UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE VACARO FOGAZZI

Aplicação mobile para visualização de grandes quantidades de objetos 3D em Realidade Aumentada

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Profª. Drª. Luciana Nedel
Co-orientador: Mestranda Mariane Giambastiani

Porto Alegre
2022

RESUMO

Existe um problema com a quantidade de lixo produzida no mundo e nos propomos a criar uma peça de "software"que ajudasse as pessoas a terem consciência do lixo que elas produzem. Para a realização deste estudo foi criado o núcleo de um "software"capaz de facilitar a conscientização do lixo produzido. Este núcleo consiste em usar realidade aumentada para instanciar modelos 3D e, com um simulador de física, fazer uma pilha de objetos comuns encontrados no lixo, como sacos de lixo, latas e caixas de leite. Para a criação de tal "software"utilizamos a "Game Engine Unity"para o sistema operacional "Android"e "Ios", fizemos uso do interpretador de modelos 3D e de seu simulador de física e adquirimos métricas sobre este núcleo. Estas métricas foram números de "colliders", números de triângulos, memória alocada, FPS(frames por segundo) e "frame time"(tempo de processamento do "frame"). Com estas informações em mão, poderemos definir gargalos no aplicativo, e, deste modo, em implementações futuras sabermos que tipos de configurações o aplicativo pode suportar.

Palavras-chave: Realidade Aumentada. Aplicativo Mobile. Lixo.

ABSTRACT

There is a problem with the amount of garbage produced in the world and we propose to create a piece of "software" that would help people to be aware of the garbage they produce. In order to carry out this study, the core of a "software" capable of facilitating the awareness of the waste produced was created. This core consists of using augmented reality to instantiate 3D models and, with a physics simulator, making a pile of common objects found in the garbage, such as garbage bags, cans and milk cartons. For the creation of such "software" we used the "Game Engine Unity" for the "Android" and "Ios" operating systems, we made use of the 3D model interpreter and its physics simulator and we acquired metrics about this core. These metrics were collider numbers, triangle numbers, allocated memory, FPS(frames per second) and frame time. With this information in hand, we will be able to define bottlenecks in the application, and in this way, in future implementations, we will know what types of configurations the application can support.

Keywords: Augmented Reality. Mobile App. Garbage.

LISTA DE FIGURAS

Figura 3.1 Menu de configurações do aplicativo 17
Figura 3.2 Exemplo de uma pilha de sacos de lixo usando a física da "Unity"e com sombras "No Shadow" 22
Figura 3.3 Exemplo de uma pilha de sacos de lixo usando a física da "Unity"e com sombras "Hard Shadow" 23
Figura 3.4 Exemplo de uma pilha de sacos de lixo sem usar a física da "Unity"e com sombras "No Shadow" 24
Figura 3.5 Exemplo de uma pilha de sacos de lixo sem usar a física da "Unity"e com sombras "Hard Shadow" 25
Figura 3.6 Exemplo de uma pilha de caixas de leite usando a física da "Unity"e com sombras "Hard Shadow" 26
Figura 3.7 Exemplo de uma pilha de caixas de leite sem usar a física da "Unity"e com sombras "Hard Shadow" 27
Figura 3.8 Exemplo de uma pilha de latas de sopa usando a física da "Unity"e com sombras "No Shadow" 28
Figura 3.9 Exemplo de uma pilha de latas de sopa usando a física da "Unity"e com sombras "Hard Shadow" 29
Figura 3.10 Exemplo de uma pilha latas de sopa sem usar a física da "Unity"e com sombras "Hard Shadow" 30

Figura 4.1 Gráfico mostrando a formação de uma pilha de Sacos de lixo usando a física da "Unity"e com configuração de sombra "No Shadow" 33
Figura 4.2 Gráfico mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "No Shadow" 34
Figura 4.3 Gráfico mostrando a formação de uma pilha de Latas de sopa usando a física da "Unity"e com configuração de sombra "No Shadow" 34
Figura 4.4 Gráfico mostrando a formação de uma pilha de sacos de lixo usando a física da "Unity"e com configuração de sombra "Hard Shadow" 36
Figura 4.5 Gráfico mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "Hard Shadow" 36
Figura 4.6 Gráfico mostrando a formação de uma pilha de latas de sopa usando a física da "Unity"e com configuração de sombra "Hard Shadow" 37
Figura 4.7 Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "No Shadow" 38
Figura 4.8 Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "No Shadow" 39
Figura 4.9 Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "Hard Shadow" 40
Figura 4.10 Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "Hard Shadow" 41
Figura 4.11 Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow" 42
Figura 4.12 Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow" 43

Figura 4.13 Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow" 43
Figura 4.14 Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow" 44
Figura 4.15 Gráfico mostrando 600 sacos de lixo sem usar a física da "Unity"com "No Shadow".....46
Figura 4.16 Gráfico mostrando 450 sacos de lixo sem usar a física da "Unity"com "Hard Shadow" 46
Figura 4.17 Gráfico mostrando 1650 latas de sopa sem usar a física da "Unity"com "No Shadow".....47
Figura 4.18 Gráfico mostrando 1500 latas de sopa sem usar a física da "Unity"com "Hard Shadow" 47
Figura 4.19 Gráfico mostrando a relação Triângulos e "colliders"com física 48
Figura 4.20 Gráfico mostrando a relação Triângulos e "colliders"sem física.....49

LISTA DE ABREVIATURAS E SIGLAS

AR	Augmented Reality
FPS	Frames Per Seconds

SUMÁRIO

1 INTRODUÇÃO 9
2 TRABALHOS RELACIONADOS 10
2.1 Realidade aumentada na educação 10
2.2 Realismo na realidade aumentada 11
3 TRABALHO DESENVOLVIDO 13
3.1 Realidade Aumentada - AR 13
3.2 Game Engine 13
3.3 Escolhendo a Unity 14
3.4 Descrição do aplicativo 15
3.5 Construindo o aplicativo 16
3.5.1 Interface 16
3.5.2 Detecção de planos e instanciação de plano no mundo virtual 18
3.5.3 Formação de pilha com física 18
3.5.4 Formação de pilha sem física 19
3.5.5 Selecionando sombras 20
3.5.6 Adquirindo métricas 20
3.5.7 Log e escrevendo métricas em um arquivo 21
4 RESULTADOS 31
4.1 Configurações do "hardware"usado nos teste 31
4.2 Metodologia usada nos teste 31
4.3 Resultados preliminares 32
4.4 Compreendendo os gráficos 32
4.4.1 Detecção de gargalos em pilhas com física 32
4.4.2 Detecção de gargalos em pilhas sem física 38
4.4.3 Flag static 42
4.4.4 Câmera dinâmica 45
4.5 Relação de número de "colliders"com triângulos 45
5 CONCLUSÃO 50
6 REFERÊNCIAS 51

1 INTRODUÇÃO

O lixo produzido no planeta tem cada vez mais chamado a atenção da sociedade. Este é um problema sério que merece nossa atenção (ALMEIDA, 2006). Precisamos conscientizar pessoas da quantidade de lixo que elas produzem. Queremos causar um impacto nas pessoas e deste modo talvez mudar alguns hábitos sobre a sua produção de lixo.

Para resolver esse problema sugerimos a criação de um "software"capaz de conscientizar as pessoas da quantidade de lixo que elas produzem.

Não iremos criar o "software"completo neste estudo, mas sim o núcleo dele e em seguida capturar métrica de sua execução.

Tal "software"usou AR(realidade aumentada), Está tecnologia nos permite colocar objetos virtuais no mundo real, através de dispositivos "mobile"como "smartphones"e "tablets".

Para criar o "software"usamos a "Game Engine Unity"e a biblioteca "AR Foundation". A "Game Engine Unity"nos provê uma grande quantidade uma funcionalidade como física, interpretador de modelos 3D, tocador de músicas e efeitos sonoros, além de uma linguagem de programação com grandes bibliotecas que facilitam e agilizam a criação de tais aplicativos. A biblioteca "AR Foundation"fornece funcionalidades para trabalhar com AR.

Com estas ferramentas criamos um "sofware"que é capaz de gerar uma pilha de lixo no mundo real. O lixo são modelos 3D de objetos normalmente encontrados no lixo, neste estudo usamos sacos de lixo, caixas de leite e latas de sopa como lixo.

Outra parte importante do "sofware"é a aquisição de métricas. Durante a execução do "sofware"capturamos métricas como FPS, número de triângulos, memória alocada e "frame time". Essas informações são usadas posteriormente para geração de gráficos que nos ajudaram a encontrar gargalos na execução do "sofware".

2 TRABALHOS RELACIONADOS

Os trabalhos relacionados neste estudo discutem o uso de AR na educação e do realismo na AR. Esses estudos são importantes pois queremos educar os usuários sobre o lixo produzido e também queremos um certo realismo no aplicativo

2.1 Realidade aumentada na educação

A AR é uma tecnologia relativamente nova, ainda em desenvolvimento, mas provou ter grande utilidade no campo da educação e em outros campos (HANTONO; et al, 2018). Queremos mostar sua capacidade de melhorar o aprendizado nas questões ambientais, tanto para estudantes ainda na escola básica como também na graduação.

Métodos convencionais de educação tendem a utilizar recursos bidimensionais por serem mais acessíveis e baratos, mas esses recursos não conseguem realmente aumentar o interesse dos alunos no aprendizado e é, neste momento, que podemos utiliza a AR como recurso educativo. O uso de AR não é exclusivo da educação, várias outras áreas já usam esta tecnologia como defesa militar, engenharia, medicina, museus, turismo, "games", entre outras(HANTONO; et al, 2018).

Melhorar a eficiência de aprendizagens e oferecer uma experiência mais envolvente para estudantes é uma possibilidade a ser explorada com a AR. Embora a AR já ser experimentada há anos, ela ainda é um desafio graças à tentativa de integração desta tecnologia com os métodos convencionais de aprendizado, exigindo mais pesquisas neste campo.

A tecnologia de AR criou alguns aplicativos interessantes que valem ser mencionados, segundo Kangdon Lee (2012): 1 - Um sistema que usa o AR é o SMART (System of Augmented Reality for Teaching) que é uma ferramenta educativa para crianças de segunda série com elementos de "video games"para apredizado, uma vez que é muito normal que crianças desta idade tenham contato com vídeo "games". 2 - Outra grande aplicação do AR são os “The MagicBook"onde a criança pode ler um livro mas tanbém usar a AR para visualisar cenas 3D no livro.

Certas pesquisas (DALIN; et al, 2017) foram realizadas mostrando grupos de estudantes que usaram a AR e outros que usaram métodos mais convencionais. Estas pesquisas mostraram que os estudantes que usaram AR tiveram um melhor entendimento do conteúdo estudado. Professores que participaram destes estudos mostraram facilidade em

implementar a tecnologia AR.

A AR pode vir a aumentar a eficiência e o foco em certas tarefas, principalmente no campo da educação e no campo de treinamento empresarial(LEE, 2012). Esta tecnologia, ainda, se mostrou eficiente no campo universitário, auxiliando estudantes na compreenção de estudos mais complexos.

Mas a tecnologia de AR atual só é possivel graças ao atual poder de processamento dos dispositivos atuais, principalmente dos "mobile". Esta tecnologia tem seu acesso mais livre, uma vez que agora é só preciso baixar o "software', graças a preços mais acessíveis, com câmeras embutidas e com a popularização de equipamentos compatíveis.

2.2 Realismo na realidade aumentada

Com seu desenvolvimento prático e científico, a tecnologia de AR ficou mais precisa e teve uma diminuição na latência de produção de imagens, produzindo, assim, uma experiência de imersão cada vez maior no usuário no mundo híbrido de real e virtual. Porém, a ideia ideal de uma realidade aumentada é que os elementos virtuais e reais coexistam no mesmo espaço.(MONTEIRO, 2019)

Infelizmente a AR sofre de alguns problemas, segundo Álvaro Monteiro (2019), sendo um deles a oclusão. Na sobreposição de um objeto do mundo virtual este deveria estar coberto por um objeto do mundo real, ocluindo as partes sobrepostas. Outro problema é a colisão de objetos virtuais com reais. Existem, assim, certos pontos problemáticos para a geração de imagens realistas, além dos dois já citados temos: gravidade, reflexão de luz, sombras, vento e som.

