouse e entradas de teclado. A navegação no ambiente acontece por meio do uso de câmeras, com características de colisão, proximidade e visibilidade, detecção e vários tipos de iluminação. Por meio de script é possível mudar dinamicamente a cena através de linguagens de programação como o JavaScript (HOPF et al., 2007, p.06).

Com relação a criação e edição de ambientes X3D, a situação é também semelhante ao VRML, onde restaram poucos programas que ainda disponibilizam recursos nativos para tal. Um destes *softwares* é o Blender que ainda hoje, em sua versão mais recente (2.78), possibilita nativamente a exportação de arquivos no formato ".x3d", no entanto, nem todos os recursos de modelagem disponíveis no Blender podem ser exportados e, portanto, a geração de cenas realistas não é possível. Existem também complementos que adicionam a função de exportação para X3D à *softwares* como o 3ds Max por exemplo, mas muitas vezes estes encontram problemas e não se comportam tão bem quanto uma função nativa.

Mesmo sendo um aprimoramento do VRML em termos de funcionalidades e capacidades, as ferramentas ainda disponíveis para desenvolvimento em X3D não proporcionam uma maneira facilitada de criação, mesmo os editores visuais como o Blender. Vale lembrar que, como os arquivos X3D são em essência códigos escritos, sua edição pode ser realizada por meio de simples editores de texto, método utilizado somente para pequenas alterações por ser bastante mais trabalho. E em se tratando do código (Figura 28) verifica-se que ele segue os princípios do VRML, todavia se apresenta de maneira mais organizada e melhor estruturada a partir do uso de *tags*.

Figura 28 - Parte de código que descreve uma cena simples em X3D, criado com o *software* Blender.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D version="3.0" profile="Immersive" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xsd:noNamespaceSchemaLocat
    <head>
        <meta name="filename" content="untitled.x3d" />
        <meta name="generator" content="Blender 2.76 (sub 0)" />
    </head>
    <Scene>
        <NavigationInfo headlight="false"
                        visibilityLimit="0.0"
                        type='"EXAMINE", "ANY"'
                        avatarSize="0.25, 1.75, 0.75"
                        />
        <Background DEF="WO_World"
                    groundColor="0.051 0.051 0.051"
                    skyColor="0.051 0.051 0.051"
                    />
        <Transform DEF="Cube_TRANSFORM"
                   translation="0.000000 0.000000 0.000000"
                   scale="1.000000 1.000000 1.000000"
                   rotation="0.000000 0.707107 0.707107 3.141593"
                   >
            <Transform DEF="Cube_ifs_TRANSFORM"
                       translation="0.000000 0.000000 0.000000"
                       scale="2.000000 2.000000 2.000000"
                       rotation="1.000000 0.000000 0.000000 0.000000"
                       >
                <Group DEF="group_ME_Cube">
                    <Shape>
                        <Appearance>
                            <Material DEF="MA_Material"
                                      diffuseColor="0.269 0.340 0.800"
                                      specularColor="0.401 0.401 0.401"
                                      emissiveColor="0.000 0.000 0.000"
                                      ambientIntensity="0.333"
                                      shininess="0.098"
                                      transparency="0.0"
                                      />
                        </Appearance>
                        <IndexedFaceSet solid="true"
                                        coordIndex="0 1 2 3 -1 4 7 6 5 -1 0 4 5 1 -1 1 5 6 2 -1 2 6 7 3 -1 4 0 3 7 -1 '
                                        >
                            <Coordinate DEF="coords_ME_Cube"
                                        point="1.000000 1.000000 -1.000000 1.000000 -1.000000 -1.000000 -1.000000 -1.0(
                                        />
                        </IndexedFaceSet>
                    </Shape>
                </Group>
            </Transform>
        </Transform>
        <Transform DEF="Lamp_TRANSFORM"
                   translation="-4.076245 5.903862 1.005454"
                   scale="1.000000 1.000000 1.000000"
                   rotation="-0.498084 -0.762016 -0.413815 1.513875"
                   >
            <PointLight DEF="LA_Lamp"
```

Sendo o X3D praticamente uma evolução estrutural do VRML, as possibilidades de aplicação de ambos são muito parecidas. Desde jogos, visualizadores de dados, museus virtuais, ambientes educativos e tantos outros. Ainda que formado por funcionalidades interessantes, hoje o X3D não mais atende os requisitos de ambientes tridimensionais para uso no Design e por designers, muito devido à baixa qualidade gráfica das cenas por ele descritas.

2.6.3. WebGL

Dos anos 90 para cá, VRML, X3D e diversas outras tecnologias proprietárias, tentaram se estabelecer como padrão definitivo de descrição de objetos e ambientes tridimensionais virtuais para Web. Foi então que em 2006, como um experimento de um engenheiro de *software* da empresa Mozilla chamado Vladmir Vukićević, nasceu o WebGL. Tecnologia que teve certificada sua primeira especificação somente em 2011, após a união de esforços de empresas como o consórcio Khronos[7], Mozilla, Google, Apple, Opera, entre outras (BORJA, 2016, p. 10; PARISI, 2014, p. 18).

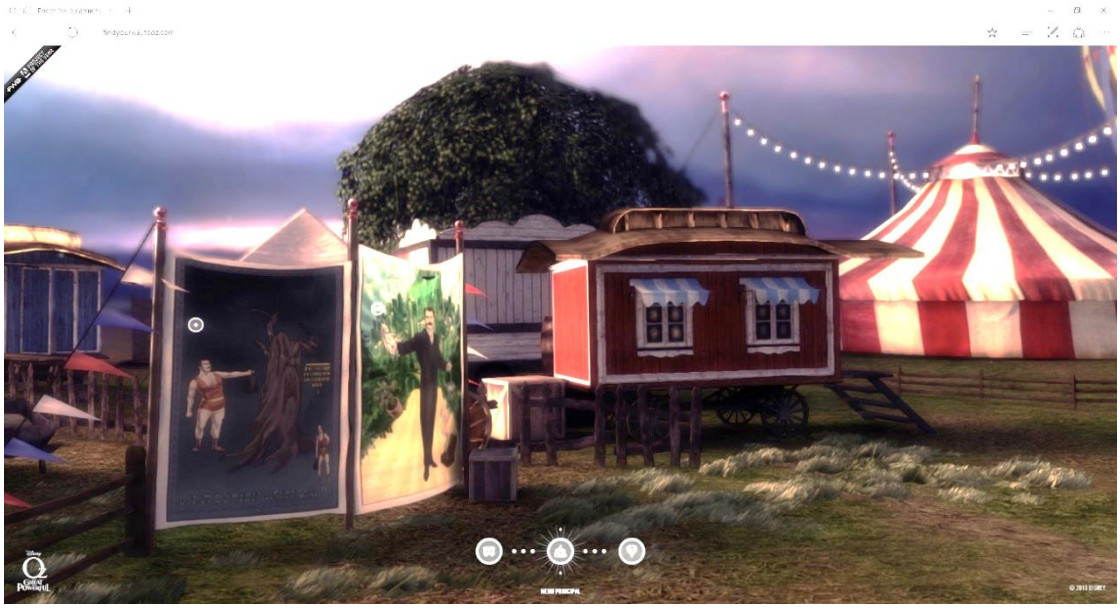
O WebGL se caracteriza como uma Interface de Programação de Aplicativos (API) multiplataforma, para renderização de gráficos 3D concebidos para Web (KHRONOS GROUP, 2014a). Nas visões de Danchilla (2012, p. xviii) e de Levkowitz; Kelleher (2012, p. 23), é um novo padrão que permite executar poderosos gráficos tridimensionais dentro de um navegador web, sem a necessidade de quaisquer apoios externos, como os *plug-ins*.

Esta tecnologia se mostra promissora porque resolve algumas das principais limitações das anteriores. Além de não serem mais necessários complementos para a visualização dos conteúdos, ela fornece funcionalidades de acesso direto ao *hardware* gráfico (GPU) dos dispositivos e as aplicações são executadas nativamente nele, permitindo uma aceleração gráfica mais eficiente, utilizando melhor a capacidade de processamento de computadores, *smartphones*, *tablets* e tantos outros (CONGOTE et al., 2011, p. 139; LOESCH et al., 2012, p. 196). Com isso, houve um salto significativo de qualidade gráfica (Figura 29) e efeitos de física das cenas tridimensionais. Cenas estas, que podem agora ser visualizadas com melhor qualidade e em uma gama muito maior de aparelhos através de navegadores web, característica enfatizada por Kenneth Russell (2015, p. 03):

> WebGL expõe o poder do processador gráfico que já está em todos os dispositivos, diretamente para a web, de maneira segura. Isso permite que aplicativos 3D sofisticados, poderosos e portáteis possam ser entregues ao usuário final, diretamente no navegador (RUSSELL, 2015, p. 03).

[7] O *Khronos Group* é uma entidade sem fins lucrativos, focada na criação de padrões abertos e livres de royalties para computação paralela, gráficos e mídia dinâmica em uma ampla variedade de plataformas e dispositivos (KHRONOS GROUP, 2016c).

Figura 29 - Aplicação interativa em WebGL com texturas e objetos detalhados, desenvolvida pela Disney para promover o filme "O Mágico de Oz".



Fonte: GOOGLE (2016).

Loesch, Christen e Nebiker (2012, p. 196) esclarecem que o WebGL é, em essência, uma interface de programação de baixo nível, com comunicação mais próxima da linguagem da máquina e familiar apenas à especialistas, onde para simplificar a criação de conteúdos, são continuamente desenvolvidos os chamados motores de alto nível ou ainda, bibliotecas. Estes abstraem a interface original e fornecem os recursos do WebGL de maneira menos complexa, utilizando uma sintaxe de programação facilitada (Figura 30).

Figura 30 - Corpo de código HTML com a descrição de uma cena simples em WebGL utilizando a biblioteca Three.js.

```
<body>

    <script src="../build/three.js"></script>

    <script>

        var camera, scene, renderer;
        var mesh;

        init();
        animate();

        function init() {

            camera = new THREE.PerspectiveCamera( 70, window.innerWidth / window.innerHeight, 1, 1000 );
            camera.position.z = 400;

            scene = new THREE.Scene();

            var texture = new THREE.TextureLoader().load( 'textures/crate.gif' );

            var geometry = new THREE.BoxBufferGeometry( 200, 200, 200 );
            var material = new THREE.MeshBasicMaterial( { map: texture } );

            mesh = new THREE.Mesh( geometry, material );
            scene.add( mesh );

            renderer = new THREE.WebGLRenderer();
            renderer.setPixelRatio( window.devicePixelRatio );
            renderer.setSize( window.innerWidth, window.innerHeight );
            document.body.appendChild( renderer.domElement );

            //

            window.addEventListener( 'resize', onWindowResize, false );

        }

        function onWindowResize() {

            camera.aspect = window.innerWidth / window.innerHeight;
            camera.updateProjectionMatrix();

            renderer.setSize( window.innerWidth, window.innerHeight );

        }

        function animate() {

            requestAnimationFrame( animate );

            mesh.rotation.x += 0.005;
            mesh.rotation.y += 0.01;

            renderer.render( scene, camera );