Na criação de um AR mais realista é conveniente o uso de uma "Game Engine", já que motores de física, detecção de colisão, texturas, iluminação e sombreamento, são funcionalidades ofertadas por ela.

Segundo Álvaro Monteiro (2019), as tarefas para a realização de uma experiência AR são:

- "Modelagem de ambiente", que consiste em fazer uma réplica virtual do ambiente real onde rodará a AR;

- "Composição de cena AR", que é o alinhamento das réplicas 3D com o ponto de vista do usuário do ambiente em questão;

- "Simulação das condições ambientais", onde o AR processa elementos de iluminação e atmosfera;

- "Definition of the scene animation and user interactions", onde temos a animação de objetos e regras sobre a interação do aplicativo.

De acordo com Zakiah Noh e Mohd Shahrizal Sunar(2009), para adquirir fotorealismo alguns pontos são importantes:

- Consistência de geometria, que implica na posição correta do objeto em relação ao mundo real;

- Iluminação consistente, onde a iluminação, sombreamento e sombras devem ser consistentes com a iluminação no mundo real;

- Consistência de tempo, onde o mundo real e o virtual estão sincronizados.

Para termos um objeto virtual indistinguível do mundo real ele deve estar corretamente iluminado e sombreado, gerando sombras precisas. Sem sombras um objeto pode parece estar flutuando, mesmo que esteja sobre um superfície. A sombra, então, pode dar a percepção de profundidade e localização de um objeto. Além disso, sombras são importantes por fornecerem informações valiosas como posição, forma e tamanho. Estas informações são importantes para tornar o objeto virtual mais realista.

3 TRABALHO DESENVOLVIDO

Nesta seção é apresentado um breve descrição do que é AR, o que é uma "Game Engine", do porque da escolha da "Unity"descrição do aplicativo e descrição de sua construção.

3.1 Realidade Aumentada - AR

AR é uma tecnologia que visa misturar o mundo real com o mundo virtual, neste caso suplementando o mundo real com objetos virtuais gerados pelo computador (VAN KREVELEN; POELMAN, 2010). Desde então esta tecnologia tem tido aplicações nas mais variadas áreas, como na educação, militar, engenharia, medicina, museus, turismo, vídeo "games"e negócios. AR pode ser definida como um sistema que incorpora três caracteristicas basicas, segundo Van Krevelen e Ronald Poelman (2010):

- Combinação do mundo virtual com o mundo real;
- Interação em tempo real;
- Localizaçao de objetos do mundo real e virtual.

Um aspecto chave para a popularização da tecnologia AR é a popularização de "tablets"e "smartphones", uma vez que todos estes dispositivos dispõem de uma câmera e poder de processamento o suficiente para gerar uma interação em tempo real.

3.2 Game Engine

Para a implementação deste projeto utilizamos uma certa quantidade de recursos prontos para uso, uma vez que não iniciamos do ponto zero. Primeiramente, utilizamos um simulador de física. Depois, um código para a interpretação dos mais variados formatos de modelos 3D encontrados no mercado. Em seguida, "shaders"para a sombreamento e a capacidade de projeção de sombras. Uma "Game Engine"forneceu estes atributos.

Existem uma grande quantidade de "Game Engines"no mercado, atualmente. Cada uma delas com características e recursos que as tornam únicas, mas também com recursos similares como carregar modelos 3D, simulador de física e iluminação, entre outros. Mas então qual "Engine"escolher? Primeiramenete este projeto não contém nenhum financi-

amento, tampouco não há nenhuma projeção de angariar valores com ele. Porém, em casos diferentes, dependendo da "Engine"escolhida a empresa que detém esta pode vir a cobrar algum tipo de taxa, dependendo do lucro obtido com o projeto. Outra característica importante na escolha da "Game Engine"é para qual plataforma ela possue suporte, pois uma "Game Engine"pode ou não suportar várias pataformas. Algumas populares são PCs, consoles, "browser"e celulares "smartphones". Neste projeto em específico usaremos a plataforma de celulares tipo "smartphone", que utilizam o sistema operacional "Android"ou "Ios", mais especificamente. Então é preciso uma "Game Engine"que suporte plataformas "Android", "Ios"e que seja gratuito.

3.3 Escolhendo a Unity

Neste momento, então, apresentaremos a "Game EngineUnity". A "Unity"tem ganho grande popularidade principalmente em projetos de mais baixo orçamento, como em estúdios independentes, e é uma "Game Engine"relativamente fácil de usar, leve e com uma comunidade bastante ativa. Na "Unity", se o rendimento finaceiro for superior a 100.000 dólares é necessário pagar uma assinatura, o que não é nosso caso, evidentemente. Devido às definições financeiras já mencionadas, à comunidade ativa, ao editor leve e ao suporte às mais variadas plataformas, neste estudo usaremos o sistema operacional "Android"e "Ios"para smartphones.

Algumas das funcionalidades importantes da "Unity"para este projeto são comentadas a seguir.

A "Unity"nos fornece uma biblioteca para lidar com a realidade aumentada, esta biblioteca é a "AR foundation". Ela coloca a disposição todas as primitivas relacionadas à realidade aumentada. As chamadas primitivas são recursos fornecidos para lidar com AR. A "AR foundation"também é uma tecnologia relativamente nova e não funciona em todos os celulares "smartphone", sendo que apenas alguns modelos, relativamente recentes, suportam esta tecnologia.

A "Unity"já dispõe de um simulador de física pronto. E coloca à disposição duas primitivas para usar a física da "Unity", são o "Collider"e o "Rigidbody". Atribuindo estes dois componentes, a "Unity"fará o resto. Os "Colliders"definem o formado do corpo do objeto e, neste estudo, estaremos usando um cubo, que é uma das primitivas fornecidas pela "Unity". Então, com a adição do "Rigidbody", poderemos adicionar movimento da física a este corpo.

Outra grande vantagem do uso da Unity é que ela aceita vários formatos de arquivo de modelos 3D, tais como .OBJ .DAE .BLEND e etc. É costumeiro, na internet, encontrarmos vários tipos de modelos 3D, por isso é tão importante que a "Unity"saiba lidar com tantos formatos.

Um aspecto muito importante são os "shaders". Os "shaders"provêem efeitos de sombreamento para os objetos. Neste estudo, estaremos usando os "shaders"tanto para dar mais realismo na execução do aplicativo, como também um "shaders"especial para projetar sombras em um plano. Os "shaders"usados no projeto serão apresentados mais adiante.

A "Unity"dispõem de 3 tipos de sombra: "No Shadow", "Hard Shadow"e "Light Shadow".

- "No Shadow"Simplesmente não projeta sombra alguma, mas pode ser muito importante caso a máquina executando não tenha potência o suficiente para executar a projeção de sombras;
- "Hard Shadow"Projeta sombras, mas de maneira menos realista que a "Soft Shadow", normalmente com sombras mais quadriculadas. Também adiciona mais triângulos à cena, ou seja, exige mais processamento;
- "Soft Shadow"Projeta sombra mais realista que o "Hard Shadow", mas tem maior custo de processamento.

A "Unity"nos fornece também "Prefabs". "Prefabs"são objetos totalmente préconfigurados que a "Unity"usa para instanciar na cena, a qualquer momento. Também permite instanciar quantos "prefabs"forem necessários. Quando for necessário modificar um objeto, simplesmente podemos modificar o "prefab"que todas as modificações passam para todos os "prefabs", em todo o projeto.

3.4 Descrição do aplicativo

O objetivo do aplicativo desenvolvido foi mostrar uma certa quantidade de lixo, usando a realidade aumentada, para conscientizar as pessoas do lixo produzido em um certo tempo. Neste estudo construimos o núcleo deste aplicativo. Com este núcleo pronto, coletamos métricas para saber até onde poderíamos colocar modelos 3D ou quantos objetos o simulador de física conseguiria suportar para, então, mostrarmos uma certa quantidade de lixo produzido.

As principais caracteristicas do aplicativo são descritas a seguir.

Um dos objetivos do aplicativo é usar o simulador de física da "Unity"para simular uma pilha de lixo. O aplicativo cria objetos a uma certa altura, então esses objetos, com a ação simulada da gravidade, acabam caindo em um plano virtual e formam uma pilha construída com o acúmulo destes objetos.

Uma alternativa ao uso de pilha com simulador de física, é colocar os modelos 3D de madeira estática, neste caso as posições dos modelos 3D não variam e não há processamento do simulador de física, uma vez que não há simulação de gravidade.

Outra função importante do aplicativo é capturar métricas durante a sua execução. Tais métricas são: FPS, mémoria alocada, número de "colliders", número de triângulos, "timer"do tempo real e "frame time".

3.5 Construindo o aplicativo

Agora que temos as descrições, tanto da "Game Engine"quanto do aplicativo, podemos entrar em detalhes de sua construção.

3.5.1 Interface

A "Unity"fornece primitivas para a criação de uma interface. Neste projeto usamos a primitiva de "Button"para a criação de botões e mostradores de métricas.

Inicialmente temos na interface quatorze botões. Os seis primeiros, no topo da tela do aplicativo, é onde são colocadas as métricas. Elas são "Frame time", "FPS", "Colliders", "Time", "Memory"e "Triangles"

Na parte de baixo do aplicativo temos mais oito botões. A 3.1 desmostra o menu de configurações do aplicativo

O botão número 1 é o "Create Plane"e é usado para a criação de um plano virtual no "mundo real". Ele pode ter dois valores, "Create Plane"e "Delete Plane", quando não temos nenhum plano instanciado o valor do botão é "Create Plane", ou seja, estamos prontos para instanciar um plano no mundo virtual, mas quando já temos um plano instanciado no mundo virtual, o valor passa para "Delete Plane", neste caso o plano no mundo virtual já está instanciado e podemos então deletar esse plano.

O botão númro 2 é o "Rest". Ele serve para ressetar a configuraçoes de tipos de

Figura 3.1 – Menu de configurações do aplicativo

objetos a serem instanciados sob algum tipo de física, ressetar o número de objetos a serem instanciados e o tipo de sombra usada, e ressetar as informações contidas na lista de informações sobre as métricas.

O botão númro 3 é o "Write". Pressionando este botão colocamos em um arquivo "csv"todas as métricas adquiridas durante uma execução. Caso o arquivo possa ser escrito com sucesso, ele muda o texto para "success in write the file", caso contrário, o texto passa a ser "error in write the file".

O botão númro 4 é o "Static Record", apaga as informações da lista de informaçoes sobre métricas e começa uma nova. Neste caso não há criação de novos objetos, ou seja, ele adquire informações de uma pilha de objetos com número fixo.

O botão númro 5 é o "Set Object", faz parte de um grupo de três botões que muda sua cor de vermelho para verde para mostrar que possui uma configuração. Neste caso o botão "Set Object"abre uma nova lista de botões, precisamente seis: "Garbage Bag with physics", "Garbage Bag without physics", "Milk Box with physics", "Milk Box without physics", "Soup with physics"e "Soup without physics". Cada botão na nova lista indica um tipo de objeto,neste caso, um saco de lixo, uma caixa de leite e uma lata de sopa. Também se usa a física da "Unity"ou simplesmente é instanciado sem física. Assim que escolher um objeto e uma opção de física, o botão fica verde e com o texto do objeto da física escolhida.

O botão númro 6 é o "Set Number", também faz parte do grupo de botões que mudam de cor. Neste caso, ele abre uma nova lista de botões com valores numéricos que indicam até que quantidade de objetos o aplicativo deve instanciar antes de parar, mudando sua cor para verde e com o texto do número selecionado.

O botão númro 7 é o "Set Shadow", também faz parte do grupo que muda de cor. Neste caso ele abre uma lista de três novos botões: "No Shadow", "Hard Shadow"e "Soft Shadow". O nome escolhido vai para o texto do botão e ele fica verde.

O botão númro 8 é o "On". Se tivermos um plano virtual instanciado e todos os botões, do grupo de botões coloridos, estiverem verdes, então o aplicativo começa a instanciar o objeto configurado e o tipo de física. O texto muda para "Off"e desta maneira a instanciação de objetos pode ser parada a qualquer momento. A captação de métricas tambem inicia.

3.5.2 Detecção de planos e instanciação de plano no mundo virtual

Antes de começar qualquer instanciação dos objetos selecionados é vital que tenhamos um plano instanciado no mundo virtual, para que os objetos colidam com este plano, ao caírem. Graças à "AR foundation"isto pode ser resolvido. No aplicativo, primeiramente adquirimos a posição da câmera, depois usamos um raio que sai da posição da câmera e verificamos se houve uma colisão nos planos da "vida real". Ocorrendo uma colisão, colocamos um marcador no ponto desta colisão e validamos como ponto válido. A cada frame fazemos esta execução, ou seja, colocamos o marcador na nova posição de modo que o usuário pode escolher a posição que lhe agrade. Uma vez que tenhamos uma posição válida (foi dectectada uma colisão com planos) o usuário finalmente pode instanciar um plano no mundo virtual que vai interagir com a física do aplicativo. Neste projeto, o plano no mundo virtual para a física é invisivel.

3.5.3 Formação de pilha com física

Uma vez que tenhamos um plano no mundo virtual para a física é interessante formar uma pilha sobre ele. Uma vez que tenhamos um plano no mundo virtual instanciado podemos usar a sua localização para saber onde instanciar os objetos que formarão a pilha. Primeiramente temos que adicionar um simulador de física aos objetos. Na "Unity"isso pode ser feito com três primitivas, "collider", "rigidbody"e "mesh render". O "collider"define o formato físico do objeto, que pode ser um cubo, uma esfera ou até um formato customizado por um arquivo de modelo 3D. Já o "rigidbody"é responsável pelo movimento na física, como a gravidade e a colisão com outro "colliders". Já o "mesh ren-

der"não é necessariamente vital, mas é necessário uma informação visual para testemunharmos o que está acontecendo. Com o objeto dotado das primitivas que mencionamos podemos colocar este objeto em um "prefab"que será instanciado múltiplas vezes. Com o "prefab"e o plano de colisão podemos finalmente começar a formar a pilha. Neste projeto pegamos a posição do plano de colisão e instanciamos cinco objetos a uma certa altura, então é só deixar o simulador de física da "Unity"trabalhar. A instanciação dos cinco objetos é feita a cada um segundo e o aplicativo continuará a instanciar objetos até que o contador de objetos chegue a um número definido. Exemplos de como são formadas as pilhas com física podem ser encontradas nas figuras 3.2 3.3 3.6 3.8 3.9. Durante a captura de métricas notamos que quando é formado pilha de lixo com física que exauri o dispositivo de modo que o gargalo seja definido pe físca, quando o "colliders"param de se mexer o FPS volta a normalidade, ou seja, exige menos processamento.

3.5.4 Formação de pilha sem física

Nem sempre gostaríamos de fazer uma pilha com física, sendo adequado ter outra opção para mostrar o lixo produzido. Esta outra opção seria mostrar os objetos sem física, ou seja, retirarmos o componente "rigidbody"dos objetos e ficarmos apenas com o "colliders"e o "mesh render". A forma como mostraremos esta nova forma de pilha é colocar os objetos em uma posição estática. Ainda precisaremos do plano físico pois precisamos de sua localização para instanciar os objetos onde o usuário quer. A maneira de como colocar os objetos segue uma matriz tridimensional, com um mesmo tamanho para a altura, largura e profundidade. O aplicativo começa colocando um objeto na posição inicial da matriz, o segundo objeto e os seguintes são instanciados a uma certa distância do anterior. Esta distância é totalmente configurável antes do aplicativo ser compilado, sendo que os objetos instanciados assim formam uma especie de cubo, pois primeiro são estanciados na linha da largura, e uma vez que a largura é exaurida ele incrementa em 1 a profundidade, seguindo o mesmo para a altura. Uma vez que a matriz esteja totalmente preenchida o aplicativo não coloca mais nenhum novo objeto. Exemplos de pilhas sem física são mostradas em 3.4 3.5 3.7 3.10

3.5.5 Selecionando sombras

Uma parte importante do aplicativo é a seleção de sombras. Primeiramente o plano no mundo virtual que nós já instanciamos tem um "shader"especial, este "shader"torna o plano invisível, mas com a capacidade de projetarmos sombra nele, ou seja, passa a impressão de que as sombras estão sendo projetadas em um plano no mundo real. A "Unity"provê três tipos de sombra, como já foi mencionado, mas durante a execução foram notadas duas situações, que descrevo a seguir. A primeira é que ao mudar da configuração "No Shadow"para "Hard Shadow"temos a adição de novos triângulos à cena, o que observado em certos casos pode deixar o aplicativo mais pesado no quesito processamento. Outro aspecto importante é que foi notado que não existe diferença entre o "Hard Shadow"e o "Soft Shadow": não importa qual escolher os resultados visuais e número de triângulos é o mesmo. É notável, neste ponto, que o uso de sombras aumenta o realismo na cena (NOH e SUNAR, 2009) embora isso aumente o consumo de processamento, pode valer a pena para um maior efeito de realismo.

3.5.6 Adquirindo métricas

O segundo ponto chave deste estudo é a aquisição de métricas. É através delas que podemos encontrar gargalos e possíveis configurações que o "hardware"possa suportar. Primeiramente, para a captura de métricas o "Unity"fornece uma API, ela se chama "Profiler Recorder", através dela a "Unity"fornece informações dos mais diferentes tipos, infelizmente a documentação sobre esta API é bastante escassa o que dificultou o seu uso. Outra métrica muito importante é o FPS, esta métrica é obtida através do "Time.deltaTime". A primeira métrica é o "Frame Time", esta métrica é relativa ao tempo de processamento do frame, esta métrica pode variar dependendo dos triângulos na cena ou do processamento da física dos objetos. A segunda métrica é o número de "colliders", esta métrica não é obtida através do "Profiler Recorder", mas sim do comando "FindGameObjectsWithTag"que indentifica todos os objetos na cena com uma determinada "flags"e coloca em um "array", deste modo sabendo o tamanho do "array"saberemos o número de "colliders"na cena. A terceira métrica é o numero de triângulos e através do "Profiler Recorder"a "Unity"pode nos informar quantos triângulos estão sendo renderizados. Este número varia bastante devido à localização e rotação da câmera. A quarta métrica é a memória, também através do "Profiler Recorder"a "Unity"nos informa a quan-

tidade de memória usada em "bytes". A quinta métrica é o FPS e esta não usa o "ProfilerRecorder", mas sim o "Time.deltaTime"que fornece o tempo entre o frames e com ele podemos determinar o número de frames por segundo. Por último, temos o tempo que é simplesmente um "timer"para ser usado como referência de tempo.

3.5.7 Log e escrevendo métricas em um arquivo

Durante a execução da simulação de pilha ou pressionando o botão "start the static record", a cada segundo é colocado em uma lista todas as métricas já mencionadas. Desta forma temos uma tabela com essas métricas. E, a qualquer momento, podemos pressionar o botão "Write"para colocar toda essa tabela em um arquivo chamado "data.csv". Este arquivo contém a informação do "log"no formato "csv"o que possibilita que ele seja exportado para um planilha do "Google Sheets"que será usado para criação dos gráficose e tabelas dos resultados.

Figura 3.2 – Exemplo de uma pilha de sacos de lixo usando a física da "Unity"e com sombras "No Shadow"

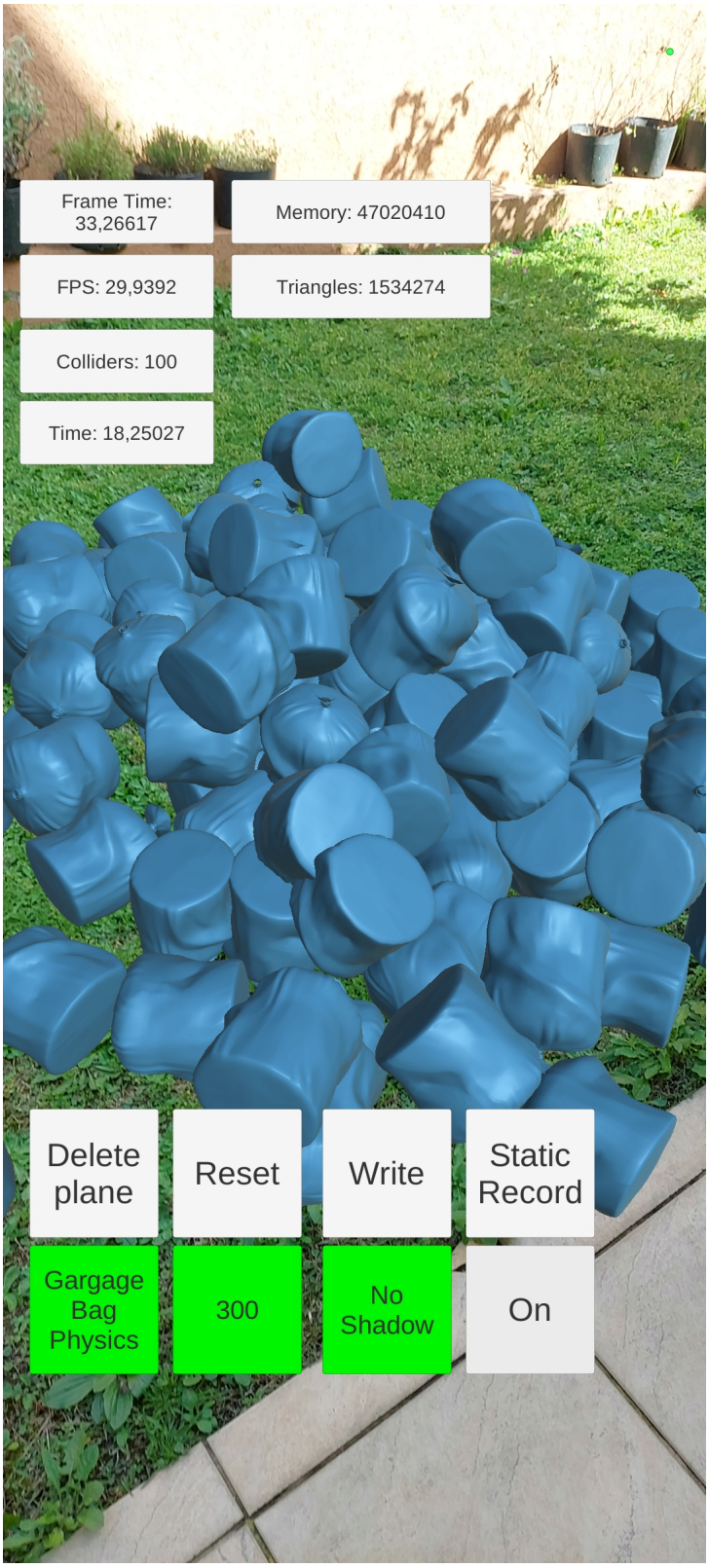


Figura 3.3 – Exemplo de uma pilha de sacos de lixo usando a física da "Unity"e com sombras "Hard Shadow"