        }

    </script>

</body>
```

Fonte: THREE.JS (2016).

Os possíveis usos do WebGL são inúmeros e crescentes à medida em que se aumenta o poder de processamento dos dispositivos, em especial os móveis (*smartphones*, *tablets* e tantos outros), possibilitando a execução de aplicações com maiores exigências computacionais, diretamente de um navegador web e adaptável ao *hardware*. Já se encontram, para execução multiplataforma e sem a necessidade de *plug-in*s através do WebGL, jogos com gráficos 3D detalhados e físicas realísticas (Figura 31), demonstrações virtuais de produtos com animações e interações

imersivas (Figura 32) e também completos *softwares online* de modelagem tridimensional (Figura 33). Todas estas aplicações eram inimagináveis até poucos anos atrás. Assim como o aumento do poder gráfico dos dispositivos, o número também crescente de novas bibliotecas, como novas funcionalidades para a descrição de cenas, só faz aumentar as possibilidades e utilidades da tecnologia.

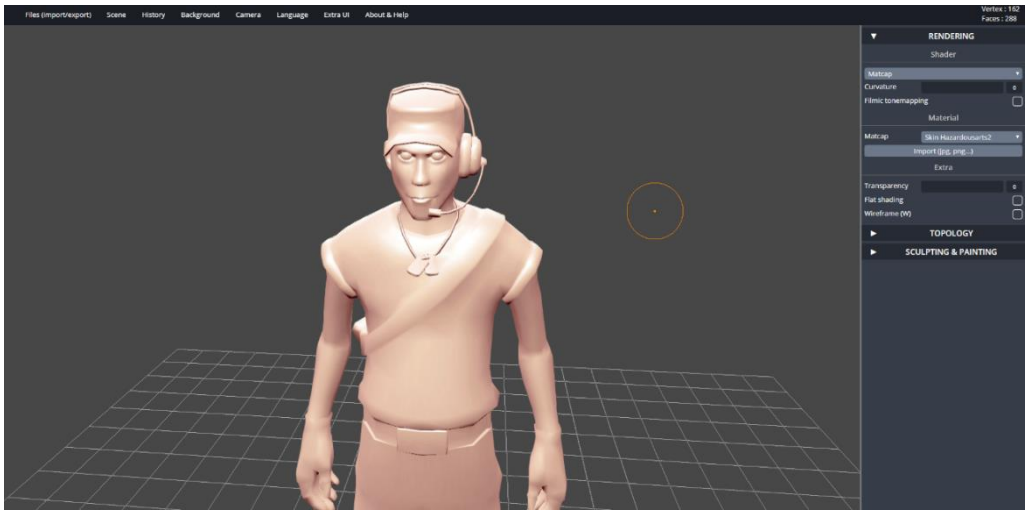
Figura 31 - Jogo transportado de plataformas móveis para execução direta em navegador *web* através do WebGL.

Fonte: MADFINGER GAMES (2014).

Figura 32 - Demonstração promocional de veículo em ambiente virtual.

Fonte: LITTLE WORKSHOP (2015).

Figura 33 - *Software online* de escultura digital inteiramente em WebGL.



Fonte: GINIER (2015).

Imagina-se que seja possível oferecer a designers uma maneira mais simples de aproximação e utilização das funcionalidades do WebGL. Ampliando suas capacidades em projetos, dando-lhes a oportunidade de contribuir com suas criações, neste universo que promete evoluir muito nos próximos anos e se tornar um dos pilares da computação gráfica (LEVKOWITZ; KELLEHER, 2012, p. 34).

Apesar do entusiasmo generalizado e as pretensões de um futuro brilhante, alguns autores temem que o WebGL possa falhar na tentativa de se tornar o padrão definitivo de representação 3D para web, como seus antecessores. Para que isso não aconteça, Brian Danchilla (2012, p. 301) diz serem necessárias três coisas: suporte de navegadores web e aparelhos, adoção da comunidade de desenvolvimento, especialmente dos desenvolvedores de jogos e melhorias continuas no desenvolvimento do próprio padrão. Espera-se que este trabalho possa contribuir para inclusão de mais designers neste meio e assim garantir a satisfação destas três necessidades, trazendo vantagens para ambos, tecnologia e profissionais.

2.7. O WEBGL EM DETALHES

Objeto principal desta pesquisa, é de fundamental necessidade o entendimento aprofundado acerca do WebGL. Algumas das potencialidades dele já

foram apresentadas, porém é necessário esclarecer quais outras tecnologias propiciaram o seu desenvolvimento e como ele se torna cada vez mais acessível por meio das bibliotecas de desenvolvimento.

A partir do entendimento básico de que o WebGL é uma tecnologia de representação de gráficos 3D em navegadores web de forma "nativa", ou seja, sem a necessidade de quaisquer complementos para sua execução, é possível defini-lo de forma mais técnica. Então, o WebGL só foi possível graças a outras três importantes tecnologias que o suportam, o OpenGL ES 2.0, o HTML5 e o JavaScript.

O OpenGL ES (*Open Graphics Library*), que chegou em sua versão 2.0 em 2007, é uma poderosa interface de programação de aplicativos (API) multilinguagem, gratuita e de baixo nível, composta por um conjunto de funções que fornecem acesso direto a muitos recursos da Unidade de Processamento Gráfico (GPU – sigla em inglês), responsável pelo processamento de vídeo de computadores, videogames, aparelhos móveis e tantos outros. Utilizando tais funções é possível descrever cenas 2D e 3D com gráficos complexos que são "lidas" e renderizadas pela GPU (KHRONOS GROUP, 2016a; OPENGL.ORG, 2010). Praticamente todo o funcionamento do WebGL faz referência ao OpenGL ES 2.0, utilizando ainda a mesma semântica para se valer justamente de uma de suas maiores e mais vantajosas funcionalidades, a multiplataforma, agora destinada à distribuição e execução de conteúdo gráfico digital para a web (BORJA, 2016, p. 09). Sendo também aberto e de baixo nível, em comunicação direta com a GPU, é chamado de OpenGL para a Web (KHRONOS GROUP, 2014b, 2016b).

Por sua vez, o HTML5, versão número cinco do HTML[8], linguagem utilizada para organizar e formatar páginas web, foi quem propiciou levar o OpenGL ES para os navegadores. Com sua primeira especificação anunciada em 2008, o HTML5 trouxe dentre muitas novidades o elemento Canvas, que basicamente fornece uma superfície para a renderização em tempo real de gráficos (imagens) diretamente no navegador. Esse trabalho de reprodução de conteúdo multimídia na internet era

[8] O *HyperText Markup Language*, HTML, é uma linguagem de marcação utilizada para desenvolvimento de páginas e aplicações web. Arquivos HTML são interpretados por navegadores web (*browsers*) de diferentes dispositivos (ORIENTE, 2015; PACIEVITCH, 2016).

antes realizado por *softwares* externos, proprietários, mais limitados e menos seguros como é o caso do Adobe Flash Player, que teve reinou por longos anos como a melhor opção do segmento.

Conforme Wallace Jackson (2016, p. 171 e 178) ao mesmo tempo em que o WebGL se utiliza da nova funcionalidade, ele também funciona como um complemento a versão número 5 da linguagem HTML. O elemento Canvas, introduzido pelo HTML5, foi estipulado primeiramente para gráficos bidimensionais (2D) e posteriormente, com a combinação com o WebGL, passou a permitir conteúdo 3D animado e interativo.

Já o JavaScript é sobretudo uma linguagem de alto nível que tem como características ser leve, multiplataforma e suportar programação orientada a objetos na forma de *scripts*[9] (MARK, 2016; MOREIRA, 2016). É utilizada pela maior parte das bibliotecas (ver item 2.7.1) de WebGL em conjunto com o elemento Canvas do HTML5 para descrever, gráficos 2D e 3D, composições, interações e animações simples e complexas (PALUCH, 2015).

As versões atuais de todos os principais navegadores web suportam aplicações desenvolvidas em WebGL, entre eles Microsoft Edge, Google Chrome, Mozilla Firefox, Opera e Safari, em suas versões para computadores pessoais e dispositivos móveis. A única restrição envolve aparelhos mais antigos que talvez não consigam executar todos os efeitos presentes em conteúdos desde tipo. Com apoio total das gigantes da tecnologia, o WebGL já se tornou o padrão mais promissor para internet, deixando para trás fortes concorrentes como o Flash da Adobe.

Com relação às limitações ainda existentes, elas tendem a ser reduzidas rapidamente e seu campo de aplicação ampliado, pois conforme o esclarecido por Dan (2014), o WebGL faz uso dos mesmos *drivers* de vídeo que são utilizados pela grande maioria das aplicações que rodam de maneira *off-line* nos dispositivos atuais, indicando que há muito a ser aproveitado ainda. Para se manter forte a tecnologia

---

[9] Linguagens de programação geralmente são projetadas para a criação de aplicações completas, porém as linguagens de *script* estão direcionadas para tarefas em menor escala como, criação de rotina de execução corriqueira (macros) ou integração entre aplicações existentes, sendo chamadas até de *glue languages* (linguagens cola, em tradução literal) por serem mais flexíveis e permitirem a junção de programas menores para se construir um grande sistema (BAZILIO, 2016, p. 02 e 04).

deve ter suas capacidades exploradas e aprimoradas, sendo os designers grandes beneficiadores do que ela tem a oferecer.

2.7.1. Bibliotecas

Com o propósito de facilitar a programação de aplicações com WebGL, são utilizadas as chamadas Bibliotecas de Programação ou *Library*, em inglês. O WebGL é uma API que se comunica diretamente com o *hardware* responsável pelo processamento de vídeo dos dispositivos através de uma linguagem de baixo nível e é por este motivo que a criação de conteúdo utilizando esta tecnologia acontece majoritariamente por meio de bibliotecas de programação.

As bibliotecas, nada mais são que conjuntos de funções e definições pré-escritas para um propósito definido e prontas para serem utilizadas em um nível mais elevado da programação por aqueles que não tem motivos para entender a linguagem "bruta" da máquina simplesmente para criar um aplicativo (ANDRADE, 2015; JÁCOME, 2010). As bibliotecas são fundamentais porque reduzem a complexidade da programação com vistas de aumentar a produtividade, proporcionando uma maneira mais fácil e compreensível de se desenvolver aplicações, no caso deste trabalho ambientes virtuais tridimensionais (MINISTÉRIO DA CULTURA, 2006). Elas abstraem funções de baixo nível, subtraindo e resumindo detalhes para, segundo Leite (2007), expressar algo de maneira mais concisa, sem que os detalhes mais complexos fiquem a mostra, tornando a escrita do código mais "humana" à medida que se distancia da linguagem de máquina. No momento da execução das aplicações essas funções abstraídas são traduzidas novamente para a linguagem reconhecida pelo *hardware*.

Muitas bibliotecas já trabalham com o OpenGL e o elemento Canvas do HTML5 para facilitar o desenvolvimento de cenas virtuais com o padrão WebGL. Neste caso, uma biblioteca ou um conjunto delas pode constituir um Motor Gráfico ou uma *Engine*, que é "um pacote de funcionalidades que são disponibilizadas para facilitar o desenvolvimento de um jogo e impedir que sua criação tenha que ser feita

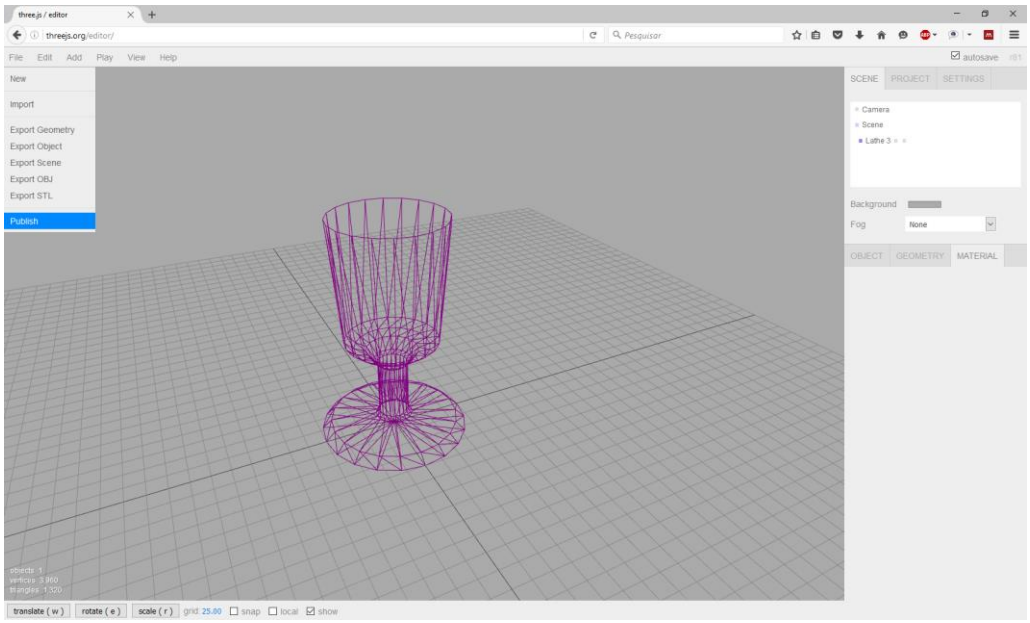
do zero" (KLEINA, 2011). Este pacote é utilizado na modelagem de gráficos 2D e 3D, adição de animações e interações como a movimentação de personagens ou o sistema de colisão entre personagem e objetos.

Como mencionado, uma biblioteca existe para propósitos definidos e dentro do WebGL muitas servem para fins bem específicos como é o caso da Awe.js, que oferece recursos para a criação de conteúdo para execução em plataformas de realidade virtual e aumentada. Outras bibliotecas possuem uma quantidade extremamente variada de recursos como criação de geometrias, câmeras e luzes, carregamento de modelos tridimensionais em diversos formatos, capacidade de edição de propriedades de materiais e texturas, criação de animações, entre outros.

Ressalta-se que algumas bibliotecas trazem consigo uma interface gráfica de desenvolvimento, semelhante a muitos dos *softwares* de modelagem tridimensionais tradicionais, para servir de editor visual dos ambientes virtuais em WebGL, deixando o código escrito mais afastado dos olhos do desenvolvedor. Muitas dessas interfaces gráficas são criados também a partir do WebGL, herdando sua capacidade de multiplataforma. Tais editores facilitam ainda mais o aprendizado e manipulação dos recursos desta tecnologia, se mostrando adequados para uso por designers sem conhecimento avançado em programação.

A mais importante e utilizada biblioteca para WebGL hoje é sem dúvidas a Three.js. Muitos dos bons projetos em WebGL são criados com a Three.js (threejs.org – página oficial), pois ela possui um número bastante grande de funcionalidades prontas para serem utilizadas em cenas virtuais de baixa ou alta complexidade visual e interativa. Ela também acompanha uma interface gráfica para edição de conteúdo *on-line* e *off-line* (Figura 34), que possibilita a importação de modelo 3D, que podem ser escalados e posicionados dentro do espaço virtual e a criação e edição de materiais, luzes e câmeras. Apesar disso, animações e interações só podem ser adicionadas diretamente no código escrito exportado pelo editor visual em formato '.html', ponto de maior dificuldade para não programadores.

Figura 34 - Interface gráfica de edição *online* de conteúdo WebGL da Three.js.



Fonte: THREE.JS (2016b).

Como já visto anteriormente, ainda existem programas de modelagem 3D e desenvolvimento de jogos, como a Unreal Engine e a Unity, que permitem a exportação de projetos criados a partir deles para WebGL. Estes utilizam em sua maioria um conjunto de bibliotecas proprietárias para isso, limitando o trabalho de criação a interface do próprio *software*.

Capítulo 3

MATERIAIS E MÉTODOS

Aqui, apresentam-se o método de pesquisa (Figura 36) utilizado para a caracterização do problema, obtenção de dados relevantes e os ensaios realizados com as tecnologias, a fim de encontrar uma solução inovadora. Neste capítulo também são descritos os materiais utilizados em cada fase do trabalho investigativo.

O presente trabalho científico é conduzido por um método de pesquisa baseado na metodologia denominada *Design Science Research* (DSR), que foi aclarada por um estudo de Dresch et al. (2013), voltado principalmente a Engenharia de Produção, mas possível de ser utilizado também no Design. Através de suas cinco etapas, o DSR compreende a identificação do problema, a definição dos resultados esperados, o projeto e desenvolvimento, a demonstração, a avaliação e por fim, a comunicação. Nestes se encaixam os sete passos realizados neste trabalho (Figura 36), conforme a Figura 35.

A primeira etapa (Conscientização) visa a compreensão da problemática envolvida. Assim, é nesta fase que se define e formaliza o problema de pesquisa e suas fronteiras. Já em Sugestão criam-se possíveis soluções satisfatórias para o problema elucidado anteriormente.

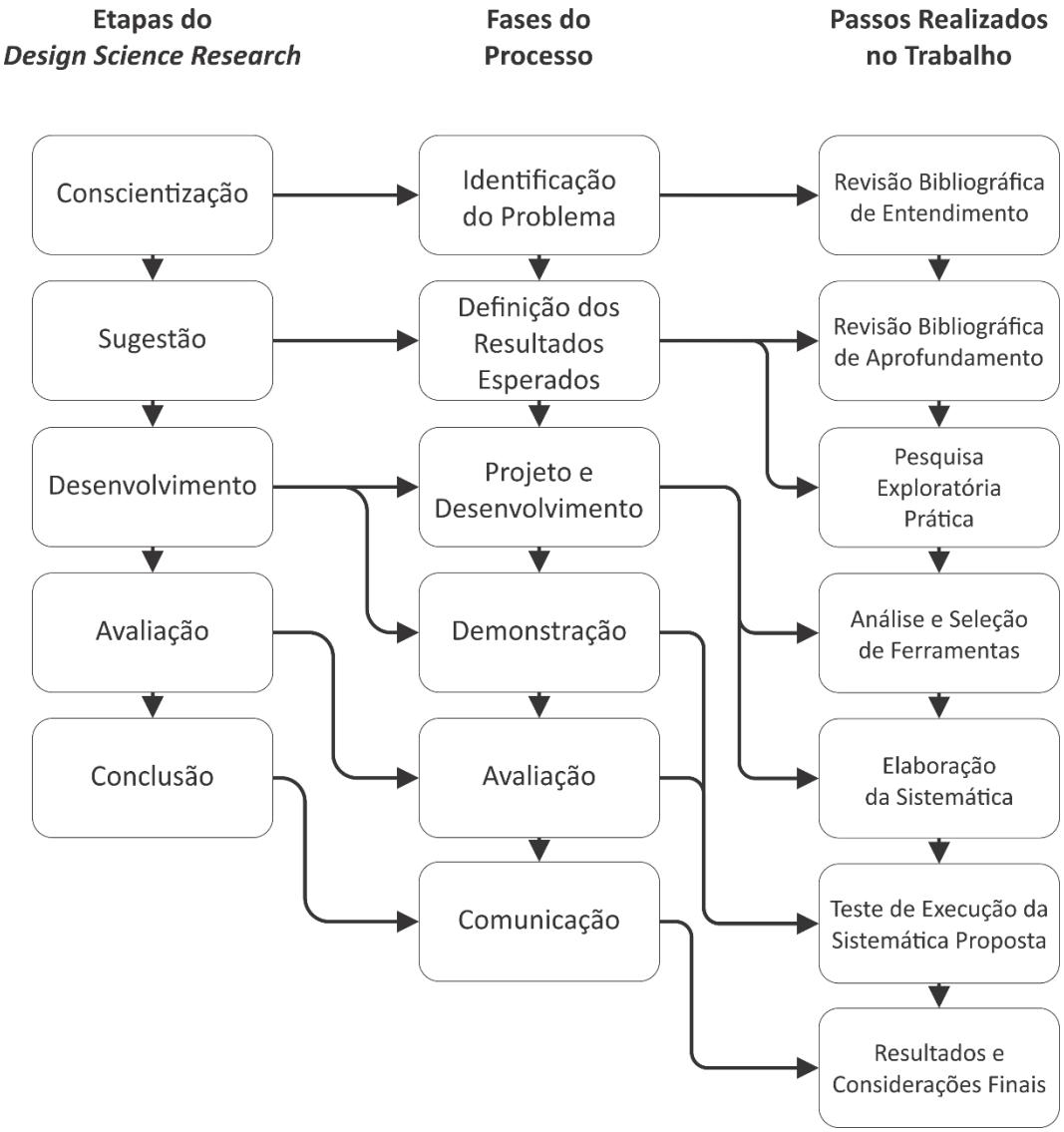
"O Desenvolvimento corresponde ao processo de constituição do artefato em si" (DRESCH et al., 2013, p. 750 apud MANSON, 2006). Ou seja, é na fase de Desenvolvimento em que se desenvolvem as soluções satisfatórias, também chamadas de artefatos[10], definidas na fase anterior com base na caracterização do problema realizada na primeira etapa.

A fase de Avaliação é quando se realiza a verificação do comportamento da solução desenvolvida no ambiente para o qual foi projetada. Para isso é necessário

[10] Segundo Dresch et al. (2013, p. 751), "a pesquisa fundamentada em *Design Science* deve produzir um artefato viável, na forma de um constructo, modelo, método e/ou uma instanciação".

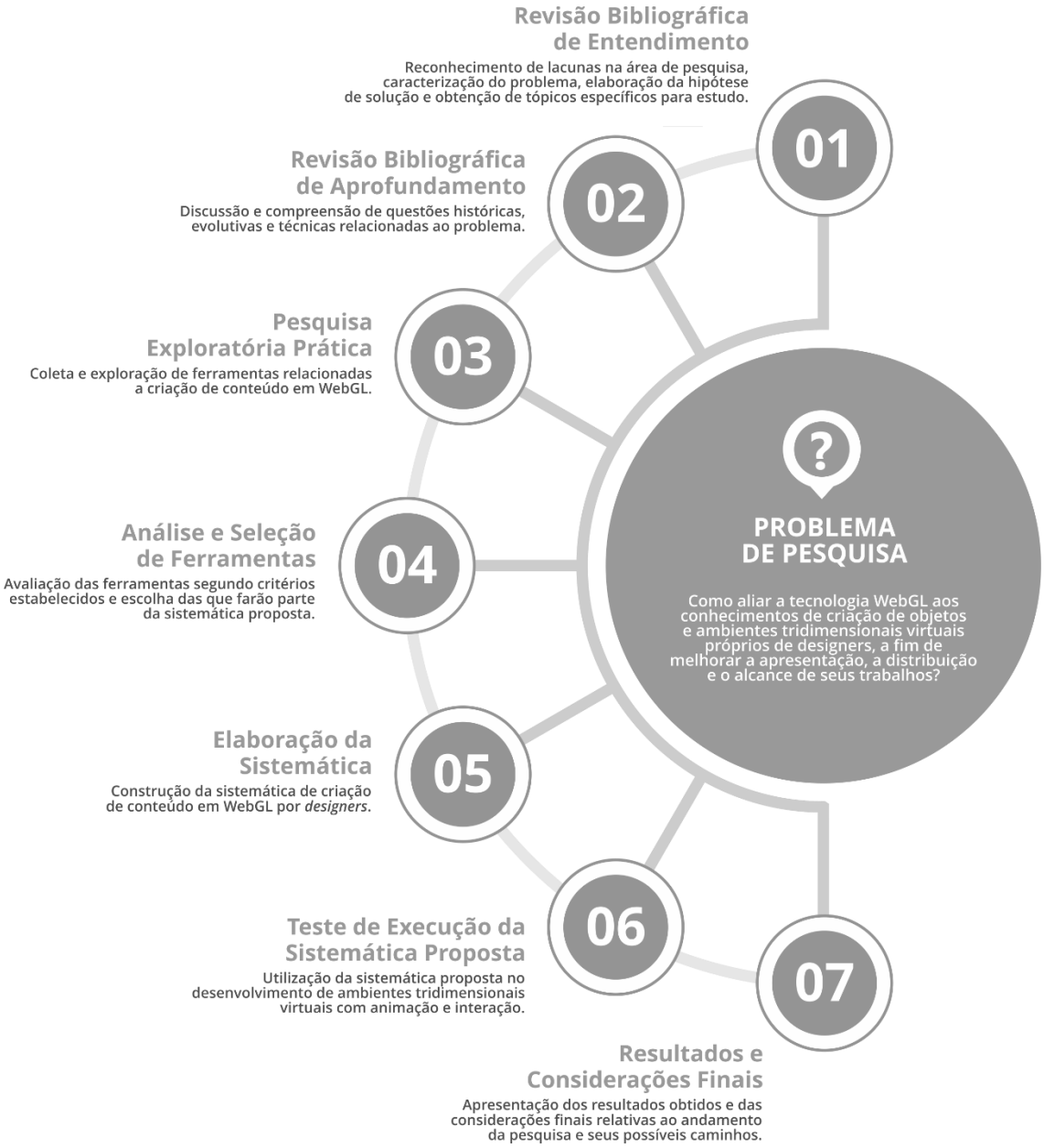
um processo composto por critérios, que validará ou não a solução criada. Por fim, na Conclusão, apresenta-se a síntese do processo de pesquisa acompanhada de uma discussão e, consequente, comunicação do mesmo à comunidade acadêmica e profissional.

Figura 35 – Etapas do *Design Science Research* adaptadas para o método de pesquisa utilizado.



Fonte: Adaptado de Dresch et al. (2013, p. 750).

Figura 36 - Método de pesquisa utilizado.



3.1. MÉTODOS

3.1.1. Revisão bibliográfica de entendimento

Primeiramente, diante do anseio de aproximar designers às capacidades de criação de projetos de ambientes virtuais interativos e atualizar aqueles que já

trabalham na área, realizou-se uma revisão bibliográfica de entendimento, objetivando traçar uma linha evolutiva das tecnologias destinadas ao desenvolvimento destes, sejam por designers ou programadores. Como retorno desta, obteve-se os principais tópicos a serem explorados a respeito das tecnologias de descrição de cenas virtuais tridimensionais, partindo do VRML, passando pelo X3D e chegando no WebGL, que configura hoje o estado da arte no que diz respeito ao tema.

Identificou-se, porém, que apesar de oferecer vantagens significativas em comparação a seus antecessores, o promissor WebGL ainda está distante de designers, que, ao menos no Brasil atual, em grande parte não possuem um conhecimento mínimo de programação, ainda que estejam familiarizados com *softwares* de modelagem 3D. Esta constatação se dá por meio da análise das grades curriculares dos oito melhores cursos de graduação em Design em universidades públicas e privadas do Brasil, segundo classificação da revista Guia do Estudante (2017).

Sete[11] dos oitos cursos analisados não possuem em suas grades curriculares qualquer disciplina ligada direta ou indiretamente ao ensino de programação aos estudantes de Design. E, ainda que um curso[12] tenha em sua matriz curricular uma disciplina (Conceitos de Informática) relacionada com a Informática, não se encontrou nada sobre seu conteúdo programático e a presença de assuntos ligados à programação.

Então, foram efetuados alguns ensaios livres de rigor metodológico, explorando-se ambientes virtuais dinâmicos, desenvolvidos a partir de VRML, X3D e WebGL, focando no reconhecimento de uma lacuna na área de pesquisa em questão. Em seguida realizou-se a caracterização do problema, com a clarificação de suas variáveis e elaboração da hipótese de solução através do método hipotético-dedutivo.

---

[11] Design (Projeto de Produto) – Pontifícia Universidade Católica - Rio (PUC Rio);
Design (Comunicação Visual) – Pontifícia Universidade Católica - Rio (PUC Rio);
Design Gráfico – Universidade Federal de Goiás (UFG);
Design – Universidade Estadual Paulista (UNESP);
Design – Universidade Federal do Amazonas (UFAM);
Design – Universidade de Brasília (UNB);
Design – Escola Superior de Propaganda e Marketing SP (ESPM SP).

[12] Design (Mídia Digital) – Pontifícia Universidade Católica - Rio (PUC Rio).

3.1.2. Revisão bibliográfica de aprofundamento

Tendo clara as condicionantes do problema, iniciou-se uma revisão bibliográfica de aprofundamento sobre as questões históricas, evolutivas e técnicas das tecnologias abordadas (VRML, X3D e WebGL). Em conjunto elaboraram-se discussões sobre assuntos elementares e de compreensão necessária para a sequência da pesquisa tais como, a tipificação e singularização da representação tridimensional virtual no Design, os campos da realidade virtual e aumentada, a qualificação dos ambientes tridimensionais virtuais em geral e a estruturação da programação orientada a objetos.

3.1.3. Levantamento de ferramentas

Após a revisão bibliográfica para obtenção de informações relevantes a análises futuras, realizou-se um levantamento das ferramentas disponíveis hoje que objetivam dar suporte à criação de conteúdos por meio do WebGL. O levantamento delas foi realizado com o auxílio de buscadores online com o Google.com, através de termos em português, inglês e espanhol. A lista de ferramentas obtidas (34 no total - Tabela 5) compreende modeladores, editores e exportadores de cenas 3D que possuem funcionamento online ou off-line. Preenchendo a necessidade de se ter instrumentos para todas as etapas da sistematização a ser proposta, da modelagem do ambiente até a sua publicação. Algumas delas consistem em um conjunto de software e complemento (plug-in), sendo este último o responsável pela exportação do conteúdo utilizando o WebGL nestes casos.

Tabela 5 – Modeladores, editores e exportadores de cenas 3D analisados.

Ferramenta	Funcionamento
Unity + Assets	*Off-line*
Blender + Blend4Web	*Off-line*
Blender + Babylon Exporter	*Off-line*
Blender + Sketchfab Exporter	*Online*
Rhino + Iris	*Off-line*
3ds Max 2017 + WebGL Exporter for Max	*Off-line*
3ds Max 2017 + SEA3D Exporter	*Off-line*
3ds Max 2017 + Three.js Exporter	*Off-line*
3ds Max 2017 + Babylon Exporter	*Off-line*
3ds Max + Sketchfab Exporter	*Online*
Autodesk Maya 2016 + Inka3D	*Off-line*
Autodesk Maya 2016 + WebGL Exporter For Maya	*Off-line*
AutoCAD 2017 + WebGL Exporter For Autocad	*Off-line*
A3DS Viewer	*Off-line*
CopperCube	*Off-line*
WebGL Publisher	*Off-line*
Atomic Game Engine	*Off-line*
FinalMesh	*Off-line*
Marmoset Toolbag	*Off-line*
Godot Engine	*Off-line*
Unreal Engine	*Off-line*
Autodesk Stingray 2017	*Off-line*
Goo Create	*Online* e *Off-line*
WebGLStudio.js	*Online* e *Off-line*
Three.js	*Online* e *Off-line*
Clara.io	*Online*
ShaderFrog	*Online*
PlayCanvas	*Online*
ThreeFab Alpha	*Online* e *Off-line*
Vizor.io	*Online*
SculptGL	*Online*
3DTin	*Online*
Cl3ver	*Online*
P3d.in	*Online*

3.1.4. Critérios de avaliação

Para comparação e seleção das ferramentas mais adequadas à proposta, foram criados dez critérios baseados nas discussões realizadas anteriormente, na experiência, citada anteriormente, com a disciplina de Design Virtual do PGDesign da UFRGS e a partir das experimentações dos próprios *softwares* levantados para a análise, visando a utilização dos programas a nível estudantil e profissional. Tais critérios (Tabela 6) servem para caracterizar as ferramentas e possibilitam verificar quais se destacam tecnicamente em cada função do processo. Também, cada critério é explicado em detalhes e sua criação justificada na sequência.

Tabela 6 - Critérios a serem considerados para a escolha das ferramentas que farão parte da sistemática proposta.

Nº	Critérios	Referências
1	Tipo de licença (gratuita, paga, avaliação, educacional...)	Disciplina de Design Virtual (PGDesign/UFGRS)
2	Funcionamento *online* ou *off-line*	Experimentação Prática dos *Softwares* Levantados
3	Sistemas operacionais suportados	Experimentação Prática dos *Softwares* Levantados
4	Possibilidade de publicação direta em WebGL (HTML) para execução em multiplataforma	(ZHANG, Xiaoyu; GRAČANIN, 2013, p. 195)
5	Exportação em código aberto ou fechado	Disciplina de Design Virtual (PGDesign/UFGRS)
6	Formatos de importação	Disciplina de Design Virtual (PGDesign/UFGRS)
7	Formatos de exportação	Experimentação Prática dos *Softwares* Levantados
8	Funcionalidade (modelagem, animação, interação)	Disciplina de Design Virtual (PGDesign/UFGRS)
9	Conhecimento prévio necessário	(GUIA DO ESTUDANTE, 2017)
10	Periodicidade das Atualizações	Experimentação Prática dos *Softwares* Levantados

No **critério de número 1** verifica-se quais os tipos de licenças os *softwares* possuem entre Gratuito, Pago, Versão Educacional Gratuita e Versão de Avaliação, podendo o desenvolvedor disponibilizar mais de um tipo para o mesmo programa. Para inclusão na sistemática a ser proposta prioriza-se os programas gratuitos ou os que oferecem versões educacionais devido ao acesso facilitado para uso no ensino. Programas somente com versões pagas não interessam em um primeiro momento e por isso tal critério é importante.

A categorização do *software* conforme seu modo de funcionamento (**critério nº 2**) entre *Online* ou *Off-line*, possibilita a identificação daqueles que podem funcionar integralmente mesmo sem conexão contínua com a internet, ainda que sejam executados em navegadores web. Em conjunto com o reconhecimento dos sistemas operacionais suportados (**critério nº 3**), formulam-se os principais requisitos para o funcionamento dos mesmos no que diz respeito à edição de conteúdo. Quanto mais abrangentes as possibilidades de execução nestes dois critérios, mais usuários poderão utilizá-las.

A capacidade de publicação direta do conteúdo criado em uma página HTML executável (**critério nº 4**) também tem grande valor, visto que diminui os passos necessários para o usuário do programa transformar a sua criação em um arquivo pronto para ser disponibilizado pela internet ou outros meios. Isso simplifica e reduz os conhecimentos prévios necessários para se chegar ao fim do processo. No caso de alguns programas que funcionam unicamente *online*, mesmo que esta publicação direta para um executável seja possível, não é disponibilizado o arquivo final de maneira *off-line*, somente sendo possível seu compartilhamento através dos serviços também on-line da empresa desenvolvedora.

O **critério nº 5** visa distinguir se o arquivo final gerado pelo programa é composto por código aberto ou fechado. Possuir código aberto significa que nele podem ser facilmente identificadas suas funções e diretrizes, possibilitando além da simples exportação do ambiente virtual para WebGL, um estudo da própria estrutura do código escrito e até mesmo alterações diretas nele, como visto no VRML. Já os com código fechado, também chamados de proprietários, são criptografados pela empresa que o desenvolve para impossibilitar a sua leitura e modificação pela

escrita. Apesar de não ser definidor para a o uso ou não do *software* na sistemática, é sempre desejável para fins educacionais que os códigos se apresentem de maneira aberta e clara.

A identificação dos diversos formatos de arquivos possíveis de serem importados para o programa (**critério nº 6**) é relevante. Quanto maior o número de arquivos suportados e a possibilidade de trabalho com os principais formatos do mercado, maior a integração com outros *softwares* que porventura possam fornecer recursos como modelos 3D, imagens, sons, etc.

A partir do **critério nº 7** são verificados os formatos trabalhados pelo programa para exportação do conteúdo desenvolvido no contexto da publicação em WebGL, desconsiderando formatos destinados a outros fins. De antemão são observados alguns possíveis formatos de arquivo como o HTML que pode existir sozinho, contendo todos os elementos do ambiente criado em um só arquivo ou pode vir acompanhado de outros arquivos que são referenciados no HTML principal e chamados quando a aplicação é posta em execução. Também comum para utilização no WebGL, os arquivos JSON carregam diversas informações da cena e podem ser inseridos em código externo para que possa ser gerado então um HTML executável por um navegador. A exportação direta em HTML simplifica o processo, contudo arquivos JSON possibilitam uma utilização aprimorada por usuários mais experientes.

Elementar, o **critério nº 8** dá espaço para a relação das principais funcionalidades dos *softwares* levantados. Um maior número de funções relacionadas aos pilares da criação de ambientes tridimensionais virtuais (modelagem, animação e interação) em um mesmo *software* pode reduzir o número de ferramentas necessárias no método proposto.

O **critério nº 9** avalia quais conhecimentos prévios são exigidos pelo *software* para desenvolver ambientes 3D animados e interativos. Em muitos casos podem ser necessários conhecimentos básicos ou avançados de programação, os quais não fazem parte das habilidades de grande parte dos designers e por isso serão evitadas. Por outro lado, pode ser preciso saber lidar com modelagem tridimensional virtual,

competência mais próxima dos profissionais desta área, não oferecendo grandes empecilhos para a sistematização.

Por último, o **critério nº 10** serve para conferir em qual período de tempo são disponibilizadas atualizações para as ferramentas. Este aspecto demonstra se o responsável pelo desenvolvimento do programa continua dando suporte ao seu produto. Uma frequência alta de atualizações pode significar uma correção de problemas e adição de funcionalidades mais rápida. Também é possível prever as chances de continuidade do suporte para evitar a utilização de programas que possam ser descontinuados em breve.

3.1.5. Requisitos da sistemática

Através dos dez critérios definiram-se os requisitos desejáveis e obrigatórios para o método proposto mais tarde (Tabela 7). Cada requisito está diretamente ligado a um dos critérios e aos resultados esperados ao final deste trabalho de pesquisa. A partir deles se torna possível a validação das ferramentas a serem utilizadas na sistemática.

Tabela 7 - Requisitos da sistemática de criação de ambientes 3D interativos por meio do WebGL.

Nº	Requisito	Condição
1	Ser composto por softwares com versões gratuitas ou educacionais	Obrigatório
2	Possibilitar a criação e edição de conteúdo sem a necessidade de conexão com a internet	Desejável
3	Funcionar nos três principais sistemas operacionais do mercado (Windows, OS X e Linux)	Obrigatório
4	Permitir publicação direta em WebGL sem ajustes avançados	Obrigatório
5	Ser o arquivo final constituído por código aberto	Desejável
6	Possibilitar a importação dos mais diversos formatos de arquivo	Obrigatório
7	Permitir a exportação do conteúdo diretamente em um HTML executável	Obrigatório
8	Abranger todas as três principais funcionalidades requeridas (modelagem, animação e interação)	Desejável
9	Não exigir conhecimento prévio de programação	Obrigatório
10	Ser constituído por softwares com suporte ativo e atualizações frequentes	Desejável

3.1.6. Seleção dos *softwares*

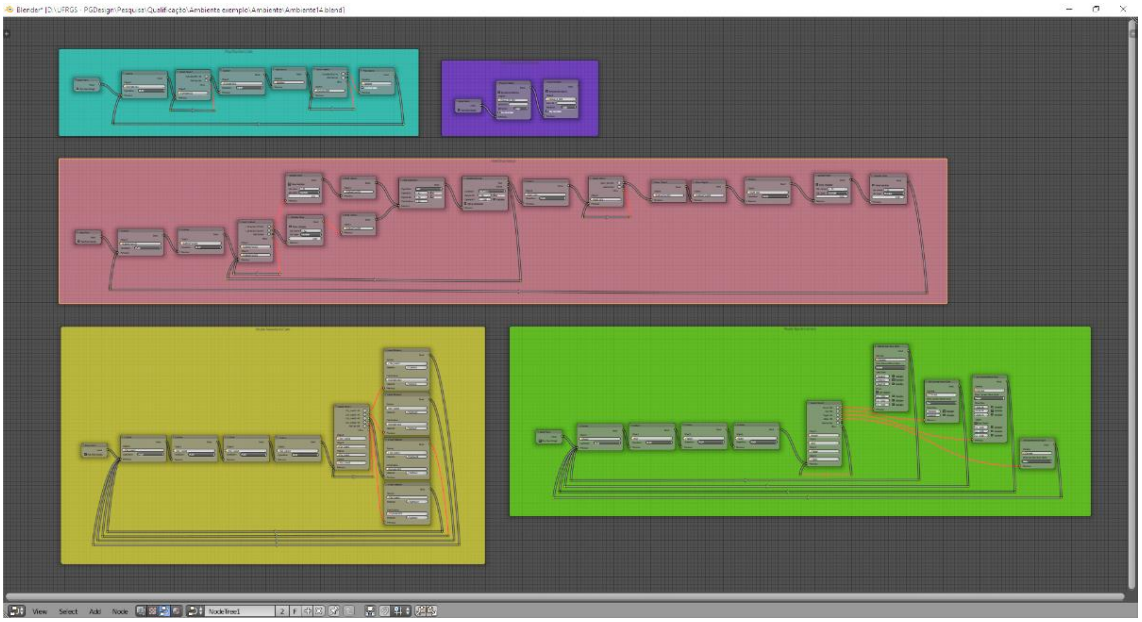
Após a construção dos critérios e requisitos, iniciou-se a análise prévia para identificar os softwares com maior potencial para atender as necessidades dos designers no momento da criação de seus ambientes e exportação em WebGL. Esta análise empírica e balizada pelos critérios estipulados se deu pela tentativa de criação e exportação de um cenário tridimensional simples composto por objetos 3D, animações e interações de baixa complexidade. Assim foram então obtidas as informações necessárias, segundo os critérios, para verificar o cumprimento ou não de cada um dos 10 requisitos para cada umas das 34 ferramentas (Apêndice A).

Infelizmente um grande número de ferramentas foram consideradas inadequadas para utilização na sistemática, visto que se mostram em desacordo com um ou mais requisitos obrigatórios estipulados para o cumprimento dos objetivos. Muitas exigem um conhecimento avançado de programação, principalmente para a criação de animações e interações, outras oferecem apenas versões pagas ou versões de avaliação com grandes limitações. Existem também aquelas que apresentaram problemas técnicos, não sendo possível chegar ao final do processo com a geração de um arquivo executável da cena.

Das 34 ferramentas analisadas apenas uma se mostrou inteiramente adequada a compor a sistemática em desenvolvimento cumprindo com todos os requisitos obrigatórios e 3 dos 4 desejáveis. Ela é constituída por dois *softwares* que trabalham em conjunto, o Blender, que une as capacidades de modelagem 3D, importação de objetos, animação e interação e o Blend4Web um complemento (*plug-in*) do Blender que proporciona a exportação facilitada do conteúdo para WebGL.

Ambos são totalmente gratuitos (critério 1), funcionam sem a necessidade de conexão direta com a internet (critério 2) e possuem versões para os três principais sistemas operacionais, Windows, Mac OS e Linux (critério 3), com suporte ativo e atualizações frequentes (critério 10). Ainda, o conjunto abrange por si só as três funcionalidades requeridas, modelagem, animação e interação (critério 8), sem que seja necessário algum conhecimento prévio de programação computacional para o trabalho com quaisquer delas (critério 9). Para isso, os softwares trazem embarcada uma linguagem de blocos (Figura 37), com a qual é possível desenvolver facilmente animações e interações de maneira predominantemente visual.

Figura 37 - Visualização geral de uma estruturação de blocos utilizando a linguagem embarcada no conjunto de *softwares* Blender + Blend4Web.



Os *softwares* permitem também a importação de objetos 3D em diversos formatos de arquivo (critério 6), o que facilita o fluxo de trabalho com outros softwares de modelagem tridimensional. E possibilitam a exportação do conteúdo 3D de forma simples diretamente em um arquivo HTML executável (critérios 4 e 7).

Apesar de não satisfazerem o requisito desejável nº 5, que diz respeito ao arquivo final ser constituído de código aberto para poder ser estudado posteriormente em forma de texto, verificou-se que as interações estruturadas em blocos suprem em parte este anseio, possibilitando uma leitura e entendimento facilitado da construção das mesmas.