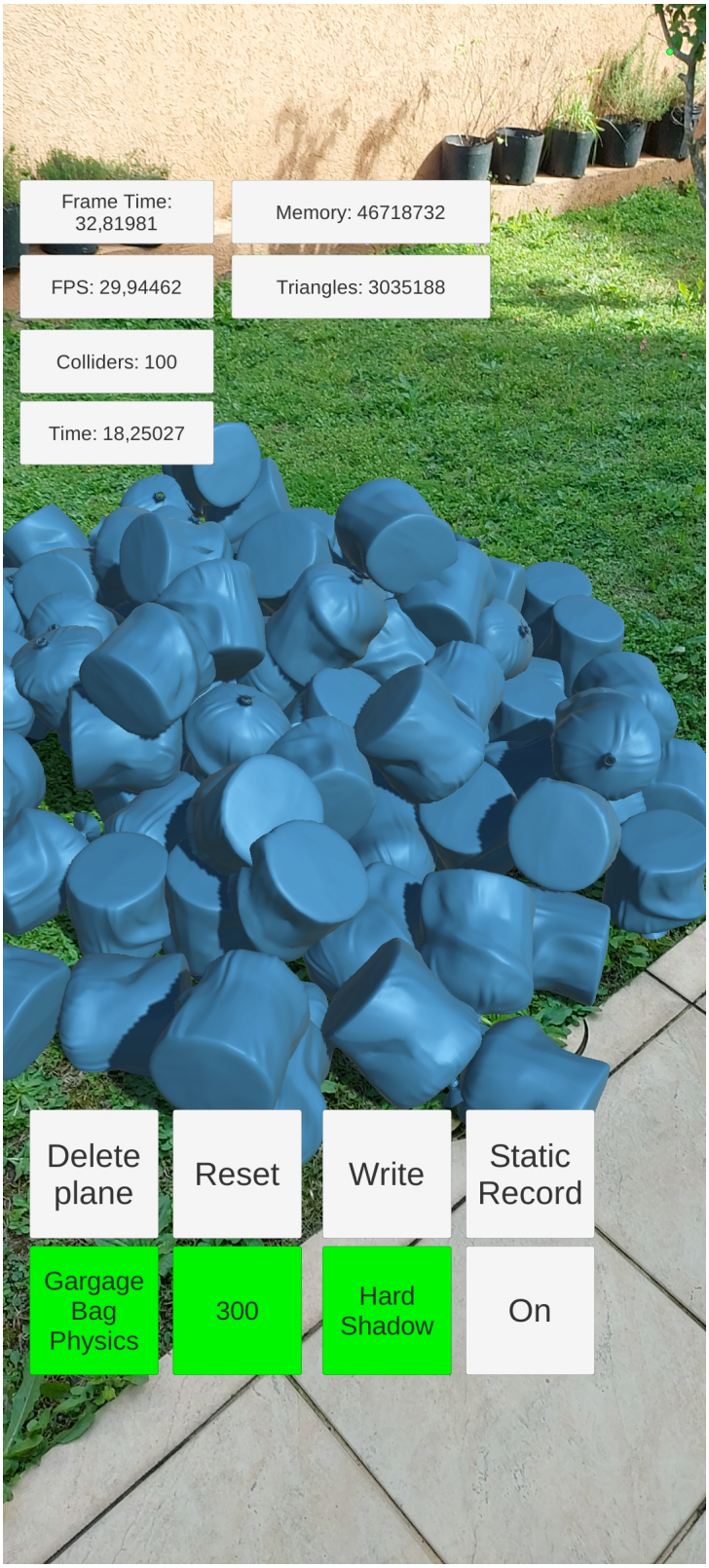


Figura 3.4 – Exemplo de uma pilha de sacos de lixo sem usar a física da "Unity"e com sombras "No Shadow"

Figura 3.5 – Exemplo de uma pilha de sacos de lixo sem usar a física da "Unity"e com sombras "Hard Shadow"

Figura 3.6 – Exemplo de uma pilha de caixas de leite usando a física da "Unity"e com sombras "Hard Shadow"

Figura 3.7 – Exemplo de uma pilha de caixas de leite sem usar a física da "Unity"e com sombras "Hard Shadow"

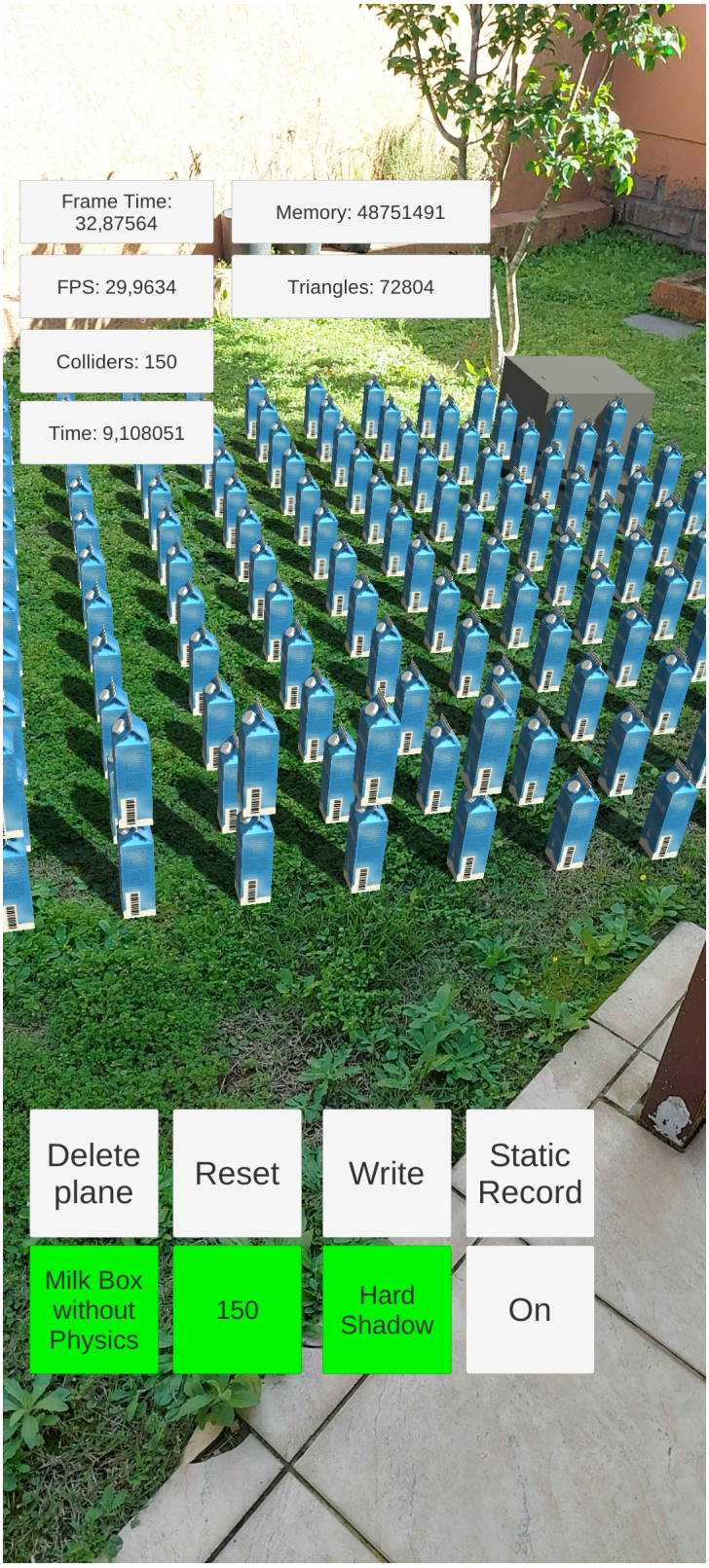


Figura 3.8 – Exemplo de uma pilha de latas de sopa usando a física da "Unity"e com sombras "No Shadow"

Figura 3.9 – Exemplo de uma pilha de latas de sopa usando a física da "Unity"e com sombras "Hard Shadow"

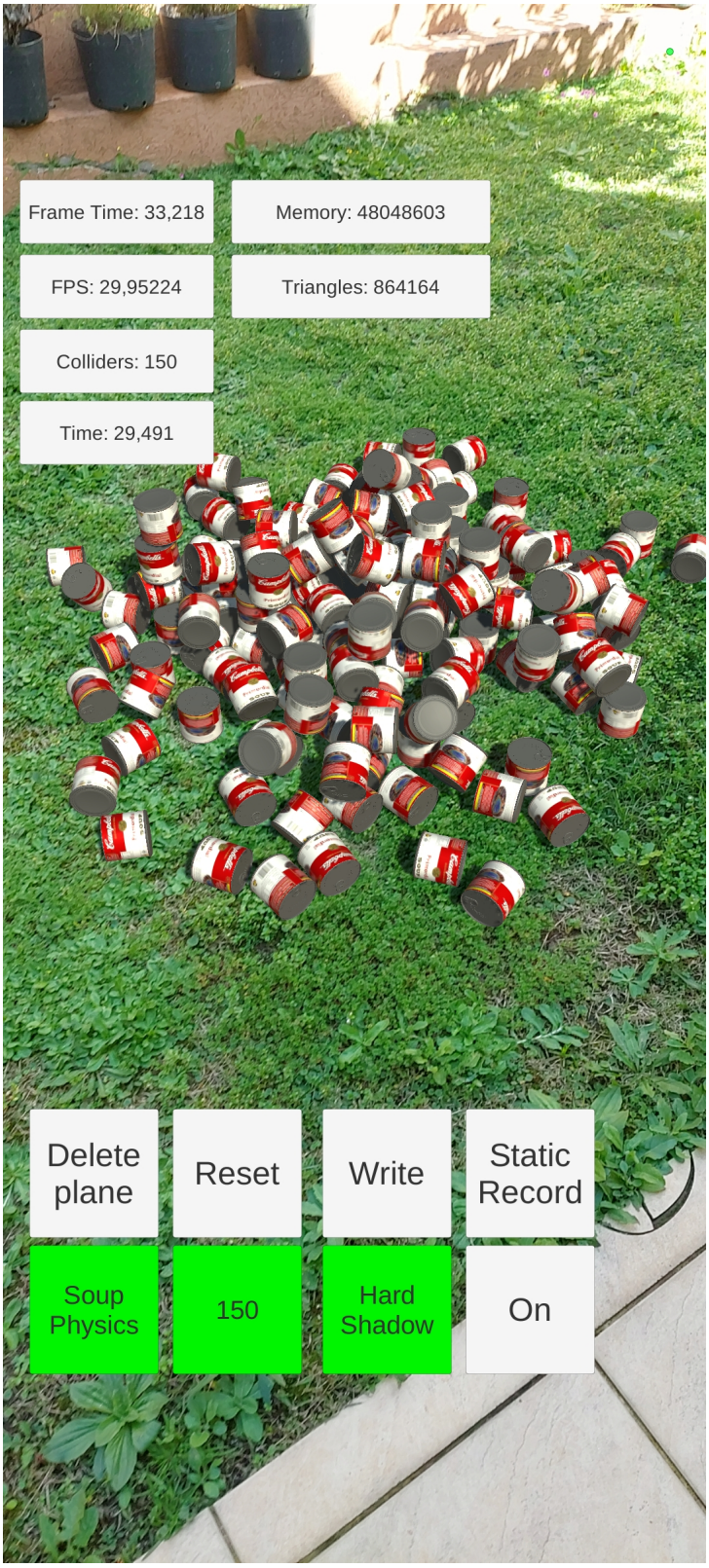
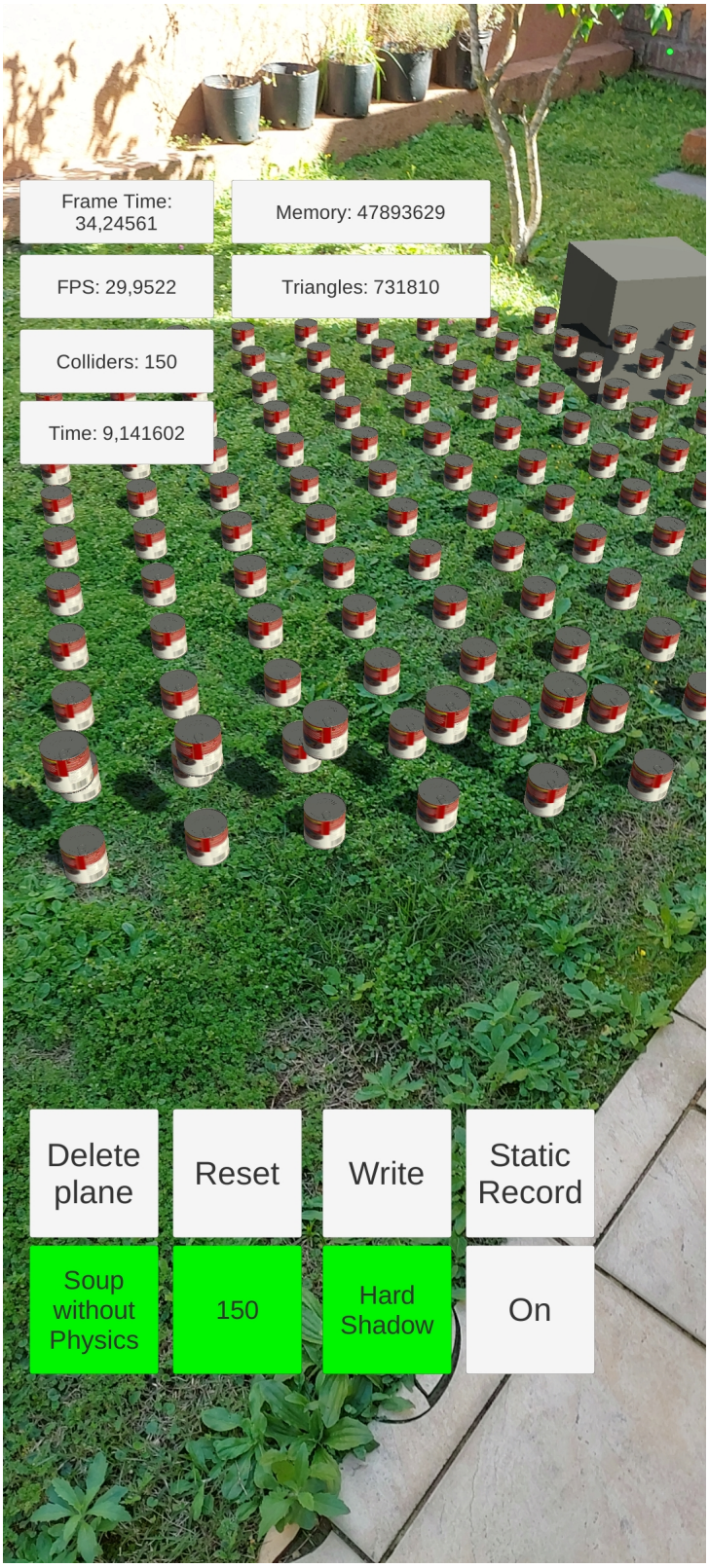