Ao passo que os softwares Blender e Blend4Web em conjunto demonstraram potencial em uma primeira análise exploratória, um novo experimento mais aprofundado foi realizado para verificar em detalhes as funcionalidades oferecidas e que possibilitariam a difusão do WebGL como tecnologia útil e aplicável aos projetos de Design Virtual e ao ensino.

O experimento compreendeu a criação de um cenário (Figura 38 e Figura 39) composto por variados objetos 3D com animações e interações desenvolvidas a partir da linguagem de blocos embutida nos programas. Objetivou-se explorar e

demonstrar a potencialidade dos principais recursos oferecidos, bem como analisar o desempenho na execução do arquivo final (HTML) em computadores e smartphones através dos navegadores web mais utilizados no mundo (Google Chrome (60,5%), Mozilla Firefox (15,6%), Microsoft Edge/Internet Explorer (15,5%)), conforme pesquisa realizada em Abril de 2016 pela empresa StatCounter (CANALTECH, 2016).

Figura 38 - Parte de ambiente demonstrativo evidenciando o modelo de um carro com a paleta de possíveis cores para ele e à direita um espaço de estar.

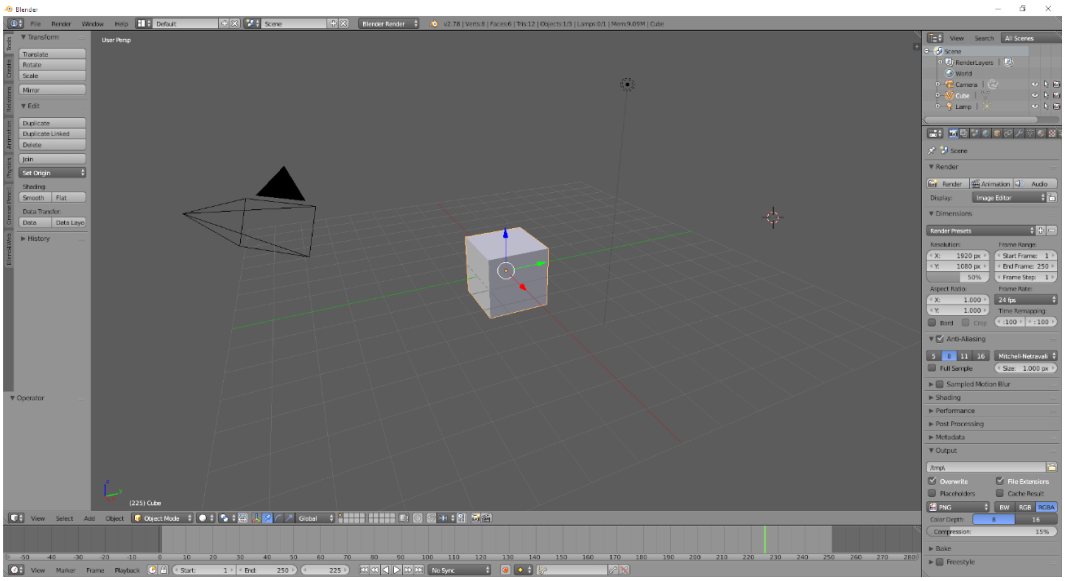
Figura 39 - Espaço de estar com interação nos objetos sobre a mesa e palavras na parede que acionam a troca do tipo de câmera.

Verificou-se que o complemento Blend4Web consegue trabalhar com a maior parte dos recursos do Blender relacionados a modelagem 3D, aplicação e edição de materiais e texturas, animações lineares, efeitos de física e partículas, iluminação, câmeras e outras configurações do espaço tridimensional, exportando com sucesso o conteúdo criado, diretamente em arquivo HTML único e executável. Essa exportação bem-sucedida, na maioria dos casos, se deve ao fato de que quando ativado, o Blend4Web retira da interface do Blender todos os recursos com os quais ele ainda não consegue trabalhar, restando para o usuário apenas aqueles suportados.

Além disso o Blend4Web adiciona ao sistema de programação visual, já incluso no Blender, novos blocos com funções de entendimento facilitado, não necessitando conhecimento prévio de programação para sua operação. A partir da relação dos blocos é possível criar inúmeras interações, entre elas, movimentação e visualização da cena; transformação de objetos, luzes e câmeras quanto a suas dimensões, posição, material, visibilidade e outros; acionamento de animações e sons; redirecionamento para páginas da web ou outros ambientes, etc. Ainda existe a possibilidade de inserção de interações mais complexas ou relacionadas a arquivos externos por meio de arquivos criados com a linguagem javascript, porém devido a este estudo ser voltado ao processo de trabalho de designers esse ponto não foi explorado.

Ressaltam-se também os recursos mais relevantes do conjunto Blender + Blend4Web relacionados à modelagem, animação e interação, observados durante as experimentações. Primeiro, o software Blender por si só possui uma interface (Figura 40) bastante semelhante aos demais programas de modelagem de modelagem 3D do mercado, sendo familiar aos profissionais que já utilizam estes. Por sua vez, o complemento Blend4Web se utiliza desta mesma interface, aproveitando-se de grande parte das ferramentas do Blender para fornecer os recursos necessários para a exportação dos ambientes utilizando a tecnologia WebGL. Dessa forma, praticamente toda a parte referente a modelagem tridimensional, configuração de luzes e câmera, o tratamento de materiais e mapeamento de texturas é suportada sem limitações significativas.

Figura 40 – Tela inicial do *software* de modelagem tridimensional Blender.



O conjunto de softwares também trabalha satisfatoriamente com vários tipos de animação, como as que compreendem as transformações de objetos como um todo (dimensão, posição, material, visibilidade...), as animações de esqueletos que permitem por exemplo a movimentação de personagens, as que trabalham com o áudio da cena que pode ser tanto reproduzido no ambiente como um todo ou com emissão concentrada em um só ponto, as que simulam os efeitos ao ar livre (água, vento, atmosfera...) e aquelas que retratam alguns efeitos de partículas como fumaça, fogo e respingos d'água.

Finalmente, as interações podem ser construídas na tela Node Editor, onde se encontra a linguagem de blocos do Blend4Web. Entre os blocos básicos disponíveis estão os que controlam sons; a visibilidade, transformações e materiais de objetos; os tipos e movimentações de câmeras; as animações presentes na cena; o realce de objetos por meio de um contorno e as ações executadas ao clique do mouse ou outro dispositivo suportado. Existem alguns outros blocos que possibilitam também guardar valores em variáveis, realizar operações matemáticas e verificar condições de igualdade. Estes últimos viabilizam a criação de interações com maior grau de complexidade sem que isso represente um aumento na dificuldade de utilização da presente linguagem.

Paralelo a isso, o conjunto Blender + Blend4Web proporciona diversos recursos nativos interessantes e coerentes com a caminho que segue a tecnologia digital atualmente. Um deles é a navegação e movimentação por meio de mouse, teclado, controles de videogames e telas de toque. Em aparelhos com giroscópio (*smartphones* e *tablets*) também é possível acessar uma visualização em tela dividida, própria para equipamentos como óculos de realidade virtual, preparando os projetos para a nova era da RV. Outros recursos nativos são:

- *Fast Preview*: Uma forma de visualização rápida do projeto no navegador web, antes de se exportar o arquivo HTML final. Junto a ela são disponibilizados alguns recursos para a identificação de erros e ajustes finos em tempo real nas propriedades da cena 3D. Ele proporciona maior controle no processo de desenvolvimento de ambientes e aplicações virtual tridimensionais.

- Otimizações: Verificou-se o suporte a otimizações comuns em tecnologias de representação anteriores como a suavização de geometrias e a cópia de elementos que compartilham informações de atributos entre si, tornando menor a exigência de processamento computacional para a execução do conteúdo e aumentando a taxa de quadros por segundo[13] (FPS, sigla em inglês) no momento de sua reprodução.

- Perfis de Qualidade Gráfica: Para todos os projetos exportados diretamente em HTML são criados automaticamente três perfis de qualidade gráfica (Baixo, Alto e Ultra) com diferentes níveis de aprimoramento de texturas, iluminação e outros. Isso amplia as chances de o ambiente ser executado com fluidez satisfatória em dispositivos de menor capacidade de processamento. Esses perfis também podem ser criados manualmente caso se deseje.

[13] "Quadros por segundo (FPS) é uma medida de como o movimento de vídeo é exibido. O termo se aplica igualmente ao vídeo de filme e vídeo digital. Cada quadro é uma imagem estática" (MICROSOFT, 2016b). Quando maior o número de quadros por segundo, maior a percepção visual de fluidez do conteúdo reproduzido.

Por fim verificou-se a fluidez de reprodução do ambiente criado nos três navegadores mais utilizados e já citados, em suas versões mais atuais. Para isso, registrou-se as taxas mínimas e máximas de quadros por segundo (FPS) no momento da execução, obtida por meio do recurso *Fast Preview* do próprio Blend4Web. Todas as análises foram realizadas em um mesmo computador e com o mesmo perfil de qualidade gráfica (Alta) para não haver possíveis influências externas. Ressalta-se ainda que o valor máximo possível de ser atingido em qualquer navegador é 60 FPS, devido a uma trava do sistema.

Tabela 8 - Taxas mínimas e máximas de quadros por segundo (FPS) durante a execução do ambiente WebGL nos três principais navegadores. Quanto maior, melhor.

	Google Chrome v.55	**Mozilla Firefox v.50**	**Microsoft Edge v.38**
Mínimo	58 FPS	55 FPS	57 FPS
Máximo	60 FPS	60 FPS	60 FPS

A partir da análise destes registros percebe-se que não há diferenças significativas quanto a fluidez de visualização do ambiente nos três navegadores, quando executada esta cena experimental. Também não é possível diferenciar visualmente as pequenas quedas na taxa. Pode-se dizer que neste teste não houve superioridade expressiva no uso de um destes navegadores para a execução de conteúdos gerados pelo conjunto Blender + Blend4Web, confirmando a possibilidade de uma distribuição mais abrangente dos mesmos.

Finalizam-se as discussões acerca da análise e seleção dos softwares e dá-se início a construção da sistemática de criação de ambientes tridimensionais virtuais por designers, que mais tarde é experienciada na criação de novos ambientes virtuais que esclarecem seus pontos fortes e fracos, que surgem como sugestões de novas pesquisas relacionadas. O presente trabalho é finalizado com a apresentação dos ambientes-teste em WebGL e a comparação deles com ambientes que utilizam o antecessor VRML. Também são discutidos os resultados e relatadas as considerações finais relativas ao andamento da pesquisa e seus possíveis caminhos.

Para possibilitar a comparação entre as cenas utilizando o WebGL e seu antecessor, utilizou-se dois *softwares* para exportar as cenas em VRML. Cada programa

gerou uma cena em VRML com diferentes características. A primeira exportação foi realizada a partir do próprio *software* Blender v.2.78c sem nenhuma alteração no ambiente e a outra com o software 3ds Max 2017, onde elas foram reconstruídas utilizando as ferramentas do programa exclusivas para VRML.

Para visualizar os arquivos ".wrl", formato padrão do VRML, foi preciso utilizar um *plug-in* que é instalado no computador e que funciona em conjunto com o navegador web do mesmo. Escolheu-se o *plug-in* Cortona3D por ser um dos únicos ativos atualmente e por executar adequadamente todos os recursos da linguagem VRML.

3.2. MATERIAIS

Descreve-se aqui os materiais, *softwares* e dispositivos empregados nas diferentes etapas desta pesquisa científica.

3.2.1. Revisão bibliográfica

Para uma revisão bibliográfica de relevância, buscou-se referências diversificadas. Foram escolhidos livros e e-books publicados que tenham sido revisados por uma equipe editorial. Artigos, monografias, dissertações e teses foram obtidas por meio de reconhecidas plataformas de indexação de conteúdo acadêmico (Periódicos CAPES, Elsevier, SciELO, Google Acadêmico e outros). Websites foram utilizados como fonte para a obtenção de informações explicativas básicas, de caráter público, quando não foi possível conseguir os dados procurados em meios de melhor reputação, sempre verificando a procedência dos mesmos.

Para organizar todo o conteúdo coletado utilizou-se o *software* Mendeley. Um gerenciador de referências acadêmicas que as ordena e classifica segundo dados inseridos pelo usuário ou obtidos automaticamente dos documentos nele cadastrados. Ele também facilita a inserção correta de citações em trabalhos acadêmicos, diminuindo drasticamente o tempo gasto neste processo.

3.2.2. Análise de ferramentas e avaliação da sistemática

Tanto as experimentações das ferramentas selecionadas para análise como os testes desenvolvidos com a sistemática foram realizados em um *smartphone* para visualização do conteúdo multiplataforma em WebGL e dois computadores *desktop*, um com sistema Windows e outro com Mac OS, para visualização e edição.

O *smartphone* é um LG G3 D855P, que conta com um processador Qualcomm Snapdragon 801 de quatro núcleos com velocidade de 2.5GHz cada, GPU Adreno 330, 2GB de memória RAM, tela IPS LCD de 5,5 polegadas, resolução de 1440 x 2560 pixels e rodando o sistema operacional Android em sua versão 6.0.

A configuração do primeiro *desktop* é a seguinte: processador Intel Core i5-4670 com quatro núcleos de 3,4GHz cada, placa de vídeo Nvidia Geforce GTX 760 com 2GB de memória GDDR5, 8GB de memória RAM DDR3 1600MHz. Este *desktop* roda o sistema Windows 10 Pro de 64 bits atualizado frequentemente.

O segundo computador é um Mac Mini (modelo final de 2012) com processador Intel Core i5 (modelo não especificado) de 2,5GHz de velocidade em cada um de seus dois núcleos, processador gráfico Intel HD Graphics 4000, 4GB de memória RAM DDR2 1600MHz, rodando o sistema operacional Mac OS Sierra com manutenção constante de suas atualizações.

Capítulo 4

SISTEMÁTICA

O processo de criação de ambientes tridimensionais virtuais (Figura 41) em desenvolvimento é suportado por três pilares (Figura 42). Estes foram definidos a partir das discussões anteriores sobre realidade virtual e análise histórica e técnica das tecnologias predecessoras de descrição de cenas virtuais. Portanto, a sistematização considera a adição destes três parâmetros ao espaço virtual, que por fim é exportado em formato executável em múltiplas plataformas seguindo os princípios do WebGL (Figura 43).

Figura 41 – Processo básico de criação de ambientes 3D virtuais com animação e interação.

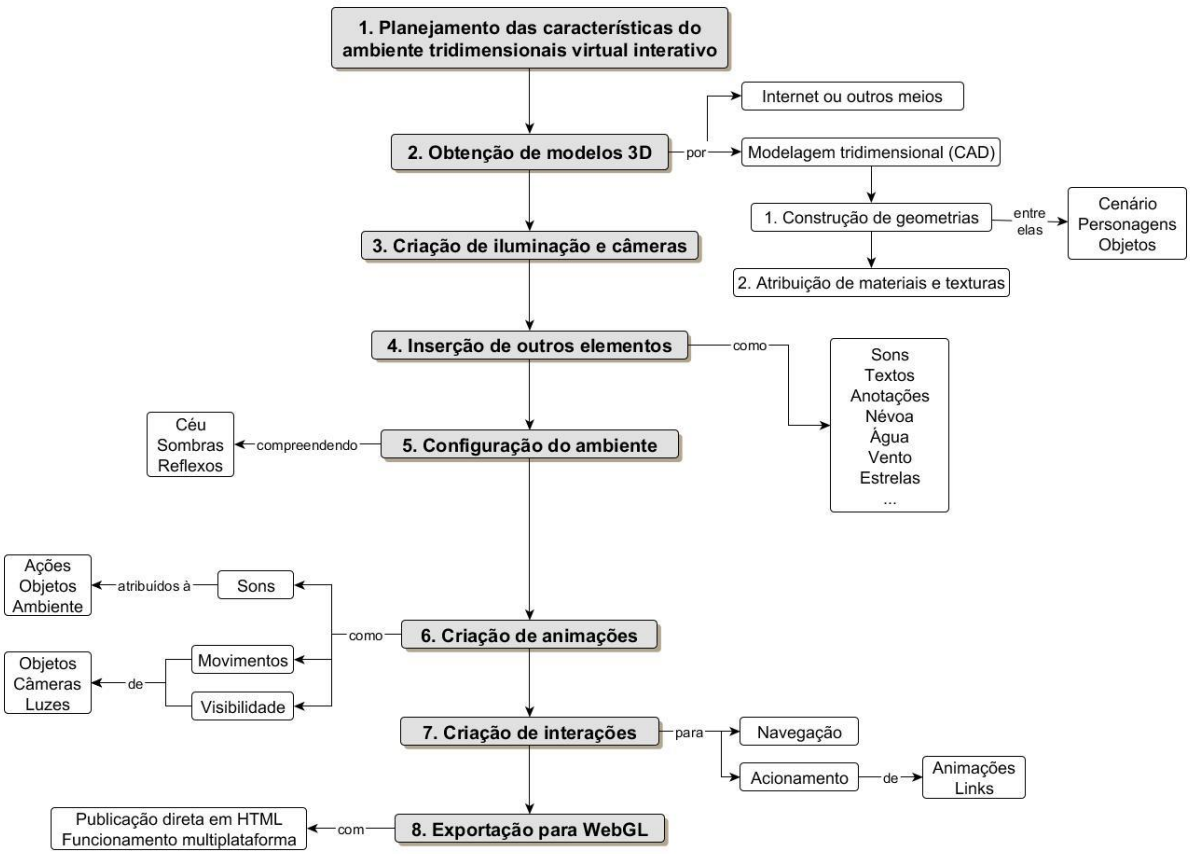


Figura 42 - A sistemática deve satisfazer três principais questões: modelagem, animação e interação.

A sistemática (Figura 43) é subdivida em onze macro fases que abrangem a obtenção e instalação dos softwares requeridos, a preparação para o uso, a apresentação da interface geral, a obtenção dos modelos tridimensionais, a inserção de luzes, câmeras, sons e textos, a configuração do ambiente e sombras, a adição de efeitos ao ar livre, a criação de animações e interações e por fim, a exportação do ambiente virtual pronto para visualização em computadores e dispositivos móveis como *smartphones*, *tablets* ou qualquer outro que possua um navegador web compatível com a tecnologia WebGL. Cada macro fase é dividida em tópicos menores que auxiliam na organização da sistemática e por consequência ajudam o usuário a localizar o assunto desejado com maior facilidade.

Salienta-se que o conjunto de softwares utilizados aqui oferece ainda mais funcionalidades do que as abrangidas por esta sistemática. Porém são exploradas somente aquelas que se encontram de acordo com os requisitos do trabalho de pesquisa e que também são relevantes à execução de projetos da área de Design Virtual.

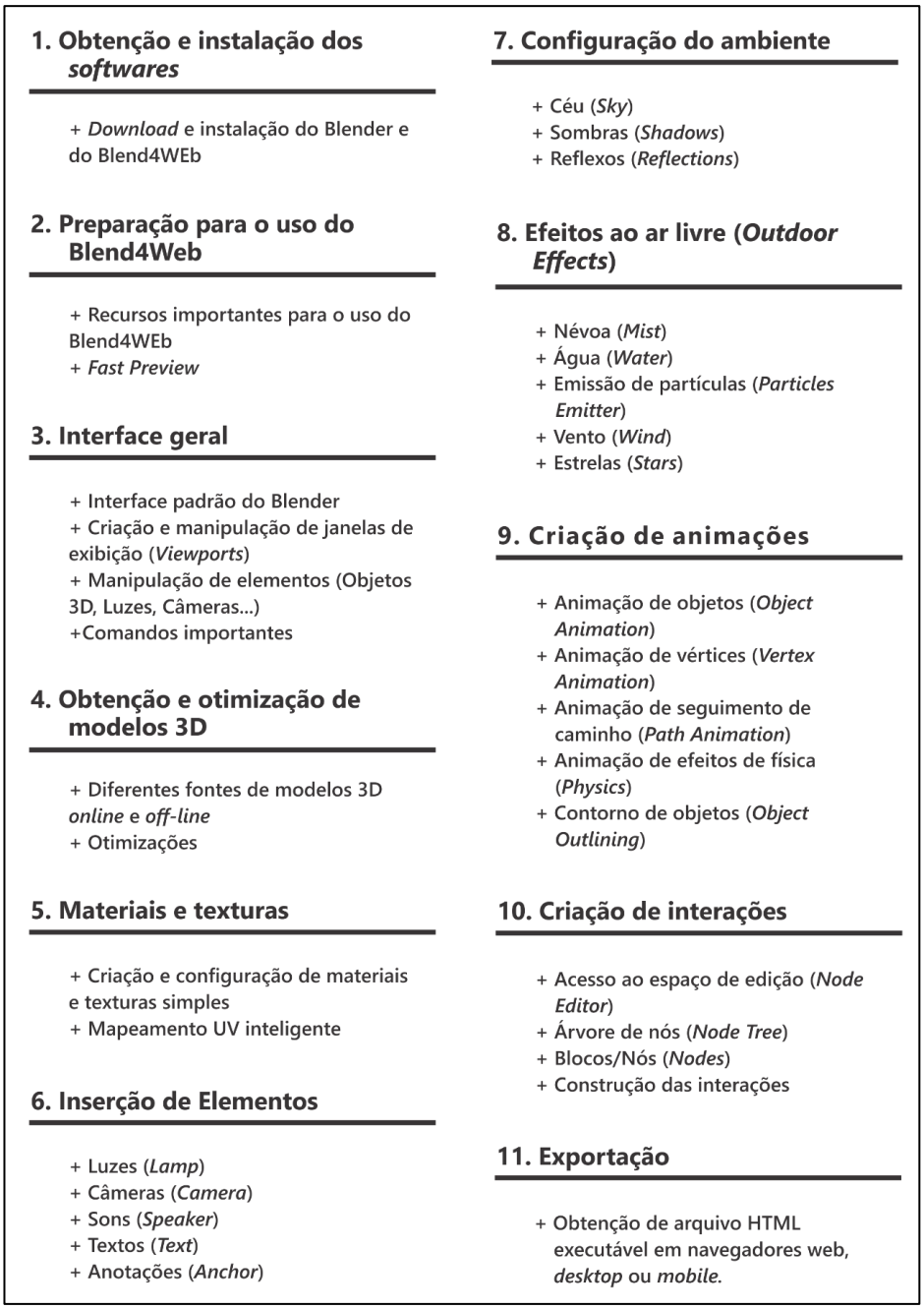
Este é um processo que visa a construção básica de ambiente virtual 3D animados e interativos por designers utilizando o WebGL. Portanto, recursos e propriedades mais avançadas podem ser estudadas e aprofundadas pelo próprio usuário posteriormente nos manuais oficiais dos programas[14] e nos diversos tutoriais

[14] Link para os manuais dos programas:
Blender - https://www.blender.org/manual/
Blend4Web - https://www.blend4web.com/doc/en/

em texto e vídeo disponíveis na internet, a fim de obter melhores resultados de acordo com as especificidades de cada projeto.

Também, ressalta-se que, em decorrência da legibilidade deste documento, somente alguns tópicos iniciais e finais da sistemática são apresentados na íntegra aqui. O conteúdo completo foi publicado na forma de um livro digital intitulado 'Ambientes Virtuais 3D Interativos – utilizando Blender + Blend4Web' pela editora Marca Visual (ISBN 978-85-61965-48-8), tendo como autores, Bruno Spanevello Pergher e José Luís Farinatti Aymone. Cada seção suprimida da sistemática traz a indicação de sua localização no referido livro e ao menos uma imagem que a ilustra.

Figura 43 – Panorama geral da sistemática proposta.



4.1. OBTENÇÃO E INSTALAÇÃO DOS *SOFTWARES*

Inteiramente gratuitos em suas versões completas[15] e constantemente atualizadas, Blender e Blend4Web são disponibilizados para *download* em seus endereços virtuais oficiais: Blender (https://www.blender.org/) e Blend4Web "CE" (https://www.blend4web.com/en/). Tanto um como o outro também possuem versões para Windows Vista ou superior (32bit ou 64bit), Mac OS (64bit) e Linux (32bit ou 64bit). Seguem os atuais requerimentos de hardware descritos na página oficial do Blender (Tabela 9).

Demais instruções a respeito da instalação dos programas podem ser encontradas no Capítulo 1 do livro já mencionado, entre as páginas 21 e 26.

Tabela 9 – Configurações de *hardware* exigidas para o funcionamento adequado dos *softwares* Blender e Blend4Web.

	Hardware **Mínimo**	***Hardware*** **Recomendado**	***Hardware*** **Ideal**
Processador (CPU)	32-bit Dual Core 2Ghz com suporte SSE2	64-bit Quad Core	64-bit Eight Core
Memória RAM	2 GB	8 GB	16 GB
Monitor	24 bit com resolução de 1280x768 pixels	24 bit com resolução de 1920x1080 pixels	2 monitores 24 bit com resolução de 1920x1080 pixels
Periféricos	Mouse ou Trackpad	Mouse de 3 botões	Mouse de 3 botões ou Mesa de desenho (Graphics Tablet)
Placa de Vídeo (GPU)	512 MB RAM e compatível com OpenGL 2.1	2 GB RAM e compatível com OpenGL 3.2	2 placas com 4GB RAM e compatíveis com OpenGL 3.2

Fonte: BLENDER (2017).

[15] No site do Blend4Web é possível encontrar duas versões gratuitas do software, a "CE" (Recomendada) que traz todos os recursos de desenvolvimento (ex.: *Scene Viewer*, *Project Manager*, *Fast Preview*) e o "*Add-on*" que possibilita apenas os recursos básicos de exportação. Existem também duas versões pagas que basicamente possuem alterações na licença de uso dos conteúdos, um suporte técnico diferenciado e uma maior biblioteca de materiais, sem acrescentar ferramentas ou recursos relevantes à criação dos cenários tridimensionais.

4.2. PREPARAÇÃO PARA O USO DO BLEND4WEB

Depois da instalação dos dois programas, deve-se atentar para dois procedimentos iniciais necessários para que o complemento Blend4Web funcione adequadamente. Esses procedimentos devem ser realizados sempre que se inicie um arquivo novo.

1. O primeiro procedimento necessário é a escolha do Blend4Web como motor de renderização do Blender no topo da janela principal (Figura 44). Assim, são habilitadas todas as opções de configuração do Blend4Web e também são assinaladas ou ocultadas da interface as ferramentas do Blender com as quais o complemento ainda não trabalha adequadamente, certificando que não haja erros, durante a exportação, pela utilização de recursos não suportados. Após habilitado, são apresentadas diversas configurações do complemento Blend4Web na área *Properties* (Propriedades em inglês), identificada por uma barra no topo contendo vários ícones (Figura 45).

Figura 44 – Alteração do motor de renderização.

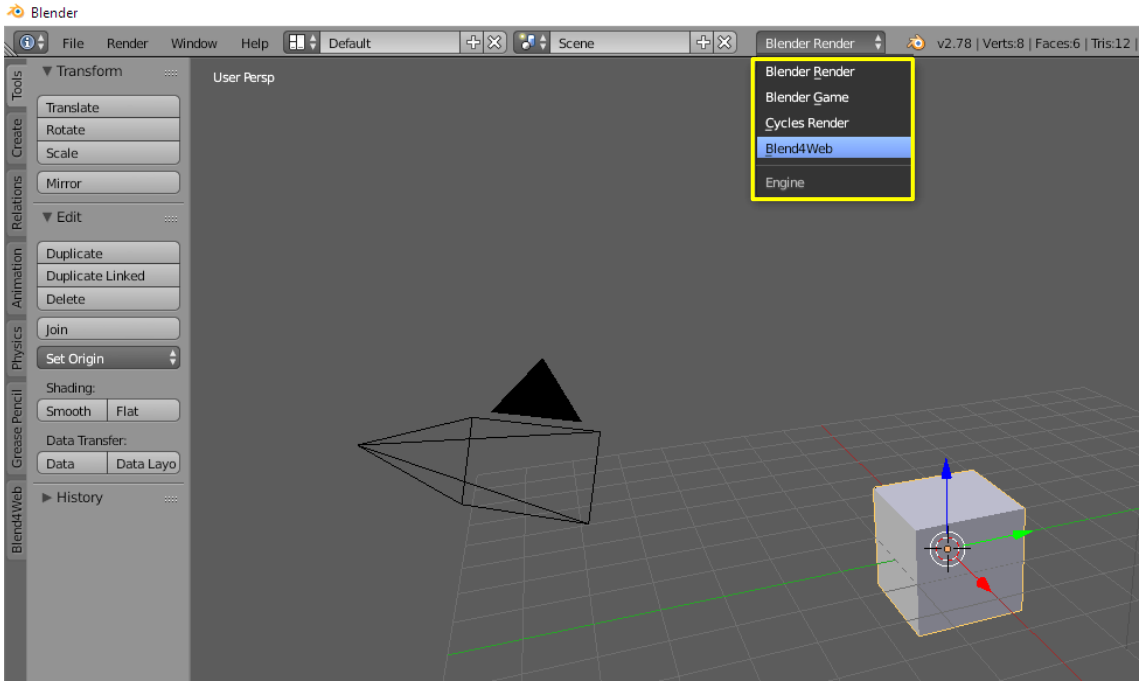


Figura 45 – Barra de ícones que identifica a janela de propriedades do Blender.

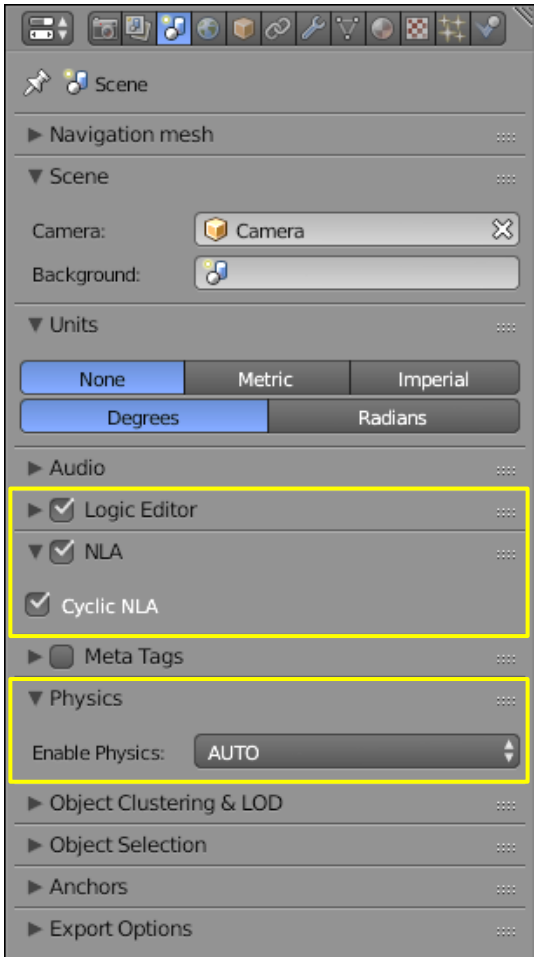
2. Antes do desenvolvimento do ambiente 3D propriamente dito, deve-se atentar para algumas opções acessadas pelo menu *Scene* (terceiro ícone da barra de propriedades - Figura 45). Para a correta exportação dos elementos 3D, animações e interações desenvolvidas deve-se habilitar as seguintes opções (Figura 46), sempre ao iniciar um novo projeto:

- ***Logic Editor***: responsável por exportar a lógica de interações criadas por meio da linguagem de blocos;

- ***NLA* (com *Cyclic NLA*)**: responsável pela exportação e execução de alguns tipos de animações;

- ***Enable Physics***: permite a exportação de efeitos de física eventualmente presentes na cena. Deve estar configurado como *AUTO* ou *ON*.

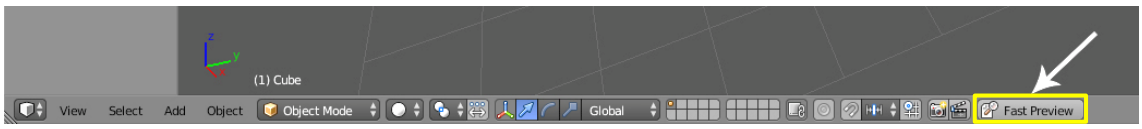
Figura 46 – Opções a serem habilitadas no menu *Scene* antes do início da criação.



4.2.1. *Fast Preview*

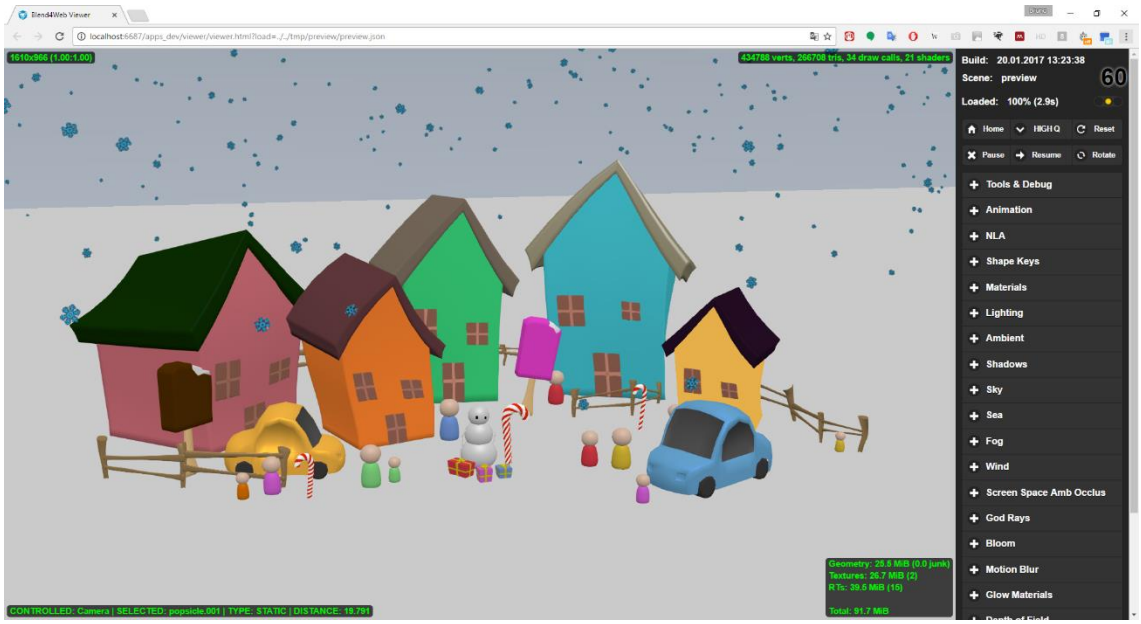
O *Fast Preview* é um recurso que se apresenta na forma de um ambiente para a realização de testes do conteúdo que está sendo desenvolvido. É acionado por meio do botão também denominado *Fast Preview* na barra inferior da janela *3D view* (Figura 47). Ele ajuda a reduzir o tempo gasto com o projeto, permitindo a realização de inúmeros testes de forma rápida e com ferramentas extremamente úteis para a identificação de erros na cena.

Figura 47 – Localização do recurso *Fast Preview*.



Ao clicar no botão *Fast Preview*, o programa realiza a exportação do projeto em desenvolvimento com as configurações do momento e em formato de um arquivo temporário, que é substituído por outro na próxima utilização do recurso. Logo após, o arquivo gerado é então aberto no navegador web padrão do computador dentro do *Scene Viewer*, um ambiente de testes onde é possível visualizar uma série de informações sobre o projeto tais como dados das geometrias, resolução da visualização, tamanho do arquivo e outros (Figura 48). Há também uma barra na lateral direita com a qual é possível realizar uma série de alterações nos valores das propriedades de vários recursos que foram inseridos na cena. Essas alterações modificam a cena ao lado em tempo real, porém não são salvas nem no arquivo gerado para testes nem no arquivo do projeto no Blender.

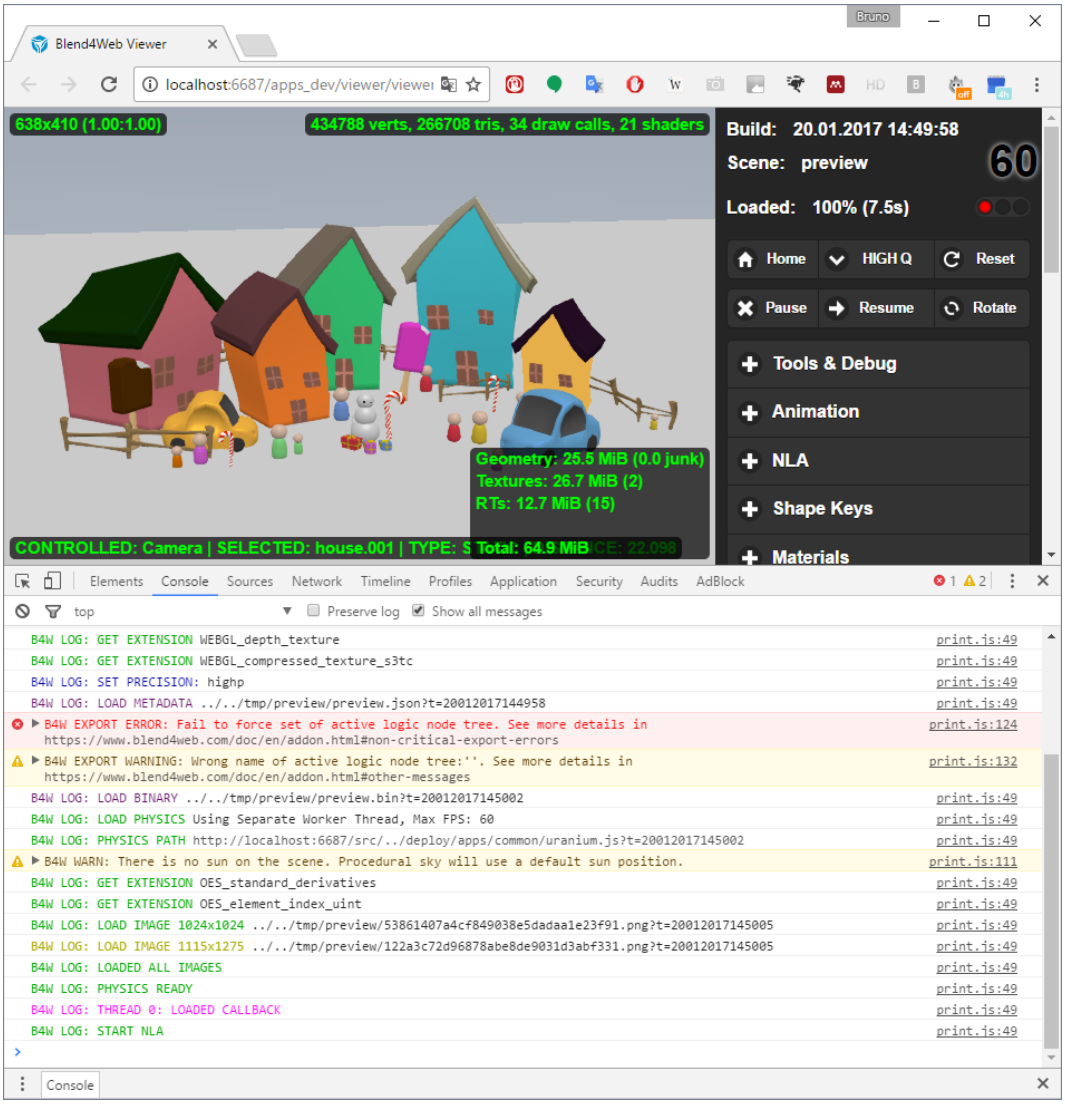
Figura 48 – *Scene Viewer* aberto pelo comando *Fast Preview* para a realização de testes.



Em se tratando de verificação de possíveis erros, existe um sistema de semáforo (Figura 49), no canto superior direito da tela, que avisa ao usuário se a cena contém problemas relacionados às restrições do programa por meio das cores **verde** (sem erros), **amarelo** (erros de menor gravidade) e **vermelho** (erros de maior gravidade). Nem sempre os erros acusados ocasionam falhas na exportação final se não corrigidos, mas podem indicar problemas na relação entre elementos e outros recursos utilizados. Para visualizar quais são os erros encontrados, é necessário abrir no navegador a janela que contém algumas ferramentas destinadas aos desenvolvedores web. Nos principais navegadores (Google Chrome, Mozilla Firefox, Safari e Microsoft Edge) essas ferramentas são acessadas pela tecla F12. Ali, geralmente na aba *Console* (Figura 50), são apresentadas as descrições dos problemas precedidas um dos seguintes termos: *B4W WARN*, *B4W EXPORT WARNING* e *B4W EXPORT ERROR*.

Figura 49 – Exemplo da sinalização da presença de erros por cores na tela do *Scene Viewer*.

Figura 50 – Aba *Console* da janela de ferramentas de desenvolvedor do navegador Google Chrome.

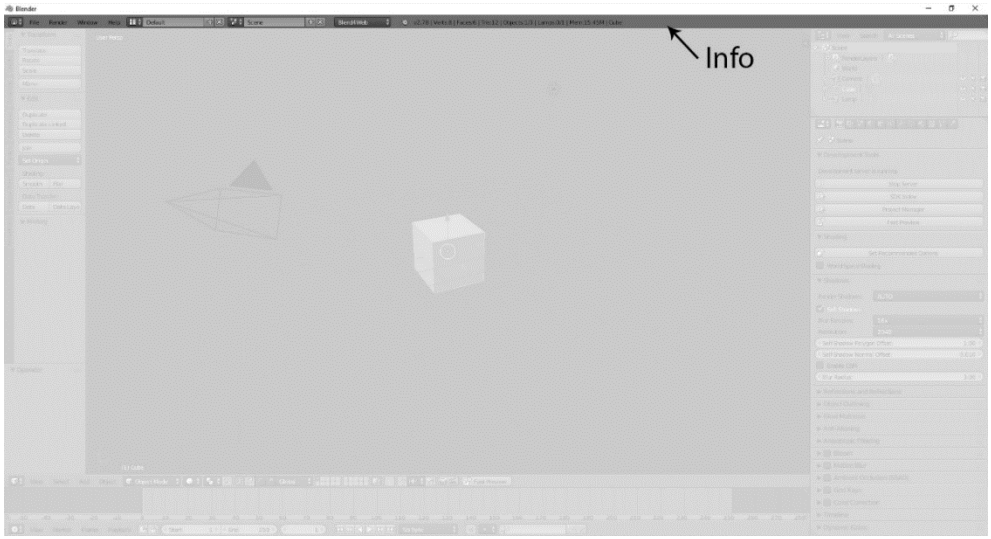


4.3. INTERFACE GERAL

O ensinamento da utilização das ferramentas e propriedades básicas de modelagem e montagem de cenas tridimensionais não faz parte deste livro. É importante, porém, facilitar a localização dos recursos apresentados. Por isso, para um entendimento geral da interface partilhada pelos *softwares* aqui adotados, segue uma breve descrição e mapeamento das áreas e janelas mais relevantes. Ressalta-se que todas as janelas do Blender podem ser movidas e dispostas, conforme a preferência do usuário e o que se apresenta aqui é baseado na disposição padrão do *software*.

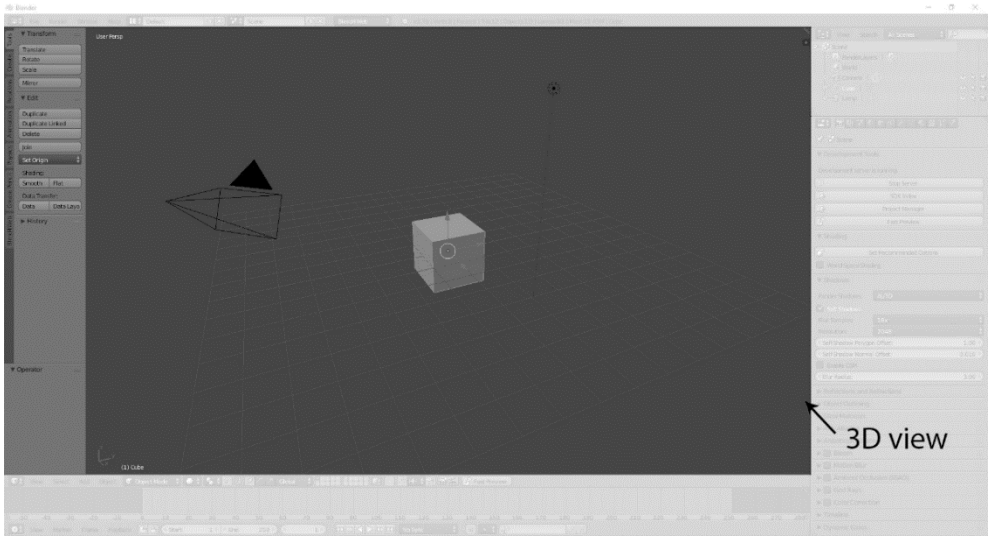
- ***Info*** (Figura 51): contém os acessos às configurações mais gerais do programa e sua interface. Traz também os atalhos para o manual de uso e algumas informações sobre o projeto em edição;

Figura 51 – Localização da barra *Info*.



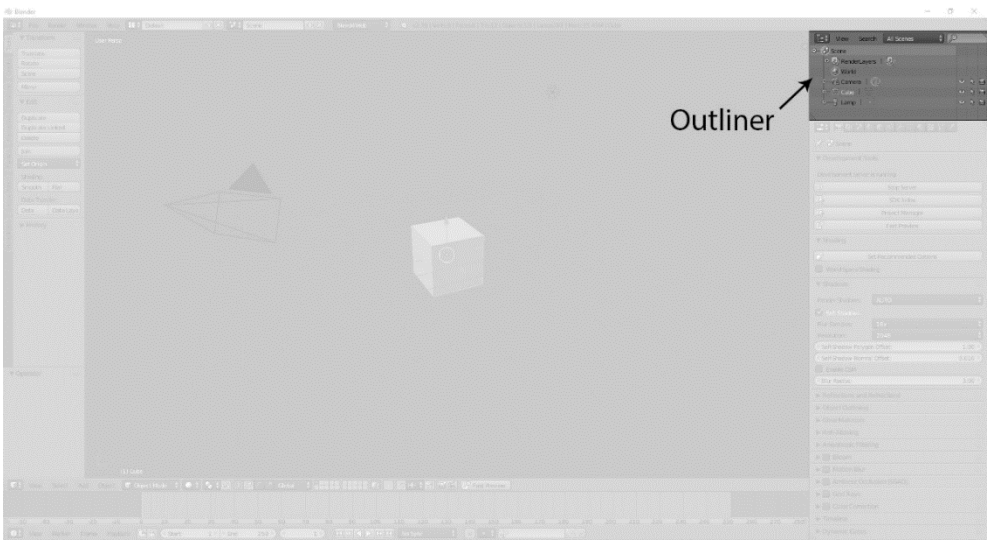
- ***3D View*** (Figura 52): é a principal janela de visualização do espaço tridimensional do programa e dos objetos 3D. Na esquerda, encontra-se um agrupamento de ferramentas de movimentação e escala de objetos, criação de geometrias, luzes e câmeras, relações entre elementos e funções básicas relacionadas a animações e efeitos de física;

Figura 52 – Localização da janela *3D view*.



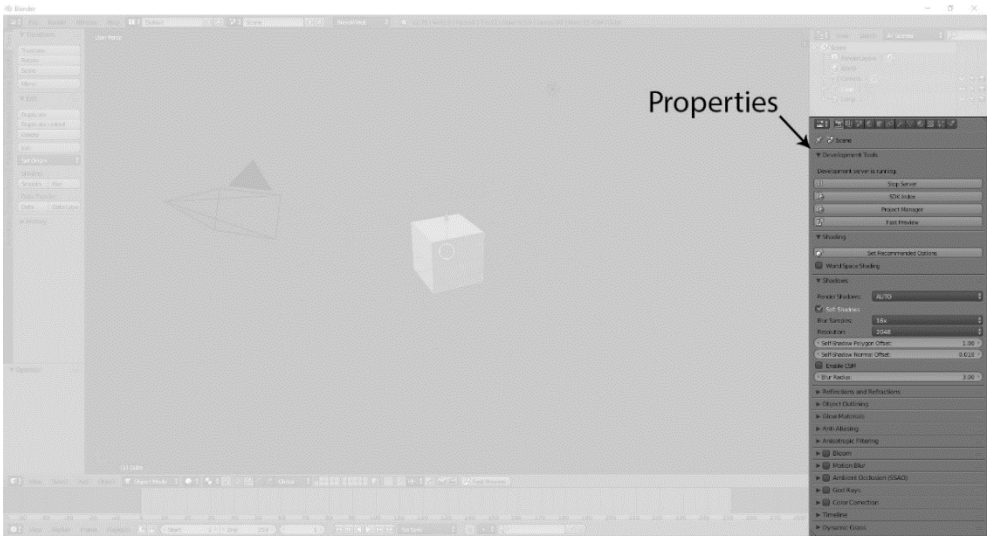
- ***Outliner*** (Figura 53): possui a lista estruturada de todos os elementos (objetos 3D, câmeras, luzes...) do projeto. Através dela é possível selecionar com precisão o elemento desejado e alterar propriedades relacionadas à visibilidade;

Figura 53 – Localização da janela *Outliner*.



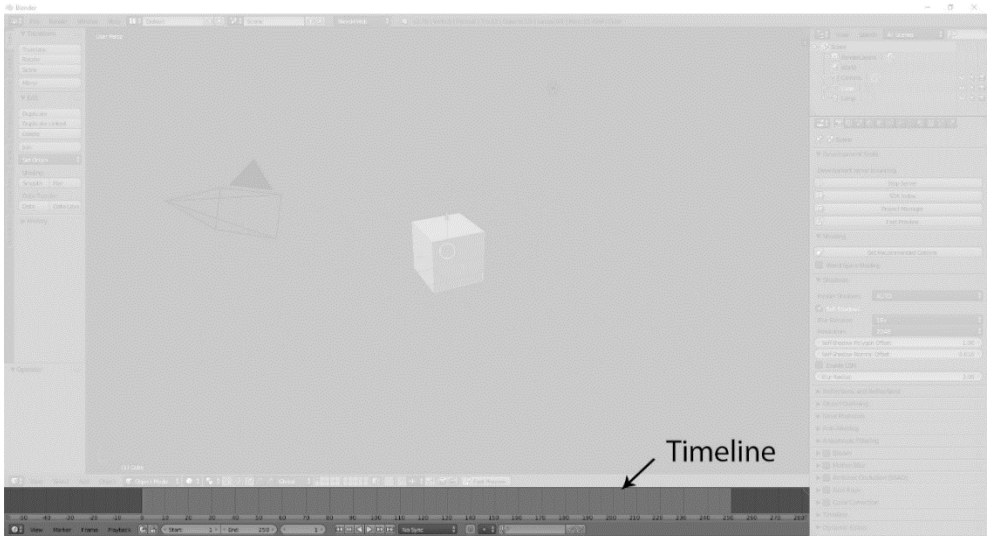
- ***Properties*** (Figura 54): contém as propriedades básicas e avançadas, divididas por abas, referentes a cada elemento criado. Também abrange as configurações do cenário e ambiente, bem como aquelas específicas do Blend4Web;

Figura 54 – Localização da janela *Properties*.



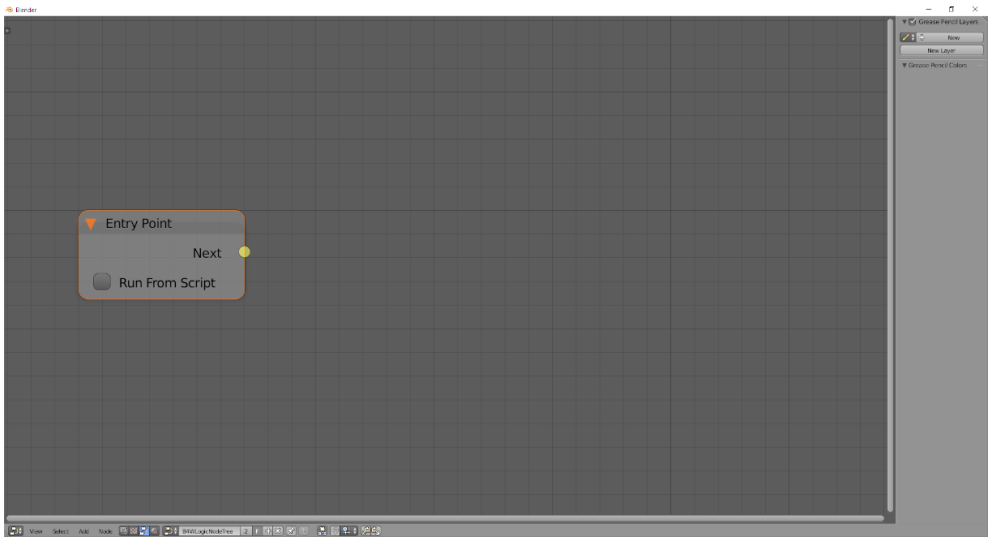
- ***Timeline*** (Figura 55): uma das áreas destinadas à edição de animações no Blender. Muito utilizada para a criação de animações mais simples, originadas pela interpolação de condições (*Keyframes*) definidas ao longo da sequência de quadros;

Figura 55 – Localização da janela *Timeline*.



- ***Node Editor*** (Figura 56): oculto logo quando é criado um novo documento no Blender. O *Node Editor* é onde se constrói as lógicas das interações através da linguagem visual de blocos do Blender acrescida de recursos próprios do Blend4Web.

Figura 56 – Janela do *Node Editor*.



4.3.1. Criação e Manipulação das Janelas de Exibição (*Viewports*)

As Janelas de Exibição, chamadas de *Viewports* em inglês, são as janelas pelas quais se visualiza a cena tridimensional que se está criando. No Blender, a janela *3D View* cumpre esse papel e, em muitos casos, é útil a presença de mais de uma *viewport* em conjunto no ambiente de trabalho do programa. Essas múltiplas janelas podem, por exemplo, auxiliar no posicionamento correto dos elementos no espaço virtual. Quando se tem mais de uma *viewport* (janela *3D View*) aberta, é possível que cada uma delas tenha uma forma diferente de visualização da cena em desenvolvimento.

Por padrão, quando se inicia um novo projeto no Blender, já é aberta uma janela 3D View. Para criar outras, existem quatro maneiras:

OBS: Estes métodos servem tanto para as janelas *3D View*, como para a abertura e duplicação dos demais tipos de janelas do *software* Blender.

1. A primeira maneira de criar janelas outras janelas *3D View* é clicando em um dos ícones triangulares localizados nos cantos inferior esquerdo e superior direito da janela 3D View aberta (também presente nos cantos de qualquer outra janela) e arrastá-lo na horizontal ou na vertical (Figura 57). Isso fará com que uma duplicação da janela 3D View seja aberta na direção em que o triângulo foi arrastado (Figura 58).

Figura 57 – Ampliação e localização dos triângulos localizados nos cantos inferior esquerdo e superior direito da janela *3D View*.

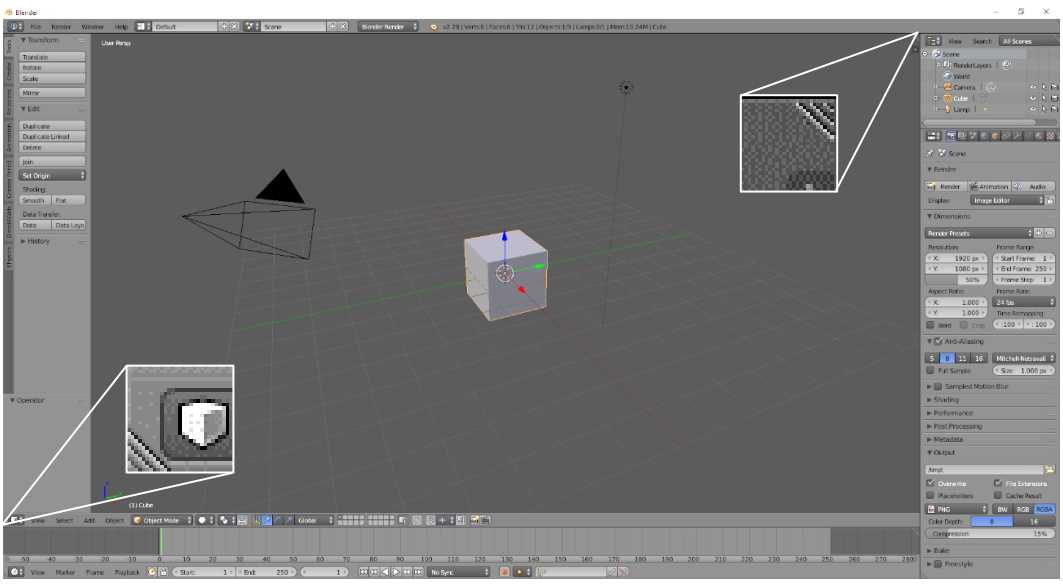
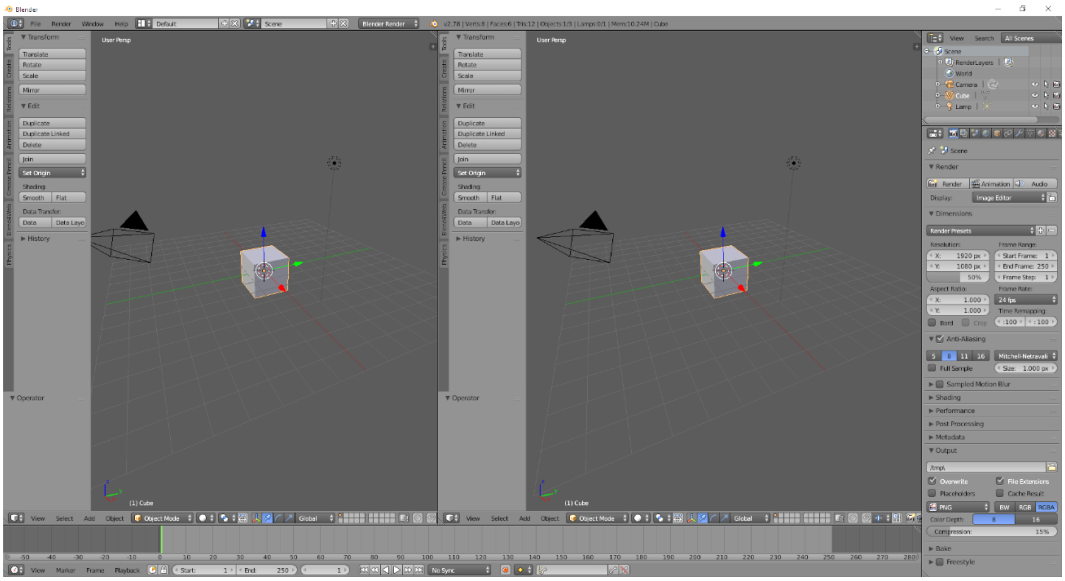
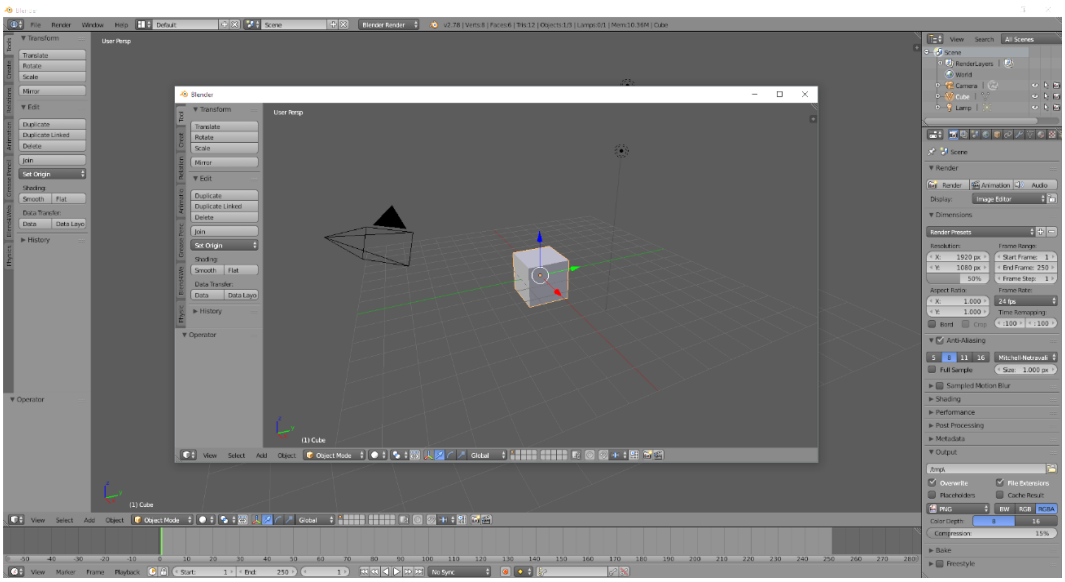


Figura 58 – Duas janelas *3D View* abertas lado a lado.



2. A segunda forma é uma variação da primeira. Mantém-se pressionada tecla SHIFT do teclado ao clicar em um dos ícones triangulares localizados nos cantos inferior esquerdo e superior direito da janela *3D View* aberta (também presente nos cantos de qualquer outra janela) e arrastá-lo na horizontal ou na vertical (Figura 57). Assim, uma cópia da janela *3D View* será aberta "destacada" das demais (Figura 59). Pode ser movida livremente e também disposta em um monitor secundário, se existir.

Figura 59 – Cópia da janela *3D View* destacada das demais.



3. A terceira forma consiste em clicar com o botão direito do mouse em qualquer linha limite (as mais externas) de qualquer janela, *3D View* ou outra. Com isso, um menu se abre (Figura 60) e, clicando na opção *Split Area,* é possível escolher alguma das janelas abertas para ser dividida em duas do mesmo tipo. A escolha do ponto exato para a divisão da janela desejada se dá por meio de uma linha, horizontal ou vertical, que acompanha o ponteiro do mouse quando este está sobre alguma das janelas (Figura 61). Para confirmar a divisão, basta um clique com o botão esquerdo do mouse na posição desejada.

Figura 60 – Menu (ampliado) com a opção *Split Area*, que se abre com o clique do botão direito do mouse entre duas janelas.

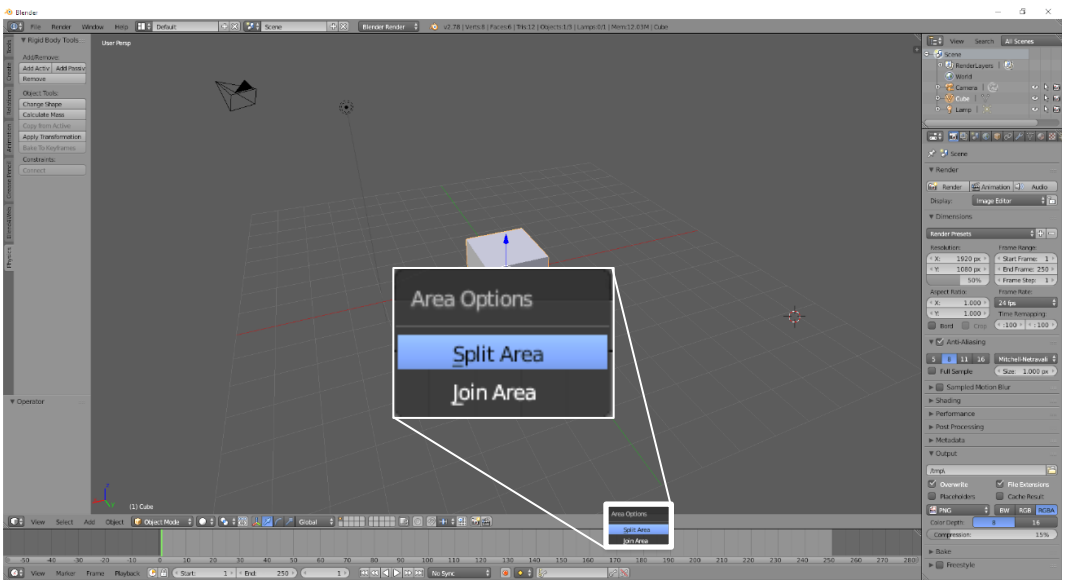
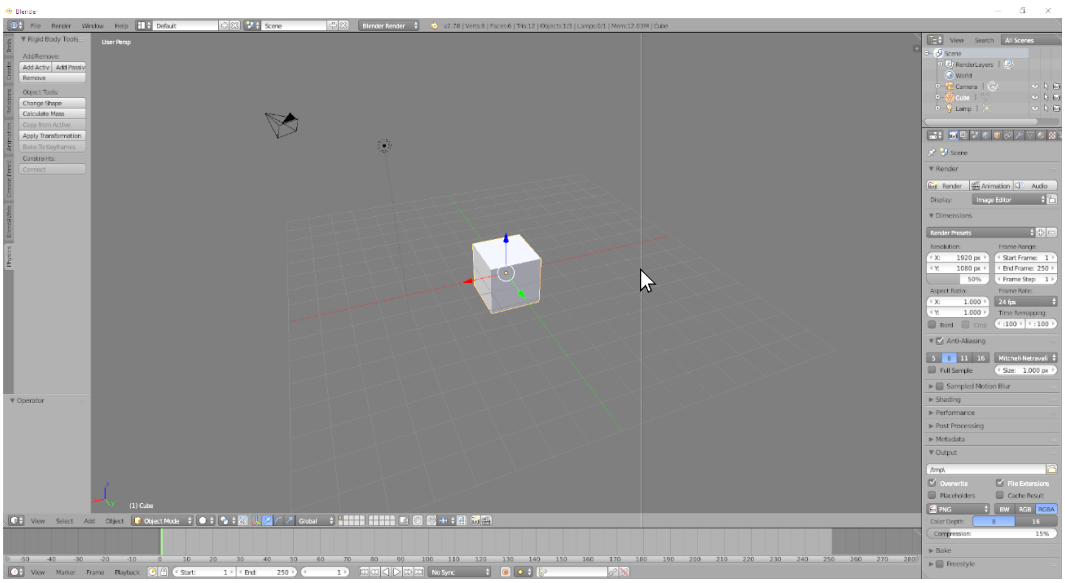


Figura 61 – Linha de divisão de janelas que acompanha o ponteiro do mouse e determina o ponto de divisão da janela.



4. A quarta e última maneira de criar cópias da *viewport* principal é substituir outra janela aberta por uma janela *3D View*. Para isso, basta clicar no botão de troca de janela (Figura 62), localizado em um dos cantos da janela que se pretende substituir (o canto varia dependendo da janela), e escolher a opção *3D View* no menu que se abre (Figura 63). A janela escolhida no menu assumirá o local e as dimensões da anterior (Figura 64). Para redimensioná-la, é só clicar com o botão esquerdo do mouse nas linhas limites da janela e, mantendo pressionado, arrastar para definir o tamanho desejado.

Figura 62 – Diferentes localizações dos botões de troca de janela (ampliados), de acordo com o tipo de janela.

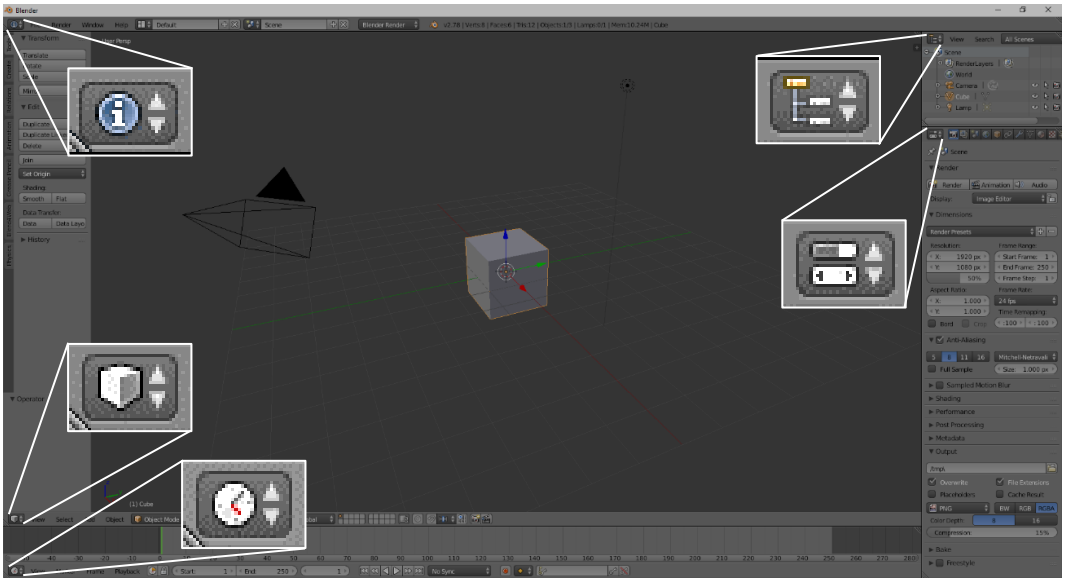


Figura 63 – Menu de seleção de janela ativa (ampliado) com seleção na opção *3D View*.

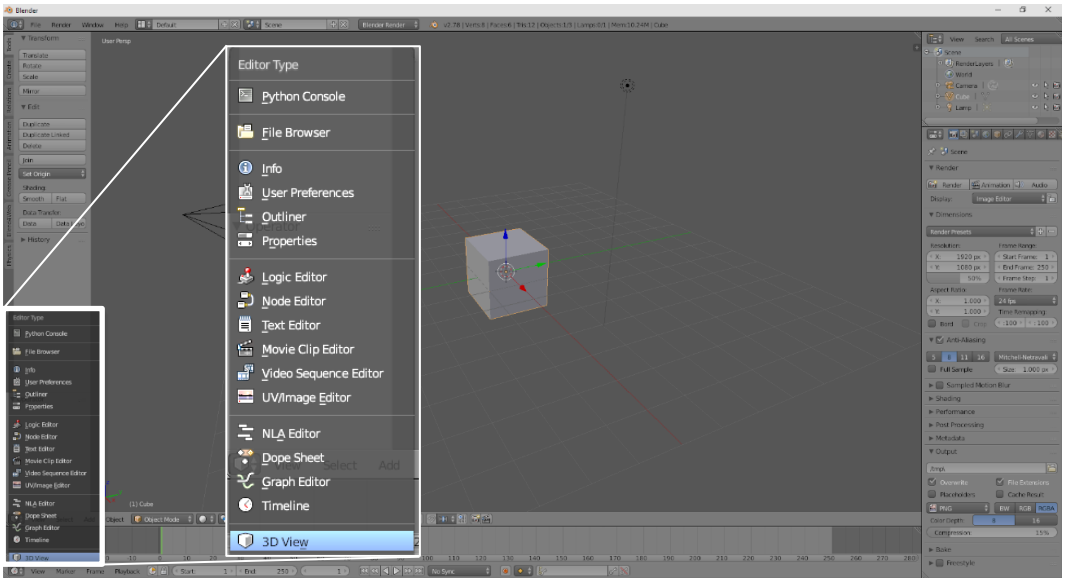
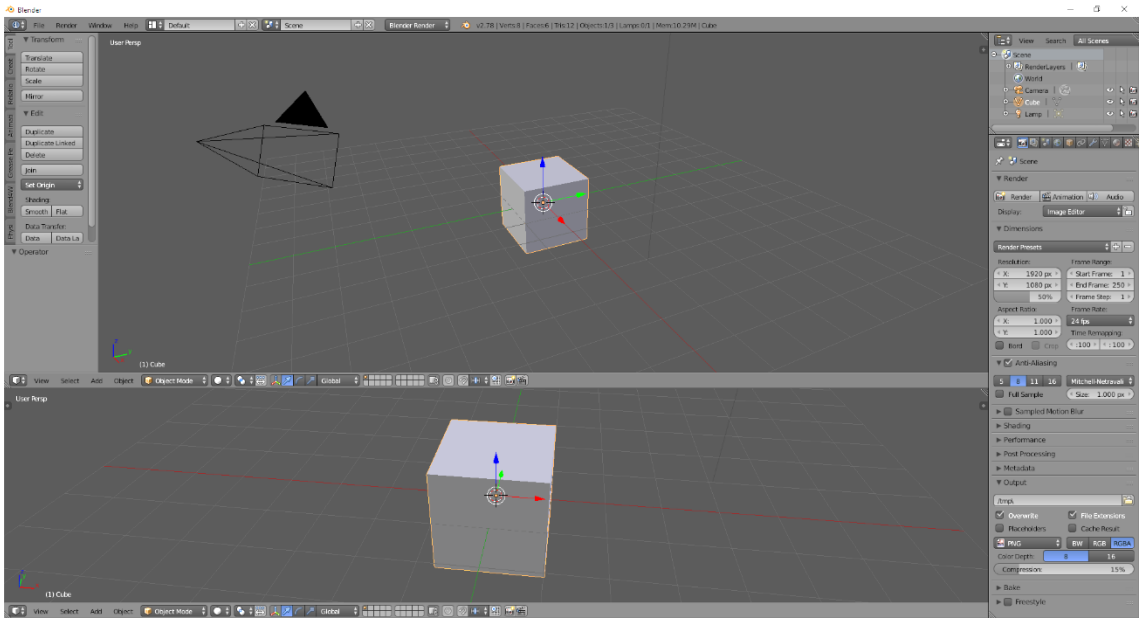


Figura 64 – Duas janelas *3D View*. A de baixo está no lugar original da *Timeline* que foi substituída e redimensionada.



Todas as janelas *3D View* abertas, bem como as demais janelas do programa, podem ser redimensionadas ao clicar com o botão esquerdo do mouse, manter pressionado e arrastar as linhas que as delimitam, tanto na horizontal como na vertical. O procedimento é o mesmo para o redimensionamento de qualquer menu lateral das janelas. Estes por sua vez, quando reduzidos até o limite externo das janelas das quais pertencem, são ocultados por completo, aparecendo em seu lugar um ícone com o sinal de positivo (+) (Figura 65), que ao ser clicado faz com que o menu reapareça. Em qualquer caso, redimensionamento de janelas ou menus, ao posicionar o ponteiro do mouse sobre as linhas limites de ambos, surge um ícone no formato de duas setas na vertical (nos limites inferior e superior) ou horizontal (nos limites esquerdo e direito) (Figura 66).

Figura 65 – Duas janelas *3D View*. A da esquerda com seu menu lateral esquerdo ocultado e em seu lugar um ícone '+' (ampliado).

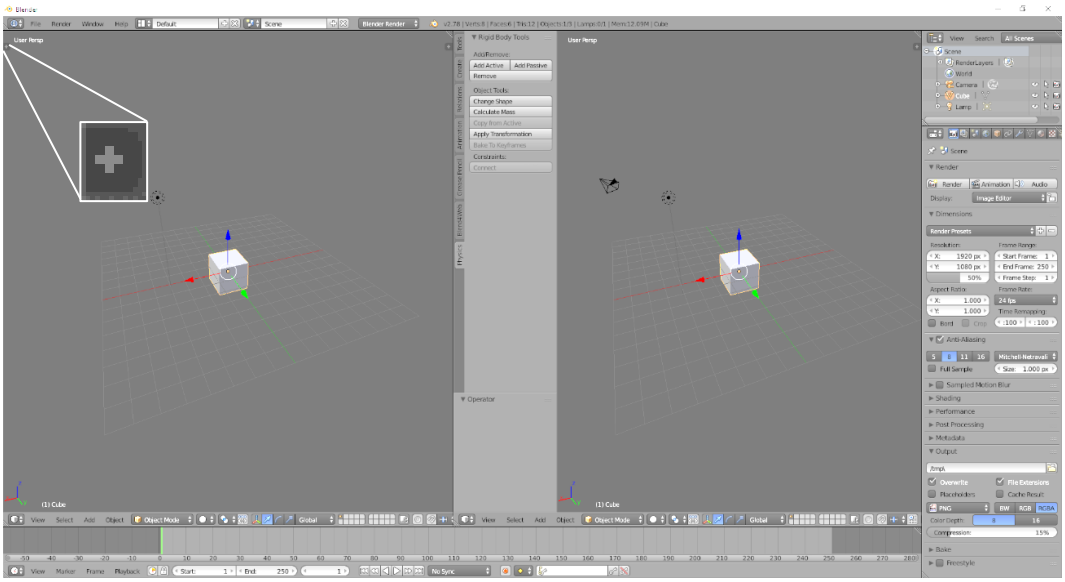
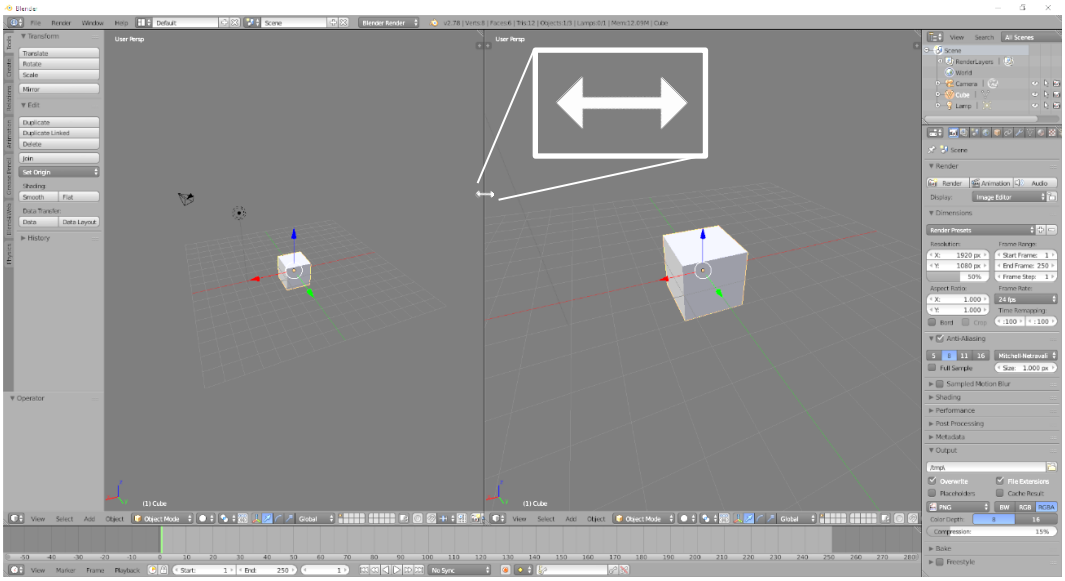


Figura 66 – Ícone (ampliado) indicando linha limite entre janelas.



Já para fechar qualquer janela, incluindo a *3D View*, deve-se clicar com o botão direito do mouse em uma das linhas limites entre duas janelas e, no menu que se abre, escolher a opção *Join Area* (Figura 67). Depois, ao passar o mouse sobre uma das janelas divididas pela linha clicada, ela escurece e surge uma seta indicando que ela será fechada ao clique do botão esquerdo do mouse (Figura 68).

Figura 67 – Menu (ampliado) com a opção *Join Area*, que se abre com o clique do botão direito do mouse entre duas janelas.

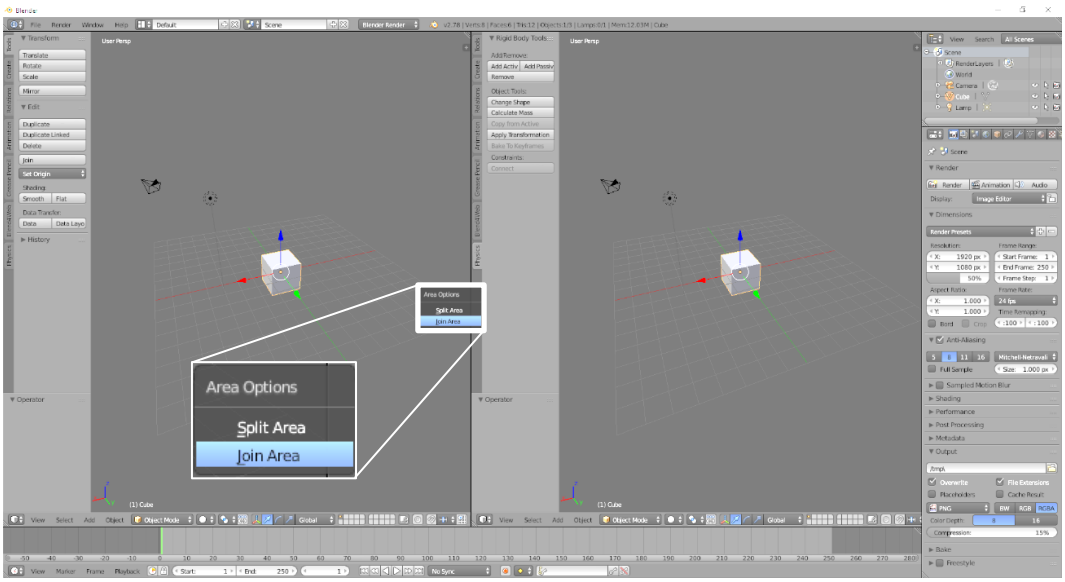
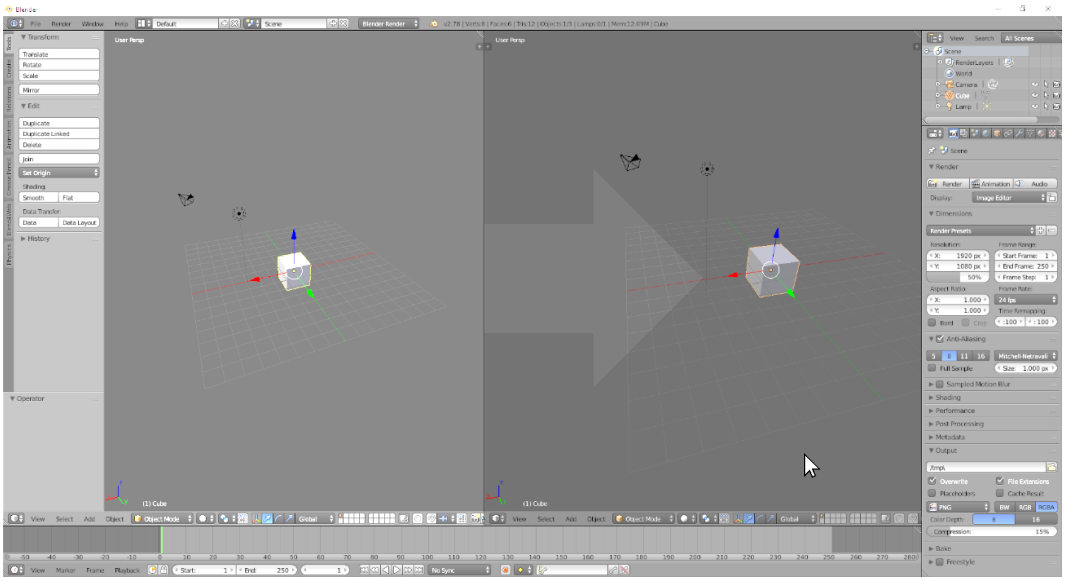


Figura 68 – Indicação da janela, selecionada com o ponteiro do mouse, que será fechada ao clique do botão esquerdo do mouse.

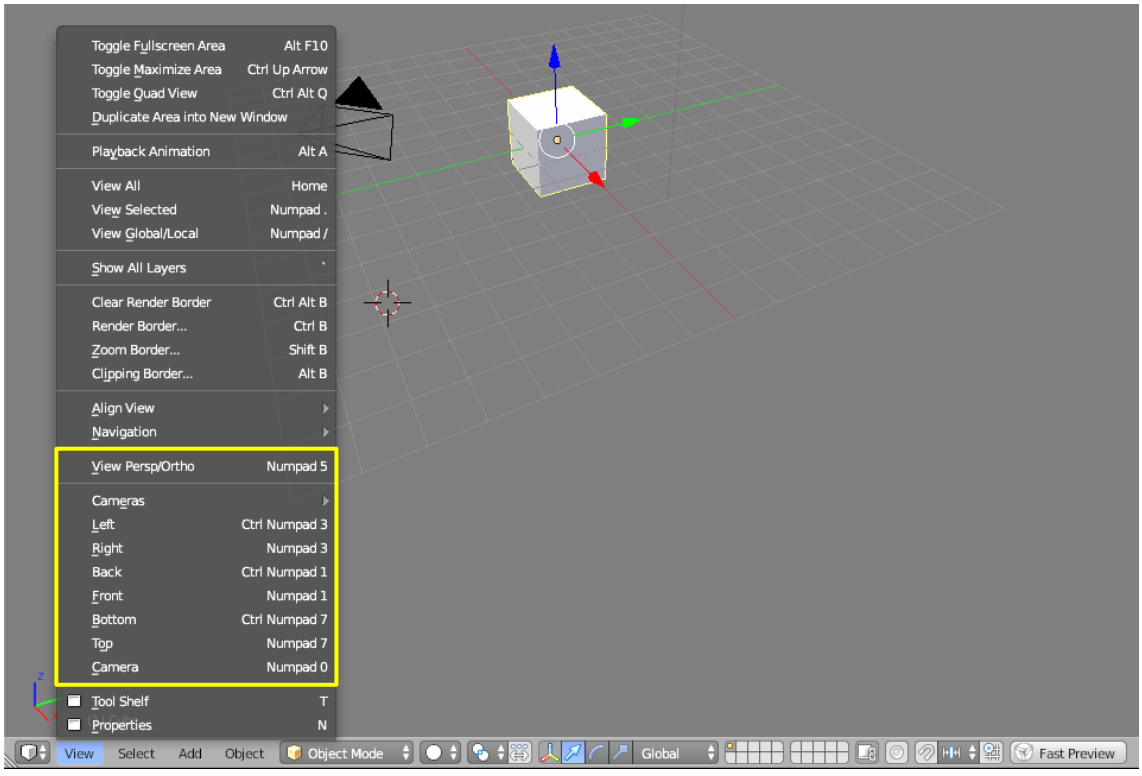


OBS: A configuração das janelas abertas (quantidade, tipo, posicionamentos, dimensões) é salva junto com o arquivo do projeto do Blender. Quando ele for aberto em qualquer computador, será iniciado com a última configuração de janelas salva.

Múltiplas *viewports*, cópias da janela *3D View*, podem apresentar para o usuário diferentes visualizações da mesma cena, assim auxiliando-o na disposição e configuração de elementos, materiais e outros. Para modificar a visualização de cada *viewport*, existem alguns menus acessados pela barra inferior da janela *3D View*.

- ***View*** (Figura 69): possui opções para alternar a posição de visualização da cena entre diferentes vistas, de acordo com a orientação do espaço virtual: *Left* (esquerda), *Right* (direita), *Back* (traseira), *Front* (frontal), *Bottom* (inferior), *Top* (superior), *Camera* (visão da câmera principal). Ainda, é possível alternar a visualização entre Perspectiva e Ortogonal (*View Persp/Ortho*);

Figura 69 – Menu *View* com marcação das opções de alteração da posição de visualização da cena.



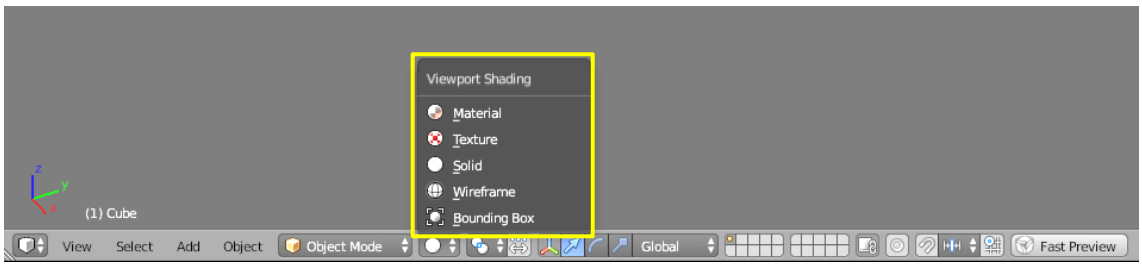
- ***Viewport Shading*** (Figura 70): Possui opções para alternar a forma de exibição dos objetos 3D entre:

  - ***Material***: que mostra uma aproximação do material aplicado aos objetos e da iluminação da cena;
  - ***Texture***: que mostra os objetos com suas texturas aplicadas;

- ***Solid***: que apresenta os objetos apenas com cores sólidas e iluminação simples;
- ***Wireframe***: que exibe somente as linhas da malha dos objetos;
- ***Bounding Box***: que exibe os objetos somente como caixas retangulares que demonstram suas dimensões máximas;
- ***Solid***: que apresenta os objetos apenas com cores sólidas e iluminação simples.

Quando se trabalha com o complemento Blend4Web são apenas essas as opções deste menu, porém sem ele há ainda mais uma opção chamada *Rendered*, que apresenta a visualização exata da cena renderizada, com iluminação e materiais em qualidade final.

Figura 70 – Menu *Viewport Shading* com as diferentes opções de exibição dos elementos da cena.



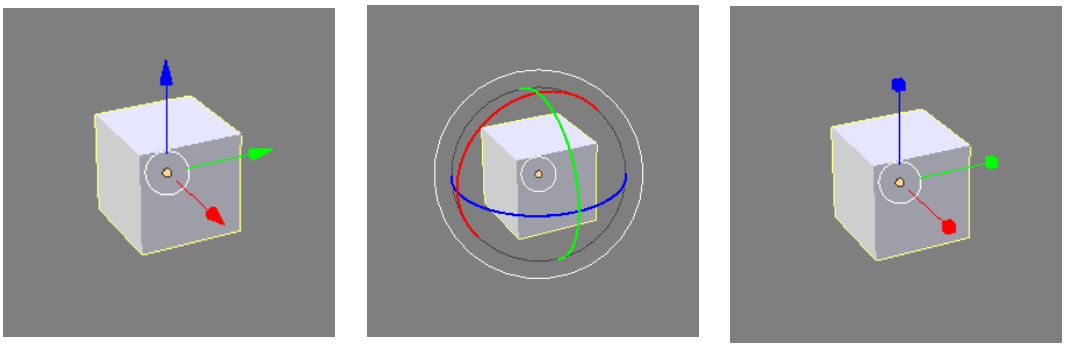
4.3.2. Manipulação de Elementos (Objetos 3D, Luzes, Câmeras...)

Usuários que nunca tiveram contato com o *software* Blender podem ter certa dificuldade na hora de manipular os elementos inseridos no espaço virtual. Por isso, é válida uma explicação das formas e configurações de Translação, Rotação e Escala presentes no programa. Estas servem tanto para objetos 3D, como para luzes, câmeras, textos, autofalantes e outros elementos.

A primeira forma de manipulação de elementos é através do *Gizmo* (Figura 71), uma representação que surge na tela quando algum elemento se encontra selecionado e serve para manipulá-lo segundo cada um de seus eixos, individualmente. Comum em grande parte dos programas que trabalham com modelagem tridimensional, ele tem sua representação alterada conforme a função determinada para ele no momento (Translação, Rotação ou Escala). Sendo constituído por setas quando em modo

Translação, círculos em modo Rotação e eixos com cubos nas pontas em modo de Escala. Cada cor do *Gizmo* representa um dos 3 eixos (X, Y e Z) do espaço tridimensional.

Figura 71 – Os três tipos de *Gizmo* segundo sua função. Da esquerda para a direita: Translação, Rotação e Escala.



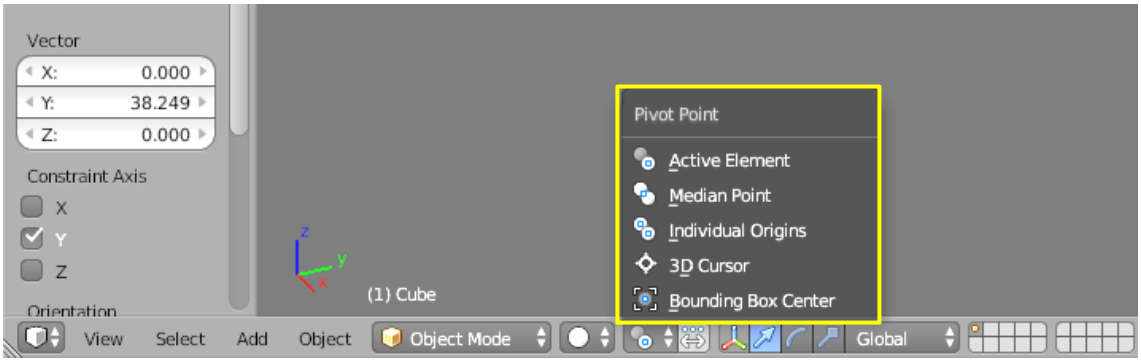
Para alternar entre os 3 tipos de *Gizmo* utiliza-se os botões denominados *Translate*, *Rotate* e *Scale*, presentes na barra inferior da janela *3D View* (Figura 72). No lado esquerdo destes botões, o botão com um ícone representando os eixos coloridos faz com que seja ocultado e exibido o *Gizmo* na tela.

Figura 72 – Localização do botão que mostra ou oculta a visualização do *Gizmo* e os que alternam entre os três tipos de *Gizmo*.

Para alterar a Translação, Rotação ou Escala do elemento selecionado, basta clicar com o botão esquerdo do mouse em alguns dos eixos coloridos do *Gizmo* e, mantendo-o pressionado, arrastar o mouse.

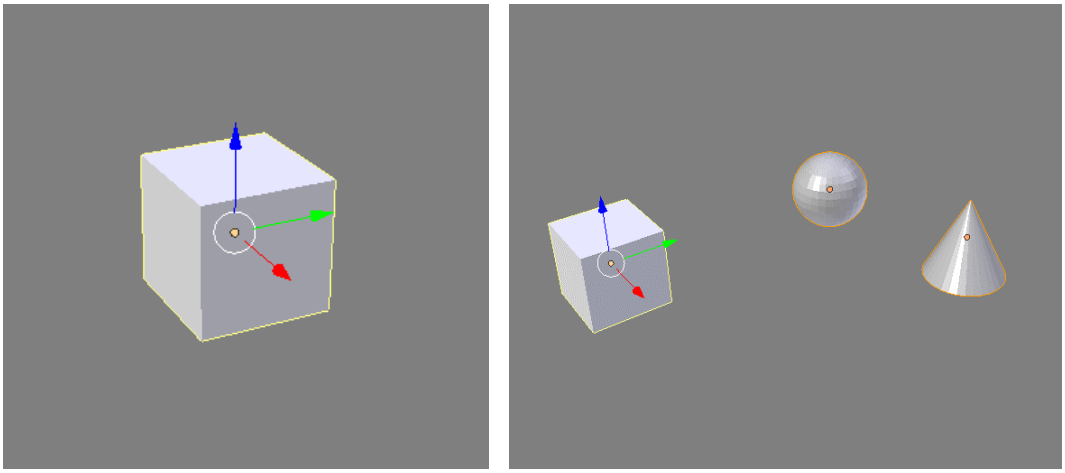
Também é possível modificar o posicionamento e a orientação do próprio *Gizmo*. Para isso, existem os menus chamados *Pivot Point* (Figura 73) e *Transform Orientation* (Figura 78), que configuram, respectivamente, as duas propriedades. Ambos os menus se encontram na barra inferior da janela *3D View* e suas opções são as seguintes:

Figura 73 – Menu de configuração do posicionamento do *Gizmo* no espaço virtual.



- ***Active Element*** (Figura 74): posiciona o *Gizmo* no centro do elemento ativo (selecionado). No caso da seleção individual de mais de um elemento em conjunto, o Gizmo se posiciona no centro do último elemento adicionado à seleção. Já no caso de uma seleção de múltiplos elementos de uma vez só, o *Gizmo* fica ao centro do último elemento manipulado antes da seleção;

Figura 74 – Representações de *Gizmos* posicionados com a opção *Active Element* ativa. Na esquerda, com somente um objeto selecionado e, na direita, três objetos selecionados em conjunto.

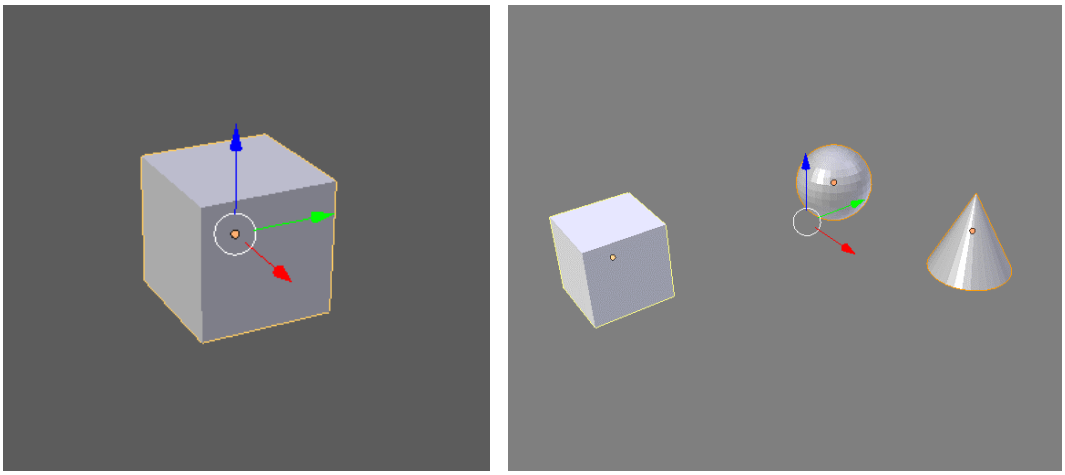


- ***Median Point*** (Figura 75): quando apenas um elemento é selecionado, posiciona o *Gizmo* no centro deste. Quando mais de um elemento é selecionado, faz com que o *Gizmo* fique no ponto médio de equilíbrio entre eles;

- ***Individual Origins*** (Figura 75): posiciona o *Gizmo* da mesma forma que o *Median Point*, porém todas as manipulações realizadas através do *Gizmo*

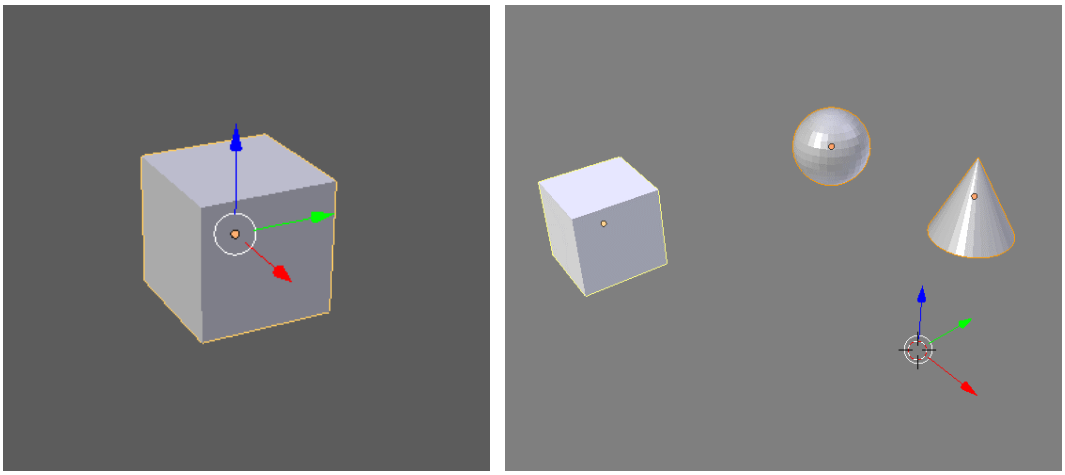
afetam os elementos como se existisse um *Gizmo* individual no centro de cada um deles;

Figura 75 – Representações de *Gizmos* posicionados com a opção *Median Point* ou *Individual Origins* ativa. Na esquerda, com somente um objeto selecionado e, na direita, três objetos selecionados em conjunto.



- ***3D Cursor*** (Figura 76): permite que o *Gizmo* seja posicionado em qualquer local que o usuário desejar. Com um ou mais elementos selecionados, basta clicar com o botão esquerdo do mouse na posição desejada para o *Gizmo*, dentro do espaço virtual;

Figura 76 – Representações de *Gizmos* posicionados com a opção *3D Cursor* ativa. Na esquerda, com somente um objeto selecionado e, na direita, três objetos selecionados em conjunto.



- ***Bounding Box Center*** (Figura 77): funciona de forma semelhante ao *Median Point*, porém quando múltiplos elementos são selecionados, o *Gizmo* se posiciona no centro de um paralelepípedo imaginário que os engloba.

Figura 77 – Representações de *Gizmos* posicionados com a opção *Bounding Box Center* ativa. Na esquerda, com somente um objeto selecionado e, na direita, três objetos selecionados em conjunto.

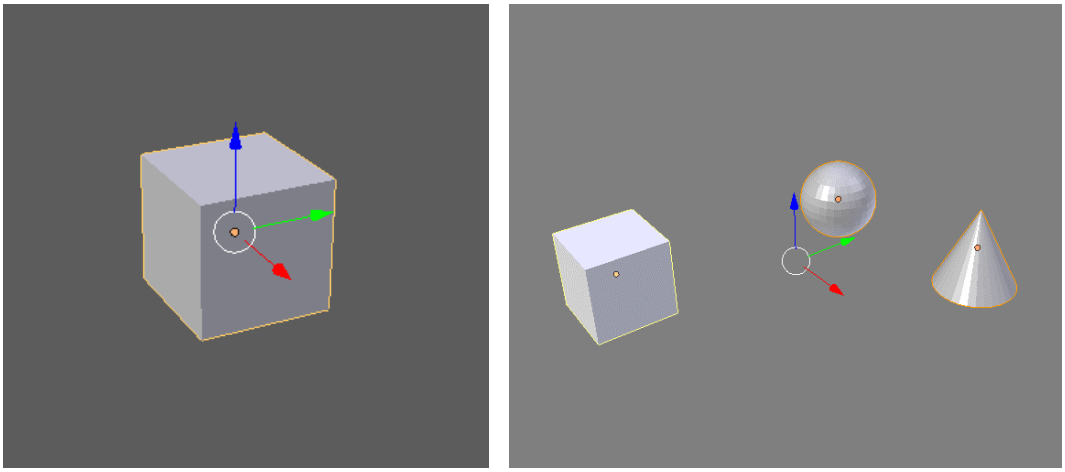
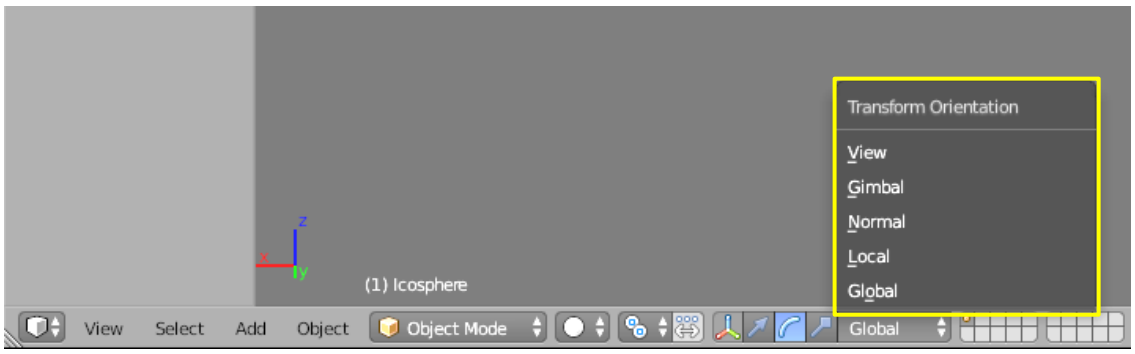


Figura 78 – Menu de configuração da orientação do *Gizmo*.



- ***View***: determina que os eixos do *Gizmo* mantenham uma orientação predeterminada independentemente da posição de visualização do elemento selecionado. Essa orientação segue as seguintes regras:

  - **Eixo Y**: Para cima / Para baixo;
  - **Eixo X**: Esquerda / Direita;
  - **Eixo Z**: Para dentro da tela / Para fora da tela.

- ***Gimbal***: faz com que o *Gizmo* se oriente e atue conforme os princípios de um *Gimbal*, um suporte articulado que permite a rotação de um objeto em torno de um único eixo (Figura 79);

Figura 79 – Representação de um suporte *Gimbal*.

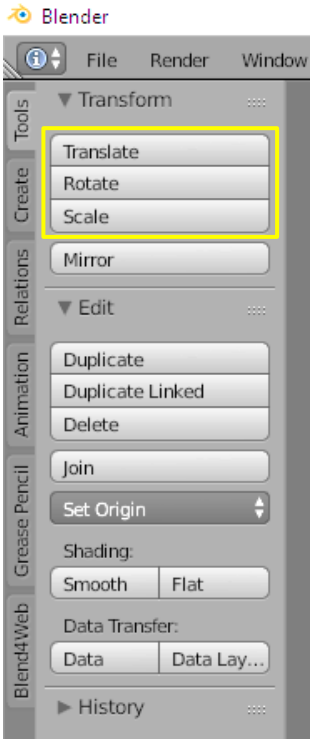
- ***Normal***: orienta o *Gizmo* de forma que o seu eixo Z corresponda com o Vetor Normal do elemento selecionado;

- ***Local***: faz com que os eixos de transformação do *Gizmo* tenham a mesma orientação dos eixos do elemento selecionado, seja ele um objeto 3D, câmera, luz ou outro;

- ***Global***: faz com que os eixos de transformação do *Gizmo* estejam com a mesma orientação das coordenadas globais, ou seja, a orientação dos eixos do espaço virtual.

Além da manipulação da Translação, Rotação e Escala dos elementos por meio do *Gizmo*, existem outras duas diferentes formas de manipulá-los. Uma delas é através dos botões *Translate*, *Rotate* e *Scale*, dispostos no menu *Tools* da janela *3D View* (Figura 80), quando um ou mais elementos estão selecionados. Estes alteram, respectivamente, a Translação, Rotação e Escala dos elementos de forma livre com o movimento do mouse, sem a fixação a algum eixo. Também podem ser acionados pelos atalhos: tecla G (*Translate*), tecla R (*Rotate*) e tecla S (*Scale*).

Figura 80 – Localização dos botões *Translate*, *Rotate* e *Scale* no menu lateral da janela *3D View*.



Para obter maior controle dessa movimentação mais livre, também é possível realizá-la de outra maneira. Com o elemento que se pretende mover selecionado, clica-se com o botão esquerdo ou botão direito do mouse e o mantém pressionado, no círculo localizado ao centro do seu *Gizmo* (Figura 81) e movimenta-se o mouse fazendo com que a posição, Rotação ou Escala do objeto (dependendo do tipo de *Gizmo* ativo) seja alterada sem a fixação em um só eixo, porém com maior precisão. Durante a movimentação o *Gizmo* é ocultado e, no caso dos modos de Rotação e de Escala, surgem ícones na ponta do ponteiro do mouse, indicando que o elemento em questão está sofrendo alterações nestas propriedades (Figura 82).

Figura 81 – Indicação da localização do círculo central do *Gizmo* de rotação.

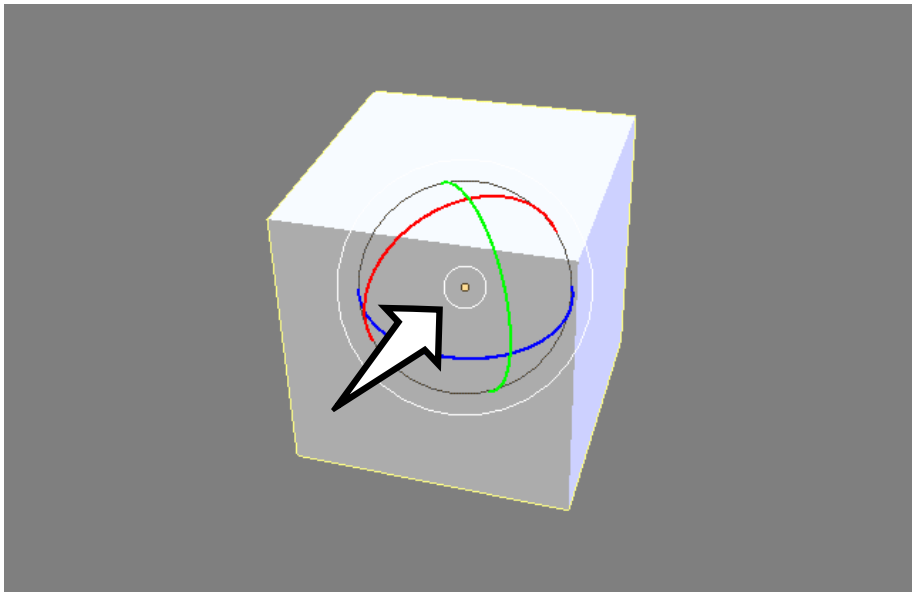
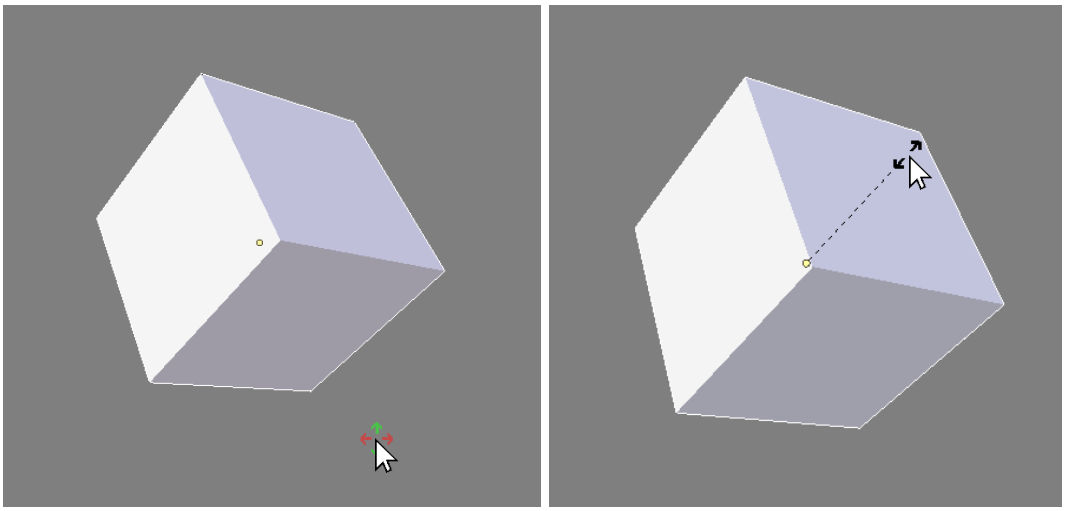
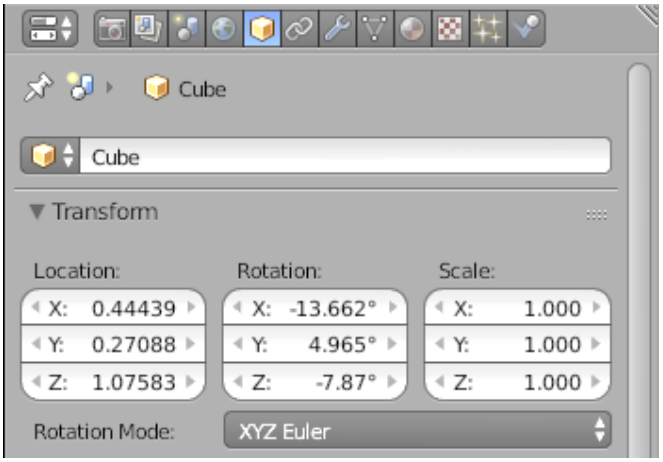


Figura 82 – Representações dos ícones que surgem quando se transforma a Rotação (esquerda) e a Escala (direita) de elementos pelo clique no círculo central do *Gizmo* e movimentação do mouse.



A última forma de manipulação de elementos é através do menu *Object* > *Transform*, dentro da janela *Properties* (Figura 83). Neste local, encontram-se as informações exatas das coordenadas de posicionamento, ângulos de rotação e fator de escala do elemento selecionado, para cada um dos 3 eixos (X, Y e Z). Estes valores podem ser alterados conforme deseja o usuário.

Figura 83 – Localização dos campos com os valores (editáveis) referentes à posição, Rotação e Escala do objeto selecionado.



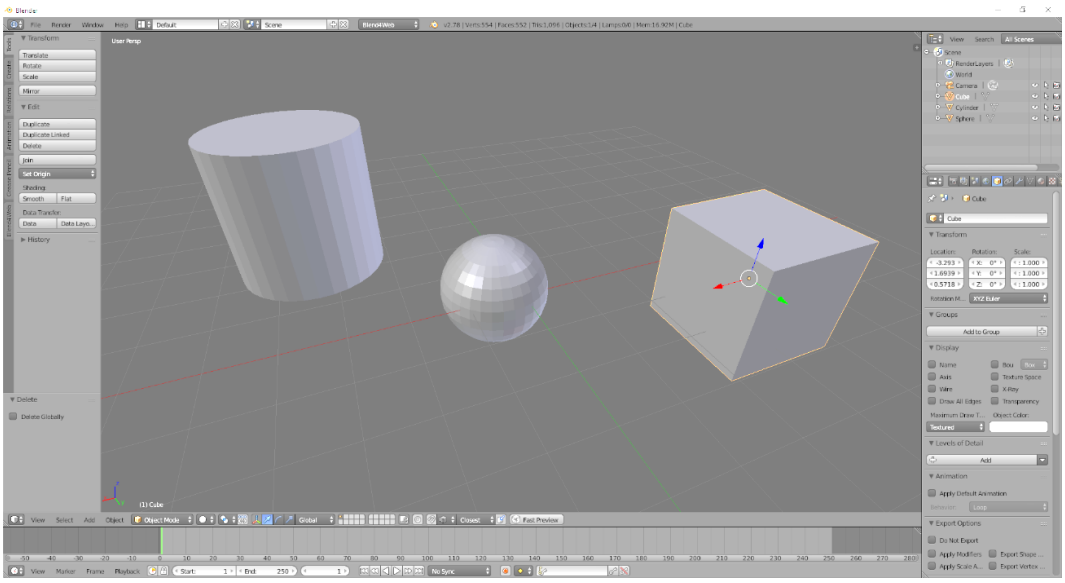
4.3.3. Comandos Importantes

Usuários não familiarizados com o Blender podem ter dificuldades em ações simples e rotineiras no início da experiência com o programa. Por isso, alguns comandos importantes são esclarecidos a seguir. Lembrando que o termo *Elemento*, utilizado diversas vezes, compreende objetos 3D, luzes, câmeras, autofalantes, textos e qualquer outra coisa inserida no espaço virtual do programa.

Selecionar Elementos

Sabe-se que um objeto está selecionado no Blender quando ele possui um contorno laranja (Figura 84) ou verde quando fizer parte de um grupo. Existem diferentes formas de selecionar um objeto isoladamente ou um conjunto de objetos:

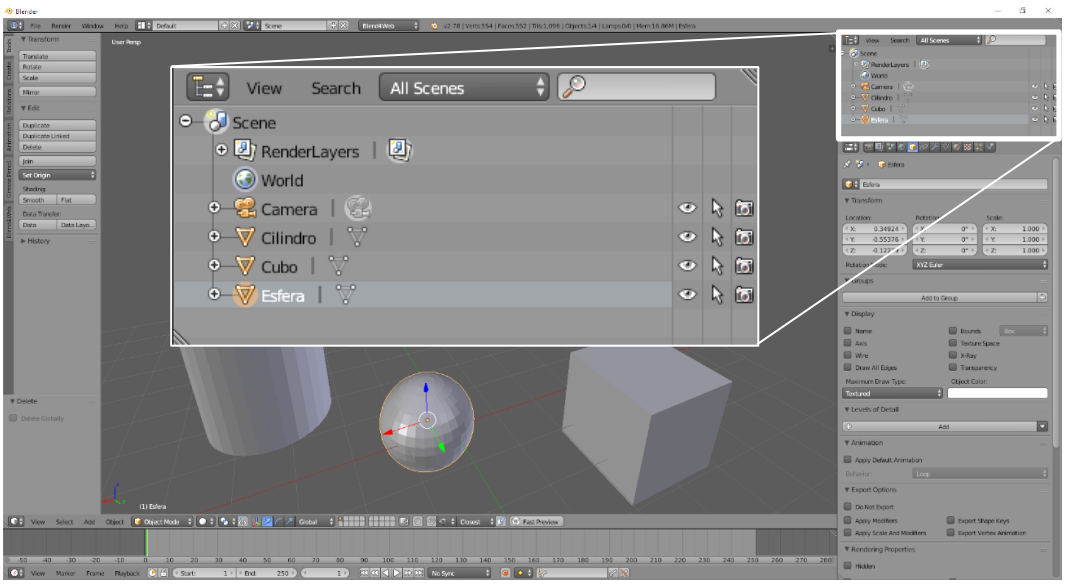
Figura 84 – Dentre os três objetos da cena (cilindro, esfera e cubo) apenas o cubo está selecionado.



- Pode-se simplesmente clicar com o botão direito do mouse no elemento que se deseja selecionar. Para selecionar mais de um elemento, é necessário segurar pressionada a tecla SHIFT e clicar com o mesmo botão do mouse neles;

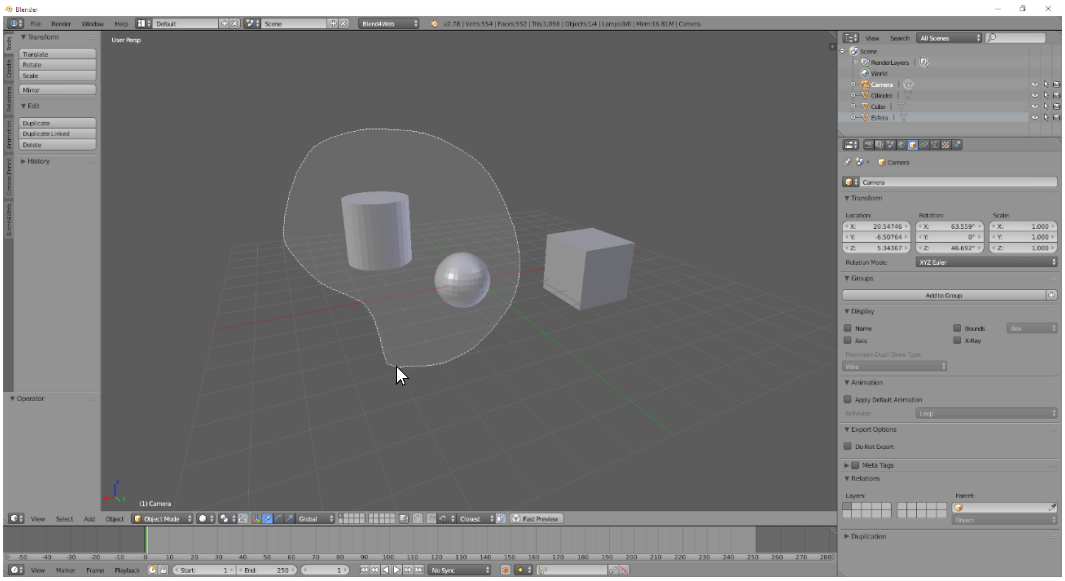
- É possível selecionar elementos clicando-se com o botão esquerdo do mouse em seu nome na lista presente na janela *Outliner* (Figura 85), que reuni todos os elementos presentes no espaço virtual do programa. Para selecionar múltiplos elementos, também deve-se manter pressionada a tecla SHIFT enquanto clica-se no nome deles com o mouse;

Figura 85 – Janela Outliner (ampliada) com o elemento *Esfera* selecionado.



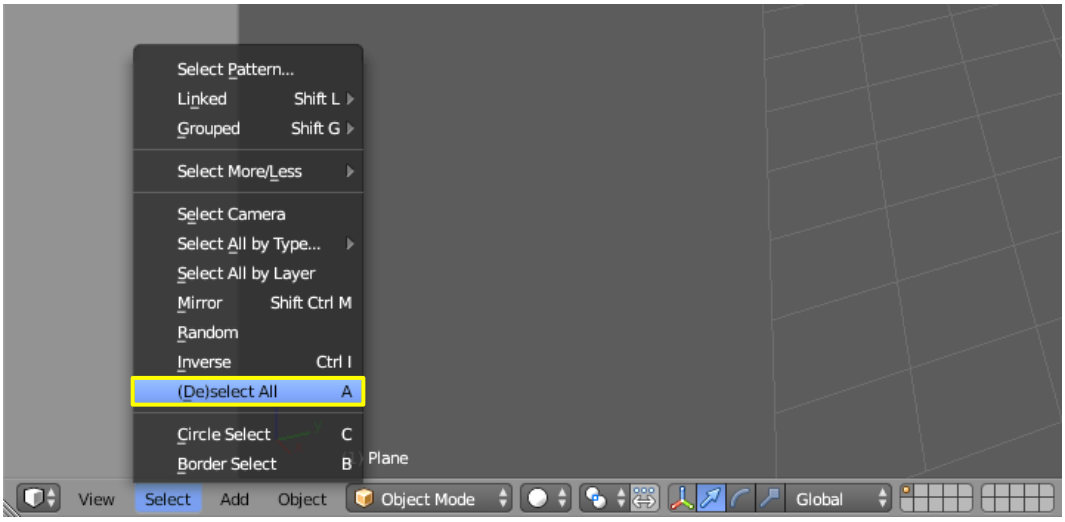
- A última maneira de selecionar elementos é segurar a tecla CTRL pressionada e, segurando pressionado também o botão esquerdo do mouse, arrastá-lo para desenhar livremente uma forma que envolva a maior parte ou por inteiro, os elementos que se pretende selecionar. Para auxiliar na criação dessa forma livre o programa apresenta na tela uma linha temporária com o desenho que está sendo feito com o mouse (Figura 86).

Figura 86 – Exemplo de desenho de forma livre para selecionar, no caso, o cilindro e a esfera.



Também é possível retirar a seleção de todos os elementos da cena se desejado, bastando pressionar a tecla A do teclado. Ainda, quando não houver nenhum objeto selecionado, pressionar a tecla A faz com que todos os elementos presentes no espaço virtual sejam selecionados ao mesmo tempo. Essa mesma ferramenta de desmarque ou seleção total, chamada *(De)select All*, é encontrada no menu *Select*, presente na barra inferior da janela *3D View* (Figura 87).

Figura 87 – Localização da ferramenta *(De)select All* na janela *3D View* > menu *Select*.



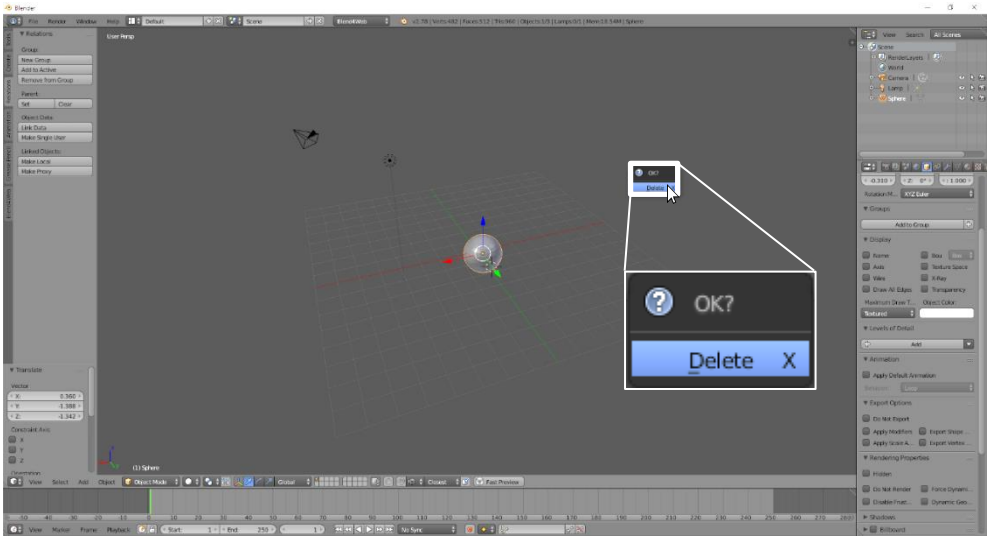
Excluir Elementos

Existe mais de uma maneira de excluir elementos do espaço virtual do programa, porém, independentemente da forma de exclusão, deve-se primeiro selecionar o(s) objeto(s) que se deseja excluir.

Depois pode-se utilizar umas das três formas de exclusão:

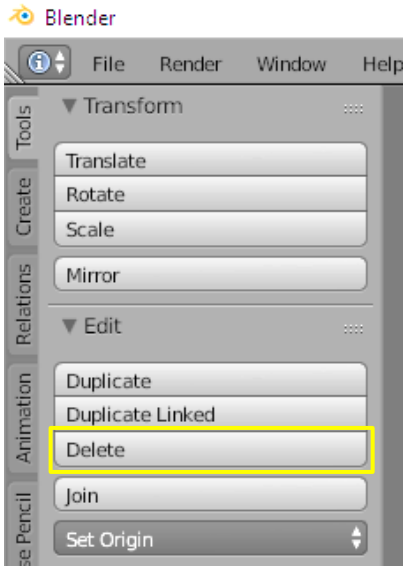
- Com o ponteiro do mouse dentro da janela *3D View*, pressionar no teclado a tecla DEL ou DELETE ou X mais a tecla ENTER, para confirmar a exclusão. Pode-se ao invés de pressionar a tecla ENTER, clicar com o botão esquerdo do mouse na opção *Delete* do menu que se abre ao pressionar a primeira tecla do comando (Figura 88);

Figura 88 – Menu (ampliado) para confirmação de exclusão do elemento selecionado.



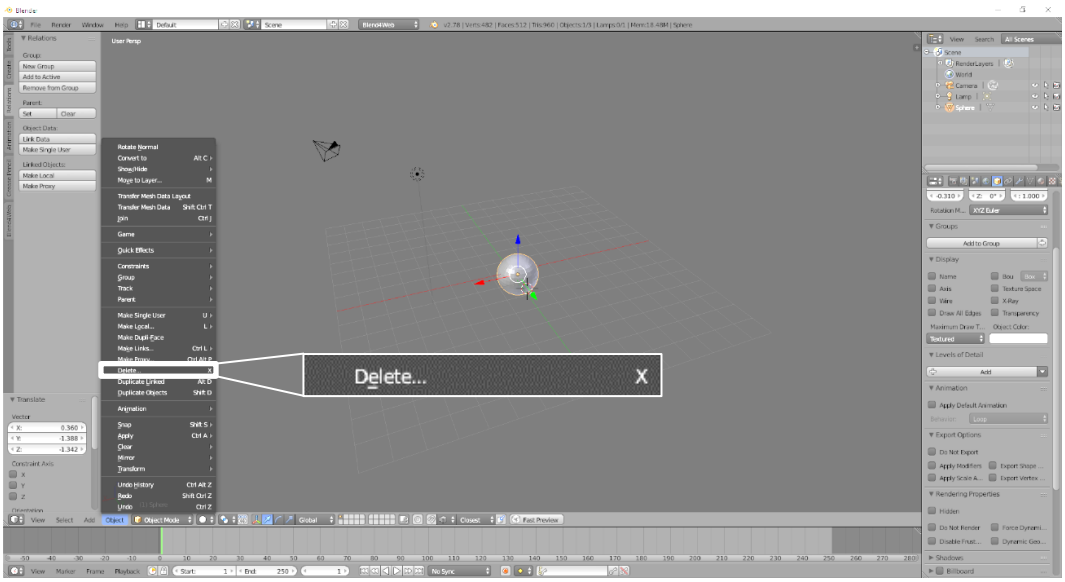
- Pode-se também excluir elementos clicando no botão *Delete* na janela *3D View* > menu *Tools* > *Edit* (Figura 89) e, depois, pressionando a tecla ENTER ou clicando com o botão esquerdo do mouse na opção *Delete*, no menu que se abre na tela, para confirmar a exclusão;

Figura 89 – Localização da opção *Delete* no menu lateral *Tools* > *Edit*, dentro da janela *3D View*.



- A terceira forma de excluir elementos da cena é escolher a opção *Delete* (Figura 90) no menu *Object*, localizado na barra inferior da janela *3D View*. Depois, confirmar a exclusão pressionando a tecla ENTER ou clicando com o botão esquerdo do mouse na opção *Delete*, presente no menu suspenso que se abre.

Figura 90 – Opção *Delete* (ampliado) no menu *Object* da Janela *3D View*.



Duplicar Elementos

Existem basicamente dois tipos de duplicação de elementos no Blender, o *Duplicate* e o *Duplicate Linked*. Ambos criam uma cópia do(s) elemento(s) selecionados, porém com algumas diferenças.

- ***Duplicate***: cria uma cópia visualmente idêntica e independente do(s) elemento(s) selecionado(s), mas todos os seus atributos (dimensão, material e outros) podem ser modificados sem que o elemento original sofra qualquer alteração. É acessado de diversas maneiras:

  - Pressionando as teclas SHIFT + D do teclado;
  - Clicando na opção *Duplicate*, no menu *Tools > Edit* da janela *3D View*;
  - Clicando na opção *Duplicate Objects*, no menu *Object* da barra inferior da janela 3D View.

- ***Duplicate Linked***: cria uma cópia com quase todos os seus atributos vinculados ao elemento original. Se um dos elementos (original ou cópia) for modificado, todos os que estiverem vinculados sofreram as mesmas alterações. Destaca-se, no entanto, que as propriedades de transformação permanecem independentes em cada um. Assim, é possível mover, rotacionar e escalar qualquer elemento sem afetar as outras cópias. É acessado de diversas maneiras:

  - Pressionando as teclas ALT + D do teclado;
  - Clicando na opção *Duplicate Linked*, no menu *Tools > Edit* da janela *3D View*;
  - Clicando na opção *Duplicate Linked*, no menu *Object* da barra inferior da janela 3D View.

  Esse segundo método de duplicação é utilizado também como forma de otimização de uma cena e seu uso é melhor explicado no item 4.4.1 (Otimizações).

Criar e Selecionar Grupos

É possível agrupar elementos que tenham algo de semelhante para facilitar, por exemplo, a seleção deles em conjunto. Para isso, primeiro seleciona-se os elementos

que farão parte do grupo e, então, utiliza-se os botões localizados no menu *Relations > Group* na janela *3D View* (Figura 91), no menu *Object > Group* na barra inferior da janela *3D View* (Figura 92) ou seus respectivos atalhos, de acordo com o desejado.

Figura 91 – Localização das opções de grupos no menu *Relations > Group*, na janela *3D View*.

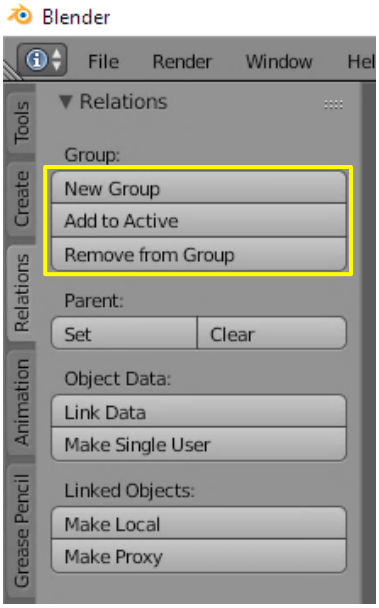
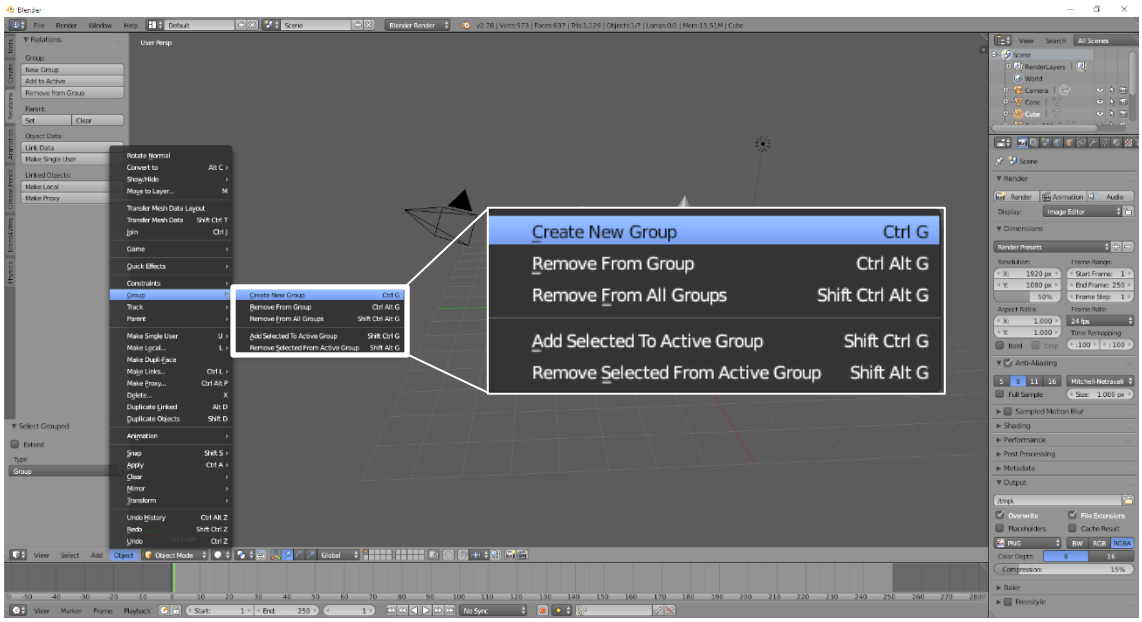


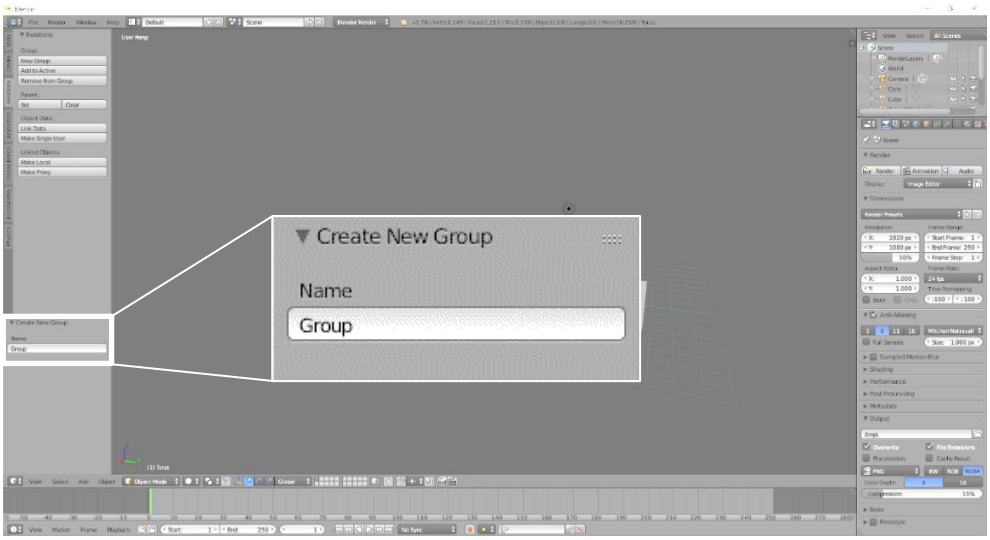
Figura 92 – Localização das opções de grupos no menu *Object > Group* (ampliado), na barra inferior da janela *3D View*.



- ***New Group*** (Figura 91) / ***Create New Group*** (Figura 92) (Atalho: CTRL + G): cria um grupo com os elementos selecionados. Por padrão ele terá o

nome *Group*, que pode ser alterado no momento de sua criação no menu *Create New Group > Name* (Figura 93), o qual se abre após o clique no botão *New Group* ou utilização do atalho;

Figura 93 – Menu para escolha do nome do grupo recém-criado.



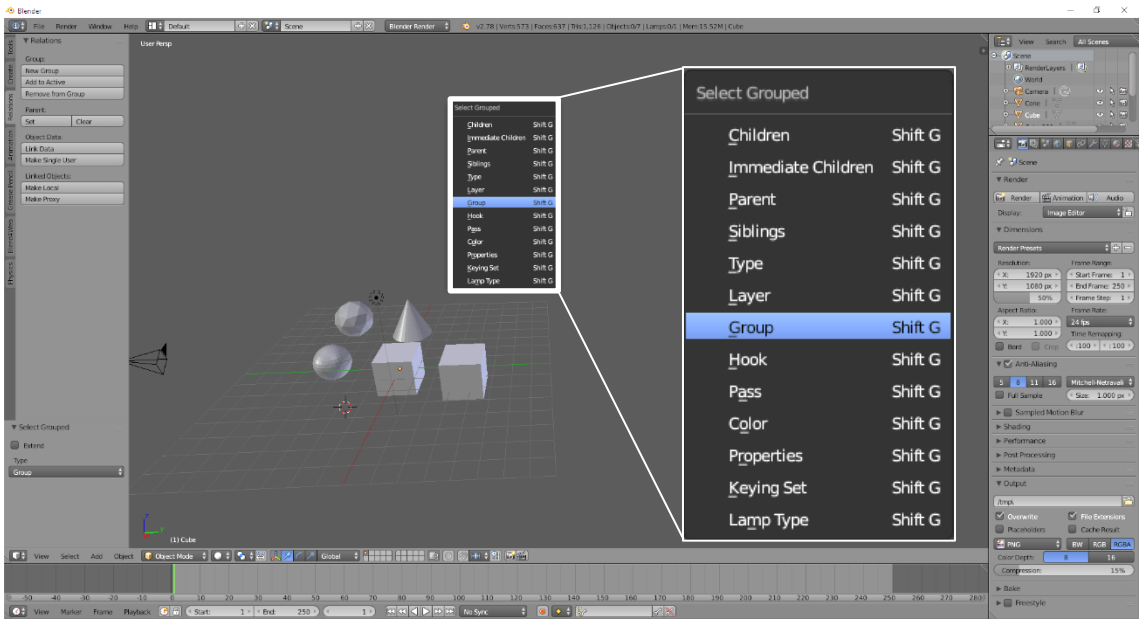
- ***Add to Active*** (Figura 91) (Atalho: SHIFT + CTRL + G): adiciona um elemento selecionado a um grupo existente, quando pelo menos um outro elemento, já pertencente ao grupo, estiver selecionado em conjunto com ele;

- ***Remove From Group*** (Figura 91 e Figura 92) (Atalho: CTRL + ALT + G): remove um ou mais elementos selecionados de um grupo ao qual ele pertence. Após o clique nessa opção, é necessário confirmar, em um menu suspenso na tela, o grupo do qual se deseja remover o(s) elemento(s);

- ***Remove From All Group*** (Figura 92) (Atalho: CTRL + ALT + G): remove um ou mais elementos selecionados de todos os grupos aos quais ele(s) pertence(m).

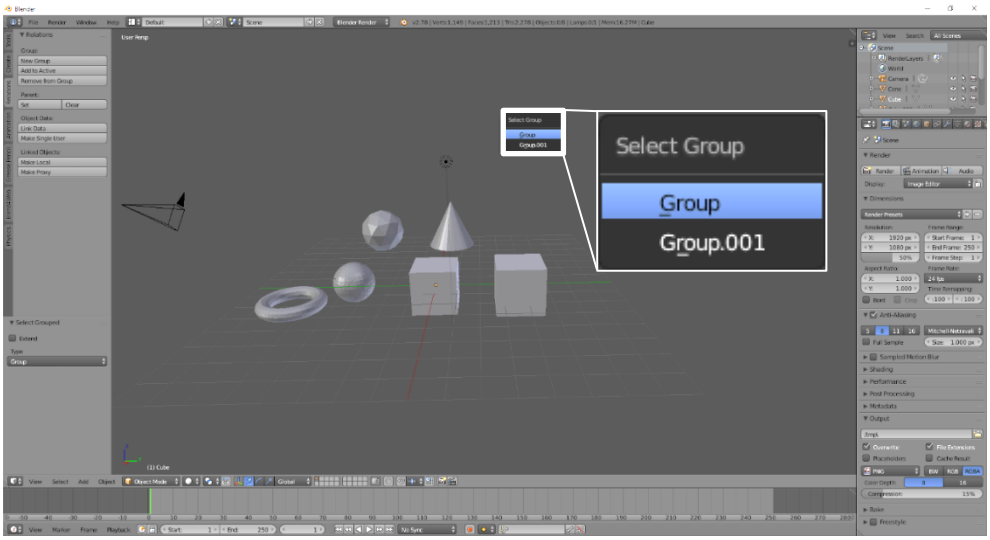
Para selecionar em conjunto todos os elementos de um grupo existente, basta selecionar um deles e utilizar o atalho SHIFT + G para abrir um menu com a opção *Group* (Figura 94), que quando pressionada com o botão esquerdo do mouse, seleciona todos os demais elementos pertencentes ao grupo.

Figura 94 – Menu suspenso que se abre ao utilizar o atalho SHIFT + G.



No caso de o primeiro objeto selecionado pertencer a mais de um grupo, quando escolhida a opção *Group* no menu, outro menu suspenso surgirá pedindo para o usuário escolher qual o grupo que ele deseja selecionar (Figura 95).

Figura 95 – Menu suspenso secundário para escolha do grupo que se pretende selecionar, no caso do elemento selecionado estar inserido em mais de um grupo.



Combinar Objetos

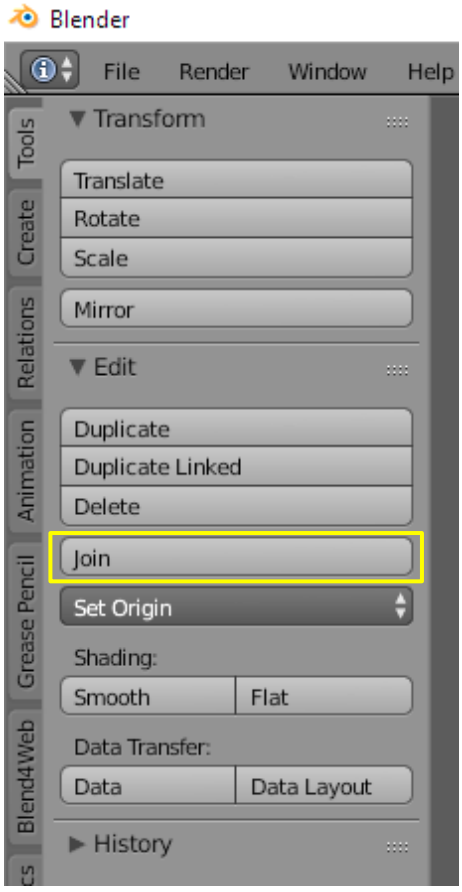
Dependendo da necessidade do usuário, é útil combinar múltiplos objetos e transformá-los em um só ao invés de criar um grupo com eles. No Blender, é possível

unir diferentes objetos que, mesmo sem a necessidade de uma ligação física entre eles, passam a se comportar como se fossem um objeto único.

OBS: Se os objetos, quando separados, possuíam diferentes materiais atribuídos a eles, após a união, mantêm seus materiais separados para edição futura pelo menu *Material*, na janela *Properties*.
O objeto unido fica com o nome do último objeto a ser selecionado antes do uso da ferramenta *Join*, podendo ser renomeado clicando com o botão direito do mouse em seu nome na lista presente na janela *Outliner* e, depois, em *Rename*.

O procedimento de combinação é simples e inicia-se com a seleção de todos os objetos que serão unidos. Depois, utiliza-se o botão *Join* (janela *3D View* > menu *Tools* > *Edit* - Figura 96) ou o atalho CTRL + J.

Figura 96 – Localização do botão *Join* na janela *3D View* > menu *Tools* > *Edit*.



Já, para separar os objetos que foram combinados, é possível utilizar o atalho CTRL + Z quantas vezes forem necessárias para que a cena retorne ao estado em que os objetos estavam ainda separados. Há vezes, porém, em que não é possível mais utilizar o atalho CTRL + Z, pois afeta todas as alterações realizadas no programa. Nesses casos, existe uma técnica com a qual é possível separar novamente tais objetos.

Inicialmente, seleciona-se o objeto que foi unido/combinado. Depois, entra-se no modo de edição (*Edit Mode*) pelo atalho TAB ou pelo menu de seleção de modos da janela *3D View* (Figura 103). Então, seleciona-se ao menos um vértice do objeto que se deseja separar dos demais (Figura 97). Finaliza-se, selecionando a opção *by loose parts*, localizada na janela *3D View* > menu *Mesh* > *Vertices* > *Separate*, quando dentro do modo de edição (Figura 98). Pode-se, então, retornar ao modo de objeto (*Object Mode*) pelo atalho TAB ou menu de seleção de modos.

Figura 97 – Vértice do cubo selecionado.

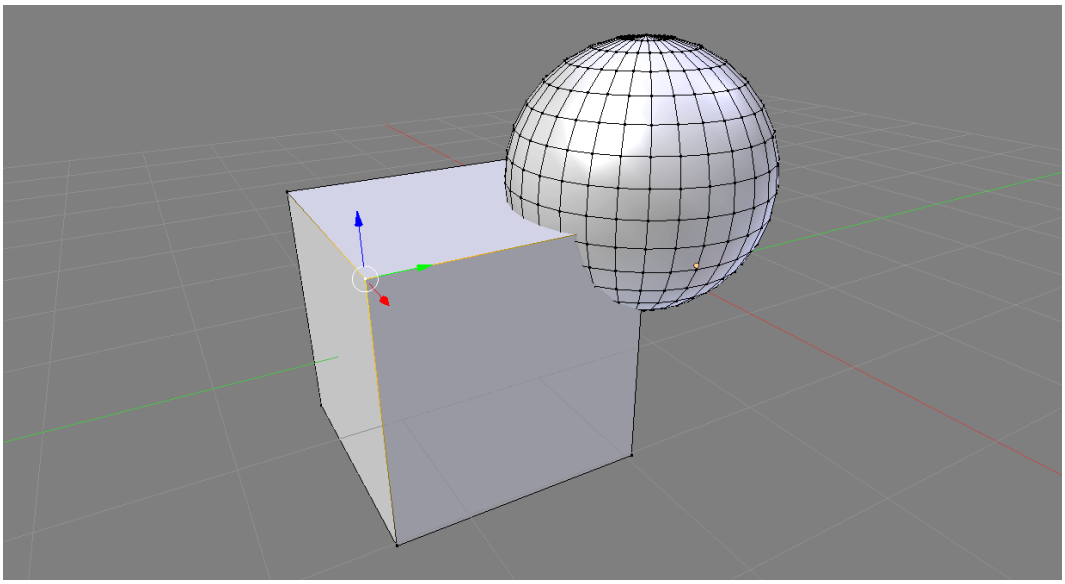
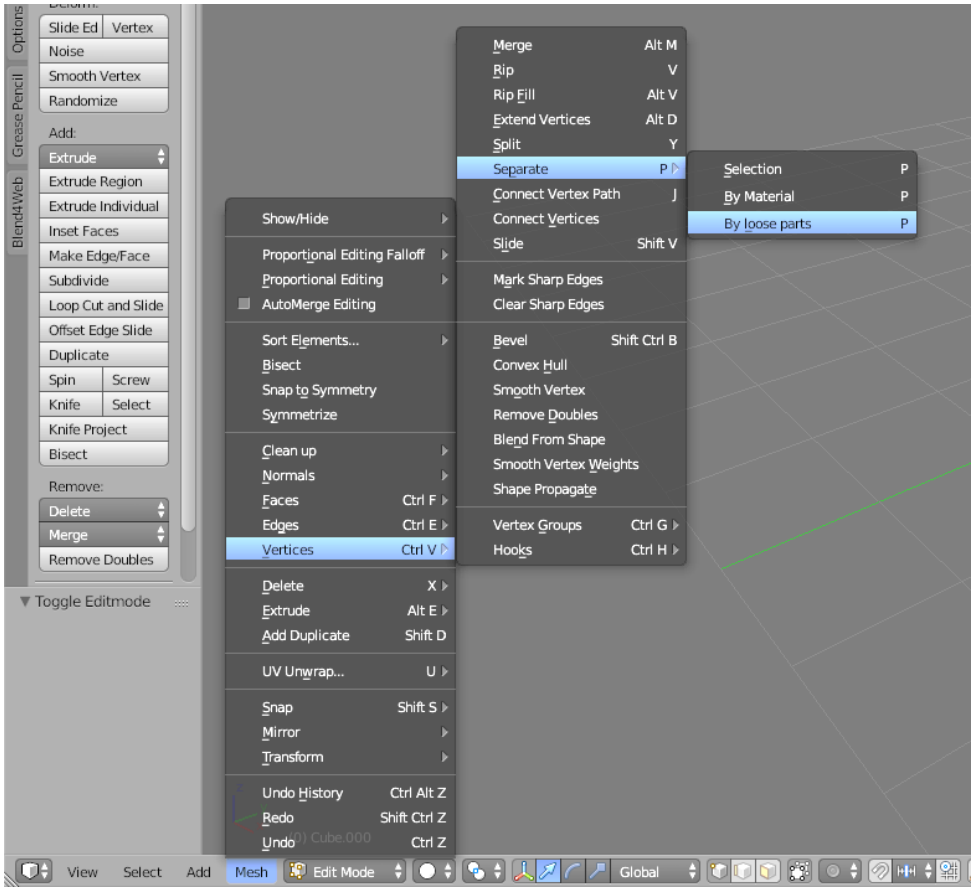


Figura 98 – Localização da opção *by loose parts* no menu *Mesh* > *Vertices* > *Separate*, dentro da janela *3D View* em modo de edição.



Movimentar a Cena (janela *3D View*) com o Mouse

A principal forma de movimentar a visualização da cena exibida na janela *3D View* é através dos botões do mouse e movimentos realizados com ele dentro dessa janela. Para isso, recomenda-se a utilização de um mouse com pelo menos três botões (esquerdo, central com rolagem e direito). Seguem as movimentações possíveis de serem realizadas com o mouse e algumas teclas auxiliares do teclado.

- **Rolagem/Zoom**: utilizando a rolagem do botão central do mouse pode-se aproximar (para frente) ou afastar (para trás) a visualização da cena. Também é possível realizar esse movimento de *zoom,* segurando pressionada a tecla CTRL mais o botão central do mouse e, em seguida, arrastando o mouse para frente (aproximar) ou para trás (afastar);

- **Órbita**: segurando pressionado o botão central do mouse e o arrastando, é possível realizar um movimento de órbita, em que o visualizador movimenta a cena como se estivesse rotacionando uma esfera ou orbitando em sua volta;

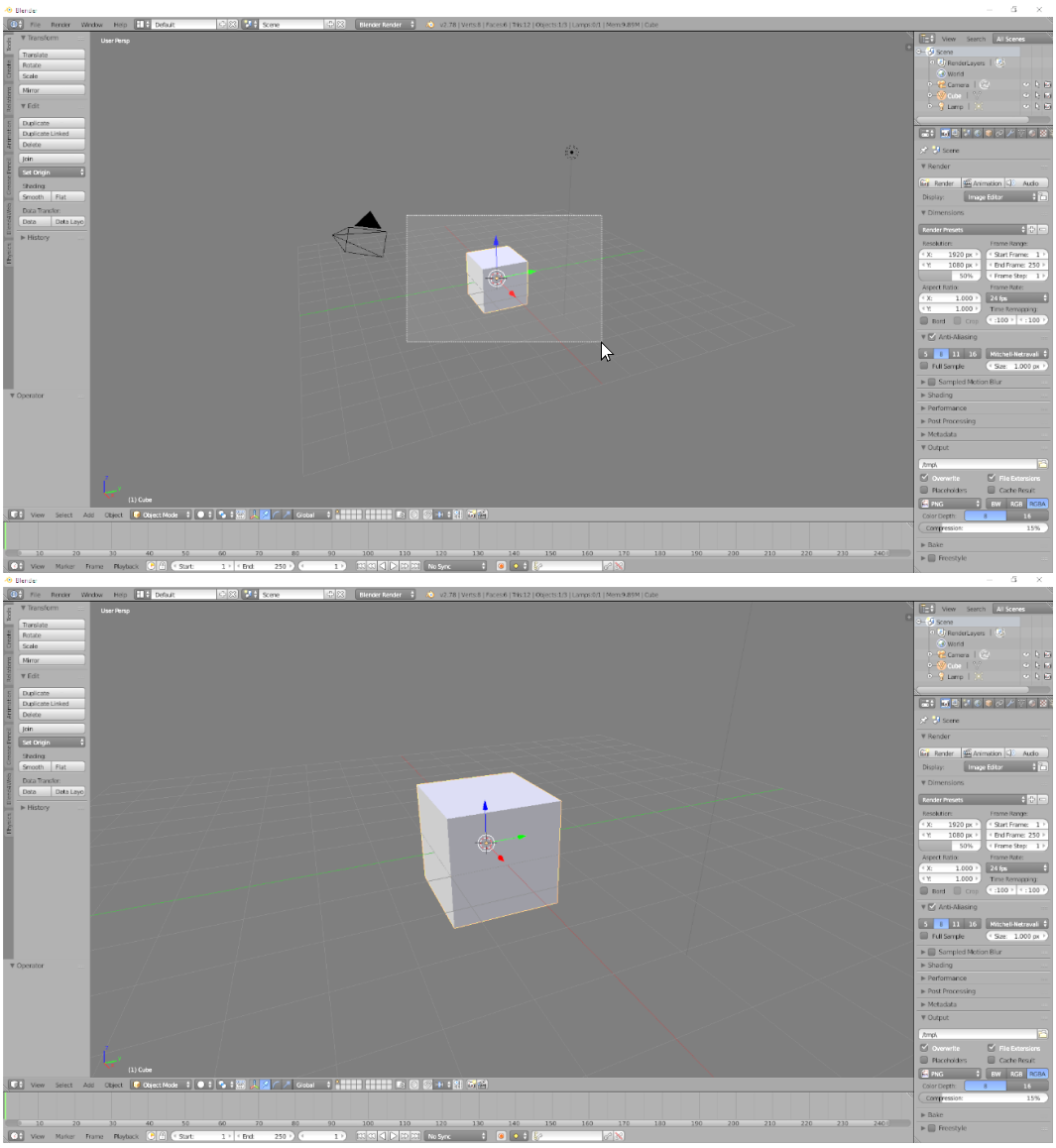
- **Movendo o panorama**: para simplesmente mover a visualização para cima, para baixo, para a esquerda e para direita sem orbitar em sua volta, segura-se pressionada a tecla SHIFT mais o botão central do mouse e, depois, arrasta-se o mouse para qualquer lado;

- **Movendo o panorama em um só eixo**: É possível mover a visualização em um só eixo. Para movê-la somente para cima e para baixo segura-se pressionada a tecla SHIFT mais a rolagem do mouse para frente (cima) ou para trás (baixo). Já, para mover para a esquerda e para a direita, a tecla a ser mantida pressionada é a CTRL mais a rolagem do mouse para frente (direita) e para trás (esquerda);

- **Rotação em um só plano**: mantendo pressionadas as teclas CTRL e SHIFT mais a rolagem do mouse, rotaciona-se a visualização para a esquerda (rolagem para frente) ou direita (rolagem para trás) em um plano paralelo ao plano da tela;

- **Definir quadro de ampliação da imagem**: pode-se também aplicar um zoom instantâneo em uma determina área da cena. Com o ponteiro do mouse dentro da janela *3D View*, pressiona-se o conjunto de teclas SHIFT + B e, depois, mantendo pressionado o botão esquerdo do mouse, desenha-se na tela um retângulo sobre da área desejada, que será ampliada e centralizada na janela (Figura 99).

Figura 99 – Em cima, retângulo demarcando área para ampliação e embaixo área ampliada.



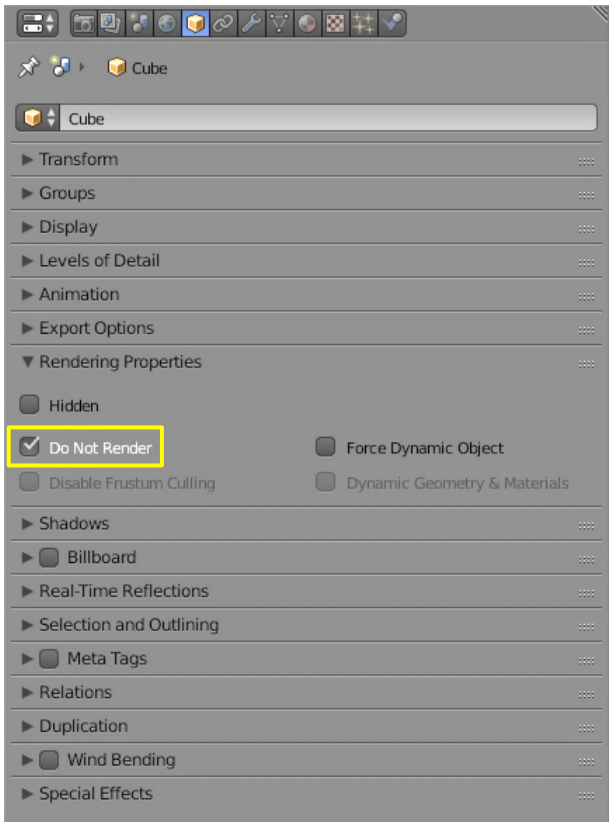
Ocultar a Renderização de Elementos

Objetos 3D presentes no espaço virtual do Blender podem estar visíveis ou ocultos na cena final exportada com o complemento Blend4Web. Um objeto definido como oculto, não será renderizado somente durante a execução da cena no navegador. Isso significa que ele não estará visível, mas todas as influências dele no ambiente, como reflexos, sombras, efeitos de física e outros, se manterão e serão visualizados normalmente.

Por padrão, todos os objetos inseridos serão visíveis após a exportação. Para desativar a visibilidade de um objeto na cena exportada, com o motor de renderização

Blend4Web ativado (Figura 44), deve-se selecioná-lo e ativar a opção *Do Not Render* no menu *Object > Rendering Properties*, dentro da janela *Properties* (Figura 100).

Figura 100 – Localização da opção *Do Not Render* na janela *Properties* > menu *Object > Rendering Properties*.



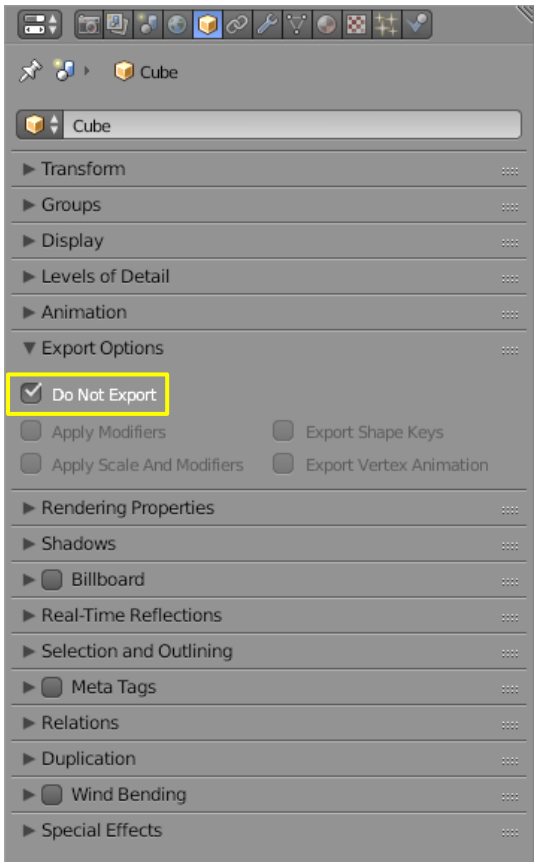
OBS: Para desativar a renderização de mais de um objeto presente no espaço virtual, é necessário realizar o procedimento de seleção e ativação da opção *Do Not Render* separadamente para cada um deles. A seleção de um conjunto de objetos e ativação dessa opção uma única vez desativa somente a visibilidade do último objeto selecionado do conjunto.

Bloquear a Exportação de Elementos

Além de ocultar um elemento na visualização do arquivo final exportado, pode-se fazer com que elementos não sejam exportados junto com o resto do ambiente virtual. Um objeto 3D, câmera, luz ou outro pode ter sua exportação desativada,

primeiro o selecionando e, depois, ativando a opção *Do Not Export*, encontrada na janela *Properties* > menu *Object* > *Export Options* (Figura 101).

Figura 101 – Localização da opção *Do Not Export* na janela *Properties* > menu *Object* > *Export Options*.



OBS: Para desativar a exportação de múltiplos elementos inseridos no espaço virtual, é preciso realizar o procedimento de seleção e ativação da opção *Do Not Export* em separado para cada um deles. A seleção de um conjunto de elementos e ativação da opção uma única vez desativa somente a exportação do último objeto selecionado do conjunto.

4.4. OBTENÇÃO E OTIMIZAÇÃO DE MODELOS 3D

Os objetos tridimensionais preenchem o espaço virtual e são os principais elementos dos Ambientes 3D Virtuais e os primeiros a serem inseridos na cena.

Dependendo da necessidade e propósito, eles podem ser obtidos já prontos, por meio da internet, por exemplo, ou modelados do zero através de *softwares* específicos.

Em se tratando da aquisição de objetos pela internet, existem diversos bancos de modelos 3D onde é possível encontrar variados tipos de objetos, gratuitos e pagos, em diversos formatos de arquivo. Destacam-se os de maior popularidade, diversidade e número de modelos: TurboSquid (http://www.turbosquid.com), Free3D (https://free3d.com/), 3D Warehouse (https://3dwarehouse.sketchup.com), GrabCAD (https://grabcad.com/library), Thingiverse (http://www.thingiverse.com/), Archive 3D (http://archive3d.net/) e Clara.io (https://clara.io/library).

Todos os referidos acima possuem uma grande quantidade de modelos gratuitos e é necessário atentar apenas para os formatos de arquivos disponíveis para cada modelo, que devem ser um dos suportados pelo Blender para importação: Collada (.dae), Alembic (.abc), 3D Studio (.3ds), FBX (.fbx – em suas versões mais recentes), Motion Capture (.bvh), Stanford (.ply), Wavefront (obj.), X3D Extensible 3D (.x3d/.wrl), STL (.stl), Scalable Vector Graphics (.svg). O processo específico de importação de modelos no Blender se dá pelo menu *Import*, localizado na barra *Info > File*. Nele, deve-se escolher o formato do arquivo que se pretende importar para então abrir uma nova janela onde é possível buscar no computador o arquivo desejado.

Quanto à criação integral de objetos 3D, pode-se realizá-la no próprio Blender ou em outro *software* de modelagem tridimensional que possibilite a exportação de modelos em um desses formatos. Entre eles, 3ds Max, SolidWorks, Sketchup, AutoCAD, Maya, Rhinoceros, Modo, Inventor, Cinema 4D, Zbrush e tantos outros servem como fornecedores de *inputs* (entradas) e podem atuar paralelamente a este processo de construção de ambientes 3D virtuais. Destaca-se, no entanto, que o ensino acerca do processo de modelagem e atribuição de materiais nesses e em outros programas não faz parte desta sistemática.

4.4.1. Otimizações

Existem algumas práticas com relação aos modelos 3D criados ou importados no Blender que podem otimizar o desempenho dos ambientes virtuais exportados. Assim, isso exige menos capacidade de processamento dos dispositivos utilizados para a

visualização do conteúdo e diminuindo o tamanho do arquivo final. Os quatro tipos de otimização (Suavização de Superfícies, Edição de Texturas, Duplicação Ligada e Níveis de Detalhes) fazem parte do Capítulo 1 do livro 'Ambientes Virtuais 3D Interativos' entre as páginas 65 e 72.

4.5. MATERIAIS E TEXTURAS

Os materiais virtuais buscam ter propriedades e funcionamento próximos aos de materiais do mundo físico. Ambos, descrevem a resposta da superfície de um objeto à luz e também contêm informações sobre sua transparência, refletividade, parâmetros físicos e assim por diante.

As texturas de objetos virtuais também funcionam de forma semelhante às texturas de objetos físicos. Elas são imagens aplicadas às superfícies dos objetos virtuais e trabalham em conjunto com as características dos materiais atribuídos a eles. Tecnicamente, os *pixels* da imagem são atribuídos aos pontos da superfície do objeto usando um mapeamento de textura. Por essa razão, às vezes são chamadas de mapas.

O conjunto de programas Blender + Blend4Web possui uma grande variedade de parâmetros para configuração de materiais e texturas. Os procedimentos básicos para a criação deles e aplicação em objetos 3D que possuam apenas materiais e texturas simples estão presentes no Capítulo 1 do livro no intervalo entre as páginas 72 e 90.

4.6. INSERÇÃO DE ELEMENTOS (LUZES, CÂMERAS, SONS, TEXTOS E ANOTAÇÕES)

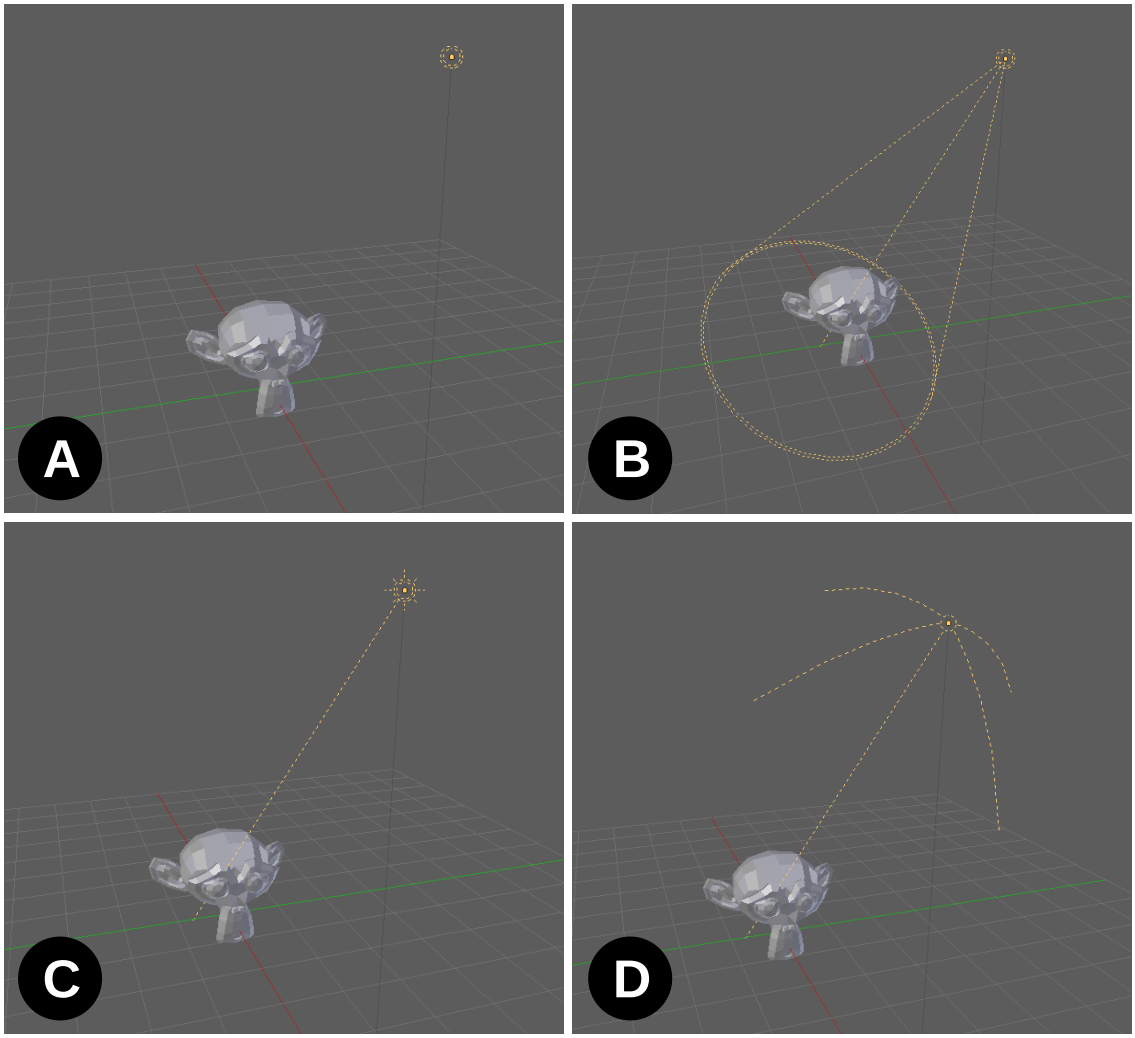
Luzes e câmeras são elementos essenciais em um ambiente virtual 3D, porque são responsáveis pela iluminação e captura da cena, respectivamente. Já os sons, os textos e as anotações, mesmo não sendo fundamentais, não deixam de ser importantes para a experiência de visualização de muitos cenários interativos.

O processo de inserção e configuração básica desses elementos no espaço virtual do Blender é apresentado entre as páginas 93 e 106 do livro (PERGHER; AYMONE, 2017).

4.6.1. Luzes (*Lamp*)

A iluminação de um ambiente provê dos diferentes tipos de luzes presentes nele, também chamadas de lâmpadas. Os quatro tipos de luzes do Blender, que são suportados também pelo Blend4Web (*Point*, *Spot*, *Sun* e *Hemi* - Figura 102), estão explicados em detalhes no Capítulo 2, página 93 a 96, do livro 'Ambientes Virtuais 3D Interativos'.

Figura 102 – Os quatro tipos de luzes do Blender suportados pelo Blend4Web. (A) *Point*, (B) *Spot*, (C) *Sun* e (D) *Hemi*.



4.6.2. Câmeras (*Camera*)

O elemento câmera é essencial para uma cena virtual. Todo o conteúdo referente a sua inserção e configuração está no livro 'Ambientes Virtuais 3D Interativos' (Capítulo 2 - página 96 a 99). Também no livro, há uma explicação sobre a técnica de fixação de elementos em relação à câmera (p. 100 e 101).

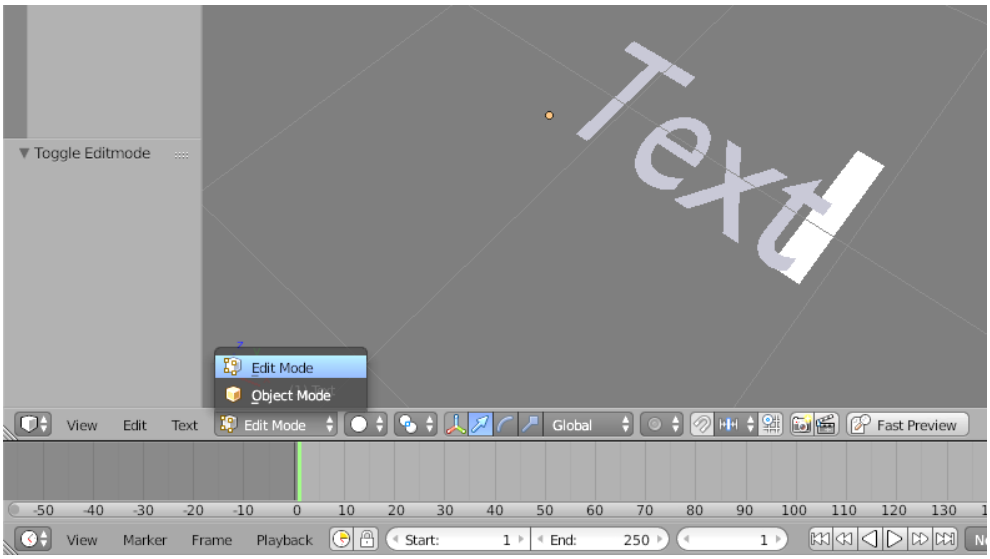
4.6.3. Sons (*Speaker*)