Figura 3.10 – Exemplo de uma pilha latas de sopa sem usar a física da "Unity"e com sombras "Hard Shadow"



4 RESULTADOS

Nesta seção discutiremos os dados coletados das métricas já mencionadas, todos no formato de gráficos e tabelas, todos os dados foram extraídos do uso do aplicativo.

4.1 Configurações do "hardware"usado nos teste

Todos os testes feitos neste artigo usaram um "smartphone"Samsung Galaxy a52. Seus dados técnicos são:

- *Processador*: 2x 2.3 GHz Kryo 465 Gold + 6x 1.8 GHz Kryo 465 Silver
- *Chipset*: Snapdragon 720G Qualcomm SM7125
- *GPU*: Adreno 618
- *RAM*: 6 GB
- *Resolução da Câmera*: 9238 x 6928 pixel

4.2 Metodologia usada nos teste

Todos os resultados dos testes são oriundos das métricas capturadas já mencionadas. Para a demonstração dos resultados serão apresentados gráficos com as métricas capturadas.

As principais métricas para medir o desempenho instantâneo do aplicativo são o FPS e o "frame time". Estas métricas são espelhadas, ou seja, quando o FPS desce o "frame time"sobe. Usaremos estas métricas para dizer até onde podemos colocar "colliders", triângulos e memória na cena.

Para diferenciar um gargalo de triângulos de um gargalo de "colliders"usamos a posição da câmera. Se o gargalo for o número de triângulos então o FPS vai baixar, mas quando a câmera for movida de modo que não mostre mais triângulos então o número de triângulos vai baixar e o FPS voltará à normalidade. Caso o gargalo seja o simulador de física, quando movemos a câmera de modo que não vemos mais triângulos, porém o FPS ainda continuar baixo, então sabemos que o gargalo é a física.

Os modelos 3D usados nestes testes foram um saco de lixo com mais de 16314 triângulos, uma caixa de leite com 148 triângulos e uma lata de sopa com 1600 triângulos.

4.3 Resultados preliminares

Foi através dos resultados preliminares que descobrimos os pontos de gargalo e então definimos uma janela para os gráficos. Também foi descoberto que ao usar "hard shadow"averia um adição de triângulos a cena .

4.4 Compreendendo os gráficos

Seram apresentados gráficos normalizados para provar as relações entre as métricas, além trechos das tabelas usadas para a geração dos gráficos. A linha vertical roxa nos gráficos indica o ponto de gargalo em determinada configuração.

4.4.1 Detecção de gargalos em pilhas com física

Os três primeiros gráficos, 4.1 4.2 4.3, mostram a formação de uma pilha de objetos 3D usando "colliders"e "rigiddbody", ou seja, utilizando a física da "Unity". Podemos notar que as curvas de FPS e "frame time"tem períodos de estabilidade, ou seja, o FPS continua alto enquanto o "frame time"continua baixo, e este é um ponto importante porque queremos saber até onde é estável colocar triângulos e "colliders".

Um dado muito importante são os últimos pontos de estabilidade. Estes pontos podem ser encontrados nas tabelas que formaram os gráficos. O FPS é considerado estável quando seu valor é por volta de 29,9 e o "frame time"é por volta de 33 ms. Estes pontos nos informam a configuração máxima possível sem o FPS ficar abaixo de 29,9, ou seja, até onde podemos colocar objetos no aplicativo sem que ele fique lento.

Fragmentos das tabelas usadas para a geração os gráficos podem ser vistas em 4.1 4.2 4.3. Estes fragmentos contém o último ponto de estabilidade de cada gráfico.

Os últimos pontos estáveis são:

- *Sacos de lixo*: 335 colliders, 4960228 triângulos, 49145680 memória, 29,96635 FPS, 34,74287 frame time.
- *Caixas de leite*: 735 colliders, 200914 triângulos, 57109274 memória, 29,95453 FPS, 31,85948 frame time.
- *Latas de sopa*: 815 colliders, 2239832 triângulos, 56083878 memória, 29,94702 FPS, 34,26259 frame time.

Figura 4.1 – Gráfico mostrando a formação de uma pilha de Sacos de lixo usando a física da "Unity"e com configuração de sombra "No Shadow"

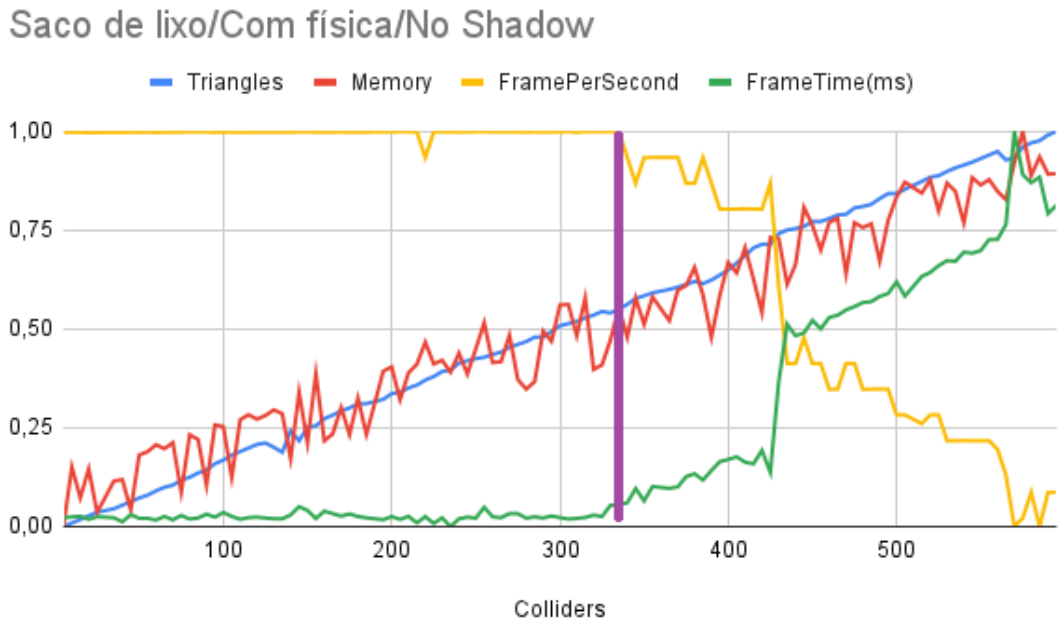


Tabela 4.1 – Fragmento da tabela mostrando a formação de uma pilha de Sacos de lixo usando a física da "Unity"e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
320	4829716	47985268	29,95853	33,65768
325	4911286	48069344	29,96029	33,51755
330	4878656	48535414	29,96231	34,65836
335	4960228	49145680	29,96635	34,74287
340	5074426	48586418	28,96349	34,91403
345	5204938	49326982	27,95995	36,37549
350	5270194	48833412	28,95494	35,12009

Devido aos métodos já mencionado para indentificar se o gargalho é gráfico ou físico podemos inferir que os gargalos da caixa de leite e da lata de sopa são a físicos, enquanto o gargalo do saco de lixo é gráfico.

Os próximos três gráficos a seguir, 4.4 4.5 4.6 tem a mesma proposta que os anteriores, mas agora com a configuração de sombra em "Hard Shadow". Vale lembrar que o "Hard Shadow"adiciona mais triângulos à cena, o que pode mudar os gargalos de física para gráfico.

Segundo as tabelas 4.4 4.5 4.6 os novos últimos pontos de estabilidade são:

- *Sacos de lixo*: 165 colliders, 5090754 triângulos, 47526214 memória, 29,94872 FPS, 35,56477 frame time.