Basicamente, os sons no Blender podem ser de dois tipos: *Positional Sound*, quando o som é emitido de uma posição específica do espaço virtual, onde é mais intenso, e *Background Sound*/*Music*, quando o som é projetado para todas as direções com a mesma intensidade. A adição de sons no ambiente virtual do Blender acontece através do elemento *Speaker* (Autofalante em inglês) e os detalhes sobre ele se encontram no Capítulo 2 do livro 'Ambientes Virtuais 3D Interativos', nas páginas 101, 102 e 103.

4.6.4. Textos (*Text*)

O Texto é um elemento de natureza bidimensional, entretanto pode ganhar uma terceira dimensão pelo processo de modelagem 3D virtual chamado *Extrude* (Extrusão em inglês). As instruções de como criar um texto no Blender, através do elemento chamado *Text* (Figura 103), estão nas páginas 103 e 104, do capítulo 2 do livro 'Ambientes Virtuais 3D Interativos'.

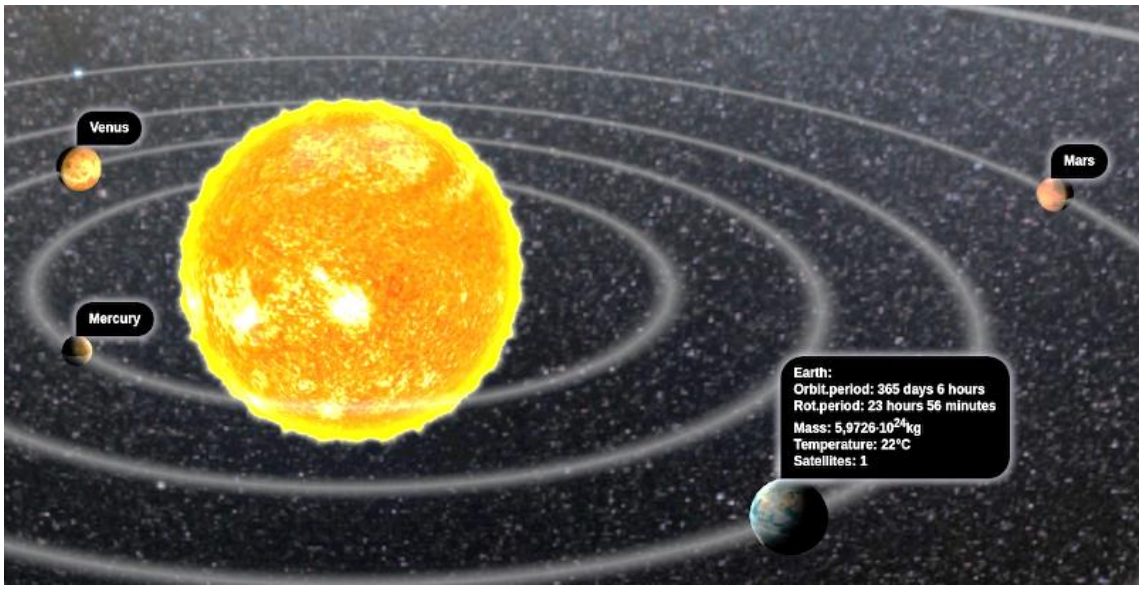
Figura 103 – Elemento *Text*, com texto padrão, sendo editado no Blender.



4.6.5. Anotações (*Anchor*)

O recurso *Anchors* pode ser usado para anexar anotações a objetos 3D (Figura 104). Seu uso é melhor explicado nas páginas 105 e 106 (Capítulo 2) do livro 'Ambientes Virtuais 3D Interativos'.

Figura 104 – Exemplo de aplicação de anotações aos elementos do ambiente virtual.



Fonte: BLEND4WEB (2016).

4.7. CONFIGURAÇÃO DO AMBIENTE

As configurações do ambiente que envolve o cenário são inerentes ao espaço virtual e, por isso, não estão atreladas a nenhum elemento específico. Elas definem diversas características do plano de fundo da cena e de sua iluminação. Em separado estão as opções que ativam e ajustam as sombras e os reflexos. Estes elementos trazem maior realismo ao ambiente virtual, transmitindo informações sobre a posição dos objetos e das fontes de luz. O conteúdo do Capítulo 3 do livro 'Ambientes Virtuais 3D Interativos' é destinado inteiramente a este assunto.

4.7.1. Céu (*Sky*)

O céu é basicamente o plano de fundo do espaço virtual. É assim chamado, pois gera um degradê que simula um verdadeiro céu e linha do horizonte. Não é um elemento que precise ser inserido ou criado, apenas tem de ser ativado e configurado, segundo algumas propriedades básicas.

Dois tipos de céus são apresentados e detalhados no livro 'Ambientes Virtuais 3D Interativos' (página 109 a 111), o Céu Simples e o Céu Procedural. Este último também habilita a visualização de uma espécie de sol (Figura 105) que aparece em uma posição predefinida, mas que pode ser alterada com a inserção de uma luz do tipo *Sun*.

Figura 105 – Céu do tipo Procedural com simulação de sol.

4.7.2. Sombras (*Shadows*)

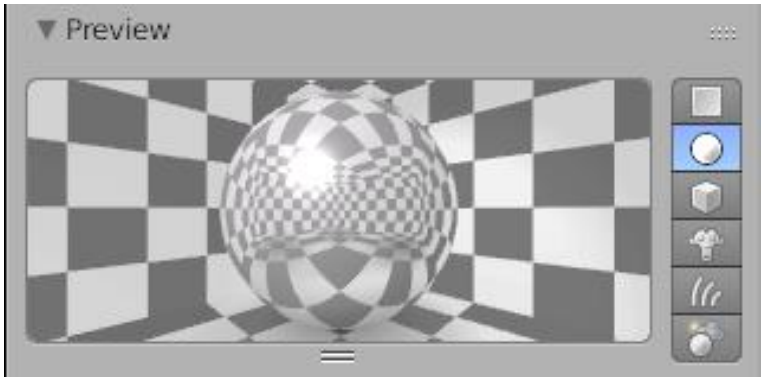
As sombras (Figura 106) são importantes para um ambiente virtual realístico, contudo forçam a realização de muito cálculos em tempo real para que o efeito seja exibido na tela. Por isso, alguns dispositivos podem ter dificuldades para executar de forma fluída um ambiente com grande número de sombras projetadas, sendo necessária a realização de ajustes em suas configurações visando maior fluidez na visualização do ambiente posteriormente, no navegador web. Todo o conteúdo da sistemática sobre as sombras está entre as páginas 111 e 114 do livro 'Ambientes Virtuais 3D Interativos'.

Figura 106 – Sombras projetadas pela coroa, cálice e dados em um outro objeto.

4.7.3. Reflexos (*Reflections*)

Assim como as sombras, os reflexos (Figura 107) não são essenciais para que um ambiente virtual exista, mas também proporcionam uma maior sensação de realismo para a cena. A configuração deste recurso se encontra nas páginas 114 a 117 do livro 'Ambientes Virtuais 3D Interativos'.

Figura 107 – Pré-visualização, no Blender, de um material com alto grau de refletividade.



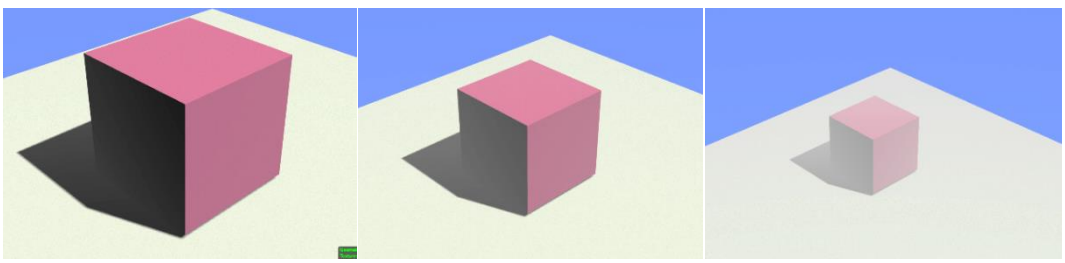
4.8. EFEITOS AO AR LIVRE (*OUTDOOR EFFECTS*)

Existem também alguns recursos adicionais denominados *Outdoor Effects* (Efeitos ao ar livre em tradução literal) que conferem ainda mais realismo adicionando, por exemplo, névoa, superfícies de água, vento, estrelas e emissão de partículas para simular chuva, neve fumaça, etc. Estes efeitos compõem o Capítulo 4 (p. 121 a 140) do livro 'Ambientes Virtuais 3D Interativos'.

4.8.1. Névoa (*Mist*)

O Mist, também chamado de Fog, é um recurso existente desde de tecnologias anteriores, como o VRML. Ele serve para simular uma névoa ou neblina que paira sobre a cena (Figura 108). E seu funcionamento é explicado nas páginas 121 e 122 do livro 'Ambientes Virtuais 3D Interativos'.

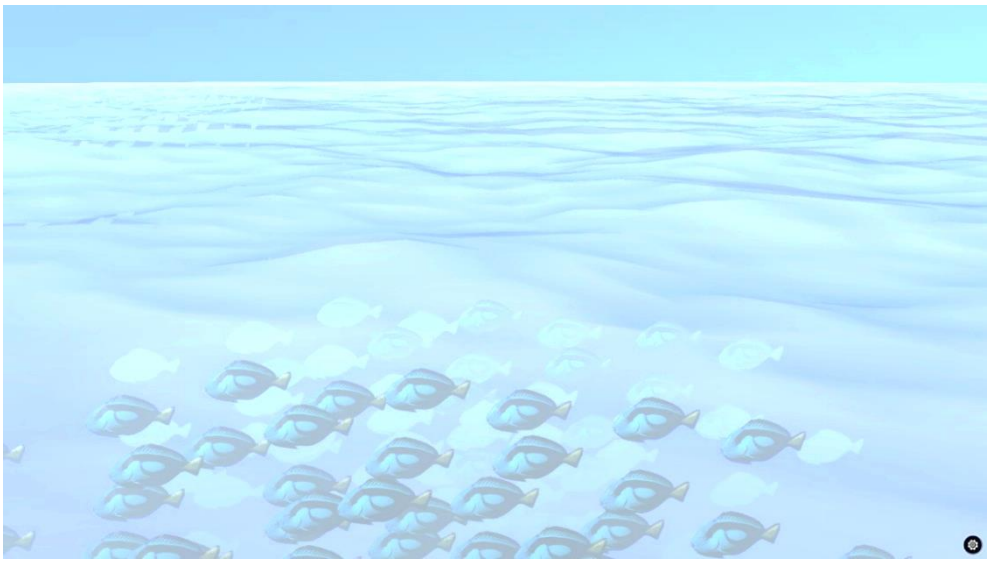
Figura 108 – A *Mist* faz com que a cena perca visibilidade à medida que se afasta a câmera dela, devido a névoa que fica cada vez mais densa.



4.8.2. Água (*Water*)

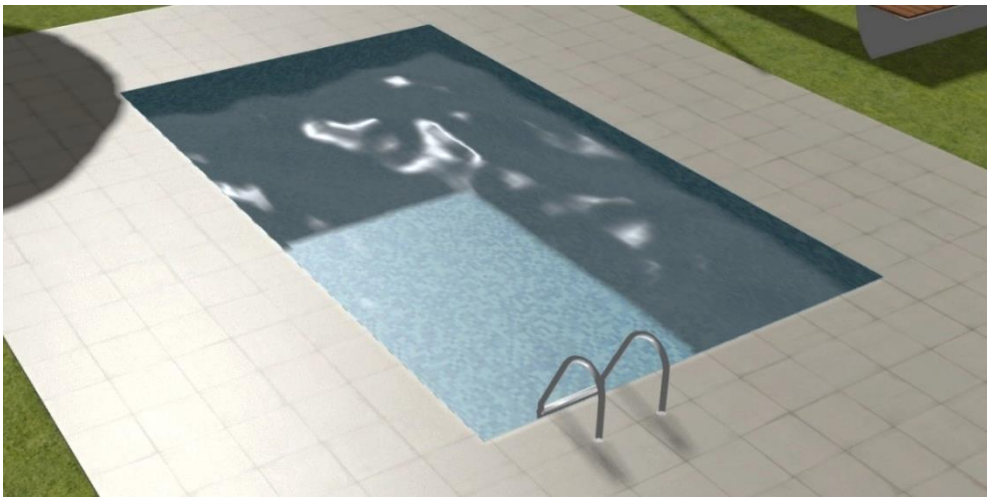
O Blend4Web possibilita também a criação de uma simulação animada de água aplicada na forma de um material a um objeto (Figura 109), geralmente um *Plane* (Plano em inglês). Sendo ideal para a simulação de amplas superfícies de água ao ar livre (oceano, lago...).

Figura 109 – Simulação de água gerada a partir do recurso *Water*.



Também, através de outra técnica, pode-se criar facilmente um efeito estático de água com ondas aplicado a um plano de dimensões definidas, que não se estende até o infinito. Ele pode ser posicionado no local pretendido simulando a superfície da água em reservatórios como uma piscina, por exemplo (Figura 110).

Figura 110 – Exemplo de efeito de água estático aplicado em uma piscina virtual.



Os dois tipos de simulação de água são detalhados no livro ‘Ambientes Virtuais 3D Interativos’, entre as páginas 122 e 129.

4.8.3. Emissão de Partículas (*Particles Emitter*)

O Sistema de emissão de partículas destina-se a criar uma simulação de fenômenos causados pelo movimento de um grupo de objetos pequenos (partículas) como fumaça, chuva, folhas caindo e tantos outros (Figura 111). Para que o efeito aconteça, é necessário um emissor, um objeto tridimensional ou um plano, que definirá a localização e a direção do fluxo de saída das partículas.

A inserção dos elementos necessários e a configuração do recurso de Emissão de Partículas são explicadas da página 130 a 135 do livro ‘Ambientes Virtuais 3D Interativos’.

Figura 111 – Exemplo de aplicação do efeito emissão de partículas (neve caindo sobre vilarejo).

4.8.4. Vento (*Wind*)

Outro efeito muito interessante é o *Wind* que, quando presente na cena, simula a ação do vento em ambiente externo ou interno, funcionando mais ou menos como

um ventilador (Figura 112). Esse vento criado afeta outros elementos interagindo com o movimento de objetos com o recurso *Wind Bending* (não detalhado nesta sistemática) ativado, com a dinâmica do sistema de partículas deslocando as partículas emitidas e com a dinâmica das ondas (*Waves*) do recurso *Water*.

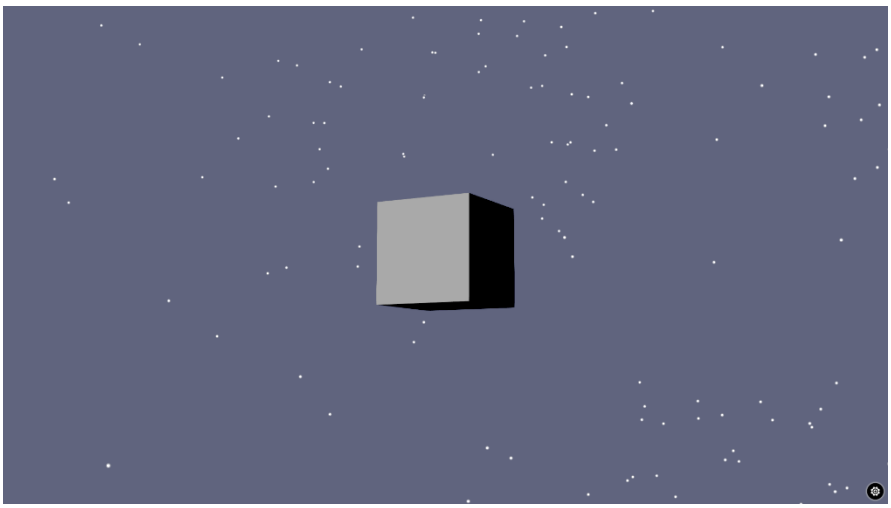
Nas páginas 136 e 137 do livro 'Ambientes Virtuais 3D Interativos' estão as instruções de uso deste recurso, que, como já explicado, pode ser combinado com outros, dependendo da intenção do criador do ambiente virtual.

Figura 112 – Exemplo de utilização do recurso *Wind* na simulação de uma bandeira ao vento (3 quadros da animação).

4.8.5. Estrelas (*Stars*)

Algo que pode ser útil em cenários representados à noite é a criação de um céu estrelado (Figura 113). Ele pode ser facilmente criado utilizando-se um tipo diferente de material aplicado a um objeto 3D que envolva a cena. A técnica de utilização deste recurso é apresentada entre as páginas 137 e 140 do livro 'Ambientes Virtuais 3D Interativos'.

Figura 113 – Exemplo de céu estrelado circundando a cena composta por um cubo.



4.9. CRIAÇÃO DE ANIMAÇÕES

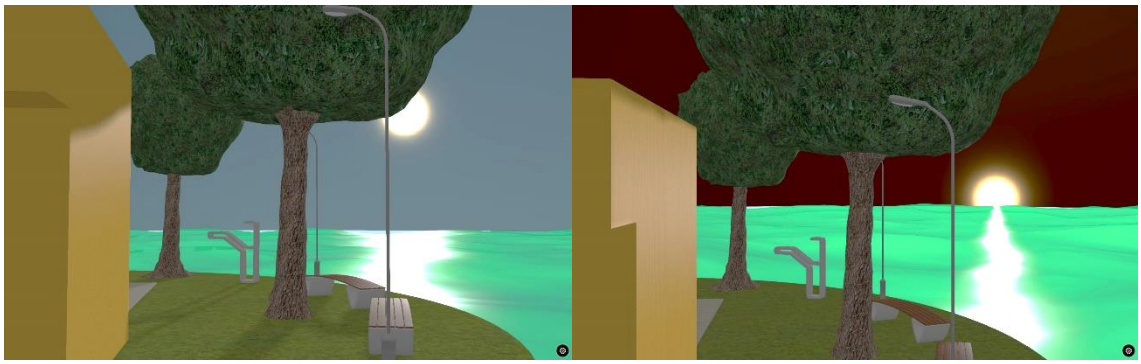
O Blend4Web suporta alguns tipos de animações, entre elas a já referida Emissão de Partículas (item 4.8.3). A Animação de Objetos, a Animação de Vértices e a Animação por Seguimento de Caminho são outros exemplos em conjunto com um processo de animação, utilizando-se o recurso de efeitos de física de corpos rígidos do Blender. Também é abordado o efeito chamado *Object Outlining*, que aplica um contorno luminoso e animado ao redor de objetos. As animações de esqueletos, de fontes de áudio e de alteração de valores de materiais, apesar de serem suportadas pelo Blender e Blend4Web, não atendem aos requisitos estipulados previamente e não são comtempladas por esta sistemática.

O conteúdo sobre animações na íntegra constitui o Capítulo 5 (p. 143 a 174) do livro 'Ambientes Virtuais 3D Interativos'.

4.9.1. Animação de Objetos (*Object Animation*)

Em geral, uma animação ocorre quando parâmetros de um objeto são modificados ao longo do tempo. A Animação de Objetos é utilizada quando o objeto é transformado como um todo, por exemplo, quando ele se desloca inteiramente de um lugar para o outro (Figura 114) ou quando é rotacionado ou escalado, sem sofrer deformação que modifique a sua malha. Todos os passos para criação deste tipo de animações estão também no livro 'Ambientes Virtuais 3D Interativos', entre as páginas 143 e 147.

Figura 114 – Dois quadros de uma animação de objetos simulando um pôr do sol através da alteração da localização da luz do tipo *Sun* ao longo do tempo.

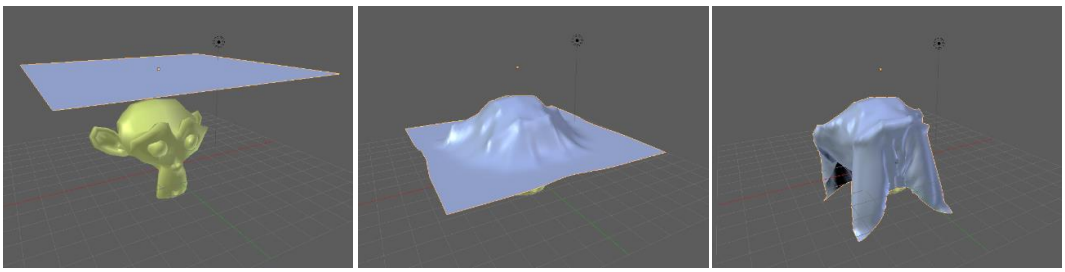


4.9.2. Animação de Vértices (*Vertex Animation*)

Enquanto a Animação de Objetos grava os estados do objeto por inteiro, conforme sua Localização, Rotação e Escala por meio dos *Keyframes*, a Animação de Vértices grava alterações em sua geometria. Nesse caso, cada quadro da animação é constituído por uma malha do objeto em um estado de transformação distinto. Devido a isso, não é recomendado fazer uma longa animação desse tipo com objetos de geometria muito detalhada, uma vez que pode aumentar bastante o tamanho do arquivo final e a necessidade de processamento do *hardware*. A simulação de uma bandeira que balança ao vento ou de um tecido que cai sobre um objeto e se deforma (Figura 115) são dois exemplos de animações possíveis com alteração de malha.

O processo de desenvolvimento de animações de vértices é explicado no intervalo da página 147 a 159 do livro 'Ambientes Virtuais 3D Interativos'. Nesta parte ainda há a apresentação de relevantes técnicas, como a inserção de imagens como planos e a definição de pontos de fixação de tecidos simulados.

Figura 115 – Quadros de uma animação simulando tecido, utilizando a animação de vértices.

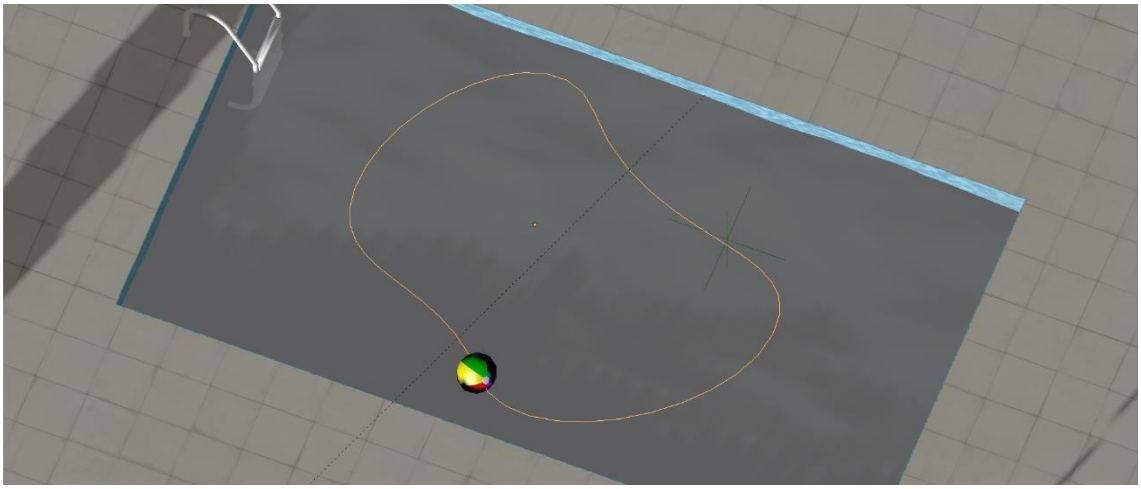


4.9.3. Animação de Seguimento de Caminho (*Path Animation*)

Este é um tipo de animação bastante comum nos *softwares* de animação tridimensional desde de tecnologias de representação passadas, como o VRML. Consiste em fazer com que um ou mais objetos percorram um caminho definido por uma linha ou curva (Figura 116). O processo de criação de animações de objetos que se deslocam

por uma curva é simples e, no livro ‘Ambientes Virtuais 3D Interativos’, vai da página 160 a 166.

Figura 116 –Uma bola e um caminho (curva), pelo qual ela percorrerá por meio da animação de seguimento de caminho.



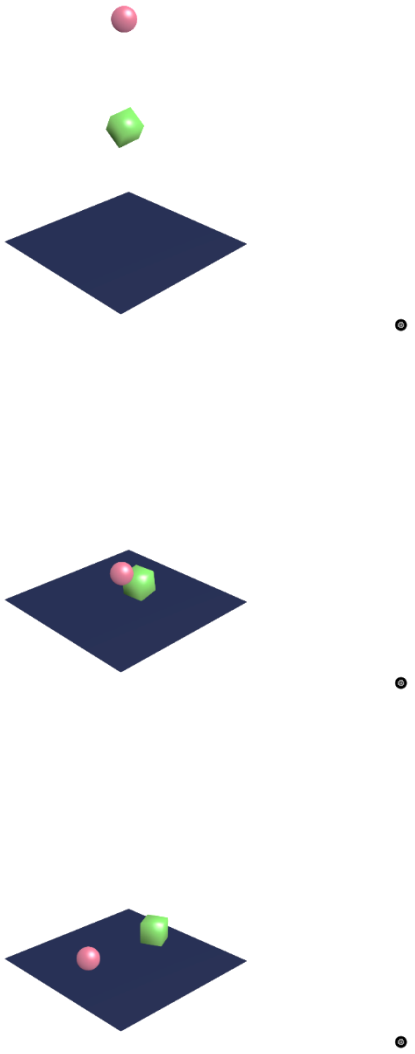
4.9.4. Animação de Efeitos de Física (*Physics*)

Esse tipo de animação simula efeitos simplificados de física de colisão de corpos rígidos sob o efeito da gravidade (Figura 117). Destaca-se, contudo, que apesar da geração de simulações desse tipo a partir das ferramentas do Blend4Web não exigir grandes esforços e elas serem devidamente visualizadas dentro do Blender e pelo recurso *Fast Preview*, os desenvolvedores do complemento Blend4Web optaram pela não incorporação dessas simulações junto aos demais elementos no arquivo HTML único, quando exportado. Com isso, caso se deseje criar efeitos de física que sejam manipuláveis durante a execução do ambiente no navegador, é necessário exportar o ambiente no formato JSON e realizar os procedimentos necessários de incorporação dele e do motor gráfico em um arquivo HTML. Esse processo, além de gerar mais de um arquivo final necessário para visualização, também exige certo conhecimento prévio de programação HTML e, por isso, não é tratado aqui.

Para solucionar essa questão sem eliminar inteiramente as possibilidades de uso dos efeitos de física que o conjunto de programas disponibiliza, existe uma forma de

transformação dessas simulações em animações mais comuns, menos flexíveis, compostas por *keyframes*. Esta técnica de geração da animação de efeitos de física e conversão dela em uma animação de objetos está relatada no livro 'Ambientes Virtuais 3D Interativos' entre as páginas 167 e 171.

Figura 117 – Quadros da animação de um cubo e uma esfera caindo e colidindo em um plano.



4.9.5. Contorno de Objetos (*Object Outlining*)

O Blend4Web possui um recurso chamado *Object Outlining*, que aplica um contorno colorido e luminoso que pisca ao redor de um determinado objeto. Pode ser útil para indicar que determinado objeto da cena é clicável (Figura 118), por exemplo.

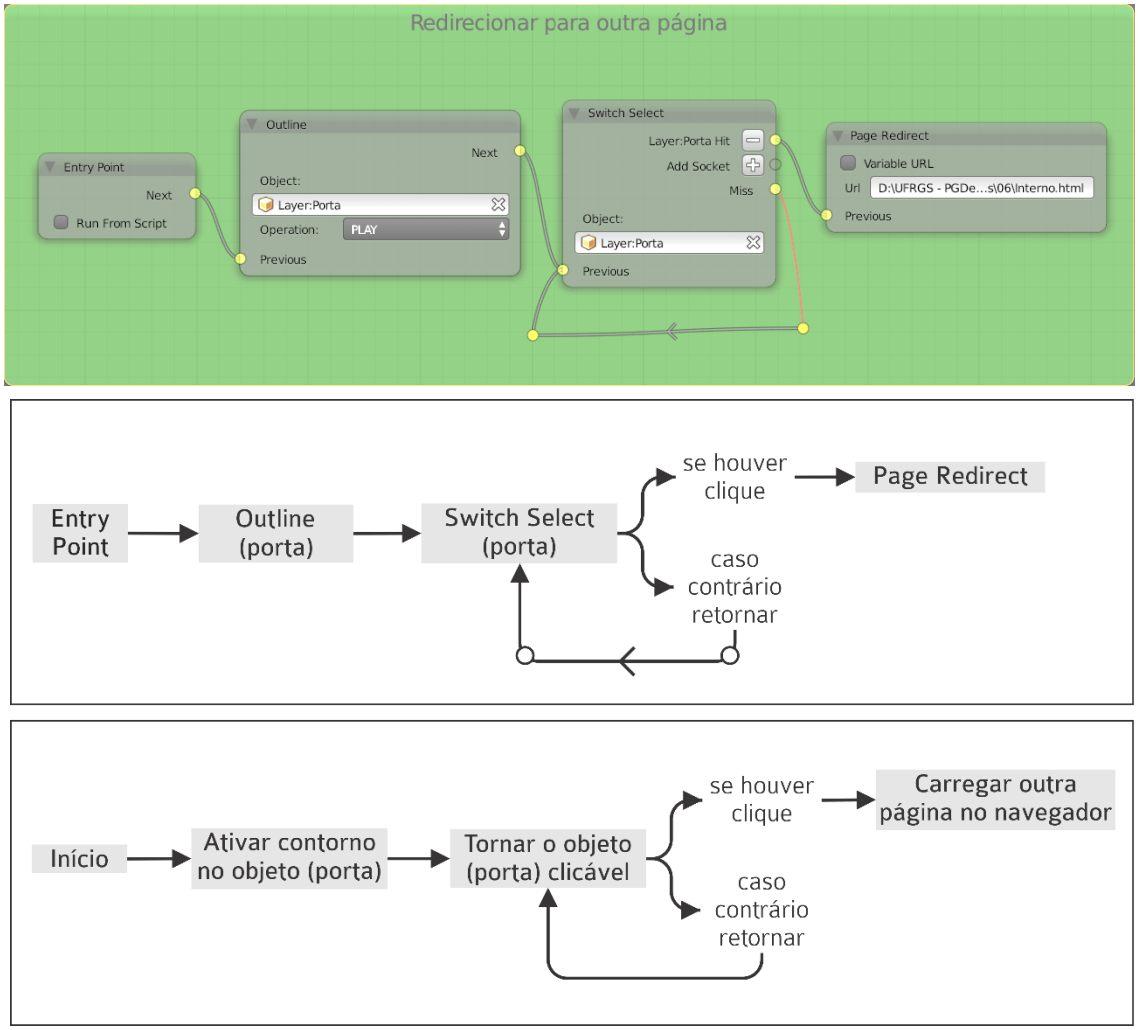
Este é mais um recurso opcional, mas que pode ser útil dependendo do ambiente e das interações em desenvolvimento. Ele é descrito por completo da página 171 a 174 do livro ‘Ambientes Virtuais 3D Interativos’.

Figura 118 – Vasos em cima da mesa possuem um contorno que os realça, indicando que algo acontecerá caso sejam clicados pelo usuário.

4.10. CRIAÇÃO DE INTERAÇÕES

O sistema facilitado de desenvolvimento de interações trazido pelo complemento Blend4Web à linguagem de blocos já presente no Blender é um dos grandes diferenciais deste conjunto de programas. Apesar de existirem outros *softwares* com propostas semelhantes e que também utilizam linguagens visuais, este não exige nenhum conhecimento prévio de programação computacional para criação de interações de variadas complexidades (Figura 119).

Figura 119 – Exemplo de interação simples representada de três diferentes formas. De cima para baixo, conjunto de blocos da linguagem utilizada nos programas, diagrama simplificado do conjunto de blocos (cinza) com ações do usuário e, por último, diagrama com as ações literais de cada bloco (cinza) e do usuário.

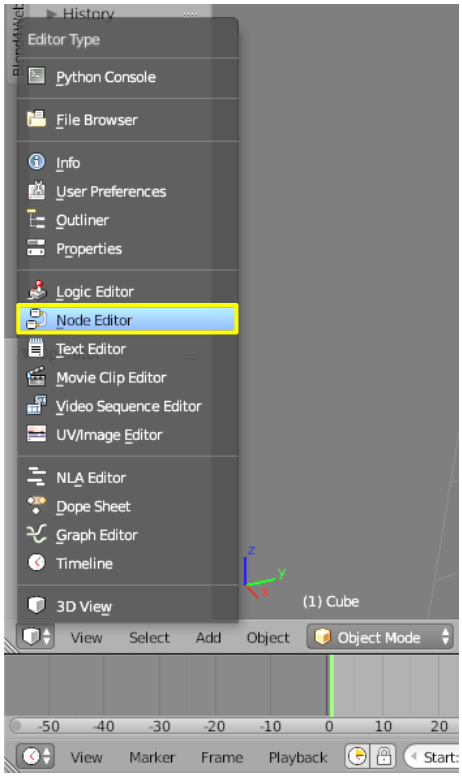


4.10.1. Acesso ao *Node Editor*

Toda a construção das interações por meio da linguagem de blocos se dá na janela *Node Editor* (Figura 56) e é possível acessá-la de três maneiras, podendo ser utilizada qualquer uma destas três formas para acessar as outras janelas apresentadas no item 4.3 (Interface Geral).

1. O primeiro jeito de acessar a janela *Node Editor* é clicando no botão *Area Type*, localizado sempre em um dos cantos de cada janela do Blender e representado por duas setas (uma para cima e outra para baixo) mais o ícone correspondente a janela que está ativa (Figura 120). Assim, abre-se um menu para escolha da janela *Node Editor* que substituirá a janela ativa naquele local;

Figura 120 – Seleção da janela *Node Editor* pelo menu de janela ativa, acessado pelo ícone *Area Type* no canto inferior esquerdo da janela *3D view*.

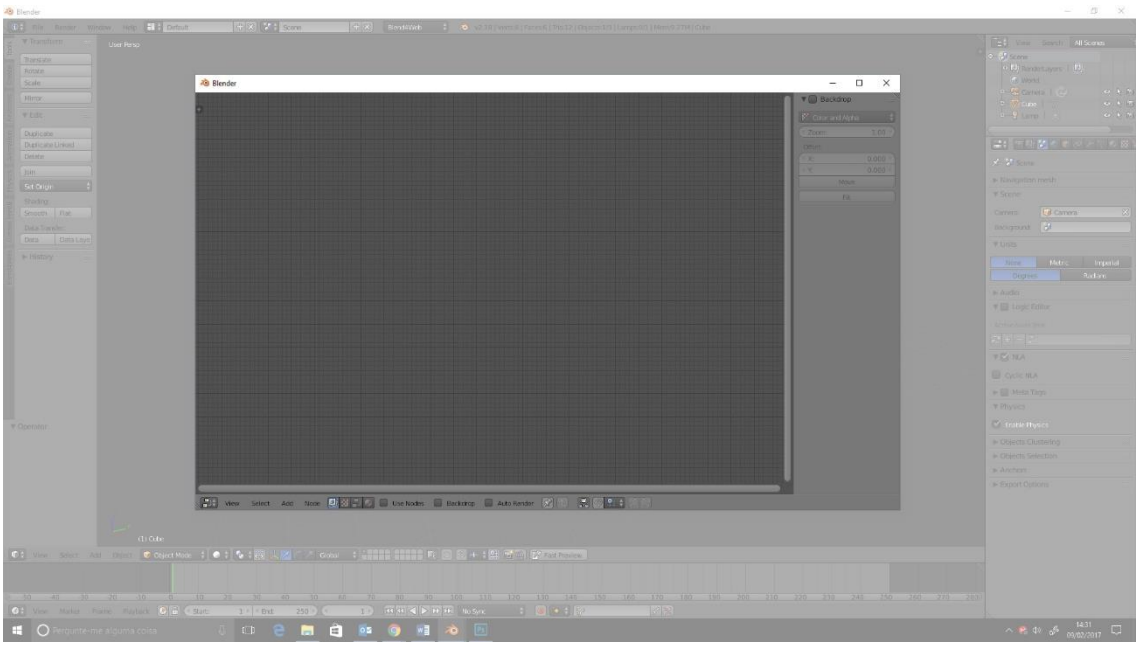


2. A segunda forma de acesso à *Node Editor* é manter clicado e arrastar para um dos lados o botão triangular representado por três linhas diagonais, localizado também em um dos cantos de todas as janelas do Blender (Figura 121) e muitas vezes ao lado do botão *Area Type*. Isso faz com que uma cópia da respectiva janela ativa se crie ao seu lado sem substitui-la. Depois, é só utilizar o primeiro método explicado para então alterar uma das duas janelas idênticas para a *Node Editor*;

Figura 121 – Indicação do botão utilizado para abertura de uma nova janela.

3. Já, o terceiro método é uma variação do segundo. Ao invés de apenas clicar e arrastar o botão triangular (Figura 121), pressiona-se ao mesmo tempo a tecla SHIFT, fazendo com que uma cópia da janela se abra separada das demais do programa, tornando-se independente (Figura 122) e podendo ser disposta onde parecer mais adequado ao usuário, até mesmo em um segundo monitor, se for o caso. O passo seguinte é também alterar a janela criada para a *Node Editor* como no primeiro método.

Figura 122 – Janela *Node Editor* criada separada das demais.

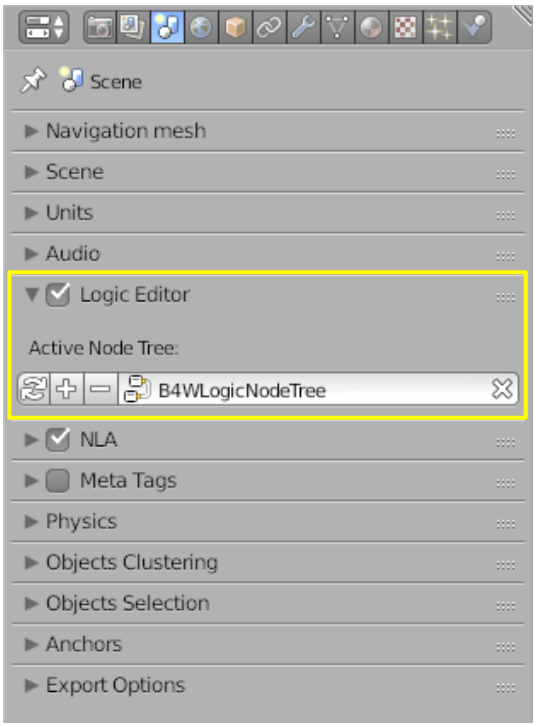


4.10.2. Árvore de Nós (*Node Tree*)

Os blocos, também chamados de Nós, utilizados para a criação da lógica das interações são dispostos em uma *Node Tree* (Árvore de Nós em inglês), podendo existir mais de uma em um mesmo projeto, mas apenas uma pode estar ativa por vez.

1. A primeira coisa a se fazer é clicar no ícone *Add new logic node tree* (sinal de “+”), em *Properties* > menu *Scene* > *Logic Editor* > *Active Node Tree* (Figura 123), para criar uma nova Árvore de Nós, denominada por padrão *B4WLogicNodeTree*;

Figura 123 – Local de criação da Árvore de Nós.



2. Antes de começar a inserir os blocos na Árvore de Nós criada, deve-se torná-la ativa na janela *Node Editor*. Para isso, seleciona-se o botão *Blend4Web logic*, presente na barra inferior da janela *Node Editor* e, em seguida, clica-se no botão denominado *Browse Node Tree to be linked* do lado esquerdo do botão *New*. Um menu será aberto com a Árvore de Nós criada anteriormente (Figura 124). Selecionando-a aparecerá na área de edição da janela *Node Editor* o primeiro e indispensável bloco chamado *Enter Point* (Ponto Inicial) (Figura 125). Após esse procedimento, está tudo pronto para inserir novos blocos e formar as lógicas das interações desejadas.

Figura 124 – Local de ativação da Árvore de Nós na janela *Node Editor*.

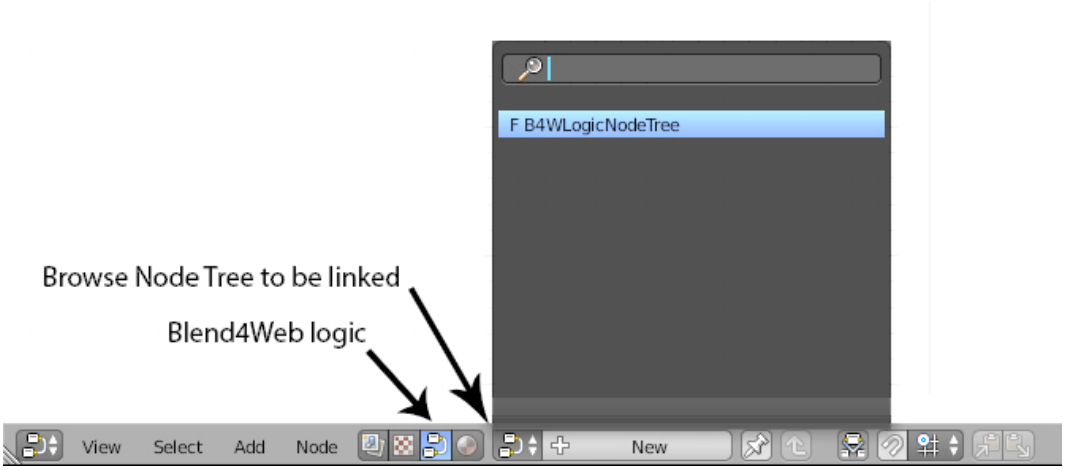
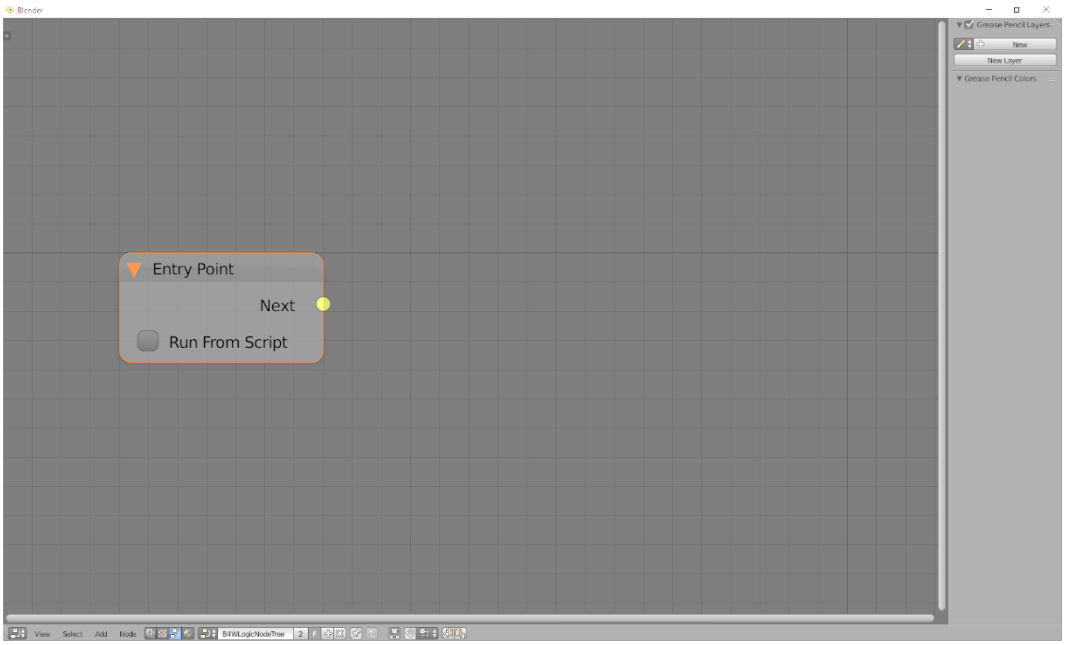


Figura 125 – Janela *Node Editor* já com o bloco de Ponto Inicial.

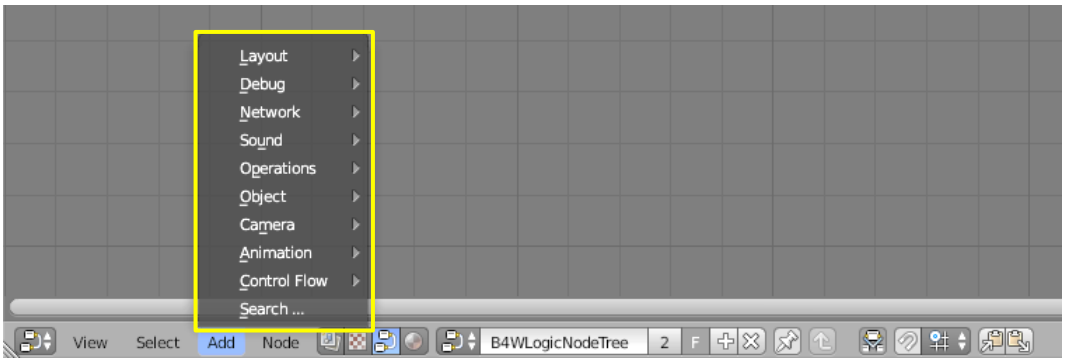


4.10.3. Blocos/Nós (*Nodes*)

Com a Árvore de Nós criada e ativa, pode-se começar a inserir na área de trabalho da janela *Node Editor* os blocos necessários para a criação das lógicas de interação. Os blocos disponíveis estão divididos em submenus (Figura 126), agrupados conforme seus propósitos. Com exceção de alguns poucos blocos que exigem conhecimentos mais avançados do que os pretendidos para esta sistemática, cada bloco será apresentado individualmente antes da explanação sobre as relações possíveis entre eles e a construção das interações propriamente ditas (item 4.10.4).

OBS: Conforme novas versões do Blend4Web são lançadas é possível que os blocos sejam reorganizados em outros submenus que não os apresentados aqui. Também é possível que novos blocos, com novas funções, sejam adicionados no futuro. Lembrando que a versão utilizada aqui foi a 17.08.

Figura 126 – Menu de adição de blocos divido em submenus (Atalho: SHIFT + A).



OBS: Os alertas presentes nas imagens de alguns blocos, representados por um sinal de exclamação dentro de um triângulo amarelo, indicam e descrevem ações necessárias dentro de cada bloco. Eles somem assim que o bloco em questão estiver devidamente configurado.
Além disso, os pontos amarelos presentes nas margens esquerda e direita dos blocos são os pontos de entrada (esquerda) e de saída (direita) de dados do bloco, servindo também para a ligação com os demais blocos. Essa ligação é realizada por meio de "fios" que se ligam a esses pontos e serão melhores explicados na sequência.

Ainda, para o entendimento pleno do funcionamento das interações como um todo é preciso compreender o que é uma Variável (*Variable*) no campo da programação, termo repetido diversas vezes na descrição dos blocos a seguir. Pode-se imaginar que a memória de um computador é composta por diversos espaços de armazenamento, assim como um grande arquivo com várias gavetas. Uma variável pode ser entendida como uma dessas "gavetas", podendo cada uma guardar apenas um dado (nesse caso, um número ou uma sequência de caracteres). Essas variáveis precisam ser criadas ao longo da programação e identificadas com um nome, que é utilizado para indicar ao computador a variável a ser acessada em determinado momento. Se chamam variáveis porque, em geral, os dados nelas contidos podem ser alterados e armazenados novamente conforme for necessário (MANZANO; OLIVEIRA, 2005, p. 24 e 25).

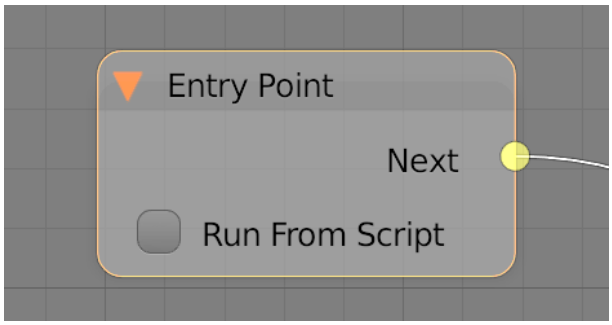
Dentro do menu de adição de blocos (Figura 126), o submenu ***Control Flow*** (Controle de Fluxo em inglês) agrupa os blocos relacionados diretamente com o controle

do fluxo da execução dos procedimentos de lógica. Possui quatro blocos com funções distintas, porém o *JS Callback* não será aqui explorado por motivos já esclarecidos.

- ***Entry Point*** (Ponto Inicial - Figura 127):

  - É utilizado apenas para determinar o local onde a lógica é iniciada e, por esse motivo, é o único que não suporta entrada de dados, somente saída (*Next*);
  - É necessário a presença de pelo menos um *Entry Point*, mas é possível adicionar quantos forem necessários, um para cada procedimento de lógica criado dentro da Árvore de Nós;
  - Sua propriedade *Run From Script* não é relevante aqui.

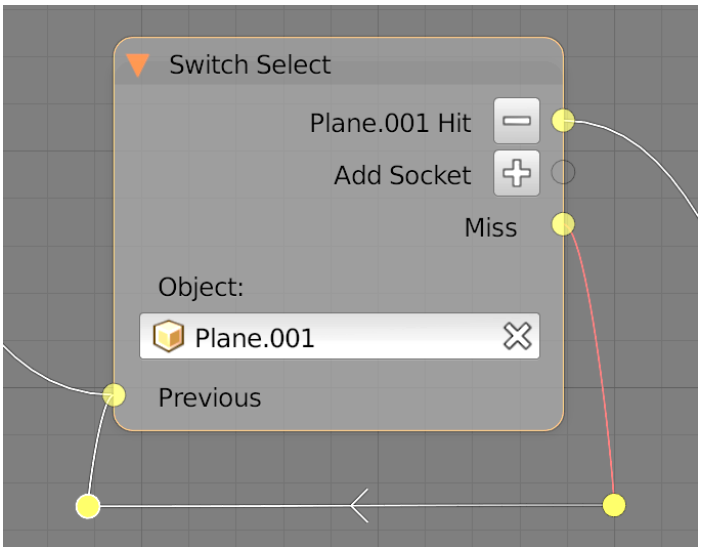
Figura 127 – Bloco Entry Point (Ponto Inicial).



- ***Switch Select*** (Interruptor de Seleção - Figura 128):

  - Pode ser utilizado para determinar que um ou mais objetos presentes na cena possam ser clicados, criando um ponto de saída para cada um com o nome do objeto mais a palavra *Hit* (Ex.: *Cube Hit*). O que estiver após esse ponto de saída será executado caso o objeto seja clicado, já o que estiver após o ponto de saída *Miss* será executado quando nenhum objeto listado no bloco for clicado;
  - Para que a execução se mantenha dentro desse bloco enquanto os objetos nele determinados não sejam clicados, deve-se utilizar o chamado *Reroute*, encontrado em *Add > Layout > Reroute*. Os *Reroutes*, explicados em detalhes na sequência, são pontos de ligação que podem ser adicionados fora dos blocos e, nesse caso, utilizados para se criar um caminho de retorno do ponto de saída *Miss* até o ponto de entrada *Previous*.

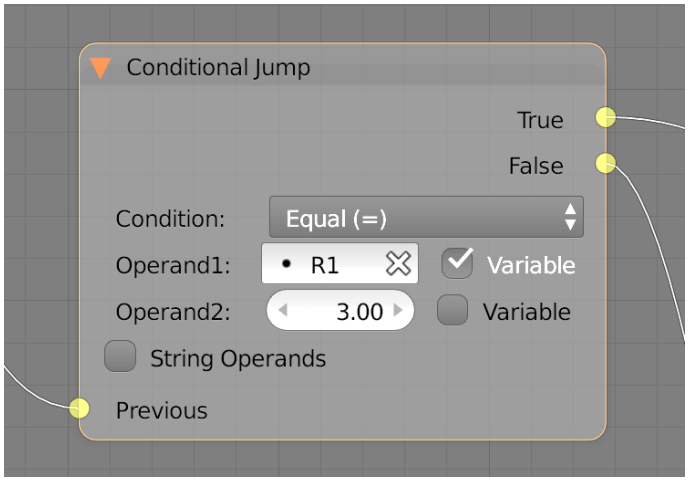
Figura 128 – Bloco *Switch Select* (Interruptor de Seleção) e caminho de retorno com pontos *Reroute*.



- ***Conditional Jump*** (Pulo Condicional - Figura 129):

  - Pode ser utilizado para realizar a verificação de uma condição que, caso seja verdadeira, dá seguimento à execução pelo ponto de saída *True* e, caso seja falsa, dá seguimento pelo ponto de saída *False*;
  - Na opção *Condition,* é possível definir a condição que deve ser atendida entre os valores determinados para que a relação seja considerada verdadeira. As condições são as seguintes: Igual (=), Diferente (!=), Menor que (<), Maior que (>), Menor ou igual que (<=) e Maior ou igual que (>=);
  - Os valores utilizados na relação segundo a condição escolhida são determinados nos operadores *Operand1* e *Operand2*, podendo um ou até mesmo os dois serem substituídos por valores presentes em variáveis anteriormente criadas. Para isso, basta ativar a opção *Variable* no operador e selecionar a variável desejada;
  - Ativando a opção *String Operands,* é possível relacionar textos, que também podem ser guardados em variáveis, ao invés de números;
  - Com esse bloco, pode-se também utilizar a técnica do *Reroute*, a mesma do bloco *Switch Select*, para o caso de o resultado ser falso (*False*), fazendo com que a execução retorne até um bloco anterior, se assim for o desejado.

Figura 129 – Bloco *Conditional Jump* (Pulo Condicional).



O submenu ***Animation*** (Animação em inglês) reúne os blocos relacionados às animações presentes na cena. Possui quatro blocos, porém serão aqui explorados só os dois que trabalham individualmente com as animações dos objetos e emissões de partículas. Os demais blocos controlam animações NLA e, como elas não são aprofundadas nesta sistemática, eles também não têm suas funções descritas aqui.

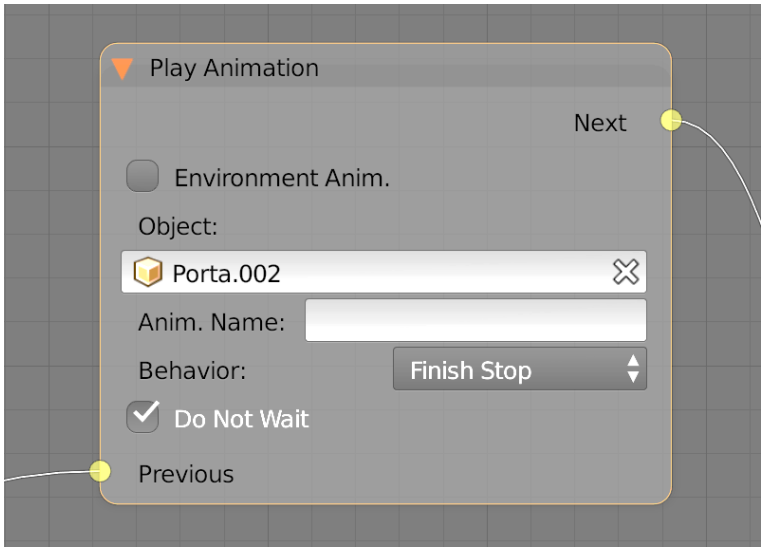
- ***Play Animation*** (Reproduzir Animação - Figura 130):

  - Pode ser utilizado para dar início à reprodução de uma animação de determinado objeto;
  - A opção *Environment Anim.* não é relevante aqui;
  - Clicando na barra abaixo de *Object*, escolhe-se em uma lista o objeto que possui a animação que se pretende iniciar;
  - Em *Anim. Name,* se especifica o nome da animação a ser reproduzida no caso de o objeto em questão ter mais de uma animação atrelada a ele ou no caso de animação de emissão de partículas, onde é colocado o nome do sistema de partículas ao qual se pretende dar início (Ex.: *ParticleSystem*, *ParticleSystem 2*);
  - Em *Behavior,* define-se o comportamento da reprodução da animação que pode ser: *Finish Stop*, quando o objeto animado se mantém em seu estado final após o término da reprodução; *Finish Reset*, quando o objeto animado retorna à posição inicial após o término da reprodução; *Loop*, quando a reprodução se repete indefinidas vezes

automaticamente até que seja determinada sua parada com um bloco posterior;

- A última opção (*Do Not Wait*), quando ativada, determina que a execução dos blocos subsequentes ocorra mesmo que a animação ali configurada ainda não tenha terminado;
- Ressalta-se que um bloco *Play Animation* deve ser adicionado para cada objeto animado da cena quando se desejar iniciar cada animação.

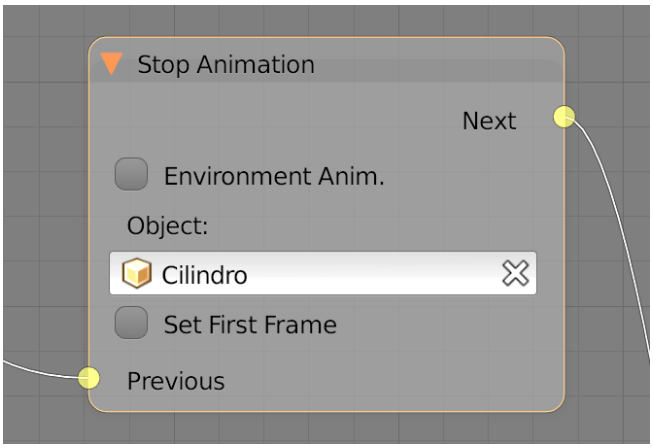
Figura 130 – Bloco *Play Animation* (Reproduzir Animação).



- ***Stop Animation*** (Parar Animação - Figura 131):

- Pode ser utilizado para interromper a reprodução de uma animação de determinado objeto;
- A opção *Environment Anim.* não é relevante aqui;
- Assim como em *Play Animation*, a barra abaixo de *Object* serve para selecionar o objeto com a animação que se pretende parar;
- A opção *Set First Frame* faz com que o objeto volte ao seu estado inicial (primeiro quadro) assim que é interrompida a animação.

Figura 131 – Bloco *Stop Animation* (Parar Animação).



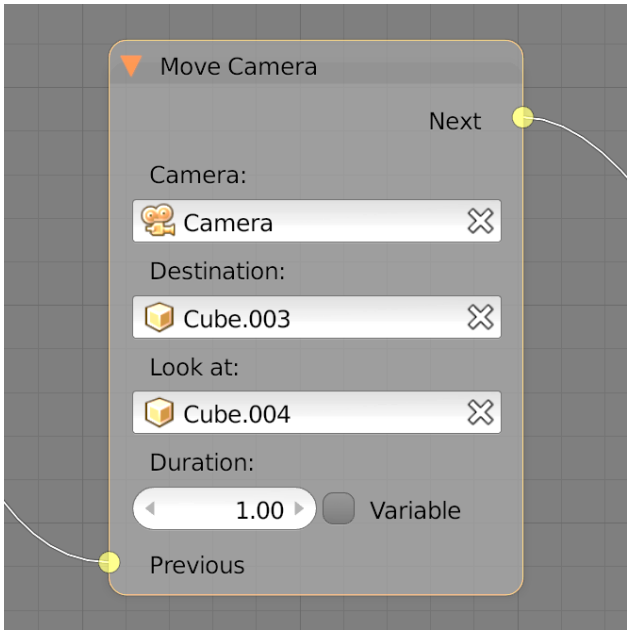
No submenu ***Camera*** (Câmera em inglês) encontram-se os blocos que realizam alterações nas câmeras da cena. Possui somente três blocos que alteram a posição, o estilo de movimentação e os limites dessas câmaras.

- ***Move Camera*** (Mover Câmera - Figura 132):

  - Pode ser utilizado para mover uma câmera de sua posição original até o centro de um objeto determinado e direcionar seu foco para outro. O movimento gerado é o resultado de uma interpolação suave entre o estado inicial e o final;
  - Mesmo que o Blend4web não ofereça nenhum recurso de troca entre câmeras durante a visualização, nem mesmo sendo possível exportar mais de uma câmera, com esse bloco é possível simular a presença de múltiplas câmeras no ambiente. Basta determinar diferentes pontos para aonde a câmera irá e diferentes focos, utilizando um bloco para cada diferente movimentação. Pode-se ainda utilizar outros objetos da cena como “gatilhos”, através do bloco *Switch Select*, para o acionamento dos blocos *Move Camera*;
  - Em *Camera,* escolhe-se a câmera que será movida;
  - Em *Destination,* define-se um objeto que terá a sua posição copiada pela câmera. As coordenadas da câmera serão as mesmas do objeto escolhido após o término do movimento;
  - Em *Look at,* seleciona-se o objeto para o qual a câmera estará apontada depois de ser movida;

- Já em *Duration,* define-se o tempo (em segundo) que a câmera levará para realizar a movimentação. Quando definido para 0 (zero), a câmera simplesmente muda sua posição do início para o fim sem nenhum estado intermediário. Pode-se ainda utilizar o valor de alguma variável para determinar esse tempo, basta selecionar a opção *Variable* e escolher a variável desejada.

Figura 132 – Bloco *Move Camera* (Mover Câmera).



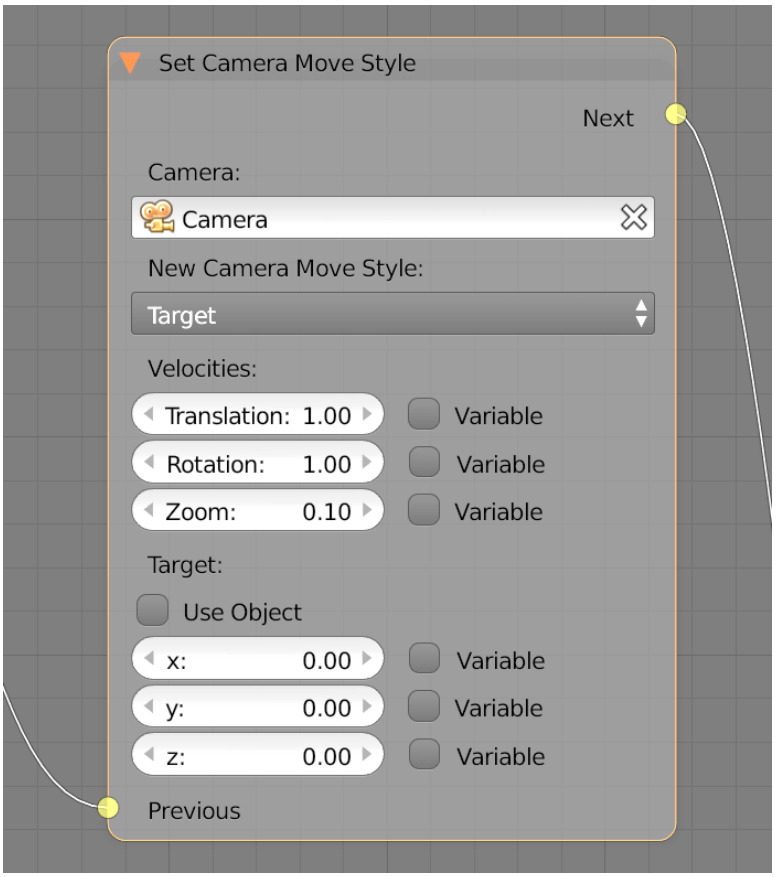
- ***Set Camera Move Style*** (Definir Estilo de Movimento da Câmera - Figura 133):

- Pode ser utilizado para alterar o estilo de movimento de uma câmera;
- Em *Camera,* escolhe-se a câmera que terá o seu estilo de movimentação alterado;
- Em *New Camera Move Style,* define-se o estilo de movimentação que a câmera escolhida adotará quando a programação passar por esse bloco. É possível escolher entre os quatro estilos (*Hover*, *Eye*, *Target*, *Static*) da seção 4.6.2 (Câmeras);
- As demais opções desse bloco variam conforme o estilo de movimentação selecionado. Para o estilo *Static* não há nenhuma configuração adicional.
- Quando presentes, as demais opções configuram a velocidade (*Velocities*) e o ponto alvo da câmera (*Pivot* ou *Target*). Para *Pivot* e

Target é possível selecionar um objeto ao invés de coordenadas, utilizando a opção *Use Object*.

- Há também a possibilidade de se utilizar valores guardados em variáveis para as propriedades desse bloco. Basta selecionar a opção *Variable* ao lado da propriedade desejada.

Figura 133 – Bloco *Set Camera Move Style* (Definir Estilo de Movimento da Câmera).



- ***Set Camera Limits*** (Definir Limites de Câmera - Figura 134):

  - Pode ser utilizado para definir os limites de movimentação de uma câmera;
  - Em *Camera,* escolhe-se a câmera para a qual se definiram limites de movimentação;
  - As propriedades restantes podem ser ativadas ou desativadas conforme o desejado. Seus valores definem os limites de movimentação, segundo os parâmetros do recurso de limitação do movimento de câmeras, encontrados na janela *Properties* > menu

Camera > submenu *Camera Move Style > Camera Limits* e apresentados no item 4.6.2 (Câmeras):

- *Distance Limits*: define os valores do parâmetro *Distance Limits* para as câmeras com estilo de movimentação do tipo *Target* e *Hover*;

- *Hor. Rotation Limits*: define os valores do parâmetro *Horizontal Rotation Limits* para as câmeras com estilo de movimentação do tipo *Target* e *Eye*;

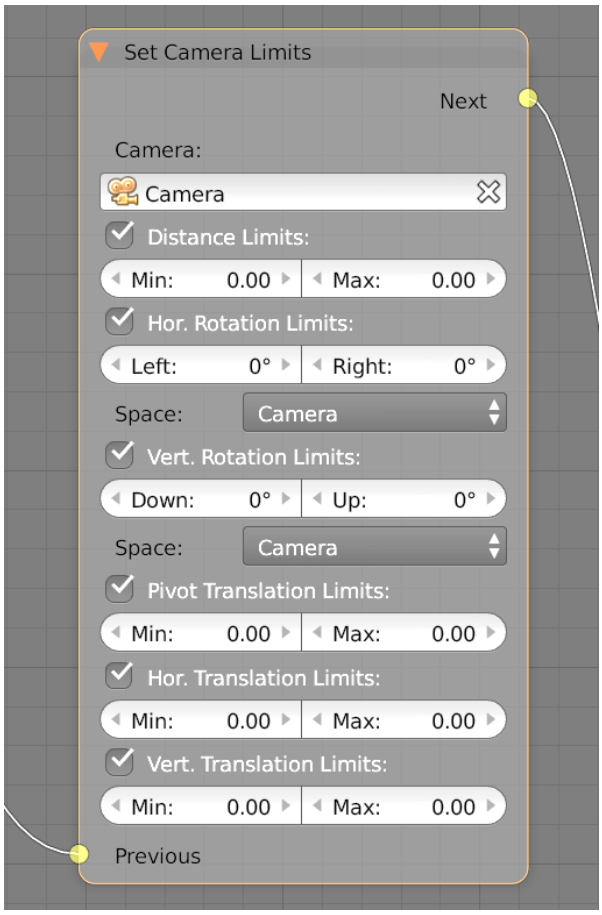
- *Vert. Rotation Limits*: define os valores do parâmetro *Vertical Rotation Limits* para as câmeras com estilo de movimentação do tipo *Target*, *Hover* e *Eye*;

- *Pivot Translation Limits*: define os valores do parâmetro *Pivot Translation Limits* para as câmeras com estilo de movimentação do tipo *Target*;

- *Hor. Translation Limits*: define os valores do parâmetro *Horizontal Translation Limits* para as câmeras com estilo de movimentação do tipo *Hover*;

- *Vert. Translation Limits*: define os valores do parâmetro *Vertical Translation Limits* para as câmeras com estilo de movimentação do tipo *Hover*;

Figura 134 – Bloco *Set Camera Limits* (Definir Limites de Câmera).

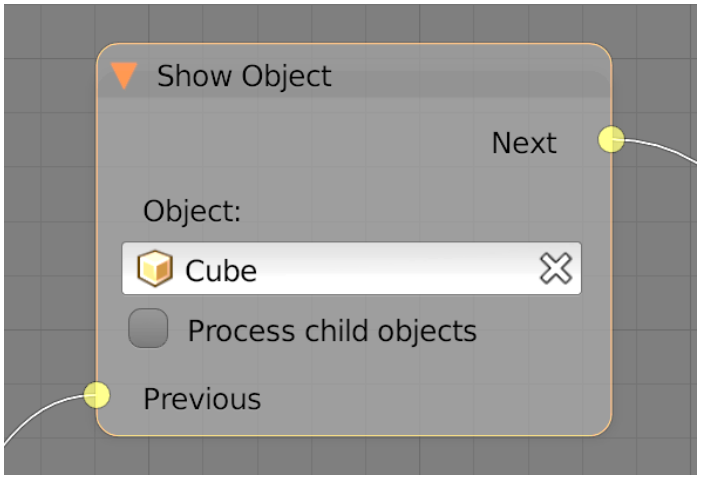


O submenu ***Object*** (Objeto em inglês) traz uma série de blocos que trabalham a visibilidade, movimentação, transformações e materiais de objetos dentro da linguagem visual. Possui oito blocos, porém dois (*Apply Shape Key* e *Set Shader Node Param*) exigem conhecimentos não explorados nesta sistemática e, por isso, não estão presentes aqui.

- ***Show Object*** (Mostrar Objeto - Figura 135):

  - Pode ser utilizado para mostrar objetos ocultos. Torna ativa a visibilidade de objetos na cena;
  - Em sua única propriedade, *Object*, define-se o objeto que se deseja tornar visível.

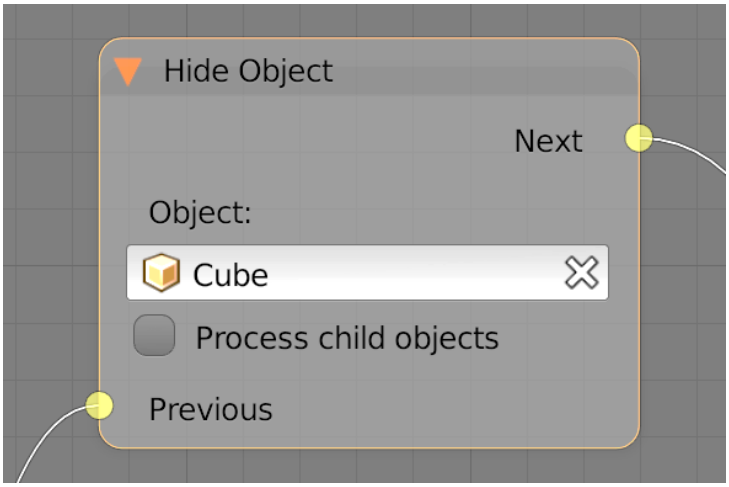
Figura 135 – Bloco *Show Object* (Mostrar Objeto).



- ***Hide Object*** (Ocultar Objeto - Figura 136):

  - Com função exatamente contrária ao *Show Object*, esse bloco pode ser utilizado para ocultar objetos, desativando a visibilidade desses objetos na cena;
  - Também com uma única propriedade chamada *Object*, define-se o objeto que se deseja tornar invisível.

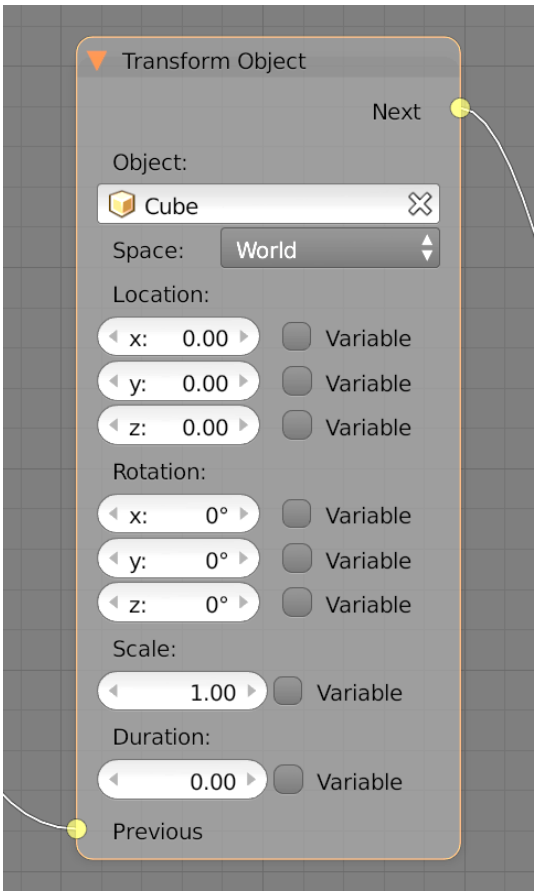
Figura 136 – Bloco *Hide Object* (Ocultar Objeto).



- ***Transform Object*** (Transformar Objeto - Figura 137):

  - Pode ser utilizado para transformar a localização, o tamanho e os ângulos de Rotação de objetos ao longo de um período de tempo definido;
  - Em *Object*, define-se objeto que se pretende transformar;
  - Em *Space*, escolhe-se o espaço de coordenadas que será usado para transformar o objeto. Existem três opções: *World*, *Local* e *Parent*;
  - Em *Location,* são definidas as coordenadas da localização do objeto ao fim da transformação;
  - Por sua vez, em *Rotation,* são escolhidos os ângulos de Rotação do objeto ao fim da transformação;
  - Em *Scale,* especifica-se o tamanho do objeto após transformado.
  - Em *Duration,* define-se o tempo (em segundos) que levará a transformação. Quando o valor for 0 (zero) a transformação se dará imediatamente;
  - Nesse bloco, também é possível utilizar valores de variáveis (opção *Variable*) para as propriedades de transformação.

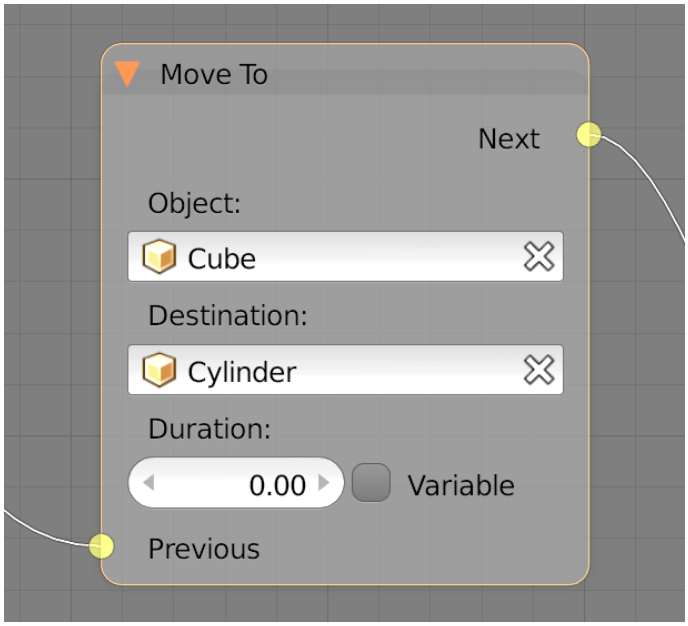
Figura 137 – Bloco *Transform Object* (Transformar Objeto).



- ***Move To*** (Mover Para - Figura 138):

  - Pode ser utilizado para mover um objeto até um elemento alvo, que pode ser tanto outro objeto 3D como também uma fonte de luz, uma câmera, etc. Após movido o objeto, este terá as mesmas coordenadas do alvo;
  - Em *Object,* escolhe-se o objeto que se movimentará;
  - Em *Destination,* seleciona-se o elemento que servirá de alvo para a movimentação do objeto;
  - Em *Duration,* define-se o tempo (em segundos) que objeto levará para realizar o movimento para o novo local. Se definido como 0 (zero) o objeto simplesmente muda sua posição instantaneamente. Pode-se também utilizar aqui um valor guardado em uma variável previamente criada (opção *Variable*).

Figura 138 – Bloco *Move To* (Mover Para).

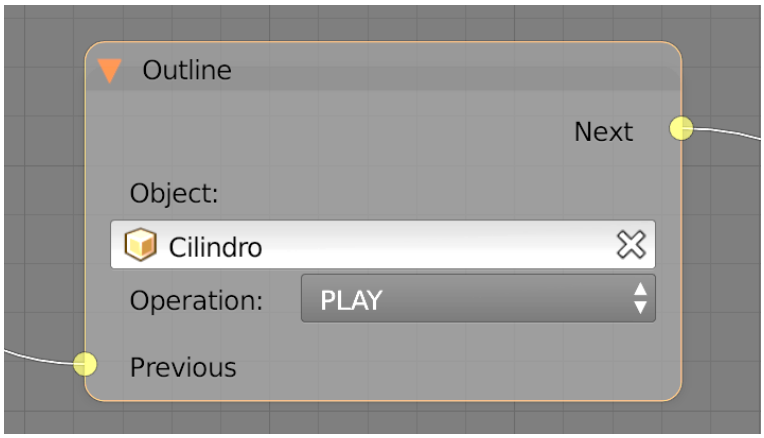


- ***Outline*** (Contorno - Figura 139):

  - Pode ser utilizado para controlar o efeito de contorno de objetos (*Object Outlining* – item 4.9.5), que surge em volta de determinado objeto quando clicado ou em qualquer momento desejado;

- Em *Object,* determina-se o objeto do qual se deseja controlar o contorno;
- Em *Operation,* escolhe-se a operação realizada pelo bloco quanto ao contorno do objeto escolhido. São três opções: *Play*, que determina o início da visualização do contorno; *Stop*, que determina o fim a interrupção do efeito; *Intensity*, que determina a intensidade do efeito de contorno de acordo com um valor numérico que pode ser definido ali mesmo ou vir de uma variável (opção *Variable*).

Figura 139 – Bloco *Outline* (Contorno).



- ***Inherit Material*** (Herdar Material - Figura 140):

- Pode ser utilizado para copiar um material de um objeto do cenário e aplicá-lo em outro objeto específico (Figura 141);
- Em *Source,* escolhe-se o objeto que terá o seu material copiado. Assim que for definido, surgirá logo abaixo um menu (*Material*) para definir qual dos materiais do objeto em questão será copiado. Mesmo que exista mais de um material criado para cada objeto, apenas um poderá ser escolhido;
- Em *Destination,* define-se qual objeto herdará o material copiado. Abaixo, em *Material*, seleciona-se um material específico desse objeto que terá seus atributos substituídos por aqueles copiados.

Figura 140 – Bloco *Inherit Material* (Herdar Material).

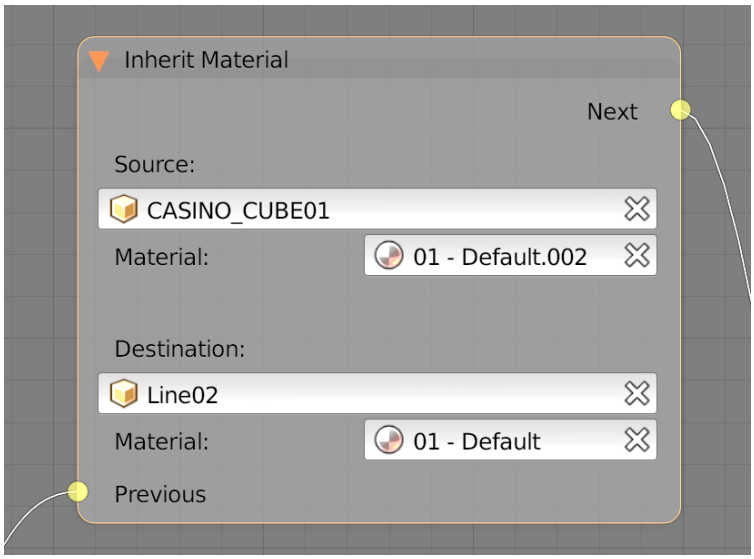
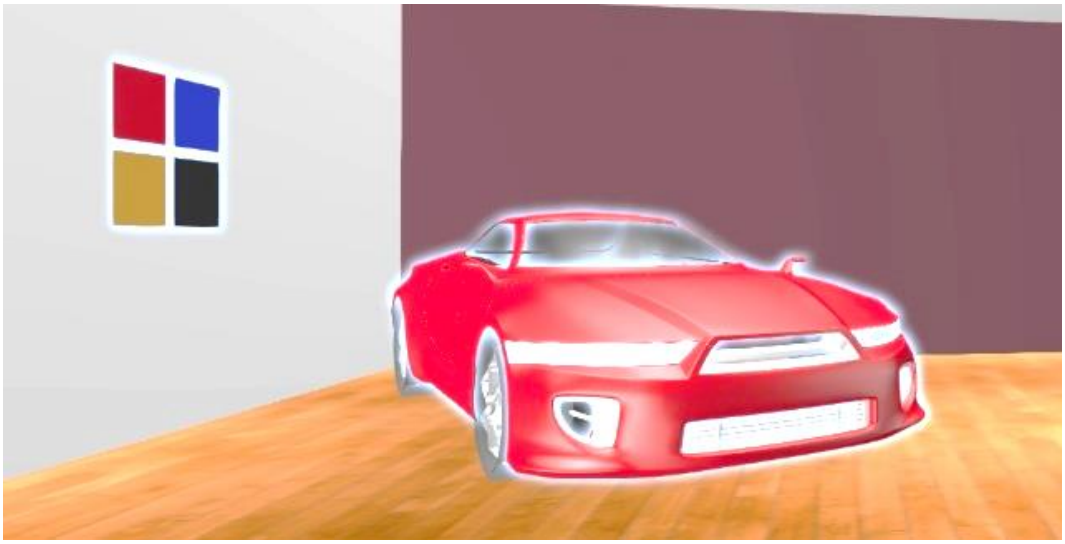


Figura 141 – Exemplo de aplicação do bloco *Inherit Material*. Ao clicar em um dos quadrados da paleta de cores (esquerda), seu material é instantaneamente aplicado à carenagem do carro.

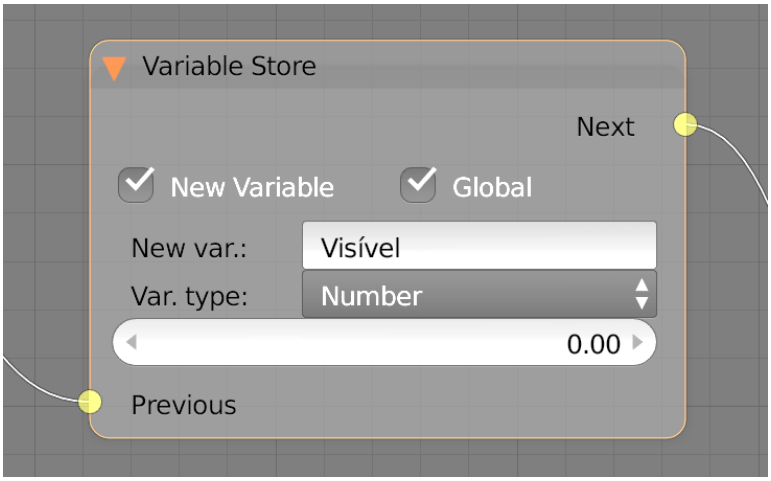


No submenu ***Operations*** (Operações em inglês) estão três blocos responsáveis pela realização de funções muito importantes para a construção de lógicas de programação mais elaboradas para as interações. Com eles pode-se realizar diferentes operações com valores numéricos e sequências de caracteres, bem como criar variáveis possíveis de serem utilizadas por outros blocos da linguagem.

- ***Variable Store*** (Armazenar Variável - Figura 142):

  - Pode ser utilizado para salvar um valor numérico (*Number*) ou uma sequência de caracteres (*String*) em uma variável. Por sua vez, os valores guardados em variáveis podem ser utilizados nos diferentes campos já vistos em outros blocos e em operações de cálculo;
  - Com esse bloco pode-se criar uma nova variável, ativando a opção *New variable* e definindo um nome a ela (*New var.*) ou alterar o valor armazenado em uma variável já existente;
  - No momento da criação de uma nova variável é possível ativar também a opção *Global*, fazendo com que a variável em questão seja acessível e possível de ser modificada em qualquer parte da programação dentro da Árvore de Nós;
  - No caso de já existirem variáveis criadas ao longo da lógica em desenvolvimento conectada a esse bloco, o campo *Var. name* é onde seleciona-se a que se deseja modificar;
  - Em *Var. type,* define-se o tipo de dado que será ou está guardado na variável. Podendo ser um valor numérico (*Number*) ou uma sequência de caracteres (*String*);
  - Já, o campo abaixo de *Var. type* é o local onde se determina o valor a ser assumido pela variável, quando a execução da programação passar por esse bloco.

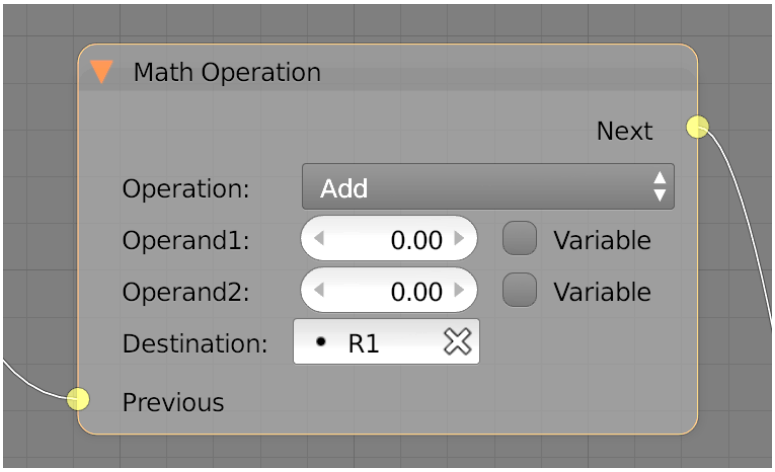
Figura 142 – Bloco *Variable Store* (Armazenar Variável).



- ***Math Operation*** (Operação Matemática - Figura 143):

  - Pode ser utilizado para executar uma operação matemática e armazenar o resultado em uma variável, previamente criada, do tipo *Number*;
  - Em *Operation,* seleciona-se o tipo de operação matemática a ser realizada com os operadores (*Operand1* e *Operand2*). São cinco as opções: *Random*, que gera um valor aleatório maior que o *Operand1* e menor que o *Operand2*; *Add*, que soma os operadores; *Multiply*, que multiplica os operadores; *Subtract*, que subtrai o *Operand2* do *Operand1*; *Divide*, que divide o *Operand1* pelo *Operand2*;
  - Tanto o *Operand1* como o *Operand2* podem ser valores numéricos ou uma variável (opção *Variable*);
  - Em *Destination,* escolhe-se uma variável já criada para armazenar o valor resultante da operação matemática realizada nesse bloco.

Figura 143 – Bloco *Math Operation* (Operação Matemática).



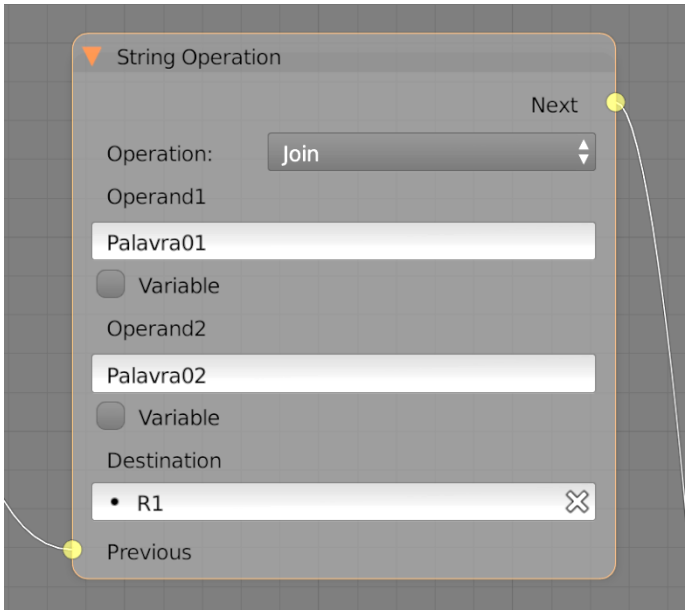
- ***String Operation*** (Operação de Sequências de Caracteres - Figura 144):

  - Pode ser utilizado para executar uma operação com duas sequências de caracteres (*Strings*) e armazenar o resultado em uma variável previamente criada;
  - Em *Operation* seleciona-se o tipo de operação a ser realizada com os operadores. Dependendo do tipo escolhido, algumas opções abaixo podem ser adicionadas ou alteradas. São cinco as opções de operações: *Join*, que une as sequências de caracteres dos dois

operadores; *Find*, que procura no *Operand2* a sequência de caracteres do *Operand1*, caso encontre gera o valor -1; *Replace*, que substitui o *Operand2* pela união do *Operand1* e do *Operand3*; *Split*, que divide a sequência do *Operand1* em duas, usando a sequência do *Operand2* como marca de divisão; *Compare*, que compara os operadores segundo uma condição determinada (Igual, Diferente, Menor que, Maior que, Menor ou Igual a, Maior ou Igual a), caso o resultado seja verdadeiro gera o valor 1 (um) como resultado, caso contrário resulta no valor 0 (zero);

- Qualquer operador pode ser tanto uma sequência de caracteres como uma variável (opção *Variable*);
- Em *Destination,* escolhe-se uma variável já criada para armazenar o valor resultante da operação de *Strings* realizada nesse bloco.

Figura 144 – Bloco *String Operation* (Operação de Sequências de Caracteres).



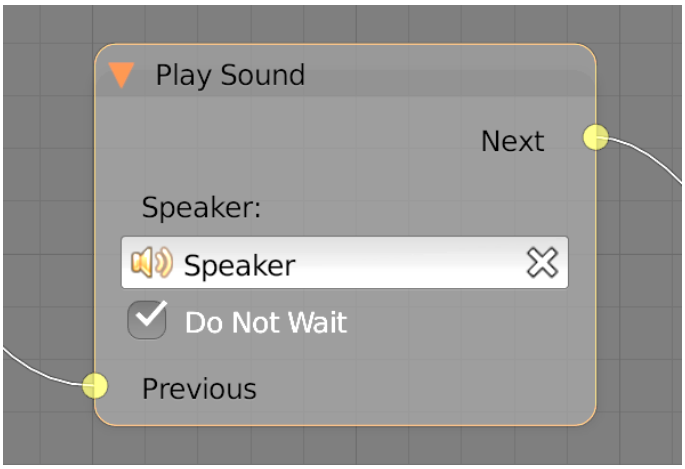
O submenu ***Sound*** (Som em inglês) engloba dois blocos responsáveis pelo controle da reprodução de sons atribuídos a elementos do tipo *Speaker* (Autofalante).

- ***Play Sound*** (Reproduzir Som - Figura 145):

  - Pode ser utilizado para iniciar a reprodução de um som atrelado a um autofalante específico;

- Em *Speaker,* seleciona-se o autofalante que será reproduzido;
- A opção *Do Not Wait*, quando ativada, determina que a execução passe para os blocos subsequentes, mesmo que a execução do som em questão ainda não tenha terminado e ainda o mantém reproduzido.

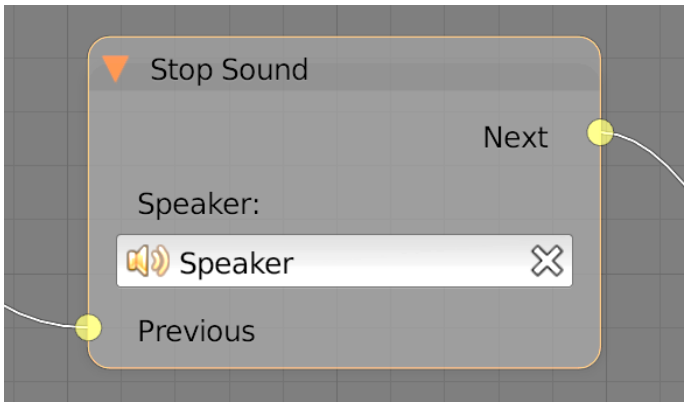
Figura 145 – Bloco *Play Sound* (Reproduzir Som).



- ***Stop Sound*** (Parar Som - Figura 146):

- Pode ser utilizado para interromper a reprodução de um som atrelado a um autofalante específico;
- Em *Speaker,* seleciona-se o autofalante que terá a reprodução interrompida.

Figura 146 – Bloco *Stop Sound* (Parar Som).



No submenu ***Network*** (Rede em inglês), existem blocos relacionados à obtenção de valores de páginas externas à cena e redirecionamentos para outros links, páginas web ou arquivos disponíveis dentro do próprio dispositivo visualizador. Apesar de serem quatro blocos, apenas um (*Page Redirect*) será explorado, por ser o único de acordo com os preceitos desta sistemática.

- ***Page Redirect*** (Redirecionamento de Página - Figura 147):

  - Pode ser utilizado para redirecionar o navegador web para outra página HTML ou arquivo presente tanto na internet como no próprio dispositivo visualizador. Esse bloco não possui nenhum parâmetro de saída e, portanto, marca o final de um caminho, já que o ambiente em execução será substituído por outra página;
  - Caso se pretenda redirecionar o navegador para uma página HTML ou um arquivo de qualquer outro tipo acessado pela internet, basta adicionar o endereço eletrônico no campo *Url*;
  - Já para o redirecionamento para arquivos presentes no dispositivo visualizador (Computador, Smartphone, Tablet), basta colocar no campo *Url* seu caminho completo (Ex.: D:\Pasta\Exemplo.pdf);
  - Esse bloco possibilita por exemplo, a criação de uma interação com um elemento na cena (Ex.: porta) que quando clicado leve até outro ambiente também desenvolvido a partir do Blender + Blend4Web. Essa técnica pode também ser utilizada como forma de otimização do processamento computacional necessário para a visualização da cena. Quando o ambiente desenvolvido conter muitos elementos e efeitos que o tornem difícil (“pesado”) de ser processado pelo *hardware*, pode ser dividido em mais de um ambiente conectados, utilizando esse bloco e aliviando assim a carga sobre ele;
  - Observa-se que utilizando caminhos de arquivos *off-line* do dispositivo e não endereços eletrônicos (sites), nem sempre o redirecionamento funciona quando se está na janela de testes do recurso *Fast Preview* e, por isso, recomenda-se realizar uma exportação em HTML (item 4.11) para testá-lo;
  - Ainda, o caminho Url desejado pode estar guardado em uma variável. Nesse caso, utiliza-se a opção *Variable URL* e seleciona-se a variável em questão.

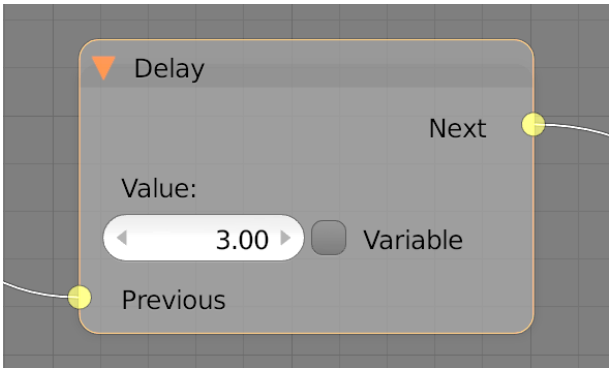
Figura 147 – Bloco *Page Redirect* (Redirecionamento de Página).

Já o submenu ***Time*** (Tempo em inglês) traz quatro blocos que pretendem controlar questões relacionadas ao tempo nas interações. Aqui é apresentado apenas um (*Delay*), enquanto os demais, com funções mais específicas, podem ser explorados pelo usuário por conta própria, caso deseje.

- ***Delay*** (Atraso - Figura 148):

  - Pode ser utilizado para realizar um atraso depois do bloco que o antecede e antes de ir para o próximo bloco;
  - O tempo desse atraso (medido em segundos) é o indicado em *Value* e pode ser substituído pelo valor de alguma variável, criada anteriormente com o bloco *Variable Store*, ativando a opção *Variable* e, depois, escolhendo a variável desejada.

Figura 148 – Bloco *Delay* (Atraso).



O submenu ***Debug*** (Depurar em inglês) possui atualmente apenas um bloco chamado Console Print.

- ***Console Print*** (Impressão de Console - Figura 149):

  - Pode ser utilizado para "imprimir" valores de variáveis em conjunto com um texto adicional (*Message*) na aba Console da janela de ferramentas de desenvolvedor do navegador (Figura 150);
  - Ele se destina a testes e análises da execução da programação. A partir dos valores impressos por esse bloco na aba *Console,* é possível saber o valor exato presente em determinada variável no instante em que a execução passar por ele;
  - Em *Message,* escreve-se a mensagem que se deseja que anteceda a impressão dos valores;
  - Abaixo de *Message,* escolhe-se a variável que contém o valor que se pretende imprimir. Para isso, ela deve ter sido criada anteriormente, caso contrário é exibida a mensagem *No var source* no local;
  - Pode-se ainda imprimir o valor de mais de uma variável ao mesmo tempo. Basta clicar no botão com o símbolo de um "+" (mais) e escolher, no campo que será criado, a variável que se deseja adicionar. Subtrai-se uma variável da lista clicando no botão "-" (menos) ao lado dela.

Figura 149 – Bloco *Console Print* (Impressão de Console).

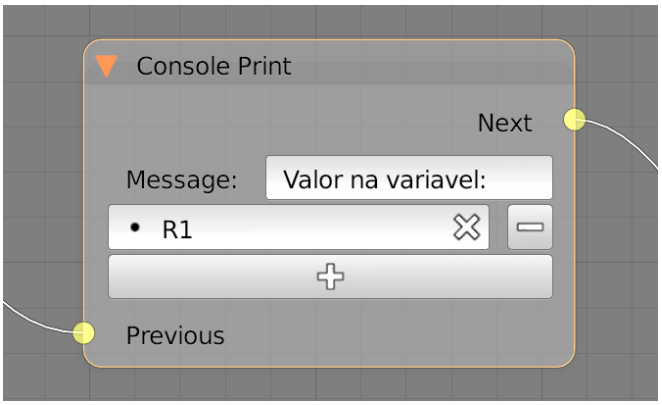
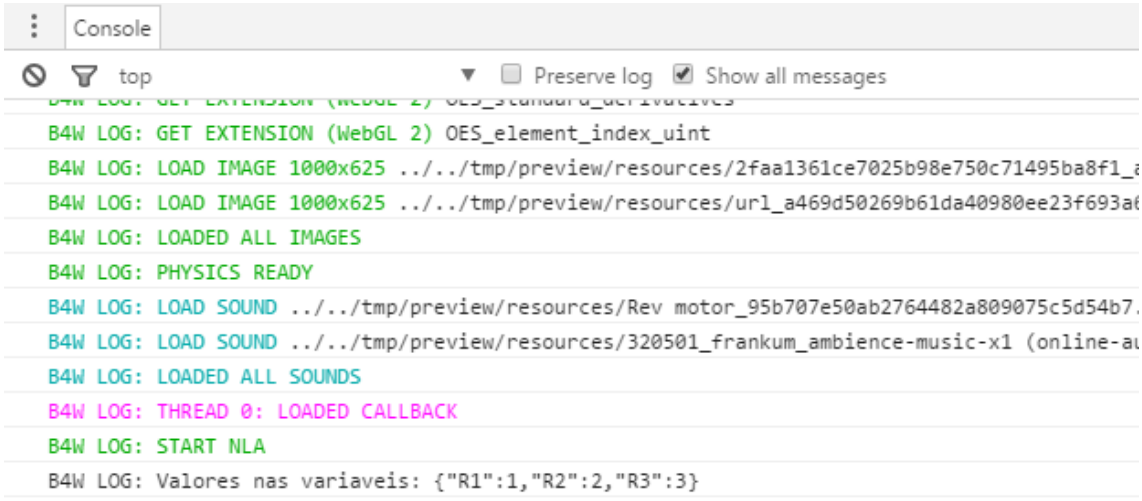


Figura 150 – Exemplo de mensagem impressa (última linha) na aba *Console* do navegador, utilizando o bloco *Console Print*.

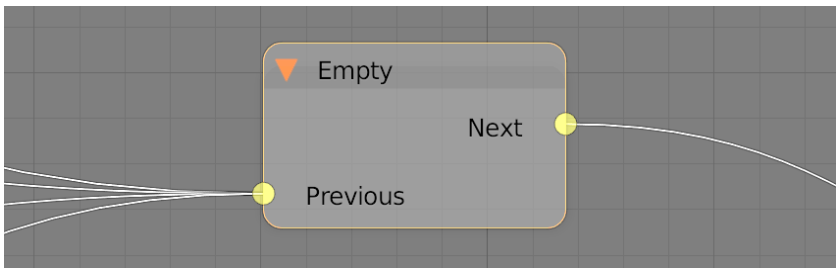


O último submenu, ***Layout*** (Leiaute em inglês), agrupa os blocos encarregados de auxiliar na organização da Árvore de Nós. Possui três blocos que podem ser utilizados para facilitar a leitura e o entendimento dos conjuntos de lógicas de programação.

- ***Empty*** (Vazio - Figura 151):

  - Não executa nenhuma operação por conta própria, sendo apenas um bloco de passagem simples. Pode ser utilizado para combinar vários segmentos de lógica (“fios”), que possuam o mesmo destino, em um único caminho.

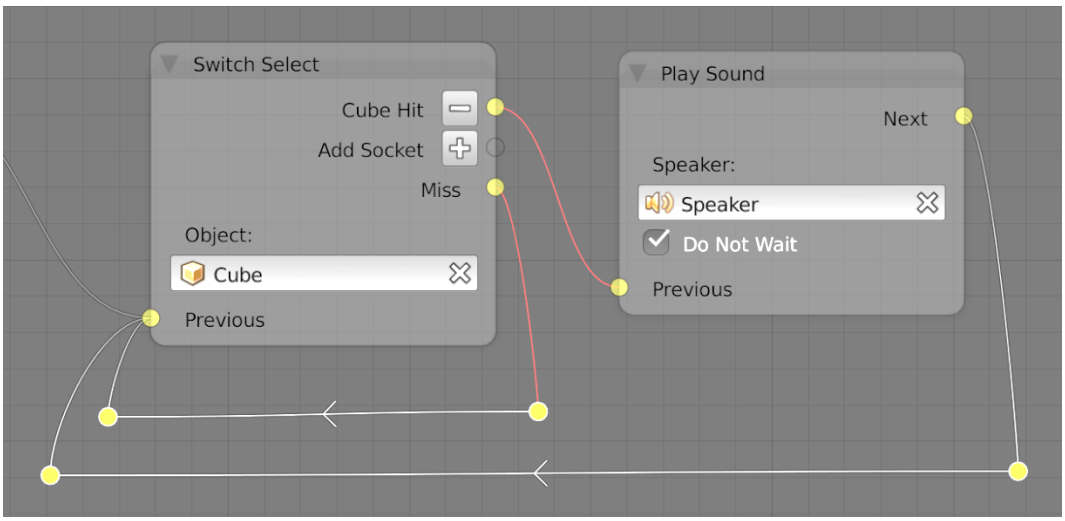
Figura 151 – Bloco *Empty* (Vazio).



- ***Reroute*** (Redirecionar - Figura 152):

  - Basicamente esses são pontos que direcionam a execução da lógica de programação para o ponto de entrada de outro bloco ou outro ponto *Reroute*;
  - Podem ser utilizados para criar estruturas cíclicas, como o exemplo dado na descrição do bloco *Switch Select* (Figura 128), ou para organizar os caminhos dos blocos, facilitando seu entendimento;
  - Após inseridos no espaço da Árvore de Nós, funcionam exatamente como os pontos de entrada e saída dos blocos, podendo ser unidos por "fios" de ligação;
  - No caso da utilização de mais de um ponto de redirecionamento para a construção de um caminho, os "fios" entre *Reroutes* apresentam setas indicando o sentido do fluxo da execução pela rota criada. Esse sentido é sempre de um ponto de saída até um ponto de entrada do mesmo bloco ou de outro.

Figura 152 – Exemplo de utilização de pontos *Reroute* no redirecionamento de caminhos.



- ***Frame*** (Quadro - Figura 153):

  - É basicamente um quadro redimensionável que possui um título personalizável centralizado no topo;
  - Pode ser utilizado como moldura para grupos de blocos (Figura 154), delimitando-os e identificando-os;

- Posiciona-se sempre atrás dos outros blocos e é possível redimensioná-lo de duas maneiras. A primeira, clicando em um de seus lados ou cantos e arrastando em qualquer direção, ampliando ou reduzindo o seu tamanho. E a segunda, arrastando os blocos desejados para dentro do Frame, fazendo com que ele os englobe e ajuste automaticamente o seu tamanho;
- Para movê-lo basta manter pressionado o botão direito do mouse sobre ele e, então, arrastá-lo para outro local;
- Quando o *Frame* estiver selecionado, o seu título pode ser alterado na propriedade *Label*, localizada no menu disposto à direita da janela *Node Editor* (Figura 155);
- É possível também atribuir um nome (não visível) ao *Frame*. Basta alterar o texto da propriedade *Name* no menu ao lado;
- Também há a possibilidade de atribuir uma cor a cada *Frame*. Isso o distingue visualmente um dos outros. Para isso, é necessário ativar a opção *Color* no menu à direita e, em seguida, escolher a cor desejada.

Figura 153 – Bloco *Frame* (Quadro).

Figura 154 – Exemplo de um *Frame* emoldurando um grupo de blocos responsáveis pelo controle de uma interação da cena.

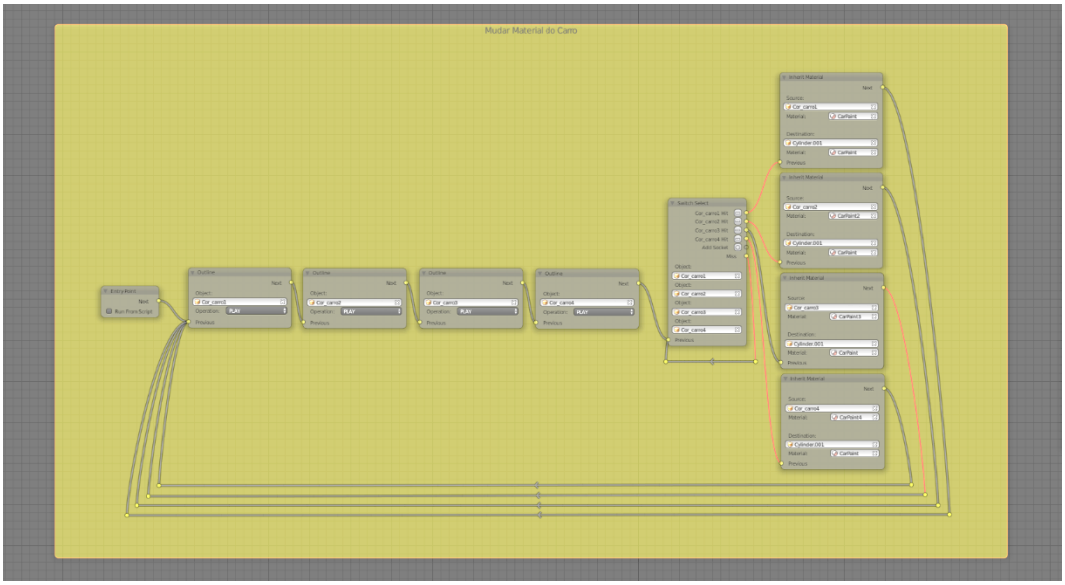
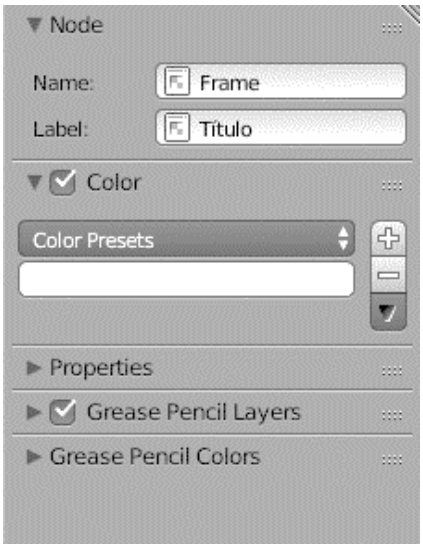


Figura 155 – Menu de propriedades do bloco *Frame*.

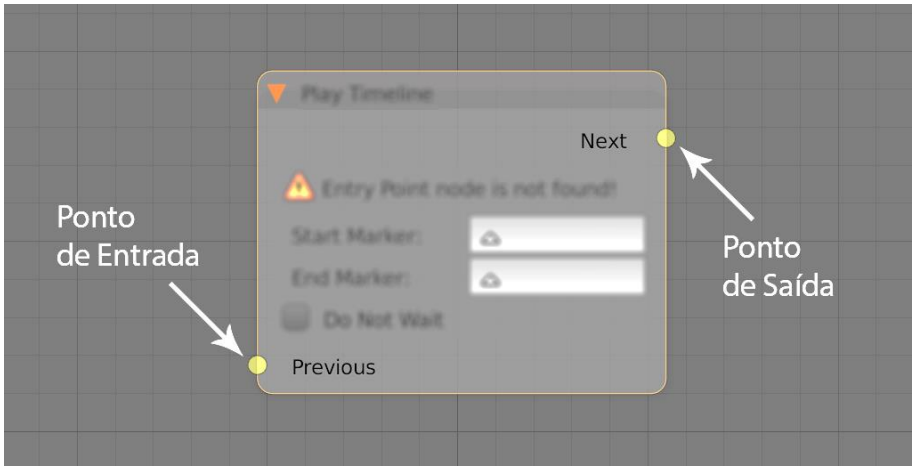


4.10.4. Construção de Interações

Conhecendo-se as funções de cada bloco da linguagem visual de programação do Blender + Blend4Web, é necessário compreender os princípios básicos de ligação entre eles para enfim construir as interações do ambiente virtual.

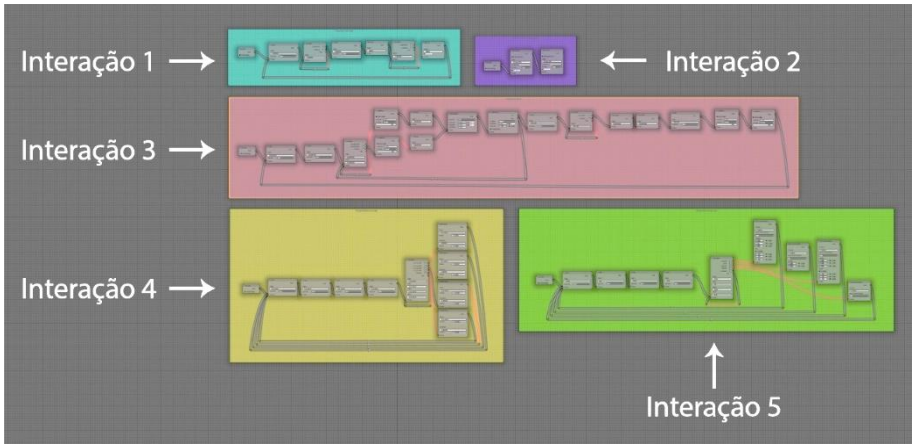
- **Ligações e Fluxo da Execução**: Como já mencionado, cada bloco tem um ponto de entrada (à esquerda) e/ou um ou mais pontos de saída (à direita) (Figura 156) e é através deles que os blocos recebem e enviam, respectivamente, informações uns para os outros. Por sua vez, as informações percorrem um caminho representado visualmente por "fios" que se conectam aos blocos justamente pelos pontos de entrada e saída. Para criá-los, basta clicar em um dos pontos e puxar o fio até outro. Sendo que nunca será conectado pontos de mesmo tipo (Ex.: entrada - entrada ou saída - saída) e a execução das programações criadas flui sempre dos pontos de saída para os de entrada;

Figura 156 – Indicação dos pontos de entrada e de saída de um bloco.



- **Múltiplos Conjuntos de Blocos**: Ressalta-se que é possível, dentro do mesmo espaço de construção, criar conjuntos de blocos totalmente separados uns dos outros. Cada um pode constituir uma interação do ambiente (Figura 157). Também é possível, porém, que os conjuntos se comuniquem em determinadas partes ou compartilhem variável globais, se assim for necessário;

Figura 157 – Exemplo de cinco conjunto de blocos, um para cada interação.



- **Necessidade de um Ponto Inicial (*Entry Point*)**: uma exigência da linguagem é que cada conjunto de blocos inicie em um *Entry Point* (Ponto Inicial) diferente, pois somente um fio pode partir de cada ponto inicial. Portanto, o *Entry Point* é essencial e marca o local onde começa cada interação, sendo executado automaticamente assim que o ambiente é carregado. Recomenda-se que esse seja o primeiro bloco a ser inserido no espaço de construção quando se inicia o desenvolvimento de uma nova interação;

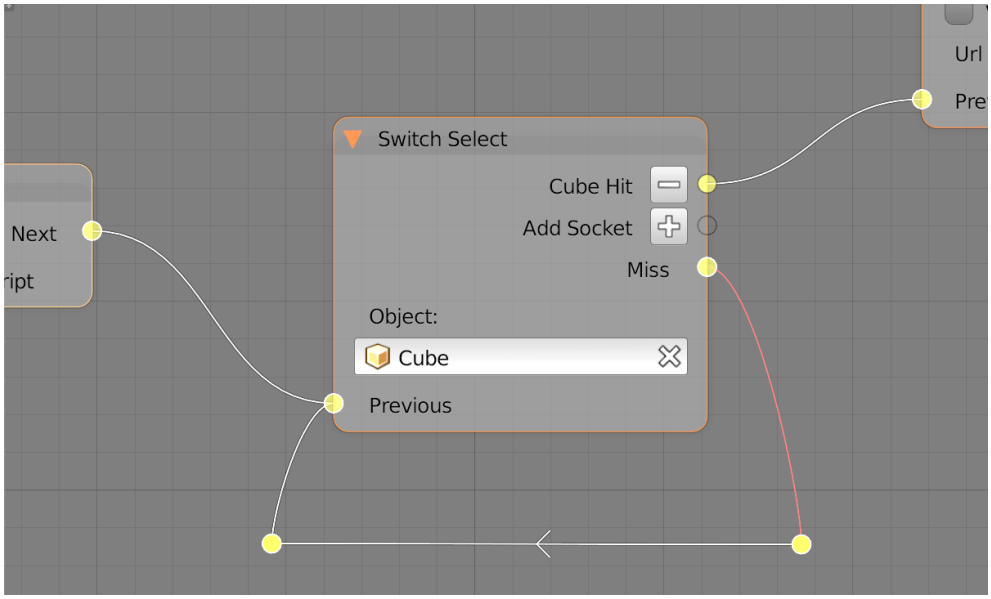
- **Redirecionamento Cíclico**: além da necessidade do Ponto Inicial, há uma técnica muito útil e, por vezes, necessária para a construção dos conjuntos de blocos, aqui chamada Redirecionamento Cíclico. Ela consiste na utilização dos pontos *Reroute* (Redirecionar) para criar um caminho de retorno de um ponto de saída até um de entrada que esteja localizado mais atrás, no mesmo bloco ou em blocos anteriores. Esse caminho contraria o sentido normal da execução (esquerda para direita), gerando um fluxo cíclico. Nota-se ainda que um único ponto de entrada aceita mais de uma ligação, o que não acontece com os de saída.

  Aplica-se essa técnica em duas distintas situações. A primeira, quando um bloco tem mais de um ponto de saída, como *Switch Select* (Interruptor de Seleção), por exemplo. Então, há duas ou mais possibilidades de seguimento para a execução da programação. Se uma das saídas ficar "aberta", sem blocos ligados em sua continuação, a execução pode travar e parar de funcionar, caso não seja possível avançar pelos demais pontos de saída disponíveis.

  Utilizando o exemplo do bloco *Switch Select*, tem-se ao menos um ponto de saída acionado pelo clique em um objeto especificado (*"Objeto" Hit*) e outro ponto de saída (*Miss*) por onde a execução flui quando o objeto em questão não é clicado. Quando a execução chega a esse bloco, em uma fração de segundo ele verifica se o objeto está sendo clicado e, caso não ocorra o clique dentro deste instante de verificação, o programa imediatamente dá prosseguimento no ponto de saída *Miss*. Nessa situação, se não houver nenhum caminho ligado ao ponto *Miss*, a execução dessa interação trava sem que seja possível seguir adiante ou reiniciá-la. Para resolver isso, é preciso criar, com a ajuda dos pontos *Reroute*, um caminho cíclico que conduza a execução do ponto de saída *Miss* até o ponto de entrada do mesmo bloco (Figura 158). Assim, enquanto não houver clique no objeto, o programa é

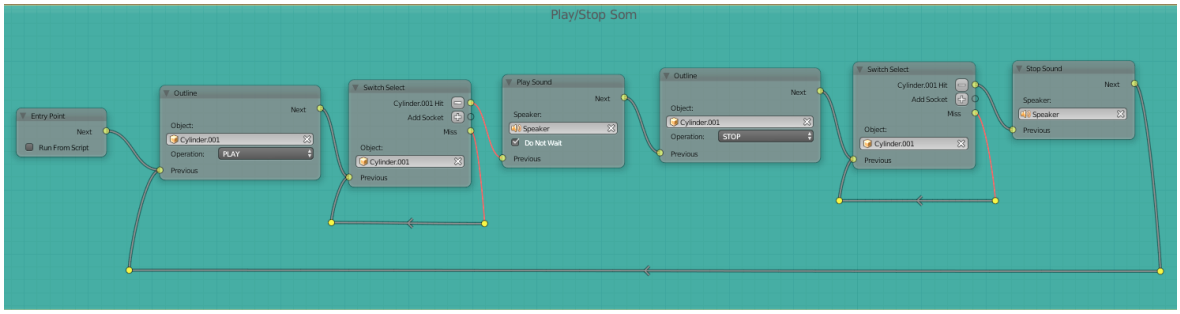
levado novamente ao início do bloco, realizando a verificação da condição continuamente.

Figura 158 – Exemplo de caminho cíclico no bloco *Switch Select*.



A segunda situação que exige a utilização da técnica de Redirecionamento Cíclico se dá, quando ao fim de um caminho de blocos, que constitui a lógica de uma interação ou em qualquer parte dele, deseje-se fazer com que a execução retorne a um bloco anterior. Incluindo a possibilidade de retorno até mesmo ao início do conjunto de blocos. Um exemplo dessa situação é quando uma interação pode ser realizada repetidas vezes. Para isso, cria-se um caminho cíclico do ponto de saída do último bloco até o primeiro ponto de entrada do conjunto (Figura 159). Caso esse ciclo não exista e o referido ponto de saída fique em aberto, a interação só é executada uma vez, permanecendo seus atores (objetos) no estado que assumiram após o último bloco.

Figura 159 – Exemplo de estrutura de blocos onde o caminho cíclico é utilizado do último ponto de saída até o primeiro ponto de entrada.



Como explicado na descrição do ponto *Reroute* (item 4.10.3), os fios entre dois deles apresentam uma seta indicando o sentido do fluxo da execução naquele caminho. Todos os caminhos, mesmo os sem seta indicativa, possuem sentido único, sempre de um ponto de saída para um de entrada. Outra observação importante diz respeito a fios que ficam vermelhos quando se cria um caminho cíclico. Isso pode ser desconsiderado desde que a programação esteja sendo executada corretamente. Nem sempre é um indicativo de que algo está errado;

- **Esboço das Interações**: para usuários com pouca experiência na linguagem de blocos, recomenda-se que antes de iniciar a inserção de blocos no programa o usuário esboce um simples esquema que represente todas as ações necessárias para que a interação aconteça. Essas ações nada mais são do que resultado da decomposição da interação em partes possíveis de serem representadas com os blocos apresentados. Para exemplificar, pode-se esquematizar a seguinte interação: iniciar a execução de um som ao clicar em um objeto e ao clicar nele novamente parar a execução desse som, adicionando ao objeto um contorno indicando que ele é clicável, enquanto o som estiver desligado, conforme a Figura 160. É possível verificar que, após a decomposição e estruturação da interação, cada pequena ação pode ser substituída por um bloco da linguagem de programação (Figura 161). Isso facilita a visualização do conjunto de blocos a ser construído (horizontalmente, de preferência) no programa.

Figura 160 – Esquema da interação de exemplo decomposta nas menores partes possíveis.

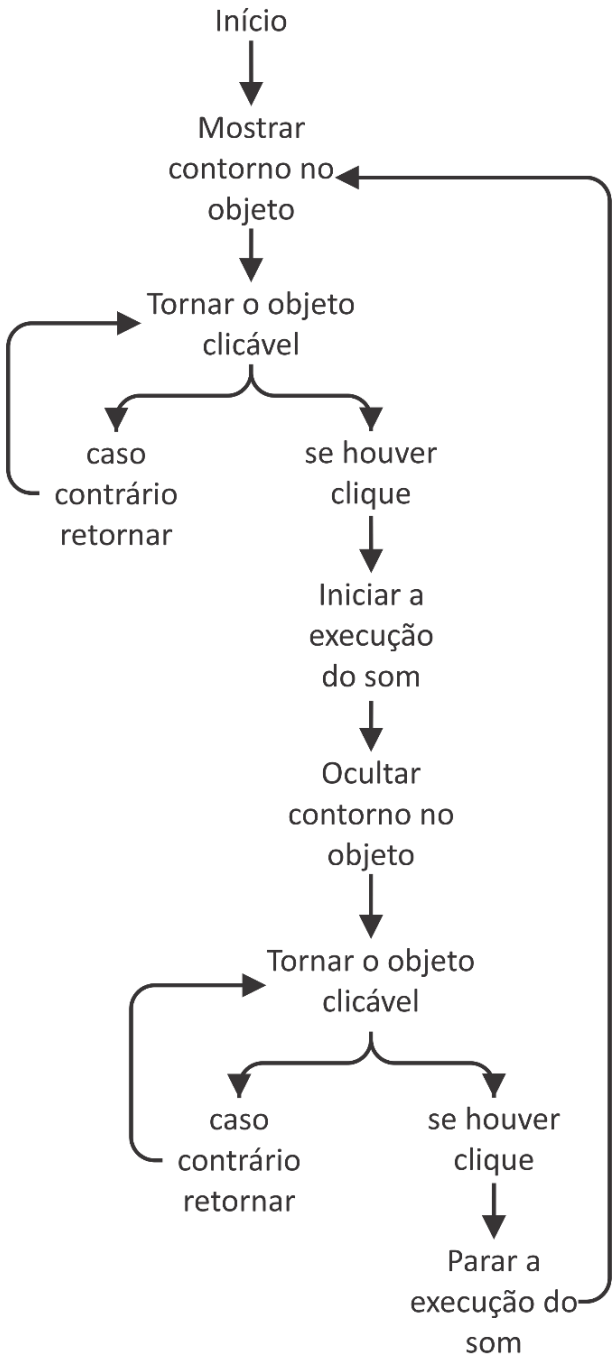
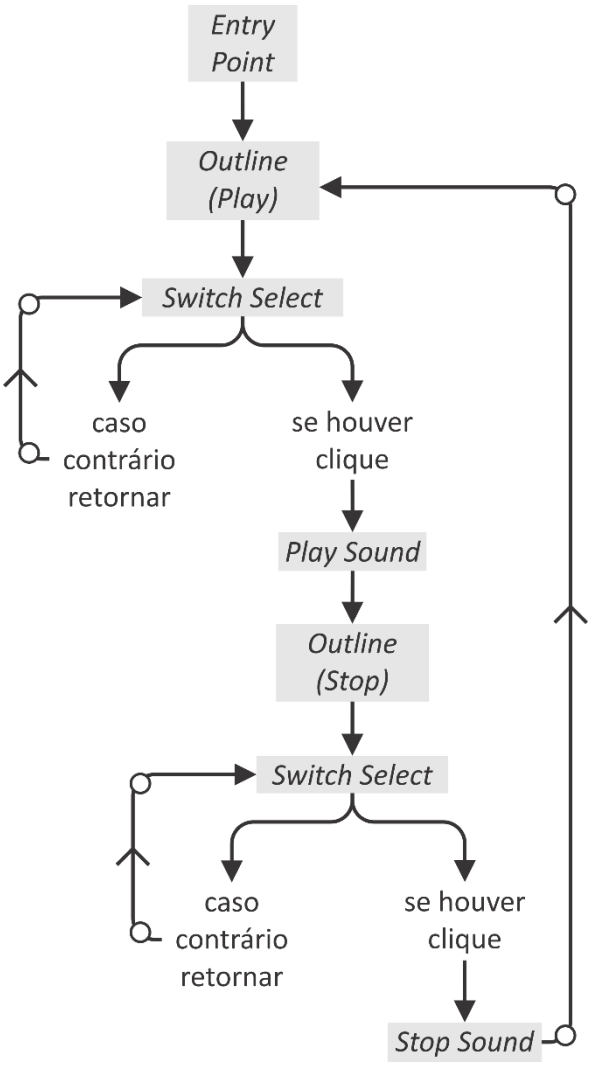


Figura 161 – Substituição das ações que compõem a interação de exemplo por blocos (em cinza) da linguagem visual de programação.



Como exemplo, é válido demonstrar ao menos uma das tantas aplicações das variáveis (*Variables* em inglês) na linguagem de blocos como forma de apresentação. A partir do exemplo dado, o usuário pode aprofundar-se e utilizá-las nos mais diversos arranjos.

Como explicitado na descrição do bloco *Variable Store* (Armazenar Variável), as variáveis criadas podem ser utilizadas na realização de operações matemáticas e em comparações entre valores. Como exemplo, tem-se a seguinte interação: acionar a animação de uma porta abrindo somente após o clique em dois interruptores distintos. Para ilustrá-la, utiliza-se o mesmo tipo de esquema de ações decompostas utilizado anteriormente (Figura 162). E, mais uma vez, substitui-se posteriormente cada ação por um bloco da linguagem

visual (Figura 163), construindo (horizontalmente, de preferência) o conjunto de blocos necessário para representar esta interação dentro do programa.

Figura 162 – Esquema das ações da segunda interação exemplo utilizando variáveis.

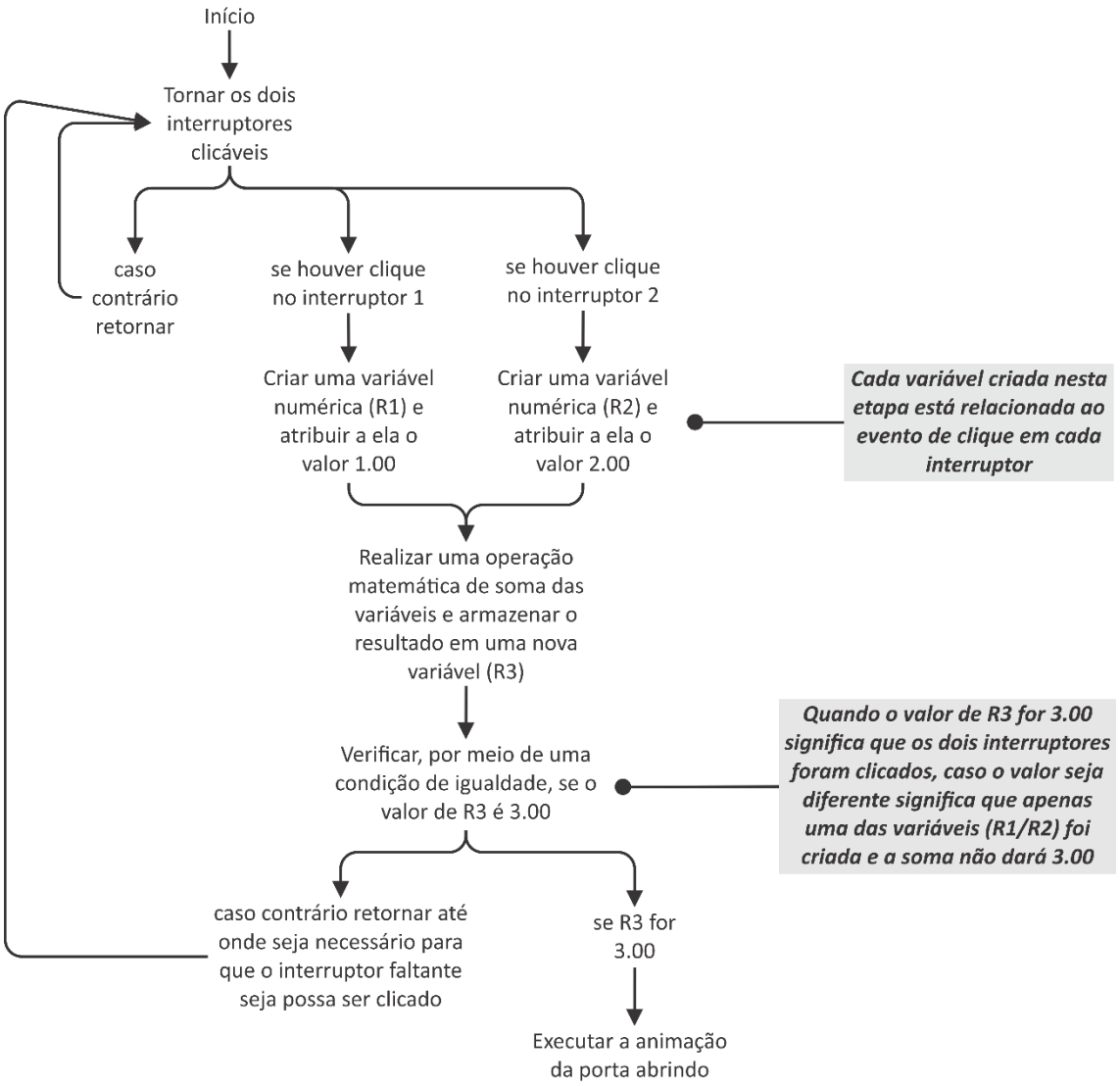
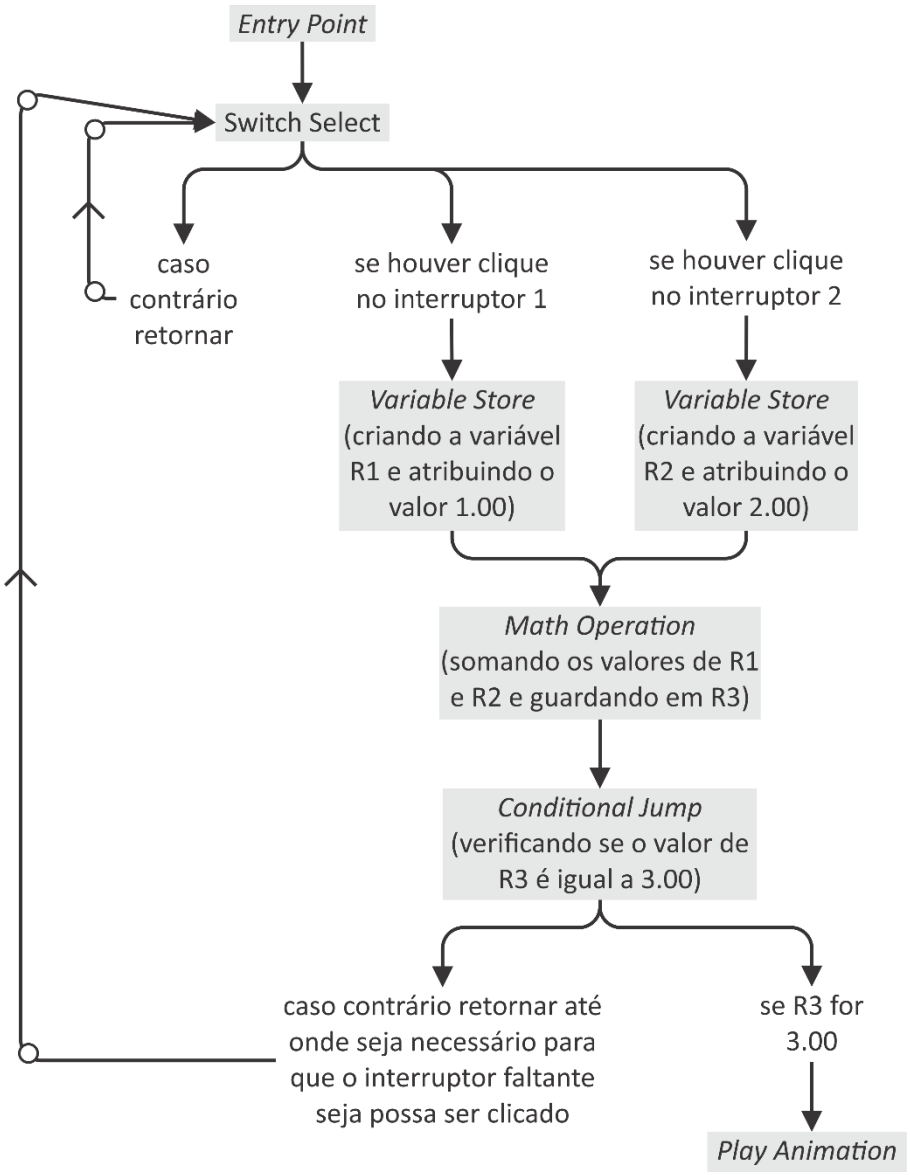


Figura 163 – Substituição das ações que compõem a segunda interação de exemplo por blocos (em cinza) da linguagem visual de programação.



4.11. EXPORTAÇÃO

A exportação do conteúdo criado com os programas Blender e Blend4Web para um arquivo HTML é a última etapa do processo de criação de ambientes tridimensionais virtuais utilizando a tecnologia WebGL. Ela pode ser utilizada somente após a conclusão do desenvolvimento do ambiente virtual tridimensional para se criar um arquivo final, porém recomenda-se que se realizem exportações regulares durante a criação do

ambiente, como forma de verificação e garantia de que todos os elementos se comportam como o planejado no arquivo em formato final.

O procedimento de exportação utilizado neste trabalho tem como objetivo converter o ambiente criado com os programas Blender e Blend4Web em um único arquivo no formato HTML. Por utilizar a tecnologia WebGL, esse arquivo é executável em navegadores web sem a necessidade de qualquer complemento (*plug-in*) ou de conexão ativa com a internet. Por isso, pode ser facilmente distribuído e visualizado em computadores, *tablets* e *smartphones*. Para realizar a exportação, basta seguir o caminho *Info > File > Export*, até a opção *Blend4Web (.html)* (Figura 164). Abrirá, então, uma nova janela para a escolha do nome no arquivo e do local no qual ele será salvo (Figura 165), restando apenas clicar no botão *B4W Export HTML*, no canto superior direito, para finalizar.

Figura 164 – Caminho até a opção de exportação em HTML.

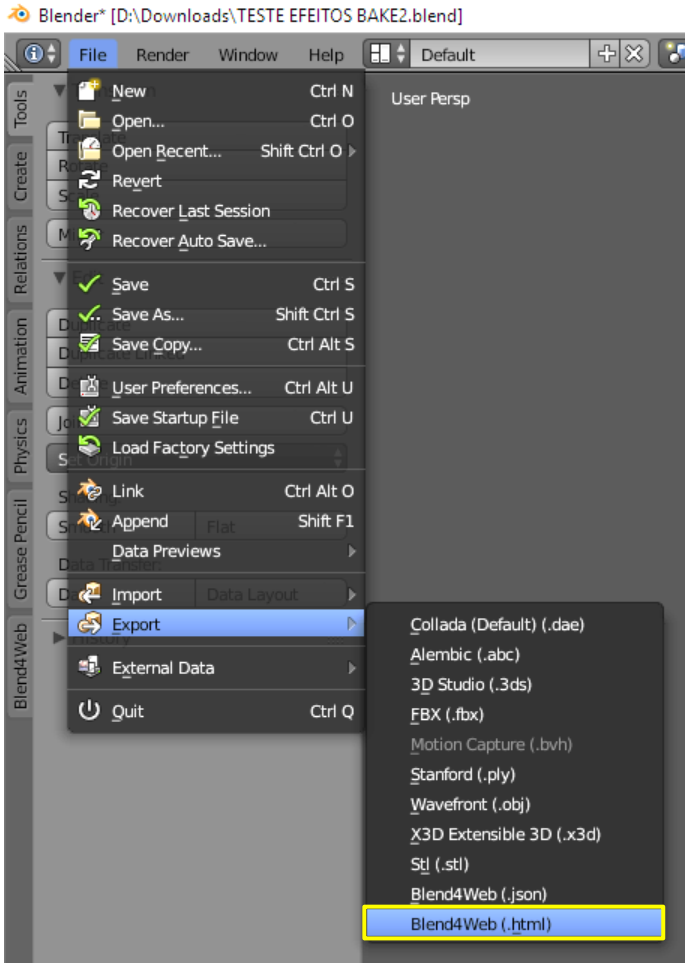
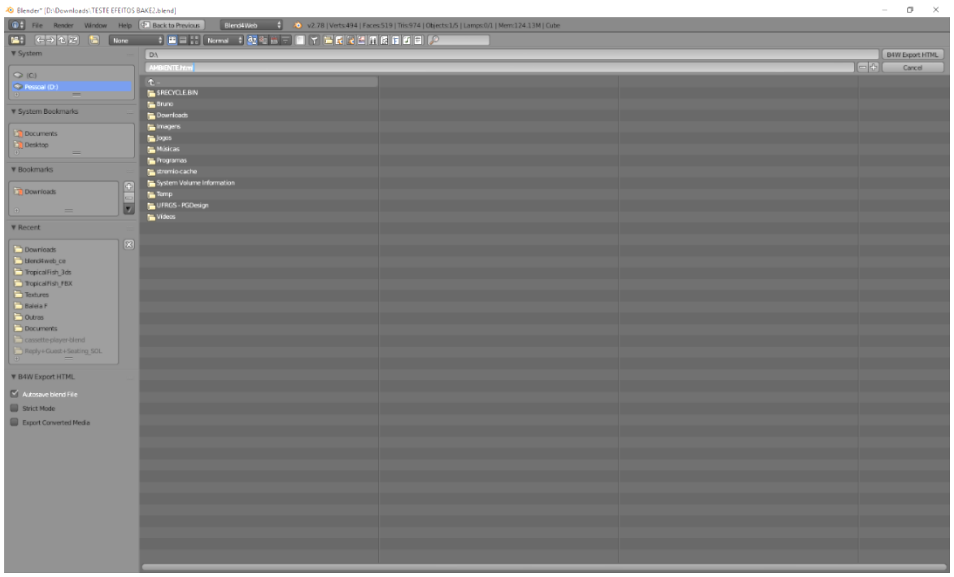
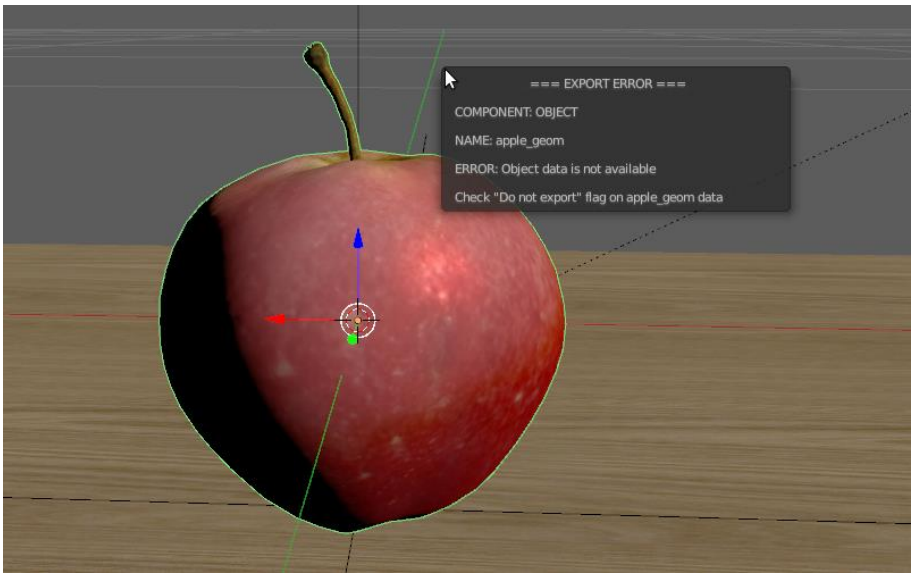


Figura 165 – Exemplo de tela apresentada para a escolha do nome do arquivo e do local para a exportação.



Estando todos os recursos corretamente configurados, conforme as informações da presente apostila, não haverá problemas nesse processo. Por isso, a importância de se utilizar com frequência o recurso *Fast Preview* e de se realizar exportações periódicas para evitar que existam erros que impeçam a exportação final. Caso algum recurso esteja mal ajustado, no entanto, uma mensagem de erro aparecerá na tela (Figura 166) com uma breve descrição do problema e um indicativo de qual elemento está causando o erro de exportação. É possível ainda obter a descrição detalhada do erro e sua provável causa acessando o manual online do software Blend4Web na seção *Add-on > Critical Export Errors* (https://www.blend4web.com/doc/en/addon.html#critical-export-errors).

Figura 166 – Exemplo de tela apresentando um erro encontrado durante a exportação.



Capítulo 5

RESULTADOS E TESTES

A fim de testar e comprovar a aplicabilidade das técnicas da sistemática proposta na criação de ambientes tridimensionais virtuais por designers, o autor deste trabalho desenvolveu ambientes experienciais utilizando todos os recursos demonstrados. Este capítulo apresenta a análise dos resultados obtidos a partir destes ambientes-teste e os desafios enfrentados durante a construção, que exigiram ajustes na exposição escrita de alguns processos.

Os ambientes-teste criados têm por propósito reunir todos os recursos do conjunto de programas Blender + Blend4Web, expostos neste trabalho. Para isso, foram concebidas uma cena interna (uma espécie de museu) e duas externas ao ar livre, uma de dia e outra à noite, interligadas e navegáveis entre si.

Ainda neste capítulo demonstra-se o atual potencial da tecnologia utilizada no neste trabalho comparando as mesmas cenas nas tecnologias WebGL e VRML. São discutidas as principais diferenças entre elas e os avanços em termos de qualidade visual e de interação.

Por fim retomam-se os requisitos que guiaram a escolha das ferramentas constituintes da sistemática proposta e verifica-se o cumprimento ou não dos mesmos conforme os resultados obtidos no teste experiencial. Esta verificação pretende validar a sistemática como ferramenta apta a ajudar na resolução do problema desta pesquisa.

5.1. AMBIENTES-TESTE

São apresentados a seguir os três ambientes criados utilizando a sistemática desenvolvida neste trabalho. Cada um concentrou um conjunto de recursos, listados e

ilustrados na sequência. Após isso são relatados os recursos comuns a todas as cenas criadas com os programas Blender e Blend4Web, que inserem a tecnologia WebGL nos atuais padrões de qualidade para ambientes voltados a execução por navegadores *web*.

Para utilizar o maior número de recursos possíveis, foram desenvolvidas cenas em diferentes condições. Um ambiente externo durante o dia (Figura 167), outro ambiente externo durante a noite (Figura 176) e um ambiente interno (Figura 183).

5.1.1. Ambiente externo durante o dia (Figura 167)

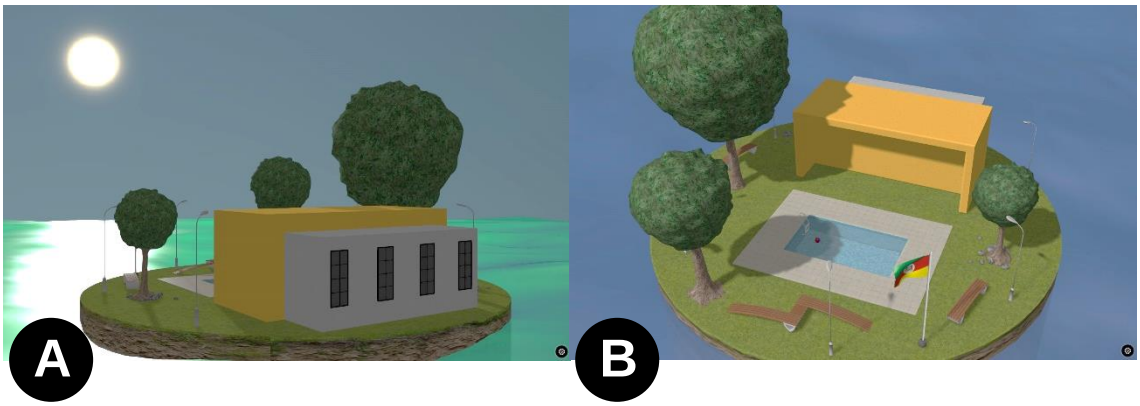
Este ambiente contempla recursos de variados tipos: Ambiente (Figura 168), Iluminação (Figura 168a), Material (Figura 169), Efeitos ao Ar Livre (Figura 170), Animação (Figura 171, Figura 172, Figura 173 e Figura 174), Interação (Figura 175), Câmera, Som e Otimização.

Figura 167 – Visão geral do ambiente externo durante o dia.

- **[Ambiente]** Céu procedural com representação visual do sol (Figura 168a);
- **[Iluminação]** Luz do tipo Sol na mesma posição da representação visual do sol no céu (Figura 168a);

- **[Ambiente]** Sombras projetadas pelos objetos a partir da luz do sol (Figura 168b);

Figura 168 – Detalhes do céu procedural (A) e das sombras projetadas pela luz do sol (B).

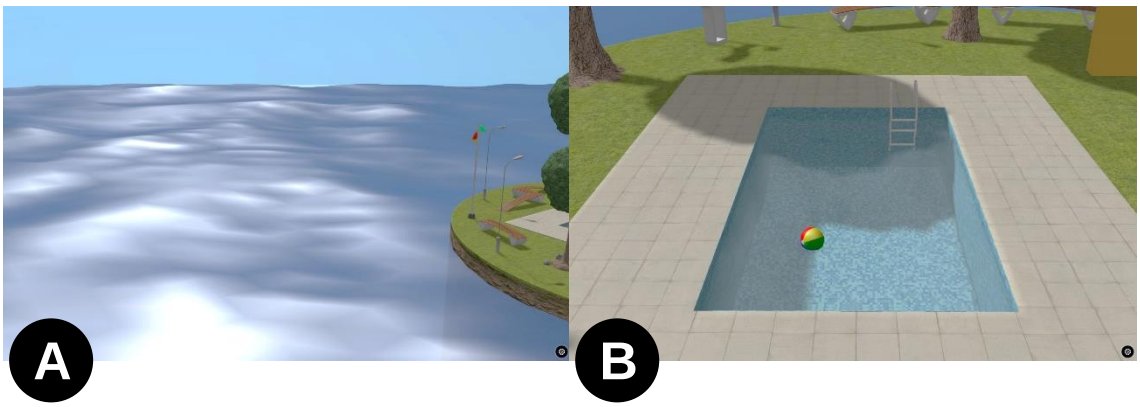


- **[Material]** Reflexos simples da iluminação em objetos com materiais mais reflexivos (oceano, água da piscina, bandeira) (Figura 169);

Figura 169 – Detalhes dos reflexos da luz do sol no oceano, na água da piscina (esquerda) e na bandeira (direita).

- **[Efeitos ao Ar Livre]** Simulação animada de oceano (Figura 170b);
- **[Efeitos ao Ar Livre]** Simulação estática de água na piscina com efeito de refração (Figura 170b);

Figura 170 – Detalhes do oceano (A) e da água da piscina (B).

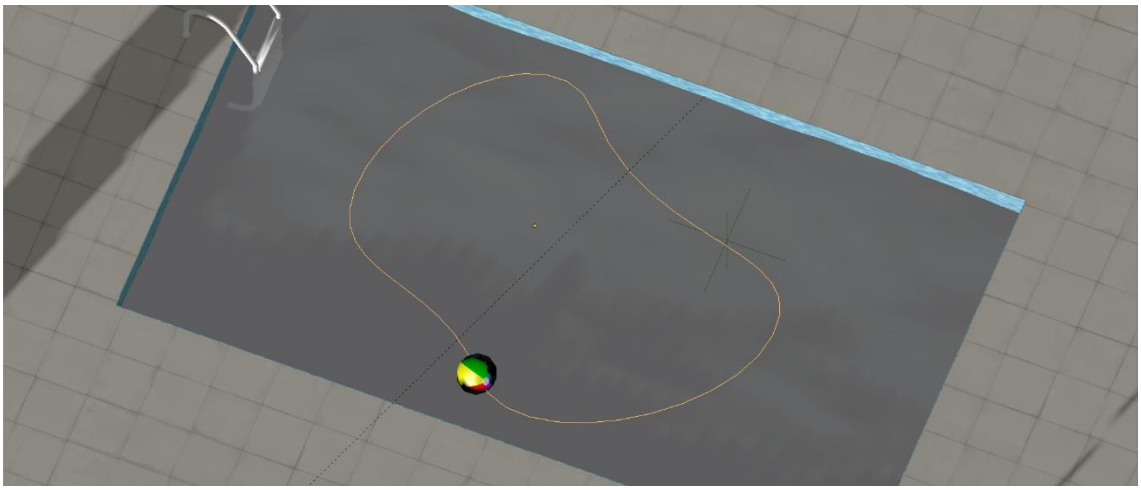


- **[Animação]** Animação de Vértices simulando uma bandeira do Rio Grande do Sul com vento forte (Figura 171);

Figura 171 – Três quadros da animação de vértices que simula o tecido de uma bandeira com vento forte.

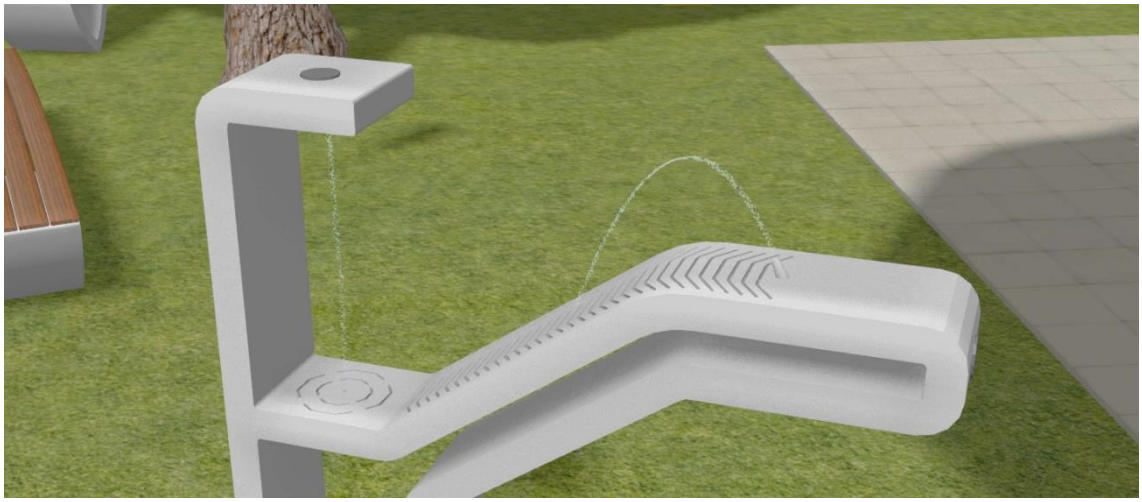
- **[Animação]** Animação de Seguimento de Caminho simulando uma bola flutuando na água da piscina (Figura 172);