Figura 4.2 – Gráfico mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "No Shadow"

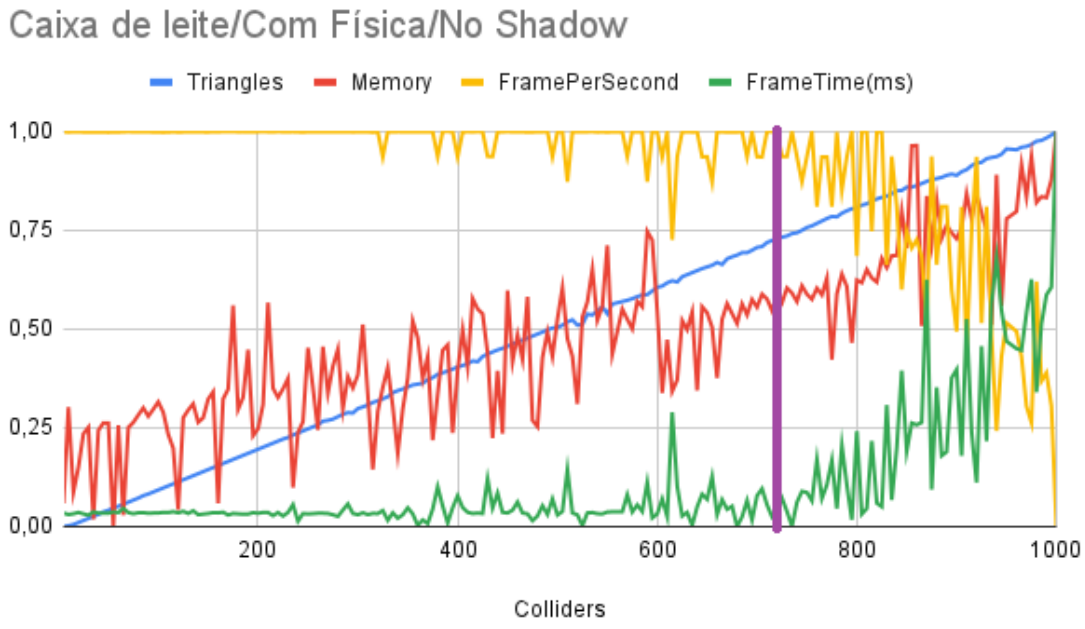


Figura 4.3 – Gráfico mostrando a formação de uma pilha de Latas de sopa usando a física da "Unity"e com configuração de sombra "No Shadow"

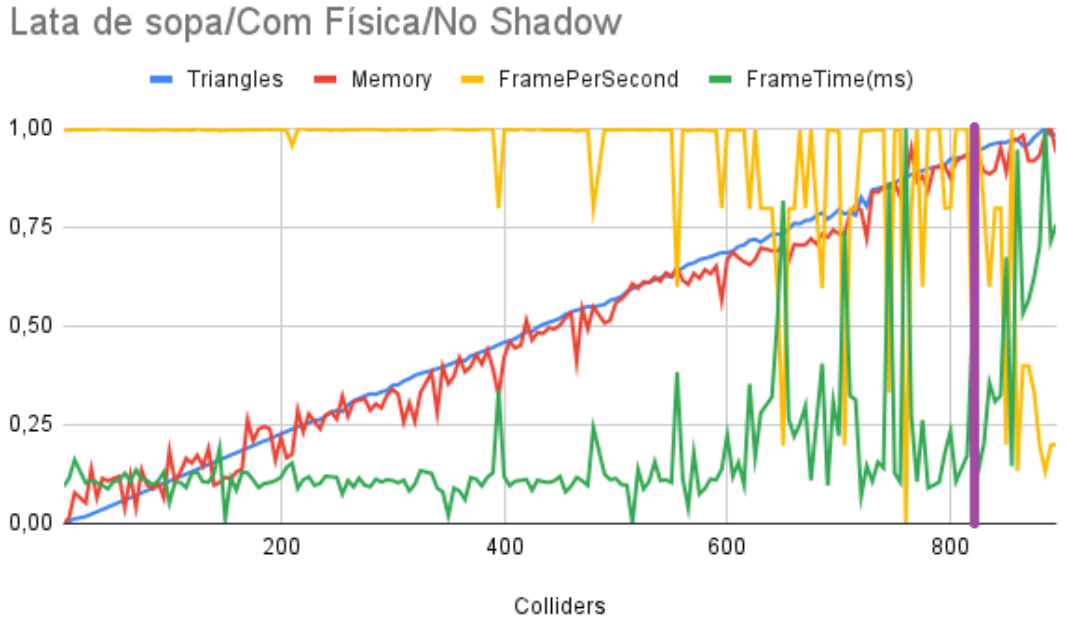


Tabela 4.2 – Fragmento da tabela mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
720	197578	57096542	29,96703	33,72012
725	198410	57008466	28,96185	35,15368
730	199246	57161338	28,96175	33,57641
735	200914	57109274	29,95453	31,85948
740	201470	57021190	28,95806	34,66367
745	203136	57183970	27,95627	35,95419
750	205084	57096754	28,96714	35,82148

Tabela 4.3 – Fragmento da tabela mostrando a formação de uma pilha de Latas de sopa usando a física da "Unity"e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
795	2188028	55741306	28,9486	34,34604
800	2222562	55371378	28,95297	35,11223
805	2228320	55873242	29,95614	33,35946
815	2239832	56083878	29,94702	34,26259
820	2259978	55485534	26,94357	38,94367
825	2277246	56029266	29,95346	33,72023
830	2288758	55577938	28,94265	34,67367

- *Caixas de leite*: 825 colliders, 440034 triângulos, 57370032 memória, 29,96243 FPS, 32,69766 frame time.
- *Latas de sopa*: 560 colliders, 3178100 triângulos, 52933467 memória, 29,94926 FPS, 33,49007 frame time.

A lata de sopa agora mudou o gargalo de físico para gráfico graças ao "Hard Shadow".

Tabela 4.4 – Fragmento da tabela mostrando a formação de uma pilha de sacos de lixo usando a física da "Unity"e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
150	4633964	47035195	29,95328	33,74599
155	4813418	47825158	29,96807	33,69581
160	4976558	48099632	29,97183	33,55874
165	5090754	47526214	29,94872	35,56477
170	5302836	47681349	28,94462	34,25604
175	5400722	47822194	28,94648	35,12514
180	5580176	47500893	27,95278	37,19634

Figura 4.4 – Gráfico mostrando a formação de uma pilha de sacos de lixo usando a física da "Unity"e com configuração de sombra "Hard Shadow"

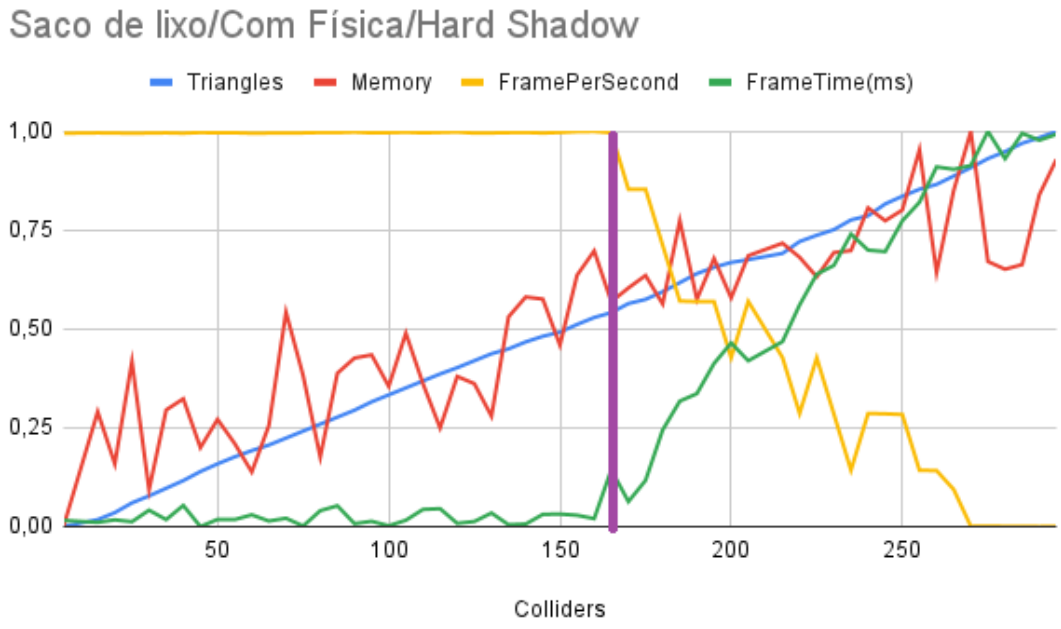


Figura 4.5 – Gráfico mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "Hard Shadow"

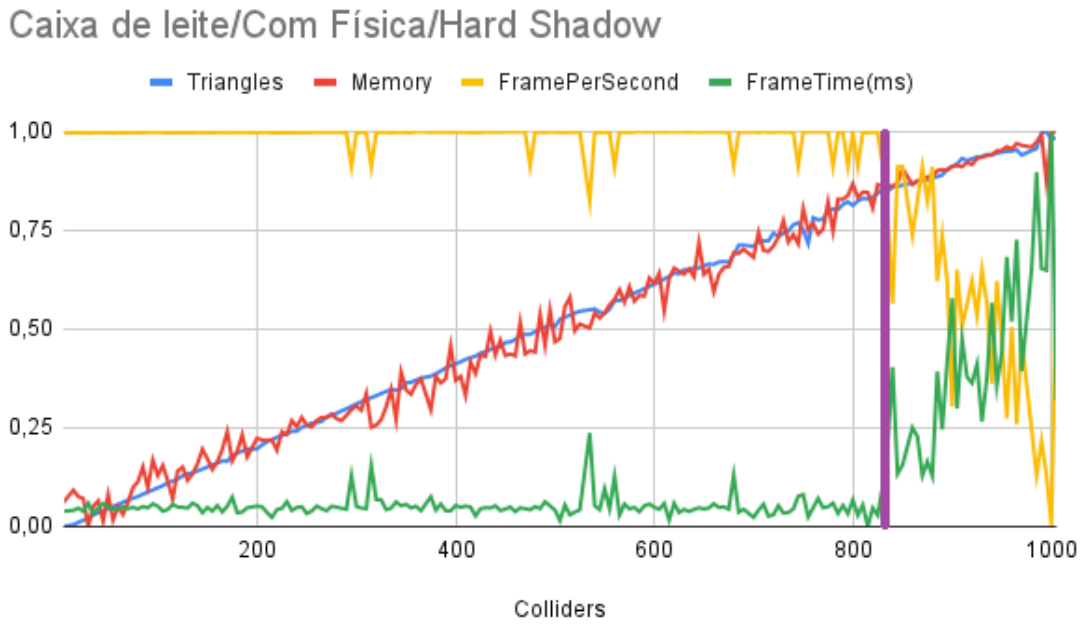


Figura 4.6 – Gráfico mostrando a formação de uma pilha de latas de sopa usando a física da "Unity"e com configuração de sombra "Hard Shadow"