Figura 172 –Visualização da bola e do caminho (curva) da animação ainda no modo de edição.



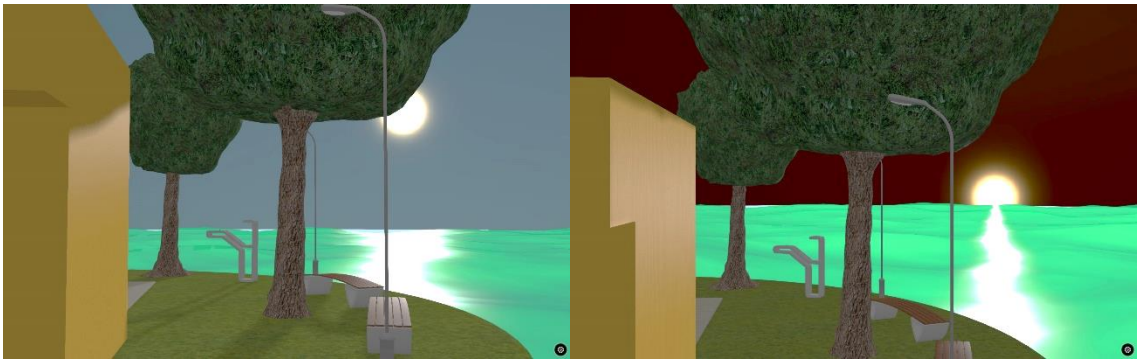
- **[Animação]** Animação de Emissão de Partículas simulando a água que sai do bebedouro (Figura 173);

Figura 173 – Emissão de partículas simulando a água que sai do bebedouro.



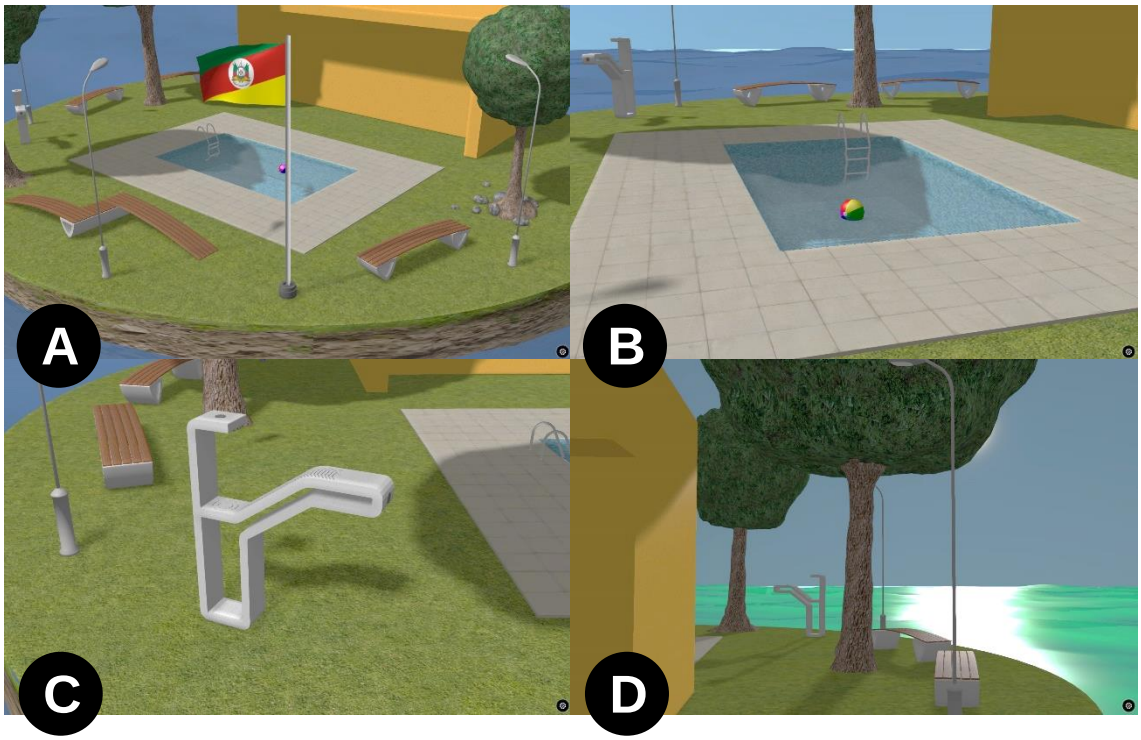
- **[Animação]** Animação aplicada à luz do tipo Sol, simulando o pôr do sol (Figura 174);

Figura 174 – Dois quadros da animação de pôr do sol.



- **[Interação]** Simulação de múltiplas câmeras através de diferentes movimentações pré-configuradas da câmera principal, acionadas por cliques em objetos específicos (bandeira, água da piscina, bebedouro, banco) (Figura 175);

Figura 175 – Visão das quatro posições da câmera quando são clicados os objetos: bandeira (A), água da piscina (B), bebedouro (C), banco (D).

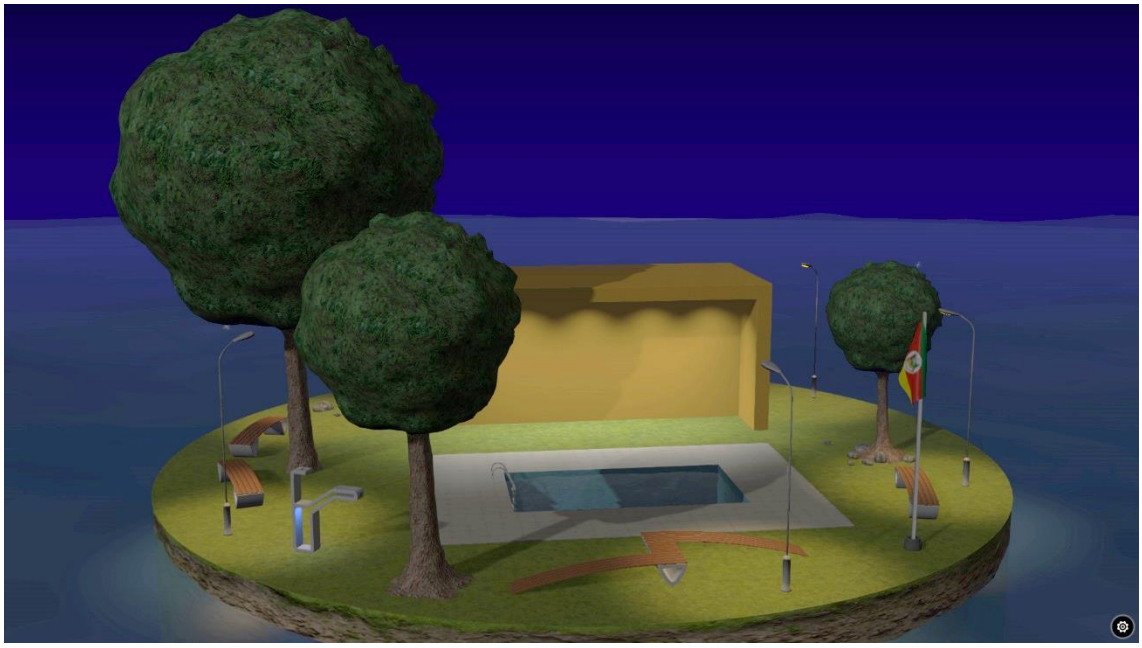


- **[Material]** Diversas texturas aplicadas aos objetos (árvores, gramado, solo, piso, bancos, bandeira, piscina, pedras);
- **[Efeitos ao Ar Livre]** Representação do movimento ocasionado pelo vento nas árvores (*Wind Bending*);
- **[Câmera]** Câmera com estilo de movimentação do tipo *Target* e com ângulos de movimentação limitados;
- **[Som]** Som de pássaros do tipo posicional sendo emitido da árvore de maior tamanho, onde tem maior intensidade;
- **[Otimização]** Cópias de objetos idênticos utilizando a Duplicação Ligada (árvores, bancos, postes, pedras);
- **[Otimização]** Suavização das malhas dos objetos (*Smooth*);
- **[Interação]** Acionamento da animação do pôr do sol através de um gatilho/botão (banco), movimentando também a câmera para uma melhor visualização;
- **[Interação]** Acionamento das saídas de água do bebedouro por meio do clique em botões;
- **[Interação]** Redirecionamento para o ambiente interno com um clique na porta do edifício.

5.1.2. Ambiente externo durante a noite (Figura 176)

O ambiente noturno é composto por recursos de: Ambiente (Figura 177 e Figura 178), Efeitos ao Ar Livre (Figura 177, Figura 178 e Figura 179), Objeto (Figura 177), Material (Figura 179), Animação (Figura 180 e Figura 181), Interação (Figura 182), Iluminação, Câmera, Som e Otimização.

Figura 176 – Visão geral do ambiente externo durante a noite.



- **[Ambiente]** Céu simples com degradê (Figura 177);
- **[Efeitos ao Ar Livre]** Simulação de estrelas no céu (Figura 177);
- **[Objeto]** Representação da lua com uma esfera posicionada na mesma direção da fonte de luz principal (Figura 177);

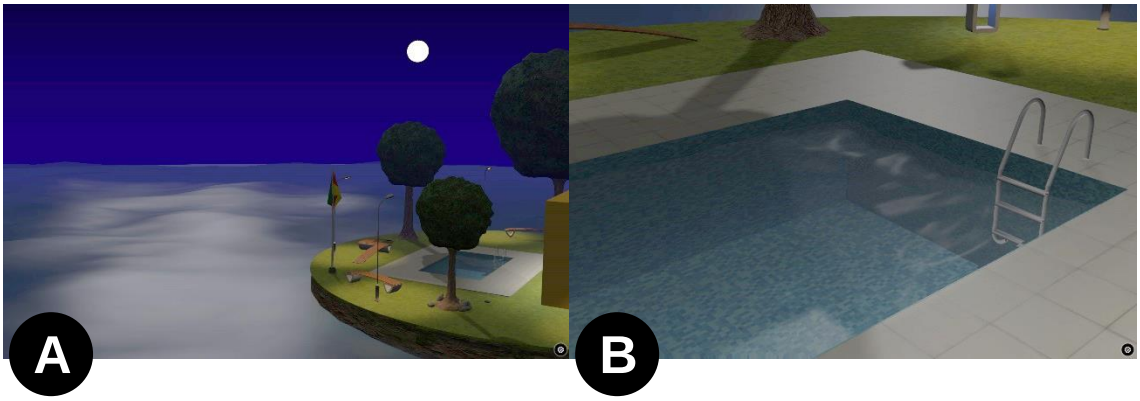
Figura 177 – Detalhe do céu em degradê com as estrelas e a lua.

- **[Efeitos ao Ar Livre]** Leve efeito de neblina que se intensifica na medida em que a câmera se afasta da cena (Figura 178);
- **[Ambiente]** Sombras projetadas pelos objetos a partir da luz da lua (Figura 178);

Figura 178 – Detalhes do efeito de neblina, que fica mais ou menos densa conforme a distância entra a câmera e o centro do cenário; sombras projetadas pela luz da lua.

- **[Material]** Reflexos simples da iluminação em objetos com materiais mais reflexivos (oceano, água da piscina) (Figura 179);
- **[Efeitos ao Ar Livre]** Simulação animada de oceano (Figura 179a);
- **[Efeitos ao Ar Livre]** Simulação estática de água na piscina com efeito de refração (Figura 179b);

Figura 179 – Detalhes dos reflexos das luzes no oceano (A) e na água da piscina (B).

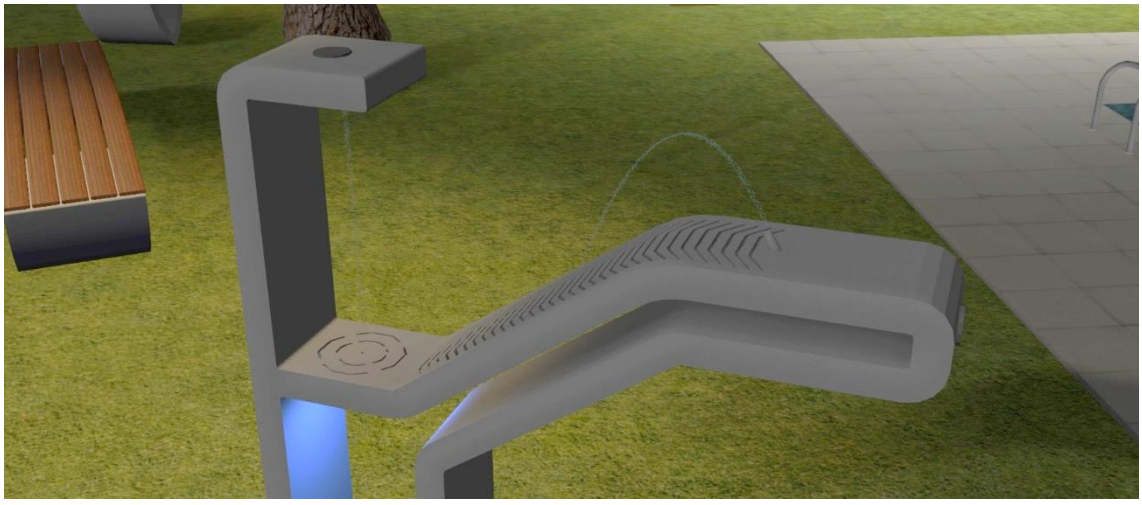


- **[Animação]** Animação de Vértices simulando uma bandeira do Rio Grande do Sul com vento fraco (Figura 180);

Figura 180 – Dois quadros da animação de vértices, que simula o tecido de uma bandeira com vento fraco.

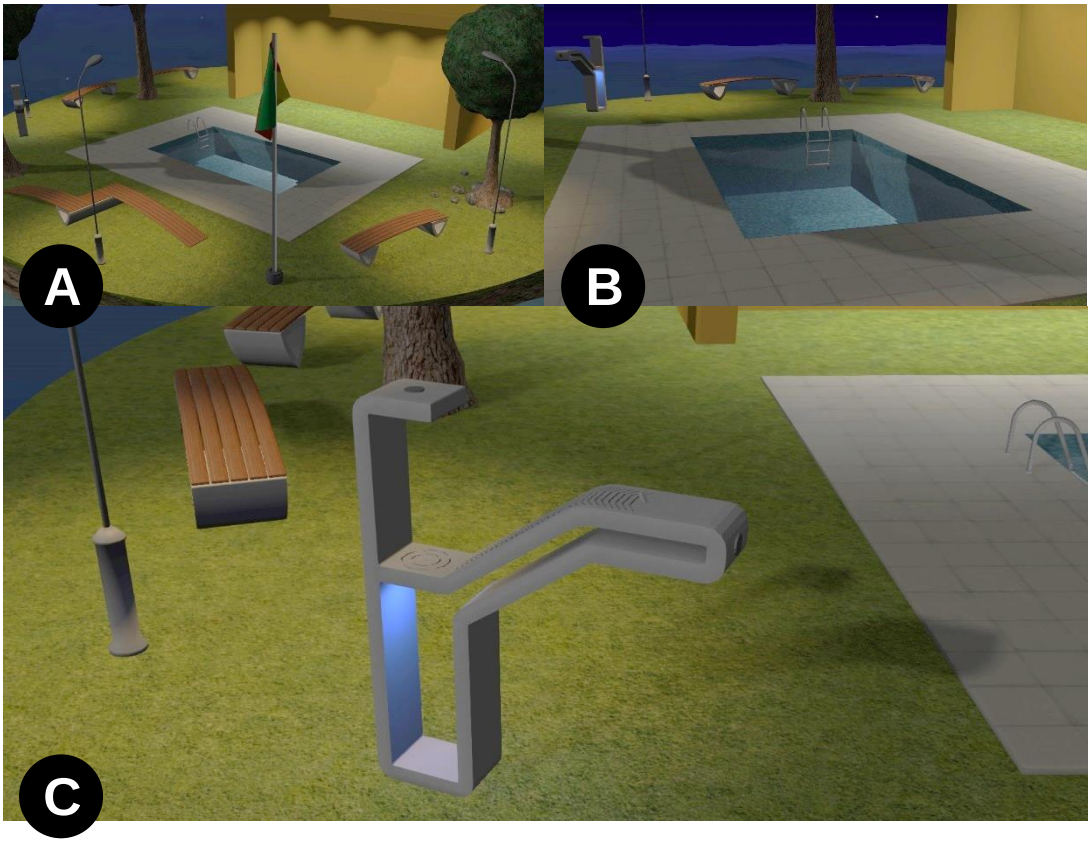
- **[Animação]** Animação de Emissão de Partículas simulando a água que sai do bebedouro (Figura 181);

Figura 181 – Emissão de partículas simulando a água que sai do bebedouro.



- **[Interação]** Simulação de múltiplas câmeras através de diferentes movimentações pré-configuradas da câmera principal, acionadas por cliques em objetos específicos (bandeira, água da piscina, bebedouro) (Figura 182);

Figura 182 – Visão das três posições da câmera quando são clicados os objetos: bandeira (A), água da piscina (B), bebedouro (C).



- **[Material]** Diversas texturas aplicadas aos objetos (árvores, gramado, solo, piso, bancos, bandeira, piscina, pedras);
- **[Iluminação]** Diferentes tipos e fontes de iluminação (lua, postes, spots no edifício, bebedouro);
- **[Efeitos ao Ar Livre]** Representação do movimento ocasionado pelo vento nas árvores (*Wind Bending*);
- **[Câmera]** Câmera com estilo de movimentação do tipo *Target* e com ângulos de movimentação limitados;
- **[Som]** Som de grilos do tipo posicional, emitido da árvore de maior tamanho, onde tem maior intensidade;

- **[Otimização]** Cópias de objetos idênticos utilizando a Duplicação Ligada (árvores, bancos, postes, pedras);
- **[Otimização]** Suavização das malhas dos objetos (*Smooth*);
- **[Interação]** Acionamento das saídas de água do bebedouro por meio do clique em botões;
- **[Interação]** Apagar das luzes do edifício quando as luzes de todos os postes estiverem apagadas;
- **[Interação]** Apagar e acender das luzes de cada poste individualmente com um clique neles;
- **[Interação]** Redirecionamento para o ambiente interno com um clique na porta do edifício.

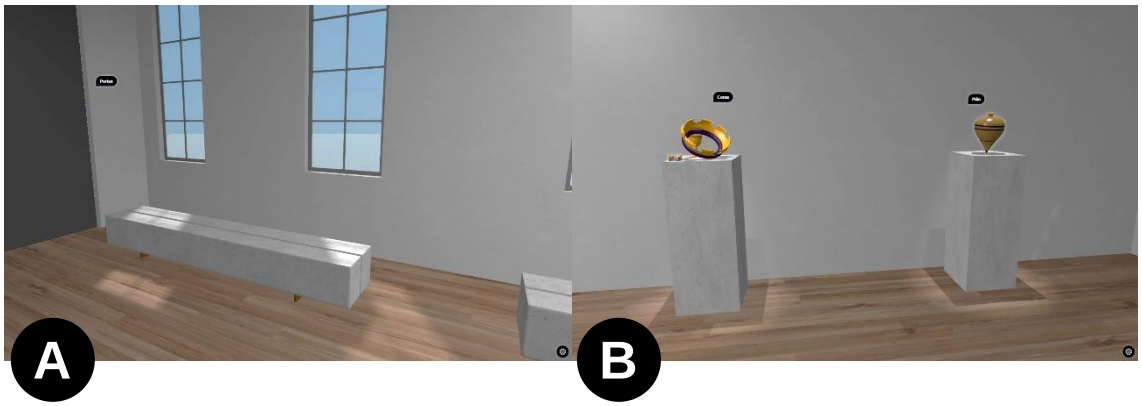
5.1.3. Ambiente interno (Figura 183)

O recursos presentes no ambiente interno são divididos nas seguintes categorias: Iluminação (Figura 184), Ambiente (Figura 185), Objeto (Figura 186), Material (Figura 187, Figura 188 e Figura 189), Animação (Figura 189), Notas (Figura 190), Câmera, Som, Otimização e Interação.

Figura 183 – Visão geral do ambiente interno.

- **[Iluminação]** Diferentes tipos e fontes de iluminação (sol/janelas, spots, lâmpadas do teto) (Figura 184);

Figura 184 – Luzes provenientes do sol (A) e dos spots acima dos expositores (B).



- **[Ambiente]** Sombras projetadas pelos objetos (Figura 185);

Figura 185 – Sombras projetadas pelos objetos em um dos expositores.

- **[Objeto]** Objetos com geometrias mais detalhadas (mulher, objeto de museu digitalizado) (Figura 186);

Figura 186 – Objetos com um número significativamente maior de polígonos em relação aos demais da cena.

- **[Material]** Textura de rugosidade aplicada ao objeto de museu digitalizado (Figura 187);

Figura 187 – Detalhe da textura de rugosidade aplicada ao objeto de museu digitalizado.

- **[Material]** Objetos (coroa, taça) altamente reflexivos, que projetam a imagem de outros objetos (dados, expositor) (Figura 188);
- **[Material]** Reflexos simples da iluminação em certos objetos (pião) (Figura 189);

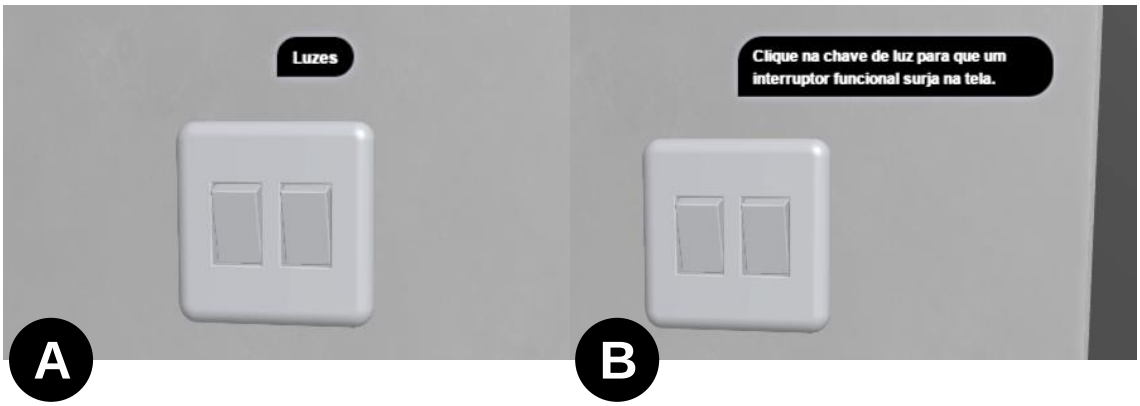
- **[Animação]** Contorno, aplicado a alguns objetos, que pisca indicando a presença de uma interação em determinado local (*Object Outlining*) (Figura 189);

Figura 188 – Coroa e taça refletindo objetos próximos (dados e expositor), devido aos materiais altamente reflexivos.

Figura 189 – Reflexos da iluminação em objetos como o pião; contorno, no pião, indicando a presença de interação.

- **[Notas]** Diversas anotações com título e descrição, indicando recursos e interações presentes na cena (Figura 190);

Figura 190 – Anotação com título (A) e descrição (B), explicando uma das interações da cena.



- **[Ambiente]** Céu procedural (visto pela janela);
- **[Material]** Diversas texturas aplicadas aos objetos (mulher, piso, rebaixo, paredes, quadro, pião, bancos, objeto de museu digitalizado);
- **[Câmera]** Câmera com estilo de movimentação do tipo *Hover* e com ângulos de movimentação limitados;
- **[Som]** Som (música) do tipo ambiente emitido com a mesma intensidade em todo o ambiente;
- **[Otimização]** Cópias de objetos idênticos utilizando a Duplicação Ligada (expositores, bancos, janelas, spots, portas);
- **[Otimização]** Suavização das malhas dos objetos (*Smooth*);
- **[Animação]** Animação de Objeto simulando o movimento de giro do pião;
- **[Interação]** Acionamento das animações de abertura das portas ao clicar nelas e consequente redirecionamento para o ambiente externo, durante o dia, após clicar nas duas portas;
- **[Interação]** Acionamento e parada da animação do pião através de cliques nele;
- **[Interação]** Interruptor de luzes funcional que surge frente à câmera ao clicar no interruptor posicionado ao lado da porta. Ele controla o acender e o apagar das luzes principais da sala e dos spots direcionados para os expositores;
- **[Interação]** Substituição do material da coroa ao clicar nos dados próximos a ela.

5.2. COMPARAÇÃO WEBGL E VRML

Analisando os ambientes-teste que utilizam o WebGL em conjunto com os mesmos ambientes no formato da tecnologia VRML, de imediato se nota a primeira limitação do VRML em relação ao WebGL. Ele exige a instalação de um plug-in no computador em que será executado, enquanto o WebGL funciona nos principais navegadores do mercado sem nenhum outro complemento. Esta limitação também impossibilita a execução de arquivos ".wrl" em dispositivos móveis, a não ser que estes possuam aplicativos específicos para este fim. Tudo isso torna mais difícil a distribuição das cenas VRML em relação ao WebGL que pode ser executado em uma gama maior de dispositivos sem grandes dificuldades.

Os ambientes Interno (Figura 191), Externo-Dia (Figura 192) e Externo-Noite (Figura 193), quando comparados os diferentes arquivos e tecnologias, apresentaram enormes discrepâncias em termos de qualidade de representação gráfica. Sendo clara, praticamente em todos os aspectos, a superioridade dos arquivos WebGL, gerado com o auxílio da sistemática elaborada neste trabalho, em relação aos outros ambientes VRML.

Figura 191 – Visão geral do Ambiente Interno nos três diferentes formatos: (a) WebGL, (b) VRML (3ds Max), (c) VRML (Blender).

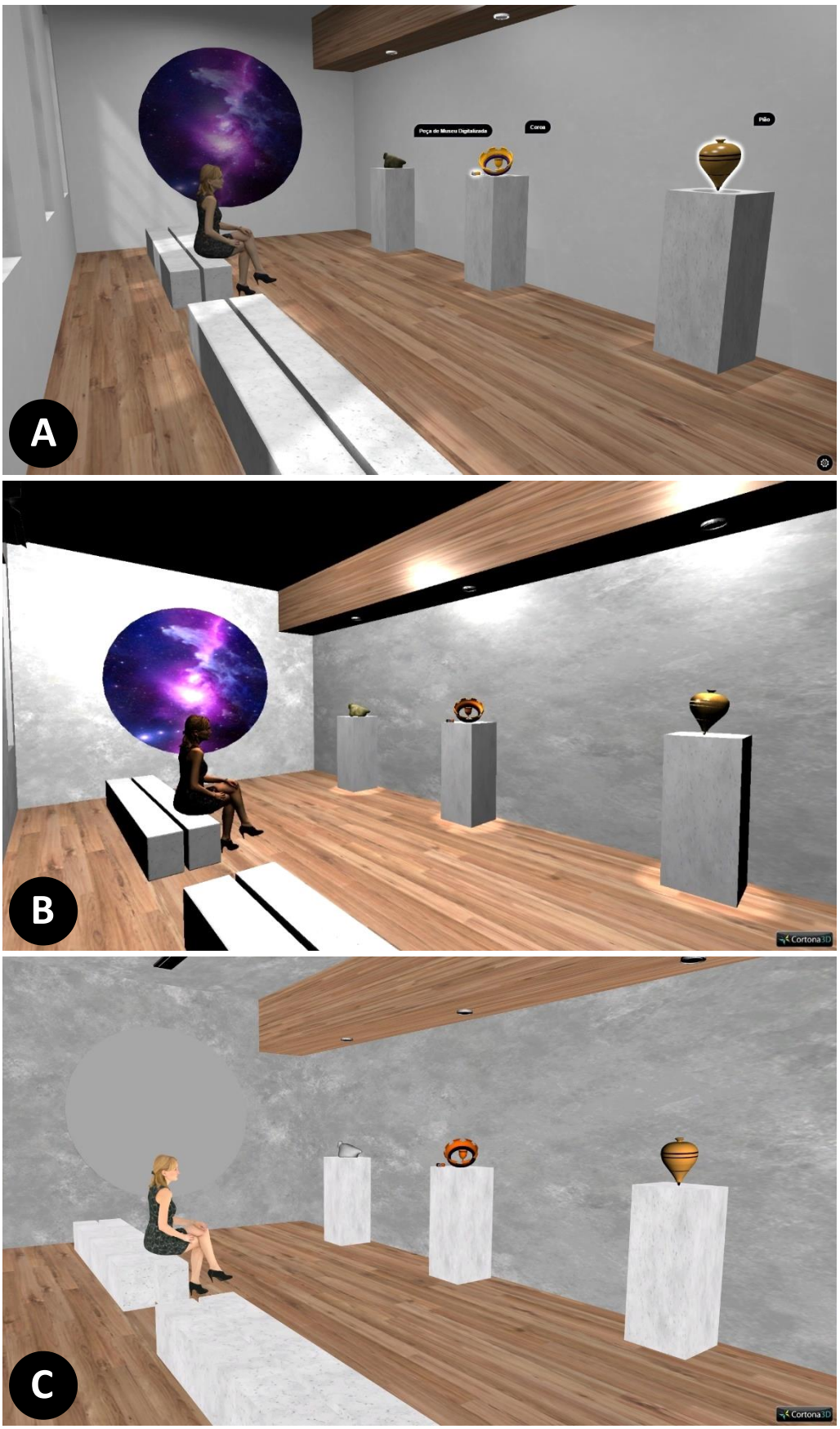


Figura 192 – Visão geral do Ambiente Externo Dia nos três diferentes formatos: (a) WebGL, (b) VRML (3ds Max), (c) VRML (Blender).

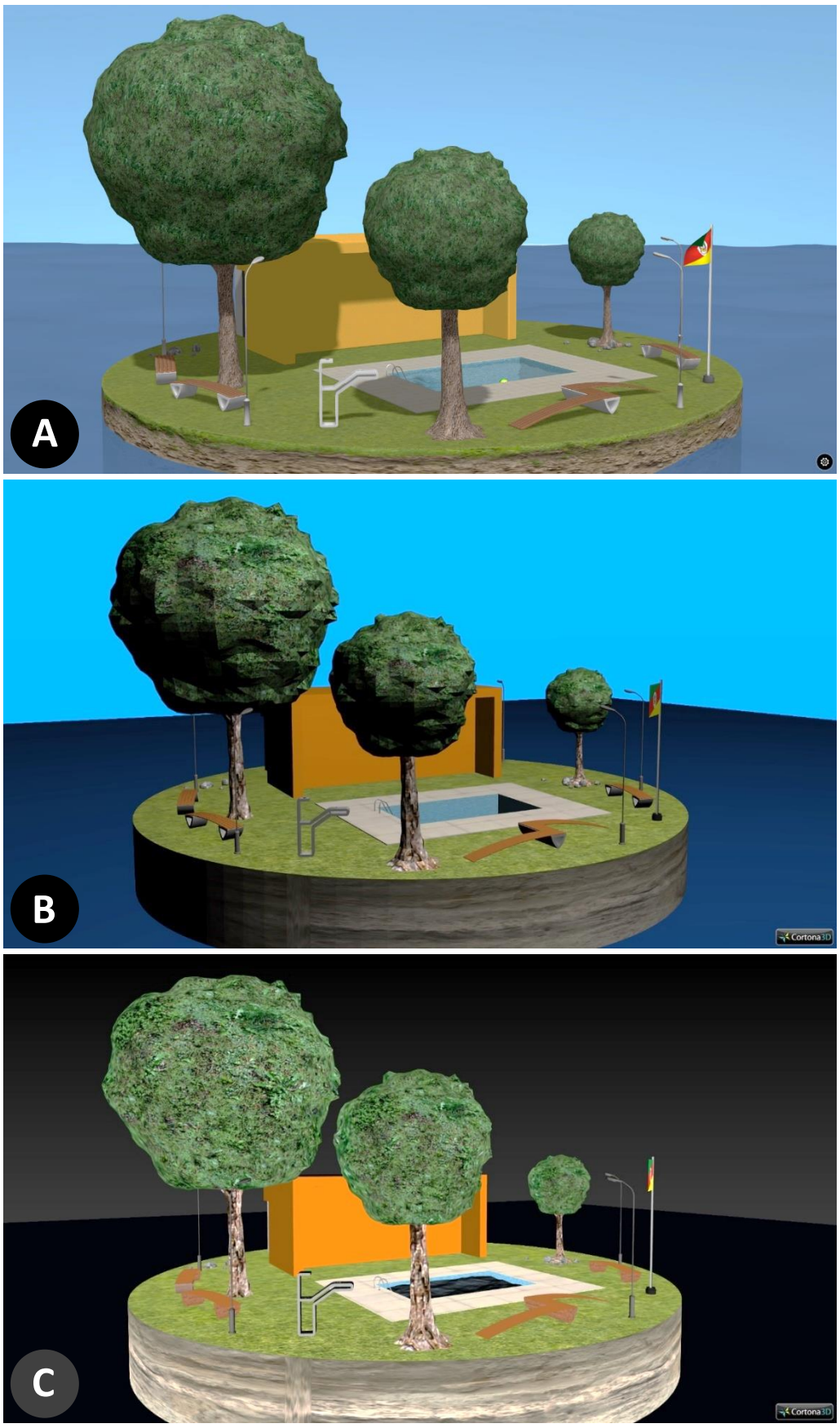
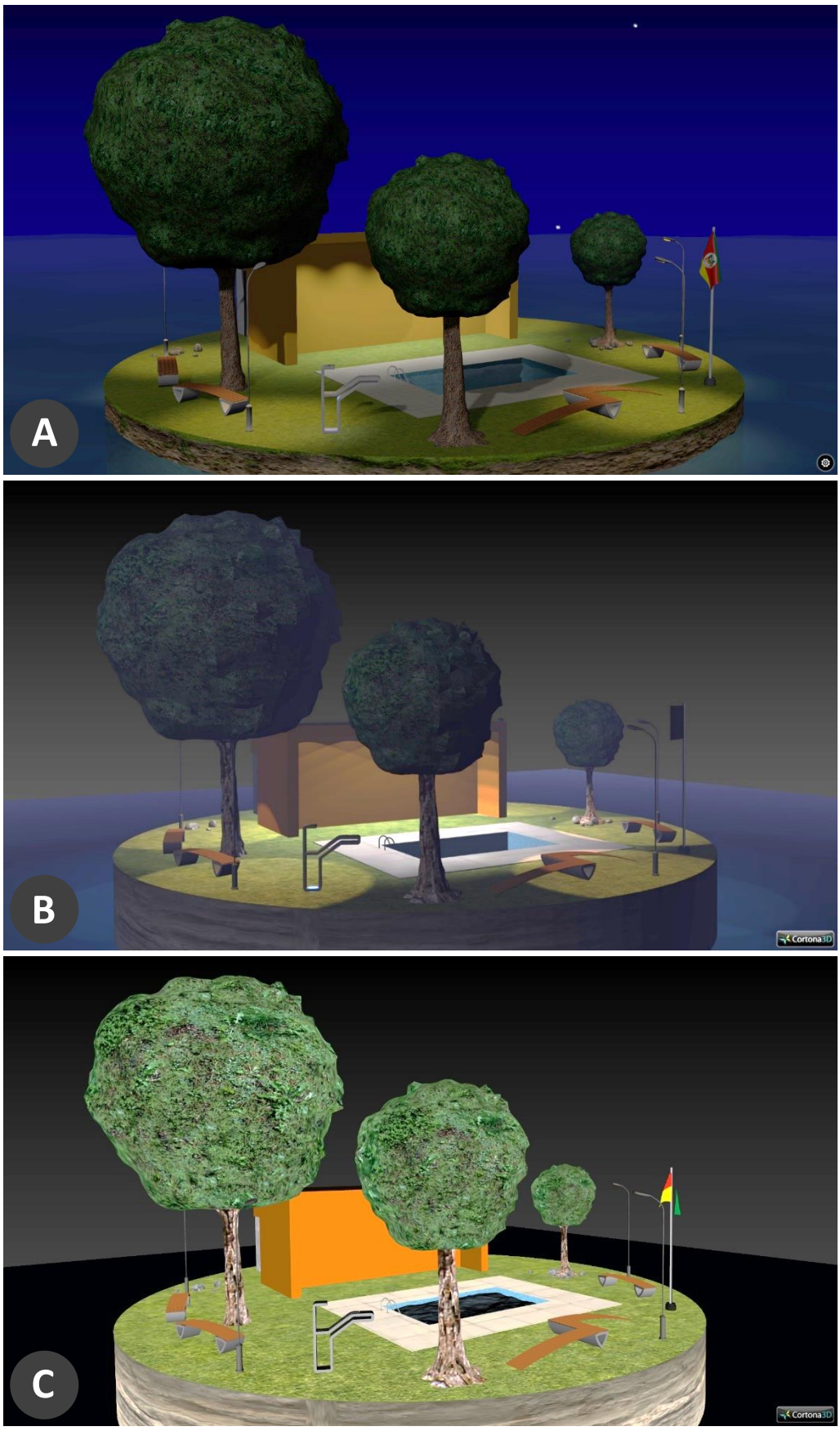


Figura 193 – Visão geral do Ambiente Externo Noite nos três diferentes formatos: (a) WebGL, (b) VRML (3ds Max), (c) VRML (Blender).



No geral, em todas as três cenas (Interno - Figura 191, Externo-Dia - Figura 192, Externo-Noite - Figura 193), quando comparados os ambientes em WebGL e VRML há diferenças claras na qualidade visual de geometrias, materiais e iluminação. Visivelmente os ambientes em VRML ficam bem abaixo do nível de realismo possível quando utilizada a sistemática desenvolvida. Aqui o mérito é da tecnologia WebGL que gerencia e aproveita melhor o processamento gráfico dos atuais computadores, *smartphones* e outros (Figura 194).

Figura 194 – Mesmo ambiente (Ambiente Externo Dia) sendo executado em diferentes dispositivos (Computador e *Smartphone*).

A discrepância entre as duas tecnologias em termos de qualidade gráfica é evidente e, no geral, pode ser analisada segundo três principais aspectos: geometria, materiais e iluminação. Além disso, existem diferenças com relação aos sons, câmeras, animações, interações e recursos extra (Tabela 10). Ainda sendo possível realizar um breve comparativo entre a qualidade geral das cenas e o tamanho final (em *megabytes*) dos arquivos dos ambientes VRML e WebGL.

Tabela 10 – Síntese da comparação entre as tecnologias VRML (Blender), VRML (3ds Max) e WebGL.

	VRML - .wrl (Blender)	**VRML - .wrl (3ds Max)**	**WebGL - .html (Blender + Blend4Web)**
Geometria	Objetos facetados, sem alisamento.	Objetos facetados, sem alisamento.	Objetos mais detalhados e com superfícies suavizadas.
Materiais e Texturas	Alguns mapeamentos de textura não suportados; Texturas compactadas e distorcidas.	Texturas sofrem compactação e distorção.	Texturas com cor, contraste e dimensão fieis às imagens utilizadas.
Iluminação	Luzes não foram exportadas.	Luzes com aspecto mais bruto e sem produção de sombras.	Iluminação e sombras com efeitos de suavização e interação realística entre diferentes fontes de luz.
Sons	Sons do tipo Posicional e Ambiente.	Sons do tipo Posicional e Ambiente.	Sons do tipo Posicional e Ambiente, com diversas propriedades configuráveis.
Câmeras	Câmeras de diferentes tipos, com configurações simples.	Câmeras de diferentes tipos, com configurações simples.	Câmeras de diferentes tipos, com configurações mais detalhadas e limitação de movimento.
Animações	Animações simples de deslocamento, rotação e escala de objetos 3D, luzes e câmeras.	Animações simples de deslocamento, rotação e escala de objetos 3D, luzes e câmeras.	Animações simples de objetos, animações de vértices, efeito de emissão de partículas e simulação de efeitos de física.
Interações	Interação somente a nível de navegação na cena.	Interações pré-definidas para o redirecionamento de páginas, reprodução de sons, acionamento por aproximação e clique e controle de animações.	Possibilidade de criação de interações de diversas complexidades, através da linguagem visual de programação dos *softwares*.
Recursos Extra	Sem outros recursos.	Sem outros recursos.	Anotações; contorno em objetos; simulação de estrelas; simulações de vento; representação de sol; simulação de água; níveis de qualidade de visualização da cena; visualização em 3D anaglifo; visualização para óculos de realidade virtual; navegação através de teclado, mouse, tela de toque, joystick e giroscópio.

5.2.1. Geometria

Em todos os ambientes VRML (Figura 191 (b) e (c), Figura 192 (b) e (c), Figura 193 (b) e (c)) os objetos não possuem um alisamento de suas geometrias, o que gera objetos facetados. Já com um melhor aproveitamento do *hardware* gráfico dos dispositivos, o WebGL consegue processar melhor as geometrias e apresentá-las com um número de detalhamento e suavização muito maior (Figura 191 (a), Figura 192 (a), Figura 193 (a)).

5.2.2. Materiais e Texturas

Os materiais e texturas nos ambientes VRML (b, c) também parecem menos realísticos quando comparados aos cenários WebGL (a). Muitas das propriedades como cor, contraste e reflexão possuem aspecto menos natural no VRML (b, c) e texturas também sofrem com distorções e diminuição de resolução no momento da exportação, utilizando esta tecnologia. A qualidade é ainda pior nos ambientes VRML exportados somente com o Blender (c).

5.2.3. Iluminação

A forma com que cada tecnologia trabalha com a iluminação talvez seja o que mais chama atenção no momento de compará-las. Assim como no tratamento das geometrias, o WebGL (a) processa melhor as informações das diferentes luzes da cena aplicando efeitos de suavização e tronando mais realista a interação dos raios luminosos com os objetos. Já o VRML (b, c), apesar de suportar praticamente os mesmos tipos de luzes que seu sucessor (Spot, Sun, Point), apresenta-as de forma mais bruta, sem grandes refinos.

Ainda relacionado às luzes, a criação de sombras projetadas e reflexos não é suportada pelo VRML, enquanto na sistemática proposta com o WebGL a ativação

destes elementos é bastante facilitada, resultando em um aspecto próximo à realidade física.

Observa-se que quase todas as luzes, bem como os céus, não estão presentes nas cenas exportadas em VRML com *software* Blender (c). Isso torna por exemplo, a visualização da cena noturna muito semelhante a diurna e diminui ainda mais o nível de realismo. Para que fosse possível visualizar estas cenas no navegador, foi necessário ativar a opção *Headlight* do plug-in Cortona3D, que habilita uma iluminação global.

5.2.4. Sons

Quanto aos sons, basicamente se apresentam da mesma maneira no VRML e no WebGL. Ambos possuem os mesmos tipos de sons (*Background* e *Posicional*) e os executam de forma semelhante. Neste quesito o WebGL leva uma pequena vantagem, pois o conjunto de programas Blender + Blend4Web oferece um maior número de opções para configurar as propriedades dos sons presentes no ambiente.

5.2.5. Câmeras

Com as câmeras acontece algo muito parecido com os sons. As duas tecnologias trazem tipos de câmera quase idênticos, porém no WebGL elas podem ser configuradas mais detalhadamente, podendo até limitar seus movimentos com facilidade. A maior diferença fica por conta da possibilidade de exportação de várias câmeras nos ambientes VRML criados com o programa 3Ds Max, enquanto nas cenas WebGL, criadas através da sistemática desenvolvida, é possível exportar apenas uma câmera por vez, podendo simular o efeito de múltiplas câmeras utilizando o bloco *Move Camera* em uma lógica de interação.

5.2.6. Animações

Quanto às animações, percebe-se que no VRML somente as animações simples de deslocamento, rotação e escala de objetos 3D, luzes e câmera foram exportadas, pois são as únicas suportadas por ele. Enquanto isso nos ambientes em WebGL, além destas, foi possível a criação de animações com modificação na malha dos objetos (Animação de Vértices), efeito de emissão de partículas e simulação de efeitos de física.

5.2.7. Interações

Em se tratando de interações, já era possível criá-las utilizando o VRML, porém para o público leigo em programação computacional as opções se restringem bastante. Existem programas como o 3ds Max que até hoje trazem ferramentas de simples inserção de interações no ambiente a ser exportado em VRML, mas estas são totalmente pré-definidas e bem limitadas. Chamadas de sensores, elas servem para redirecionar o ambiente para outras páginas HTML ou ambientes VRML quando se clica em determinado objeto (*Anchor*), determinar como serão reproduzidos os sons presentes no ambiente (*AudioClip*), determinar que ao se aproximar de certa região uma animação seja iniciada (*ProxSensor*), permitir que o clique em determinado objeto acione uma animação (*TouchSensor*) e criar controles de tempo para as animações (*TimeSensor*).

No WebGL, a situação da criação de interações mais avançadas também era destinada apenas àqueles que detinham o conhecimento de linguagem escritas de programação. No entanto, o complemento Blend4Web, que atua em conjunto *software* de manipulação tridimensional Blender, tornou mais simples esta tarefa com uma linguagem visual de bloco de fácil compreensão. Assim, as cenas em WebGL deste trabalho possuem, no total, quatorze interações diferentes, elaboradas somente com esta linguagem, sem a exigência de quaisquer conhecimentos prévios de programação computacional. Estas são apenas uma amostra das capacidades dos programas, sendo ainda possível desenvolver interações mais avançadas apenas com novos arranjos das ferramentas aqui apresentadas.

5.2.8. Recursos Extra

Verifica-se que outros muitos recursos presentes na sistemática que utiliza o WebGL não são suportados ou possuem grande complexidade de criação utilizando o VRML. Estes auxiliam na criação de ambientes com maior nível de realismo e ampliam as possibilidades de criação dos usuários. São eles:

- Criação de anotações com título e descrição;
- Criação de contorno em objetos;
- Simulação de estrelas;
- Simulações com a influência do vento;
- Representação do sol no céu;
- Simulação de água (animada e estática);
- Geração automática de diferentes níveis de qualidade de visualização da cena (*Low*, *High*, *Ultra*);
- Geração automática de visualização em 3D anaglifo[16];
- Geração automática de visualização para óculos de realidade virtual como o Google Cardboard e similares, desde que o dispositivo utilizado tenha giroscópio;
- Navegação através de teclado, mouse, tela de toque, *joystick* e giroscópio.

Além dos já citados, existem dois recursos do padrão mais antigo que se mantiveram praticamente idênticos nesta sistemática de criação com o WebGL, utilizando como comparação a criação de ambientes VRML através do programa 3ds Max. O primeiro é o *Mist* ou *Fog* (Névoa em inglês), que simula uma espécie de neblina com propriedades configuráveis, que se torna mais densa na medida em que se afasta a câmera da cena. Este pode ser observado no ambiente externo durante a noite. O segundo é a Duplicação Ligada ou Clone, um recurso de otimização do software 3ds Max e suportado pelo VRML, que cria cópias de objetos aproveitando diversas informações

[16] O 3D Anaglifo é uma dentre as diversas tecnologias de visualização tridimensional que pretendem dar ao usuário uma sensação de profundidade em imagens planas exibidas por televisores, celulares e tantos outros aparelhos. Nesta, são utilizadas cores complementares, vermelho e azul, na imagem para que quando visualizada com óculos especiais, com as mesmas cores uma em cada lente, a pessoa possa observar a profundidade esperada (SILVA; SCHULER, 2011, p. 03).

do objeto original. Isso reduz a quantidade de informações exportadas, diminuindo substancialmente a exigência de processamento para a visualização. Por isso, sempre que possível, este recurso foi utilizado para a realização de cópias de objetos 3D e luzes.

5.2.9. Qualidade Geral x Tamanho de Arquivo

Por fim, pode-se comparar o tamanho final, em megabytes (MB), dos arquivos ".wrl" (VRML) e ".html" (WebGL) (Tabela 11) desde que sejam feitas algumas considerações. Com exceção de uma cena (Ambiente Interno – VRML – 3ds Max), os arquivos exportados em VRML se mostram bem menores que o ".html" (WebGL). Contudo, deve-se ressaltar que estes são constituídos em sua quase totalidade por objetos 3D, câmeras e luzes. Com apenas uma animação de objeto simples (giro do pião) na cena do ambiente interno e sem quaisquer interações a não ser a movimentação de câmera. Sem contar a evidente menor qualidade deles, em relação ao ambiente WebGL, em todos os aspectos.

Já o arquivo ".html" traz consigo todos os recursos presentes na sistemática elaborada, com praticamente o mesmo nível de exigências de conhecimentos prévios dos usuários. Ainda, configura o que há de mais atual neste nicho em nível comercial e por tudo isso é justificável seu maior tamanho. Todavia nota-se que mesmo sendo maiores, não oferecem dificuldade compartilhamento e execução, visto que podem ser distribuídos sem problemas por meios online e offline. Para motivos de ilustração, o maior arquivo HTML, de 36,90 MB, leva em torno de 20 segundos para ser baixado com uma internet de 15 Mb/s (megabits por segundo) e ocupa apenas 0,92% de um *pendrive* com capacidade máxima de 4 GB (gigabytes), por exemplo.

Tabela 11 – Tamanhos, em megabytes (MB), dos arquivos das cenas utilizadas para comparação.

	VRML - .wrl (Blender)	**VRML - .wrl (3ds Max)**	**WebGL - .html (Blender + Blend4Web)**
Externo Dia	7,70 MB	10,00 MB	19,60 MB
Externo Noite	4,94 MB	4,84 MB	16,70 MB
Interno	18,20 MB	46,20 MB	36,90 MB

5.3. DIAGRAMAS DAS INTERAÇÕES

Semelhante ao que foi feito para ilustrar, na sistemática, o conteúdo explicativo sobre a construção de interações (item 4.10.4), reuniu-se todas as interações desenvolvidas para os ambientes-teste na forma de diagramas. Através deles pretende-se reforçar como interagem os blocos da linguagem visual do Blender + Blend4Web e fornecer mais exemplos para que os usuários possam, então, criar suas próprias interações. Estes diagramas não acompanham esta dissertação, mas podem ser encontrados no Apêndice B do livro digital 'Ambientes Virtuais 3D Interativos' (ISBN 978-85-61965-48-8), elaborado por Bruno Spanevello Pergher e José Luís Farinatti Aymone e que contém na íntegra a sistemática elaborada durante o presente trabalho de pesquisa.

Cada interação é apresentada por uma descrição e dois diagramas. Ambos os diagramas demonstram a sequência lógica dos passos necessários para constituir determinada interação no ambiente virtual. Lembrando que o número de passos e a organização estrutural dos mesmos podem variar conforme as necessidades e preferências do usuário para com os objetivos da cena. Os conjuntos de interações aqui expostos servem apenas de exemplo e inspiração.

O Diagrama de Ações retrata, de forma literal, todas as ações individuais que o programa e o usuário realizarão para que a referida interação aconteça. Já o Diagrama de Blocos substitui as ações executadas pelo programa por blocos disponíveis na linguagem visual de programação dos programas estudados. Por exemplo, a ação de *Início* do Diagrama de Ações é substituída pelo bloco *Entry Point* (Ponto Inicial) no Diagrama de Blocos. Já as ações de *Executar uma Animação* e de *Tornar um Objeto Clicável* são trocadas pelos blocos *Play Animation* (Reproduzir Animação) e *Switch Select* (Interruptor de Seleção), respectivamente.

Visando facilitar a relação entre as ações literais do programa representadas no Diagrama de Ações e os blocos de programação que as substituem no Diagrama de Blocos, estas foram inseridas em blocos preenchidos com a cinza. Assim, diferenciando-as das ações executadas pelo usuário, por meio de cliques por exemplo, que permanecem sem alterações nos dois diagramas já que são interferências/entradas externas e por isso não são representadas por blocos da linguagem.

No Diagrama de Blocos ainda foram acrescentadas representações dos pontos *Reroute* (Redirecionar) e do sentido do fluxo da execução da programação quando utilizada a técnica de Redirecionamento Cíclico, explicada no item 4.10.4. Ambas as representações são indicadas também em legenda que acompanha os diagramas.

Capítulo 6

CONSIDERAÇÕES FINAIS

A revisão bibliográfica realizada mostrou clara a importância da atualização constante dos conhecimentos, habilidades e ferramentas de projeto, de estudantes e profissionais do Design. Visando acompanhar as evoluções tecnológicas cada vez mais frequentes. No campo do Design Virtual, os últimos anos foram repletos de novidades, devido especialmente à consolidação da Realidade Virtual e da Realidade Aumentada, que transformaram significativamente as exigências na área de criação e distribuição de ambientes tridimensionais virtuais interativos.

Para atender aos atuais requisitos de criação, surgiram novas tecnologias de representação e descrição de cenas 3D virtuais, como o WebGL, que substituíram os antigos padrões e linguagens. O WebGL é promissor, sendo mais robusto e adaptável que seus antecessores. Entre as tantas novidades trazidas por ele estão, o melhor gerenciamento do processamento gráfico dos dispositivos visualizadores, viabilizando a criação de ambientes com maior qualidade visual, e a possibilidade de execução do conteúdo nos principais navegadores web da atualidade, em computadores, *smartphones* e *tablets*, dispensando o uso de quaisquer complementos (*plug-ins*).

Quando comparado a tecnologias antecessoras de mesmo propósito, a descrição de cenas tridimensionais virtuais de forma *online* ou *off-line*, os conteúdos que utilizam o padrão WebGL possuem uma ampla vantagem em relação à qualidade final e flexibilidade de execução. Porém, verificou-se que apesar de ser o sucessor natural de tecnologias como o VRML e representar o estado da arte naquilo em que se propõe, ainda é dificultada a criação de conteúdo WebGL por aqueles que não possuem conhecimentos prévios de programação computacional, como a grande maioria dos designers.

Originalmente, e na maioria dos casos ainda hoje, os ambientes virtuais WebGL eram construídos quase inteiramente através de linguagens escritas de programação, com repentinas manipulações de elementos tridimensionais de forma menos textual. Com o passar do tempo foram surgindo novos *softwares* que buscaram facilitar o desenvolvimento de ambientes 3D WebGL por leigos em programação. Entretanto, a quase totalidade dos programas encontrados não atende adequadamente a todos os requisitos deste trabalho, elaborados com base na revisão bibliográfica prévia.

De todas as 34 ferramentas levantadas, quase todas (33) tiveram de ser descartadas por não cumprirem requisitos obrigatórios do trabalho. As principais deficiências delas são a impossibilidade de criar animações e a falta de recursos de criação de interações. Mesmo quando possuíam estas capacidades, exigiam a manipulação de códigos de programação para tal. Além do mais, muitas sequer puderam ser avaliadas por completo, por não funcionaram em algum dos principais sistemas operacionais (Windows, Mac OS, Linux) em suas versões atuais ou por apenas possuírem versões pagas.

Apesar do grande número de ferramentas descartadas, restou uma, composta na verdade por dois *softwares* (Blender + Blend4Web) que trabalham em conjunto. Já de início estes *softwares* se diferenciaram dos demais por possuírem versões profissionais completas inteiramente gratuitas e com atualizações frequentes (em média a cada 3 ou 4 meses). Além disso, ambos possuem completos manuais online, amplas e ativas comunidades *online* que estão sempre dispostas a ajudar em caso de dúvidas e problemas, até mesmo com a participação dos desenvolvedores. Também é possível encontrar diversos tutoriais em texto e vídeo, que ensinam a utilizar muitos recursos.

O programa Blender junto ao seu complemento Blend4Web apresentaram resultados muito satisfatórios já nos primeiros testes experienciais. Unindo bons recursos e facilidade de uso por usuários já familiarizados com outros *softwares* de modelagem 3D, se mostraram capazes de atender todos os seis requisitos obrigatórios para a proposição da sistemática e três dos quatros requisitos desejáveis. Deixou a desejar somente por não ser de código-aberto o arquivo HTML final gerado por eles.

A partir da descoberta e exploração do conjunto de programas, não restaram dúvidas quanto a sua escolha como solução do problema de ligação de uma tecnologia (WebGL) muito mais próxima de programadores e designers. Todos os recursos disponíveis foram testados e classificados conforme os critérios do trabalho para determinar quais fariam parte da sistemática.

Uma atenção especial foi dada a um recurso muito importante dos programas, a Linguagem Visual de Programação que utiliza blocos de funções pré-definidas para a construção das mais distintas interações. Este é um recurso chave, que possibilita a criação de interações sem a necessidade de conhecimentos prévios de programação computacionais.

As capacidades do Blender, com objetos 3D, luzes, câmeras, animação e outros, aliadas as do Blend4Web, com sua facilitada linguagem visual de programação e exportação do conteúdo empregando o WebGL, surpreenderam positivamente. Com isso os três pilares da criação de ambientes 3D virtuais (modelagem, animação e interação) puderam ser contemplados consistentemente.

Após um breve tempo de experimentação dos programas já foi possível iniciar a construção da sistemática que surpreendeu pelo grande número de recursos que puderam ser desvendados e relatados para uso dos designers. Por vezes, durante o processo de construção da sistemática, o conteúdo escrito foi reformulado e novos recursos foram sendo acrescentados na medida que eram realizadas adequações segundo os requisitos do trabalho.

Outra etapa inesperadamente positiva foi a de entendimento da linguagem visual de programação por blocos. O que no início deste trabalho vinha se mostrando como um dos maiores desafios, ou talvez o maior, no final foi superado de forma muito satisfatória. Esta boa avaliação vem do não conhecimento de nenhum outro sistema de construção de lógicas de interação em cenas 3D tão facilitado e que ofereça tantas possibilidades. Com apenas algumas horas de uso, um novo usuário destes programas pode construir interações com grau elevado de complexidade utilizando por exemplo, saltos condicionais e criação de variáveis.

Tendo encontrado uma solução facilitada e ao mesmo tempo robusta para o desenvolvimento das interações, não foi mais necessária a criação de conjuntos de

interações pré-determinadas para serem disponibilizadas aos usuários, semelhante aos sensores do VRML, como o vislumbrado inicialmente. O método elucidado faz com que designers possam compreender em pouco tempo a manipulação de todos os blocos da linguagem visual através da análise de exemplos, como os apresentados no Apêndice B do livro 'Ambientes Virtuais 3D Interativos'. Ampliando as capacidades de criação dos designers neste quesito, aumentando a autonomia e flexibilidade na geração de interações completamente novas.

Após a estruturação de toda a sistemática, buscou-se aprimorar os resultados obtidos inicialmente e também testar todos os processos descritos, verificando as relações dos recursos quando em conjunto em uma mesma cena. Para isso, três novos ambientes-teste foram elaborados unindo agora todos os recursos apresentados. Os resultados, descritos no Capítulo 5, confirmaram o poder dos *softwares* e das tecnologias envolvidas. As poucas desvantagens, como a impossibilidade de exportação de mais de uma câmera e a dificuldade de utilização da simulação de efeitos de física, foram contornadas facilmente, com o bloco de interação *Move Camera* e a técnica experimental de animação criada, respectivamente. Provou-se que a sistemática proposta está apta para ser utilizada no âmbito estudantil e profissional na área do Design Virtual, sendo composta por todos os recursos presentes nas tecnologias anteriores e muito mais.

A respeito do programa Blender e seu complemento Blend4Web, as constantes atualizações de ambos, por um lado traz a correções e adição de funções com frequência e por outro, podem alterar parte do exposto neste trabalho. Contudo, pode-se perceber que as atualizações que ocorreram durante o período de desenvolvimento desta pesquisa, não alteraram nenhum recurso existente de maneira significativa. Ainda, tendem a aumentar as possibilidades de uso dos softwares no futuro, em decorrência do aprimoramento e acréscimo de funções que buscarão utilizar todas as capacidades do WebGL.

Resumindo, a busca por uma ligação frequente entre uma tecnologia muito mais próxima de programadores (WebGL) e o Design se mostrou um desafio que foi naturalmente enfrentado conforme as muitas ligações e possibilidades entre elas se salientavam. A descoberta dos *softwares* Blender e Blend4Web foi fundamental para a

construção de uma sistemática com a qual fosse possível obter resultados com ótimo nível de qualidade gráfica e funcional.

Ainda, a sistemática resultante deste trabalho foi organizada e publicada na forma de um livro digital intitulado “Ambientes Virtuais 3D Interativos – utilizando Blender + Blend4Web” pela Editora Marca Visual com o ISBN 978-85-61965-48-8. Objetiva-se com este livro, difundir o conhecimento construído de forma didática a todos os interessados no assunto.

Espera-se que este trabalho ajude designers e demais interessados, a aperfeiçoar seus ambientes virtuais tridimensionais e inseri-los em um contexto mais atual e promissor. Vislumbra-se também o potencial da sistemática, em especial a parte de construção de interações, para estimular o pensamento lógico e introduzir conceitos de programação de forma prática e incorporada aos projetos de Design.

6.1. SUGESTÕES PARA TRABALHOS FUTUROS

O assunto deste trabalho é extenso e sofre transformações contínuas. Por isso ainda há muito a ser explorado, gerando novas pesquisas e auxiliando ainda mais os designers na criação e distribuição de ambientes tridimensionais virtuais interativos. Então, sugerem-se algumas possiblidades de pesquisa para futuros trabalhos na área:

- Explorar a aplicação da sistemática proposta e do conjunto de *softwares* Blender + Blend4Web como um todo, no desenvolvimento de outras aplicações, como jogos digitais;
- Explorar as vantagens e de se utilizar códigos de programação escrita (*script*) em conjunto com o sistema visual de criação de interações dos programas Blender e Blend4Web;
- Testar o uso de diferentes dispositivos e ferramentas (Óculos de RV, Controles de Videogames, QR Codes, Arduino...) para interação com os conteúdos criados com os programas Blender e Blend4Web;

- Se necessário, realizar uma atualização ou aprimoramento da sistemática aqui proposta, adicionando novos recursos que estejam adequados aos requisitos deste trabalho;
- Elaborar novas sistemáticas de uso de diferentes programas que possam ser uteis aos designers, aprimorando suas capacidades de criação.

Capítulo 7

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, R. **Programação Estruturada e Orientada a Objetos**. Natal: [s.n.], 2013.

ANDRADE, T. Bibliotecas de programação, o que podemos fazer com elas? [S.l.], 2015. Disponível em: <http://mctechgoiania.com.br/blog/bibliotecas-de-programacao-o-que-podemos-fazer-com-elas/>. Acesso em: 16 set. 2016.

AXIOM. **Cool Google Glass Interface Up Close (Video)**. High Tech Point. Disponível em: <http://htpoint.com/news/cool-google-glass-interface-up-close-video/>.

AYMONE, J. L. F. A Otimização De Modelos Em Realidade Virtual. Rio de Janeiro: [s.n.], 2003. p. 1–11.

BAZILIO, C. Linguagens de Script. 2016. p. 1–14. Disponível em: <http://www2.ic.uff.br/~bazilio/cursos/lp/material/LinguagensScript.pdf>. Acesso em: 15 set. 2016.

BEANE, A. **3D Animation Essentials**. 1. ed. Indianapolis: Sybex, 2012.

BIGOWN. O que é linguagem de programação, IDE e compilador? [S.l.], 2015. Disponível em: <http://pt.stackoverflow.com/questions/101691/o-que-é-linguagem-de-programação-ide-e-compilador>. Acesso em: 12 ago. 2016.

BJOERKLI, L. E. A Review of Virtual Prototyping Approaches for User Testing of Design Solutions. 2014. p. 1–14.

BLENDER. Requirements. [S.l.], 2017. Disponível em: <https://www.blender.org/download/requirements/>. Acesso em: 9 jan. 2017.

BORJA, G. S. M. **Desarollo de una aplicación 3D para una tienda virtual en WebGL**. [S.l.]: Universidad Central del Ecuador, 2016. ISBN 9788578110796.

CAMPO, E. **Google Cardboard**. Crónicas Geek. Disponível em: <http://www.cronicasgeek.com/2016/01/google-cardboard-se-actualiza-con-efectos-de-audio-3d/>.

CANALTECH. O que é API? [S.l.], 2015. Disponível em: <http://canaltech.com.br/o-que-e/software/o-que-e-api/>. Acesso em: 27 ago. 2016.

_____. Firefox ultrapassa IE e Microsoft Edge e torna-se o segundo navegador mais usado. [S.l.], 2016. Disponível em: <https://canaltech.com.br/noticia/navegadores/firefox-ultrapassa-ie-e-microsoft-edge-e-torna-se-o-segundo-navegador-mais-usado-66405/>. Acesso em: 7 dez. 2016.

CONGOTE, J. *et al.* Interactive visualization of volumetric data with WebGL in real-time.

Proceedings of the 16th International Conference on 3D Web Technology - Web3D '11, 2011. p. 137–146. Disponível em: <http://dl.acm.org/citation.cfm?id=2010425.2010449>.

CUNHA, T. M. F.; MAINENTE, C. A. **Utilização de ambientes virtuais 3D no ensino de ciência da computação: estado da arte**. São Caetano do Sul: [s.n.], 2011. Disponível em: <http://www.uscs.edu.br/pesquisasacademicas/images/pesquisas/thiago_cilene.pdf>.

DALY, L.; BRUTZMAN, D. X3D: Extensible 3D Graphics Standard. **IEEE Signal Processing Magazine**, 2007. n. November, p. 130–135. Disponível em: <http://www.web3d.org/documents/specifications/19775-1/V3.3/index.html>.

DAN. WebGL Limitations. [S.l.], 2014. Disponível em: <http://stackoverflow.com/questions/24103804/what-are-the-graphical-limits-webgl-and-three-js-can-handle-gracefully>. Acesso em: 16 set. 2016.

DANCHILLA, B. **Beginning WebGL for HTML5**. [S.l.]: [s.n.], 2012.

DAVID, M. F. Programação Orientada a Objetos: uma introdução. [S.l.], 2007. Disponível em: <http://www.hardware.com.br/artigos/programacao-orientada-objetos/>. Acesso em: 29 jul. 2016.

DIGITALDEV. Linguagens de programação, para que servem? [S.l.], 2014. Disponível em: <http://www.digitaldev.com.br/linguagens-de-programacao/>. Acesso em: 12 ago. 2016.

DO, O. *et al.* Aplicação de imagem 3d, através de anaglifo, para fins de observação do patrimônio cultural religioso em recife – pe. 2011. p. 1–15.

DRESCH, A.; LACERDA, D. P.; ANTUNES JUNIOR, J. A. V. DESIGN SCIENCE RESEARCH: Método de Pesquisa para Avanço da Ciência e Tecnolgia. **Gestão Produção**, 2013. v. 20, n. 4, p. 741–761.

DUARTE, R. A. Termos e Conceitos - Computação Gráfica 3D e Maquete Eletrônica. [S.l.], 2016. Disponível em: <http://www.jrrio.com.br/computacao-grafica/termos-e-conceitos.html>. Acesso em: 20 jul. 2016.

EXFORSYS. The History of Object Oriented Programming. [S.l.], 2006. Disponível em: <http://www.exforsys.com/tutorials/oops-concepts/the-history-of-object-oriented-programming.html>. Acesso em: 1º ago. 2016.

FALCÃO, E. De L. **Publicação e acesso a conteúdos 3D através da Web: O caso do Museu3i**. [S.l.]: Universidade Federal da Paraíba, 2011.

FERREIRA, D. L. O YouTube e a popularizaçào da linguagem do vídeo. **Anais Eletrônicos do VI ENPOLE**, 2015. p. 1–8.

FERREIRA, K. G. **Teste de Usabilidade**. [S.l.]: Universidade Federal de Minas Gerais, 2002. Disponível em: <http://conteudo.imasters.com.br/3206/usabilidade.pdf>.

FUPICAT. **CN Animation**. Scratch. Disponível em: <https://scratch.mit.edu/projects/117341634/#editor>.

GHALL. **Philadelphia city model**. TurboSquid. Disponível em: <http://www.turbosquid.com/3d-models/city-philadelphia-max/631766>.

GINIER, S. **SculptGL**. Stéphane Ginier. Disponível em: <http://stephaneginier.com/sculptgl/>.

GOOGLE. **Encontre o caminho até OZ**. Chrome Experiment. Disponível em: <http://www.findyourwaytooz.com/>.

GUÉNO, F. **Une sélection de startups de Futur en Seine**. Le Phare Digital. Disponível em: <http://lepharedigital.com/2014/06/23/selection-startups-futur-en-seine/>.

GUERRA, A. R. Mercado de realidade virtual vai crescer 84,5% ao ano até 2020. [S.l.], 2016. Disponível em: <http://www.bitmag.com.br/2016/03/mercado-de-realidade-virtual-vai-crescer-845-ao-ano-ate-2020/>. Acesso em: 3 out. 2016.

GUIA DO ESTUDANTE. Design | Guia do Estudante. [S.l.], 2017. Disponível em: <https://guiadoestudante.abril.com.br/profissoes/design/>. Acesso em: 4 set. 2017.

HEMMENDINGER, D. Object-oriented programming. [S.l.], 2008. Disponível em: <http://academic-eb-britannica.ez45.periodicos.capes.gov.br/levels/collegiate/article/443530>. Acesso em: 29 jul. 2016.

HOLLIDAY, O. J. **Para sistematizar experiências**. 2. ed. Brasília: Ministério do Meio Ambiente, 2006.

HOPF, T.; FALKEMBACH, G. A. M.; ARAÚJO, F. V. O Uso Da Tecnologia X3D Para O Desenvolvimento De Jogos Educacionais. **Cintedufrgsbr**, 2007. v. 5, n. 2, p. 1–8. Disponível em: <http://www.cinted.ufrgs.br/renote/dez2007/artigos/2cTiago.pdf>.

JACKSON, W. HTML5 Quick Markup Reference. **HTML5 Quick Markup Reference**. [S.l.]: Apress, 2016, p. 171–178.

JÁCOME, J. O que é biblioteca de programação | library | lib ? O que é API | Application Programming Interface ? [S.l.], 2010. Disponível em: <https://jarbasjacome.wordpress.com/o-que-e-biblioteca-de-programacao-library-lib-o-que-e-api-application-programming-interface/>. Acesso em: 16 set. 2016.

JOAQUIM, C. Ferramentas de Modelagem 3D. [S.l.], 2012. Disponível em: <https://designdegames.wordpress.com/2012/06/04/ferramentas-de-modelagem-3d/>. Acesso em: 26 jul. 2016.

JUNG, R. L. Da C. **A arquitetura e as ferramentas digitais: uma visão do projeto arquitetônico**. [S.l.]: Univerisidade Federal do Rio Grande do Sul, 2014.

KEATS, J. **Google Glass: A Futuristic Fantasy That Already Feels Retro**. Discover Magazine. Disponível em: <http://discovermagazine.com/2014/jan-feb/69-a-glass-half-empty>.

KELNER, J.; TEICHRIEB, V. Técnicas de Interação para Ambientes de Realidade Virtual e Aumentada. *In*: KIRNER, C.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2007, p. 52–70.

KENT, A.; WILLIAMS, J. G. **Encyclopedia of Microcomputers - Volume 28**. 1. ed. Nova Iorque: Marcel Dekker, 2002.

KHRONOS GROUP. WebGL Specification. [S.l.], 2014a. Disponível em: <https://www.khronos.org/registry/webgl/specs/1.0.3/>. Acesso em: 27 ago. 2016.

_____. WebGL and OpenGL Differences. [S.l.], 2014b. Disponível em: <https://www.khronos.org/webgl/wiki/WebGL_and_OpenGL_Differences>. Acesso em: 15

set. 2016.

_____. OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. [S.l.], 2016a. Disponível em: <https://www.khronos.org/opengles/>. Acesso em: 15 set. 2016.

_____. WebGL - OpenGL ES 2.0 for the Web. [S.l.], 2016b. Disponível em: <https://www.khronos.org/webgl/>. Acesso em: 15 set. 2016.

_____. About The Khronos Group. [S.l.], 2016c. Disponível em: <https://www.khronos.org/about/>. Acesso em: 27 ago. 2016.

KIRNER, C. *et al.* Uso de Realidade Aumentada em Ambientes Virtuais de Visualização de Dados. 2004. p. 12.

_____; SISCOUTTO, R. **Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações**. [S.l.]: [s.n.], 2007a.

_____; _____. Fundamentos de Realidade Virtual e Aumentada. *In*: KIRNER, C.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2007b, p. 2–21.

_____; TORI, R. Fundamentos de Realidade Aumentada. *In*: KIRNER, C.; TORI, R.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2006, p. 22–38.

KIRNER, T. G.; SALVADOR, V. F. M. Desenvolvimento de Ambientes Virtuais. *In*: KIRNER, C.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2007, p. 90–107.

KLEINA, N. O que é engine ou motor gráfico? [S.l.], 2011. Disponível em: <http://www.tecmundo.com.br/video-game-e-jogos/9263-o-que-e-engine-ou-motor-grafico-.htm>. Acesso em: 16 set. 2016.

Programação Orientada a Objetos aula 01. LACERDA, L. C. De; RAMOS, J. M. B. Youtube, 2015. Disponível em: <https://www.youtube.com/watch?v=l-CaVliXaA0>.

LEE, P.; STEWART, D. Virtual reality: a billion dollar niche. [S.l.], 2016. Disponível em: <http://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/tmt-pred16-media-virtual-reality-billion-dollar-niche.html>. Acesso em: 3 out. 2016.

LEITE, J. Níveis de Abstração. [S.l.], 2007. Disponível em: <https://jcspl.net/2007/03/14/niveis-de-abstracao/>. Acesso em: 16 set. 2016.

LEVKOWITZ, H.; KELLEHER, C. Cloud and mobile web-based graphics and visualization. **Proceedings: 25th SIBGRAPI - Conference on Graphics, Patterns and Images Tutorials, SIBGRAPI-T 2012**, 2012. p. 21–35.

LEWIS, P. **Getting started with Three.js**. Aerotwist. Disponível em: <https://aerotwist.com/tutorials/getting-started-with-three-js/>.

LIMA, A. P. De. **Projeto de Personagens Tridimensionais e Virtuais: Validação e Adaptação de Metodologias**. [S.l.]: Uniritter, 2010. ISBN 9781617796029.

_____. **Design e práticas ágeis: Aplicação de filosofia e princípios ágeis no desenvolvimento**

de modelos tridimensionais para Jogos Digitais. [S.l.]: Universidade Federal do Rio Grande do Sul, 2015. ISBN 9788578110796.

LITTLE WORKSHOP. **New Renault Espace**. Little Workshop. Disponível em: <http://renaultespace.littleworkshop.fr/>.

LOESCH, B.; CHRISTEN, M.; NEBIKER, S. Openwebglobe – an Open Source Sdk for Creating Large-Scale Virtual Globes on a Webgl Basis. **ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, 2012. v. XXXIX-B4, n. September, p. 195–200. Disponível em: <http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B4/195/2012/>.

LOPES, E. De S. **Apostila Animação**. Beja: [s.n.], 2005.

LUZ, F. C. Animação Digital : Reflexos dos novos médias nos conceitos tradicionais de animação. **8º Congresso LUSOCOM**, 2009. p. 919–936.

MADFINGER GAMES. **Dead Trigger 2**. WebGL Games. Disponível em: <http://www.webglgames.com/deadtrigger-2/>.

MALOSSI, J. Google Cardboard, a democratização da realidade virtual. [S.l.], 2015. Disponível em: <http://joaomalossi.com/br/google-cardboard-e-a-democratizacao/>. Acesso em: 29 ago. 2016.

MANSSOUR, I. H. Introdução à VRML. [S.l.], 2000. Disponível em: <http://www.inf.pucrs.br/manssour/VRML/Intro.html>. Acesso em: 15 ago. 2016.

_____. **Paradigma Orientado a Objetos**. Porto Alegre: [s.n.], 2002. Disponível em: <http://web.b.ebscohost.com/ehost/viewarticle?data=dGJyMPPp44rp2%2FdV0%2Bnjisfk5Ie46bJOrq6xTrWkxGG%2FpOOA7enyWLOlsUmtqK5Js5a0Uq6uuE23ls5lpOrweezp33vy3%2B2G59q7RbSmsUu3qrBNtZzqeezdu33xnOJ6u9%2FngKTq33%2B7t8w%2B3%2BS7TLGtrkmyqbA%2B5OXwhd%2Fqu37z4uqM4%2B7y&hi>.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. De. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 17. ed. São Paulo: Editora Érica Ltda., 2005.

MARINS, V.; HAGUENAUER, C.; CUNHA, G. Realidade Virtual em Educação Criando Objetos de Aprendizagem com VRML. **Revista Digital da CVA**, 2007. v. 4, n. 15, p. 1–11.

MARK, K. JavaScript. [S.l.], 2016. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 15 set. 2016.

MAZURYK, T.; GERVAUTZ, M. Virtual Reality - History, Applications, Technology anda Future. 1996. p. 1–72. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780124047051000066%5Cnhttp://linkinghub.elsevier.com/retrieve/pii/B9780124047051000078%5Cn%22http://www.cg.tuwien.ac.at/research/publications/1996/mazuryk-1996-VRH/%22, %7D%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/dow>.

MICROSOFT. **Why Hololens**. Microsoft. Disponível em: <https://www.microsoft.com/microsoft-hololens/en-us/why-hololens>.

_____. Entendendo quadros por segundo (FPS). [S.l.], 2016b. Disponível em: <https://support.microsoft.com/pt-br/kb/269068>. Acesso em: 7 dez. 2016.

MINISTÉRIO DA CULTURA. Conceito: Elevar o Nível de Abstração. [S.l.], 2006. Disponível em: <http://mds.cultura.gov.br/core.base_rup/guidances/concepts/elevate_level_abstraction_7E5A12DB.html>. Acesso em: 16 set. 2016.

MIT MUSEUM. **Ivan Sutherland demonstrating Sketchpad on the TX-2**. Computer History Museum. Disponível em: <http://www.computerhistory.org/revolution/input-output/14/349/1878>.

MORAES, M. Realidade virtual acessível abre possibilidades criativas. [S.l.], 2015. Disponível em: <http://cmais.com.br/arte-e-cultura/realidade-virtual-acessivel-abre-possibilidades-criativas>. Acesso em: 29 ago. 2016.

MOREIRA, F. Introdução - JavaScript. [S.l.], 2016. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction#O_que_é_JavaScript>. Acesso em: 15 set. 2016.

MÜLLER, N. Qual a diferença entre programação estruturada e programação orientada a objetos? [S.l.], 2016. Disponível em: <https://www.oficinadanet.com.br/post/14463-qual-a-diferenca-entre-programacao-estruturada-e-programacao-orientada-a-objetos>. Acesso em: 1º ago. 2016.

MURATTAHAN. **Ratatouille**. Animasyon Gastese. Disponível em: <http://www.animasyongastesi.com/ratatouille-podcasts/>.

NADEAU, D. R. Building virtual worlds with VRML. **IEEE Computer Graphics and Applications**, 1999. v. 19, n. 2, p. 18–29.

NASCIMENTO, M. J. A. Do. **Modelagem de objetos tridimensionais para um ambiente interativo de instruções técnicas virtuais**. [S.l.]: Universidade Federal do Pará, 2010. ISBN 9788578110796.

OPENGL.ORG. OpenGL Overview. [S.l.], 2010. Disponível em: <http://www.opengl.org/about/overview/>. Acesso em: 15 set. 2016.

ORIENTE, L. HTML Básico. [S.l.], 2015. Disponível em: <http://www.linhadecodigo.com.br/artigo/81/html-basico.aspx>. Acesso em: 29 ago. 2016.

PACIEVITCH, Y. HTML. [S.l.], 2016. Disponível em: <http://www.infoescola.com/informatica/html/>. Acesso em: 29 ago. 2016.

PAGET, M. **Oculus Responds to Rift's Facebook Privacy Concerns**. Gamespot. Disponível em: <http://www.gamespot.com/articles/oculus-responds-to-rifts-facebook-privacy-concerns/1100-6438599/>.

PAL, P. K.; SARKAR, B. A language to model animation out of behaviour-embedded graphical components. **Journal of Visualization and Computer Animation**, 2002. v. 13, n. 3, p. 169–185.

PALUCH, D. Canvas tutorial - APIs Web. [S.l.], 2015. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial>. Acesso em: 15 set. 2016.

PAN, Z. *et al.* Virtual reality and mixed reality for virtual learning environments. **Computers and Graphics (Pergamon)**, 2006. v. 30, n. 1, p. 20–28.

PANISSON, E. **Gaspard Monge e a sistematização da representação na arquitetura**. [S.l.]: Univerisidade Federal do Rio Grande do Sul, 2007.

PARISI, T. **Programming 3D Applications with HTML5 and WebGL**. 1. ed. Sebastopol: O`Reilly Media, 2014.

PASCHALL, A. **Unreal Engine 4**. Unreal Engine. Disponível em: <https://www.unrealengine.com/blog/unreal-engine-4-12-released>.

PAULA, W. De P. **A animação**. [S.l.]: [s.n.], 2000. Disponível em: <www.inf.ufg.br/~eduardo/mm/livro-texto/smm8>.

PEREIRA, A. P. O que é XML? [S.l.], 2009. Disponível em: <http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>. Acesso em: 18 ago. 2016.

PERGHER, B. S.; AYMONE, J. L. F. **Ambientes Virtuais 3D Interativos: utilizando Blender + Blender4Web**. 1. ed. Porto Alegre: Marca Visual, 2017.

PRADO, G. *et al.* Virtual Reality Modeling Language. [S.l.], 1999. Disponível em: <http://ftp.unicamp.br/pub/apoio/treinamentos/vrml/vrml-about.html>. Acesso em: 15 ago. 2016.

PRATES, R. O.; BARBOSA, S. D. J. Introdução à Teoria e Prática da Interação Humano Computador fundamentada na Engenharia Semiótica. **Jornadas de Atualização em Informática, JAI 2007**, 2007. n. April, p. 263–326.

RALLYE DESIGN. **Cockpit Stradale Racing para G25-G27**. Rallye Design. Disponível em: <http://www.rallyedesign.com.br/?secao=produtos&sup=8&dep=projetos-especiais&sub=22&sub_dep=simuladores-&-cockpit>.

REVISTA EXAME. Realidade Virtual: "Produtoras de conteúdo serão responsáveis pelo crescimento do mercado VR", diz diretor do VR Studios. [S.l.], 2016. Disponível em: <http://exame.abril.com.br/negocios/dino/noticias/realidade-virtual-produtoras-de-conteudo-serao-responsaveis-pelo-crescimento-do-mercado-vr-diz-diretor-do-vr-studios.shtml>. Acesso em: 3 out. 2016.

RIBEIRO, R. D. S.; BRANDÃO, L. De O.; BRANDÃO, A. A. F. Uma visão do cenário Nacional do Ensino de Algoritmos e Programação: uma proposta baseada no Paradigma de Programação Visual. **Lbd.Dcc.Ufmg.Br**, 2012. n. Sbie, p. 26–30. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbie/2012/00127.pdf>.

RIZZO, P. A democratização da realidade virtual. [S.l.], 2015. Disponível em: <http://disneybabble.uol.com.br/br/tecnologia/democratizacao-da-realidade-virtual>. Acesso em: 29 ago. 2016.

ROGERS, Y.; SHARP, H.; PREECE, J. **Design de interação: além da interação humano-computador**. 3. ed. Porto Alegre: Bookman, 2013.

ROMERO, M. **Visão Geral do Editor de Blueprints**. Romero Blueprints. Disponível em: <http://romeroblueprints.blogspot.com.br/2014/03/visao-geral-do-editor-de-blueprints.html>.

RUSSELL, K. **Building 3D Web Applications using WebGL**. Nova Iorque: [s.n.], 2015.

SAMD0G. **When dragging an actor in a side scroller, said actor jumps around at random?** Unreal Engine. Disponível em: <https://answers.unrealengine.com/questions/23872/using-mouse-to-move-actor-isnt-working.html>.

SANTOS, R. M. S. **Ensino Da Programação Através De Programação Visual**. [S.l.]: Universidade de Lisboa, 2013.

SMITH, B. Object-oriented programming. **Advanced ActionScript 3**. 2. ed. Nova Iorque: Apress, 2014, p. 1–23.

SOARES, L. P. Web3D Consortium - X3D FAQ (portuguese). [S.l.], 1999. Disponível em: <http://www.lsi.usp.br/~lsoares/x3d/faq.html#general-1>. Acesso em: 18 ago. 2016.

SUZUKI, A. Mozilla mostra WebGL 2, o futuro dos gráficos 3D nativos de navegador. [S.l.], 2015. Disponível em: <http://www.tecmundo.com.br/web/75995-mozilla-mostra-webgl-2-futuro-graficos-3d-nativos-navegador.htm>. Acesso em: 7 out. 2016.

TAUBIN, G. *et al.* Geometry coding and VRML. **Proceedings of the IEEE**, 1998. v. 86, n. 6, p. 1228–1243.

THALMANN, D. Using virtual reality techniques in the animation process. **Virtual Reality Systems**, 1993. p. 1–20. Disponível em: <http://pdf.aminer.org/000/214/092/processes_in_a_functional_animation_system.pdf>.

THREE.JS. **Geometry Cube**. GitHub. Disponível em: <https://github.com/mrdoob/three.js/blob/master/examples/webgl_geometry_cube.html>.

_____. **Three.js Editor**. Three.js. Disponível em: <https://threejs.org/editor/>.

TORI, R.; KIRNER, C. Fundamentos de Realidade Virtual. *In*: KIRNER, C.; TORI, R.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2006, p. 2–21.

Guacamelee. UPLOA, R. Youtube, 2013. Disponível em: <https://gaming.youtube.com/watch?v=yY-d_hfDWLE>.

URBANO, V. **Project Tango: Lenovo Phab2 Pro será o primeiro smartphone**. Techenet. Disponível em: <http://www.techenet.com/2016/06/project-tango-lenovo-phab2-pro-sera-o-primeiro-smartphone/>.

VENTURA, P. O que é API. [S.l.], 2015. Disponível em: <http://www.ateomomento.com.br/o-que-e-api/>. Acesso em: 27 ago. 2016.

VIRTUAL REALITY SOCIETY. What is Virtual Reality? [S.l.], 2016a. Disponível em: <http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html>. Acesso em: 21 jul. 2016.

_____. Augmented Reality - What is it? [S.l.], 2016b. Disponível em: <http://www.vrs.org.uk/augmented-reality/>. Acesso em: 23 nov. 2016.

WEB3D CONSORTIUM. What is X3D. [S.l.], 2016. Disponível em: <http://www.web3d.org/x3d/what-x3d>. Acesso em: 15 jul. 2016.

WITHER, J.; DIVERDI, S.; HOLLERER, T. Annotation in outdoor augmented reality. **Computers and Graphics (Pergamon)**, 2009. v. 33, n. 6, p. 679–689.

YE, J. *et al.* Applications of virtual reality in product design evaluation. **… Interaction. HCI**

Applications and …, 2007. p. 1190–1199. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-540-73111-5_130>.

ZHANG, X.; GRAČANIN, D. An Approach to WebGL Based Distributed Virtual Environments. **Proceedings of the 18th International Conference on 3D Web Technology**, 2013. p. 195–198. Disponível em: <http://doi.acm.org/10.1145/2466533.2466561>.

ZHANG, X.; WU, E. Foreword to special section on Virtual Environments and Applications. **Computers & Graphics**, 2012. v. 36, n. 8, p. A13–A14. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0097849312001458>.

ZORZAL, E. R. *et al.* Visualização de Informação com Realidade Virtual e Aumentada. *In*: KIRNER, C.; SISCOUTTO, R. (Org.). **Fundamentos e tecnologia de realidade virtual e aumentada**. Porto Alegre: Editora SBC, 2007, p. 256–275.

Este trabalho foi realizado com o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por meio de bolsa de estudos.

Apêndice A

TABELA DE *SOFTWARES* ANALISADOS

Tabela contendo a listagem completa dos softwares analisados segundo os critérios estipulados.

Ferramenta	Funcionamento	Requisitos Cumpridos	Desvantagens
Unity + Assets	*Off-line*	nº 1, 2, 3, 4, 6, 7, 10	Não trabalha com edição de modelos 3D e exige conhecimento prévio em programação
Blender + Blend4Web	*Off-line*	nº 1, 2, 3, 4, 6, 7, 8, 9, 10	Arquivo final gerado não possui código aberto
Blender + Babylon Exporter	*Off-line*	nº 1, 2, 3, 5, 6	Não possibilita a criação de interações e sem exportação de arquivo HTML
Blender + Sketchfab Exporter	*Online*	nº 1, 3, 4, 6, 10	Não possibilita a criação de interações e sem exportação de arquivo HTML
Rhino + Iris	*Off-line*	nº 2, 4, 6, 7	Não possui versão gratuita ou educacional e não possibilita a criação de interações
3ds Max 2017 + WebGL Exporter for Max	*Off-line*	nº 2, 4, 6, 7, 10	O complemento necessário não possui versão gratuita ou educacional e não possibilita a criação de interações
3ds Max 2017 + SEA3D Exporter	*Off-line*	nº 1, 2, 6	Não possibilita a criação de interações
3ds Max 2017 + Three.js Exporter	*Off-line*	nº 1, 2, 5, 6	Não possibilita a exportação em direta em WebGL ou a criação de interações
3ds Max 2017 + Babylon Exporter	*Off-line*	nº 1, 2, 5, 6	Não possibilita a criação de interações e sem exportação de arquivo HTML
3ds Max + Sketchfab Exporter	*Online*	nº 1, 4, 6, 10	Não possibilita a criação de interações e sem exportação de arquivo HTML
Autodesk Maya 2016 + Inka3D	*Off-line*	nº 1, 2, 3, 4, 6, 7, 10	Não possibilita a criação de interações
Autodesk Maya 2016 + WebGL Exporter For Maya	*Off-line*	nº 2, 3, 4, 6, 7	Não possibilita a criação de interações
AutoCAD 2017 + WebGL Exporter For Autocad	*Off-line*	nº 2, 4, 6, 7	O complemento necessário não possui versão gratuita ou educacional e não possibilita a criação de interações
A3DS Viewer	*Off-line*	nº 1, 2, 4, 5, 7	Somente exporta objetos 3D
CopperCube	*Off-line*	nº 2, 4, 6, 7, 8	Não possui versão gratuita ou educacional e exige conhecimento prévio em programação
WebGL Publisher	*Off-line*	nº 2, 4, 6	Não possui versão gratuita ou educacional e não possibilita a criação de animações
Atomic Game Engine	*Off-line*	nº 1, 2, 4, 5, 6, 7	Não trabalha com edição de modelos 3D e exige conhecimento prévio em programação
FinalMesh	*Off-line*	nº 2, 4, 6, 7	Não possui versão gratuita ou educacional e não possibilita a criação de interações
Marmoset Toolbag	*Off-line*	nº 2, 4, 5, 6	Não possui versão gratuita ou educacional e somente exporta objetos 3D sem possibilitar edição ou criação de animações e interações
Godot Engine	*Off-line*	nº 1, 2, 3, 10	Não possibilita a exportação do conteúdo na versão de avaliação e exige conhecimento prévio em programação
Unreal Engine	*Off-line*	nº 1, 2, 3, 4, 6, 7, 10	Não trabalha com edição de modelos 3D e exige conhecimento prévio em programação
Autodesk Stingray 2017	*Off-line*	nº 1, 2, 4, 10	Não trabalha com edição de modelos 3D e exige conhecimento prévio em programação
Goo Create	*Online* e *Off-line*	nº 1, 2, 3, 4, 10	Não trabalha com edição de modelos 3D, exige conhecimento prévio em programação e não possibilita a visualização *off-line* do conteúdo criado
WebGLStudio.js	*Online* e *Off-line*	nº 1, 2, 3, 4, 5, 7, 10	Não trabalha com edição de modelos 3D e exige conhecimento prévio em programação
Three.js	*Online* e *Off-line*	nº 1, 2, 3, 4, 5, 7, 10	Não possibilita a criação de interações
Clara.io	*Online*	nº 1, 3, 4, 6, 8, 9, 10	Funcionamento somente online, não trabalha com edição de modelos 3D, e sem exportação de arquivo HTML
ShaderFrog	*Online*	nº 1, 3, 10	Função específica de composição de materiais e texturas
PlayCanvas	*Online*	nº 1, 3, 4, 9, 10	Funcionamento somente online, não trabalha com edição de modelos 3D, e sem exportação de arquivo HTML
ThreeFab Alpha	*Online* e *Off-line*	nº 1, 2, 3, 5, 8	Não possibilita a criação de interações
Vizor.io	*Online*	nº 1, 3, 4, 10	Funcionamento somente online, não trabalha com edição de modelos 3D, e sem exportação de arquivo HTML
SculptGL	*Online*	nº 1, 3, 4, 10	Função específica de escultura digital e sem exportação de arquivo HTML
3DTin	*Online*	nº 1, 3	Somente exporta objetos 3D
Cl3ver	*Online*	nº 3, 4	Somente trabalha com montagem de cenas para exportação, funcionamento somente online e sem exportação de arquivo HTML.
P3d.in	*Online*	nº 1, 3, 4	Somente exporta objetos 3D, funcionamento somente online e sem exportação de arquivo HTML.