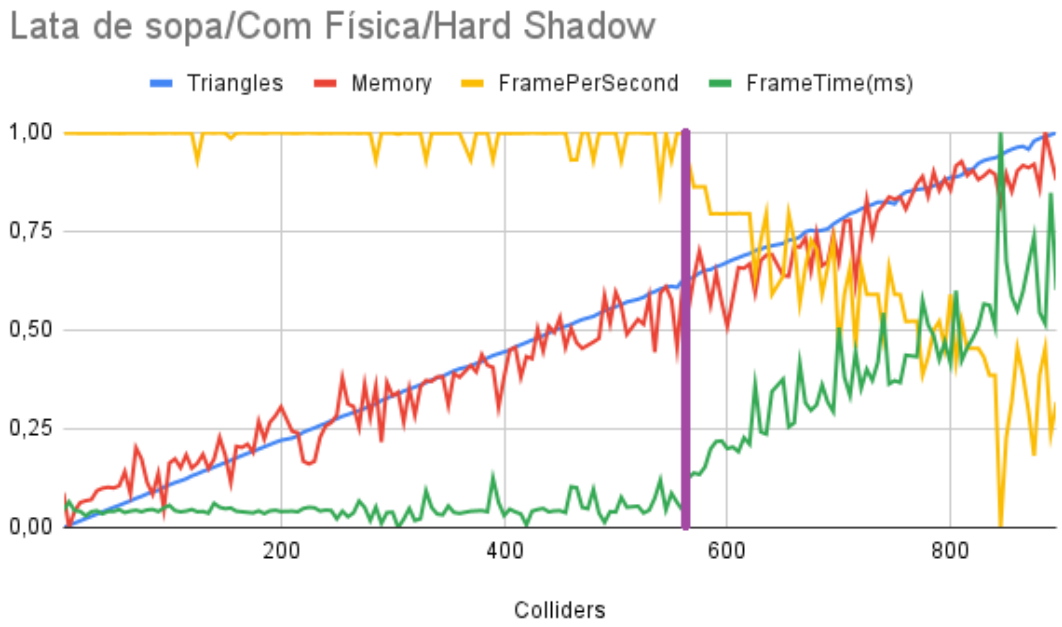


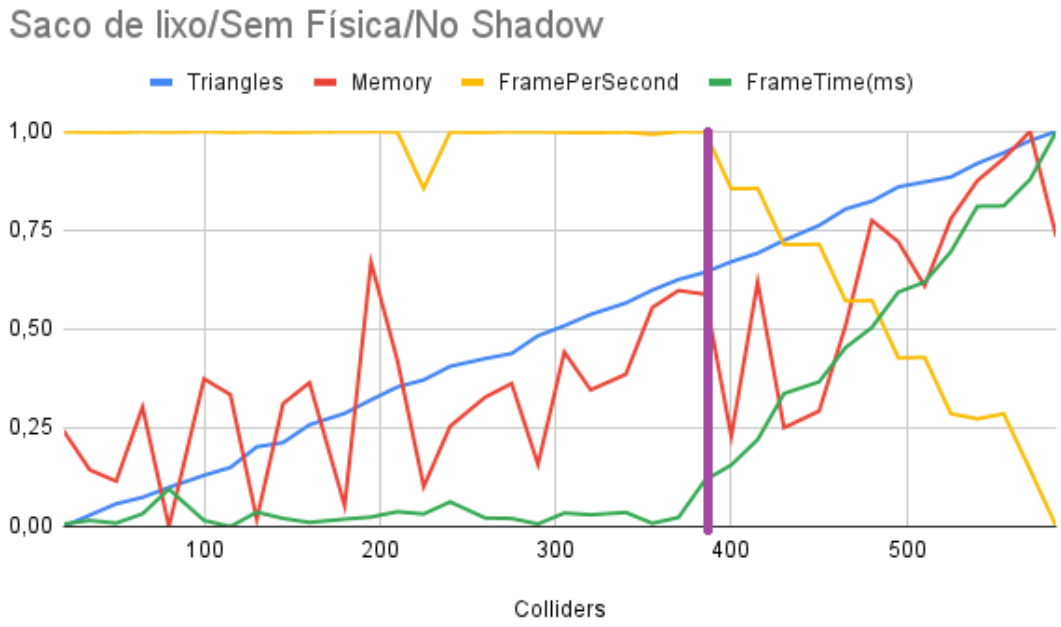
Tabela 4.5 – Fragmento da tabela mostrando a formação de uma pilha de caixas de leite usando a física da "Unity"e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
810	433362	57107151	29,96251	33,57944
815	432804	57093442	29,94901	31,71799
820	435030	56544378	29,96265	33,32907
825	440034	57370032	29,96243	32,69766
830	446148	57288352	28,9548	35,17091
835	442258	57351620	27,61942	38,62061
840	448374	57312263	24,96264	45,49874

Tabela 4.6 – Fragmento da tabela mostrando a formação de uma pilha de latas de sopa usando a física da "Unity"e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
545	3091760	52906132	29,94835	33,54906
550	3103272	52494041	28,95696	35,16244
555	3088894	50983652	29,95334	34,26047
560	3178100	52933467	29,94926	33,49007
565	3192490	52127834	28,95124	36,6318
570	3221270	53194247	27,95274	37,15585
575	3273062	54020316	27,96094	37,00756

Figura 4.7 – Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "No Shadow"



4.4.2 Detecção de gargalos em pilhas sem física

Queríamos testar apenas a potência gráfica do dispositivo, desta maneira retiramos o elemento "rigidbody", assim os objetos foram mostrados de maneira estática, ou seja, o desempenho é exclusivo do poder gráfico. Neste caso, é importante notar que não conseguimos mostrar modelos de caixas de leite o suficiente na tela para causar uma queda no FPS. Os gráficos relativos ao gargalo sem física são 4.7 4.8.

Segundo as tabelas 4.7 4.8 os novos últimos pontos de estabilidade são:

- *Sacos de lixo*: 385 colliders, 6004338 triângulos, 46758624 memória, 29,94822 FPS, 34,85459 frame time.
- *Caixas de leite*: não foi possível colocar na tela caixas de leite o suficiente para baixar o FPS;
- *Latas de sopa*: 1565 colliders, 3863052 triângulos, 50883066 memória, 29,94083 FPS, 33,61175 frame time.

Queríamos repetir o mesmo teste anterior, mas agora colocando configurações de sombra "Hard Shadow". Com os gráficos 4.9 4.10.

Segundo as tabelas 4.9 4.10, os últimos pontos de estabilidade são:

Figura 4.8 – Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "No Shadow"

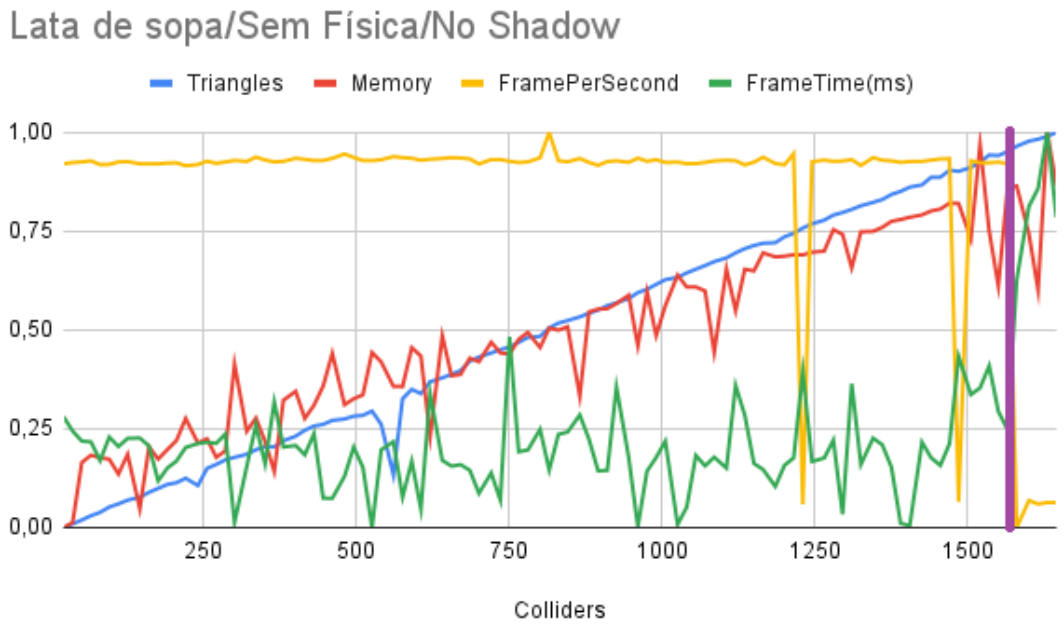


Tabela 4.7 – Fragmento da tabela mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
340	5302836	46153866	29,944	33,59962
355	5596488	46657150	29,90609	33,17533
370	5841198	46786363	29,95173	33,40402
385	6004338	46758624	29,94822	34,85459
400	6249048	45684540	28,94056	35,44729
415	6444816	46850428	28,94741	36,43564
430	6738466	45750116	27,95296	38,23131

Tabela 4.8 – Fragmento da tabela mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
1520	3753690	51654742	29,94285	33,99678
1535	3828518	50095131	29,94315	34,19985
1550	3822762	49110008	29,94498	33,7797
1565	3863052	50883066	29,94083	33,61175
1580	3917736	50908647	28,87273	34,99333
1600	3969538	50043944	28,95203	35,67187
1615	3989686	49041032	28,94191	35,83565

Figura 4.9 – Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "Hard Shadow"

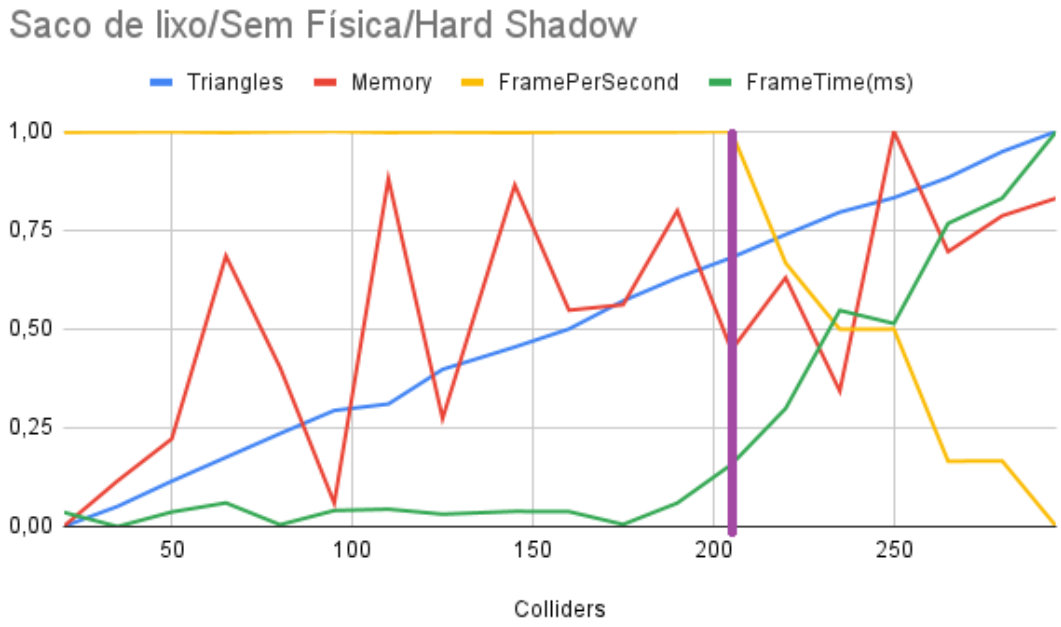


Tabela 4.9 – Fragmento da tabela mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
160	4878698	46262065	29,95666	33,30033
175	5514946	46284375	29,95661	32,84239
190	6036994	46666171	29,95777	33,60558
205	6493786	46099844	29,96477	34,98148
220	7015834	46393753	27,97029	36,97812
235	7521568	45934039	26,96421	40,4778
250	7847848	46988057	26,96735	40,00999

- *Sacos de lixo*: 205 colliders, 6493786 triângulos, 46099844 memória, 29,96477 FPS, 34,98148 frame time.
- *Caixas de leite*: novamente não foi possível colocar na tela caixas de leite o suficiente para baixar o FPS;
- *Latas de sopa*: 870 colliders, 4012724 triângulos, 49440750 memória, 29,93663 FPS, 34,62658 frame time.

Figura 4.10 – Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "Hard Shadow"

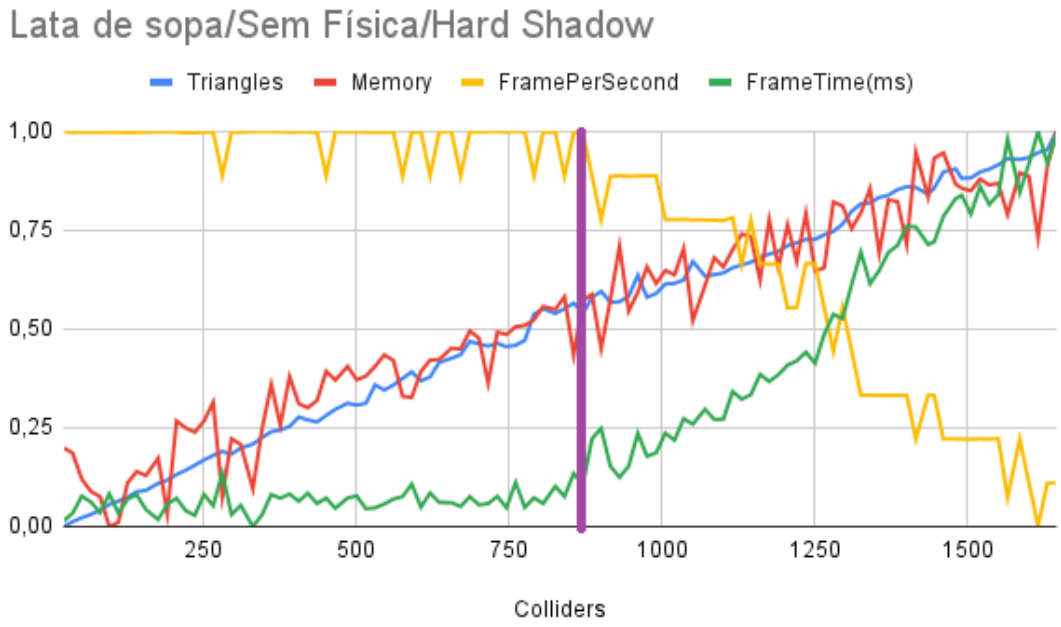


Tabela 4.10 – Fragmento da tabela mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
825	4012722	49266314	29,94947	34,53667
840	4101942	49511279	28,9413	34,01949
855	4202672	48372970	29,94238	35,19842
870	4012724	49440750	29,93663	34,62658
885	4309158	49559137	28,94604	36,97747
900	4421400	48502188	27,9467	37,50651
915	4219940	49454988	28,93708	35,57712

Figura 4.11 – Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow"

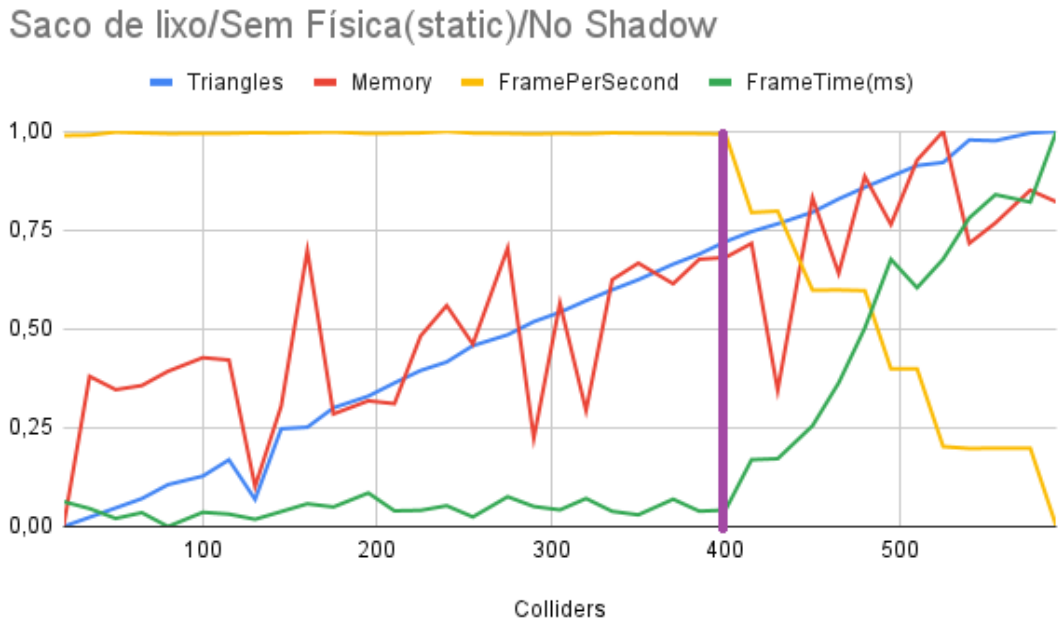


Tabela 4.11 – Fragmento da tabela mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
335	5107066	46719129	29,96049	33,35248
350	5319150	46843425	29,95567	33,24931
370	5645430	46688802	29,9531	33,71374
385	5857512	46872292	29,95038	33,35765
400	6118536	46885599	29,94461	33,39039
415	6330618	46991160	28,94803	34,88971
430	6493758	45890612	28,96552	34,91925

4.4.3 Flag static

Agora que obtivemos as informações sobre as pilha sem física, seria interessante usar a "flagstatic". A "Unity"prove para cada objeto uma "flagstatic", esta "flag"indica que o objeto não vai se movimentar durante a execução do aplicativo, deste modo economizando o processamento.

Podemos notar que os gráficos e as tabelas não mostram uma diferença significativa, o que indica que o uso do "static"não é significativo.

Figura 4.12 – Gráfico mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow"

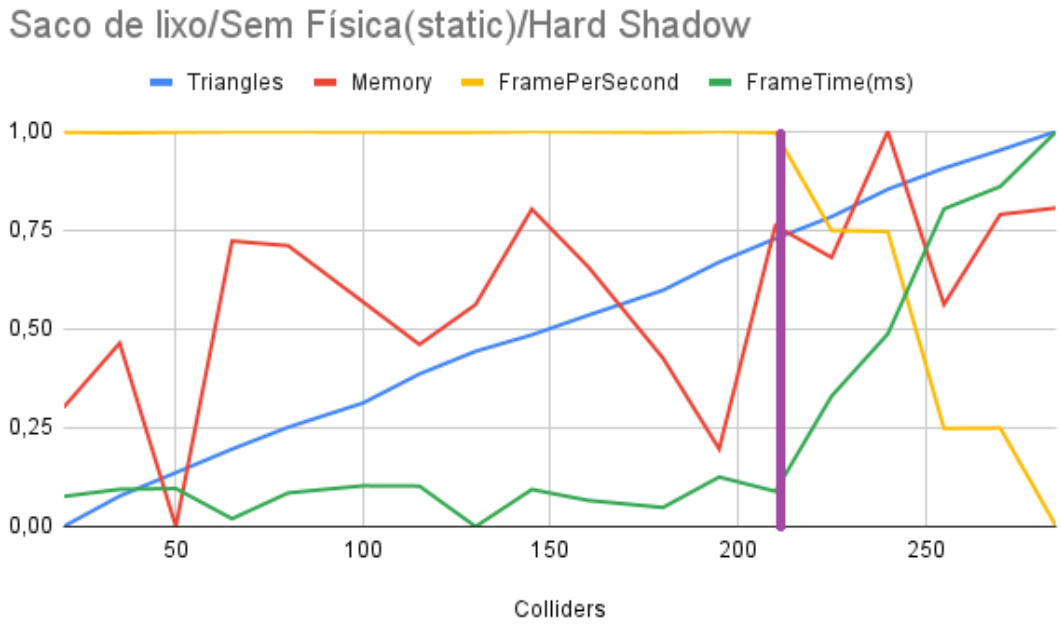


Figura 4.13 – Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow"

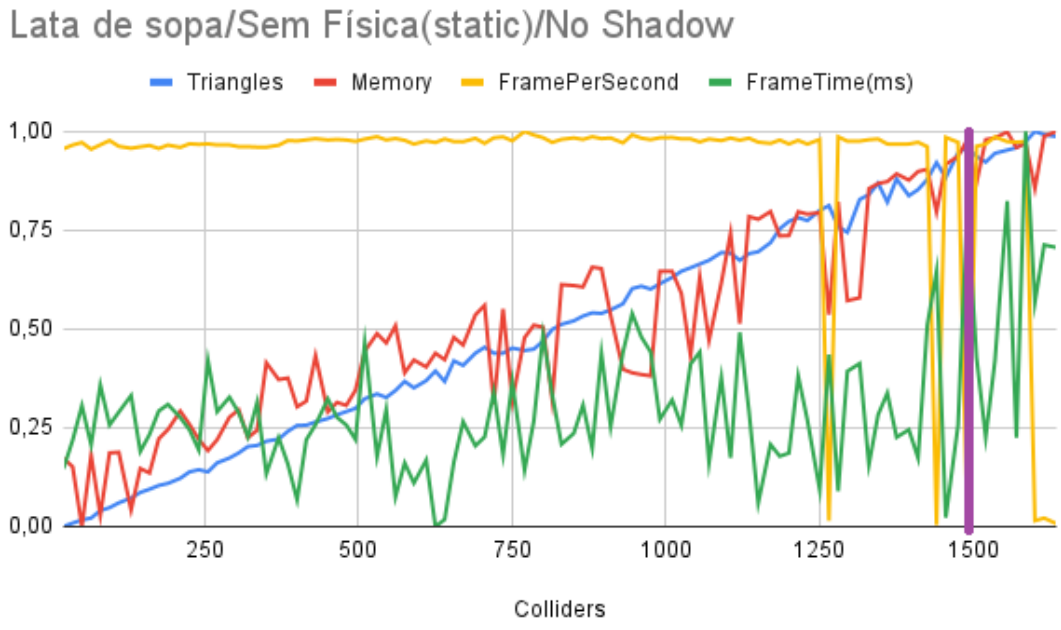


Figura 4.14 – Gráfico mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow"

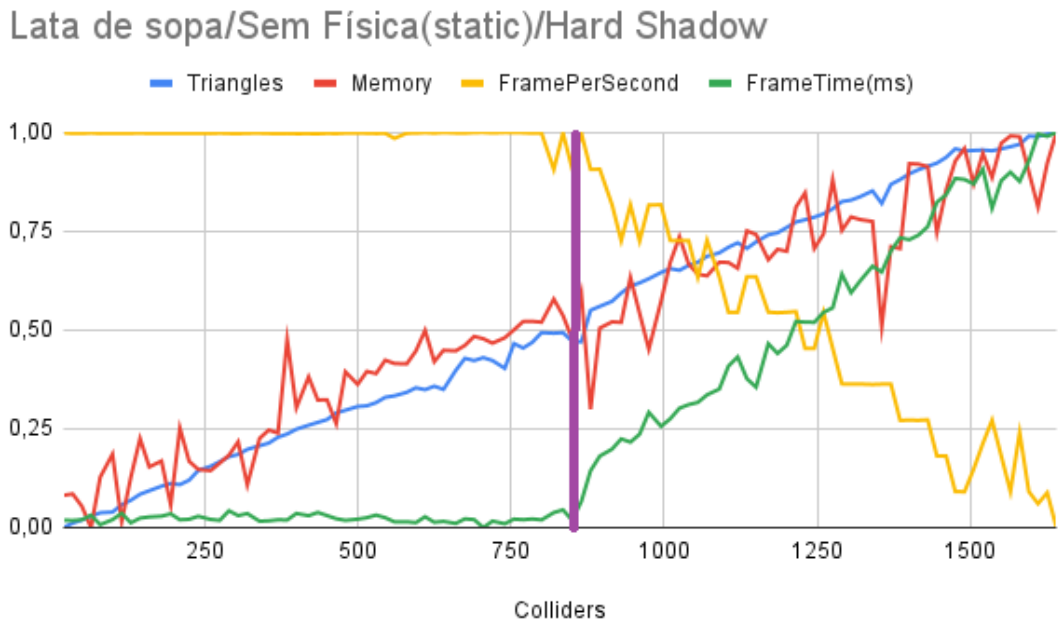


Tabela 4.12 – Fragmento da tabela mostrando a formação de uma pilha estática de sacos de lixo sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
160	4829756	46313060	29,95727	33,1849
180	5351806	45847142	29,95161	33,01741
195	5939110	45383824	29,9607	33,73845
210	6428530	46521227	29,94962	33,40058
225	6885322	46358987	28,95903	35,6542
240	7456312	47001143	28,94901	37,12788
255	7896790	46119667	26,95618	40,0893

Tabela 4.13 – Fragmento da tabela mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "No Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
1520	3960906	50806487	29,94865	33,34608
1535	4055880	50832675	29,96774	33,92118
1555	4090416	50924903	29,95534	35,12038
1570	4116316	50684904	29,95377	33,34236
1585	4182510	50749072	29,95552	35,64391
1600	4291876	50059186	28,95926	34,36008
1615	4257338	50856993	28,96616	34,79121

Tabela 4.14 – Fragmento da tabela mostrando a formação de uma pilha estática de latas de sopa sem usar a física da "Unity"(com "static") e com configuração de sombra "Hard Shadow"

Colliders	Triangles	Memory	FramePerSecond	FrameTime(ms)
820	4260232	49760777	28,93699	33,84132
835	4271740	49418913	29,95329	34,01652
850	4087564	48927817	28,94923	33,34594
865	4076040	49863091	29,94176	34,55691
880	4755248	47465332	28,9404	36,49633
895	4838708	49156701	28,94191	37,41916
915	4948086	49287567	27,95453	37,86044

4.4.4 Câmera dinâmica

Conseguimos informações sobre os últimos pontos estáveis conforme os "colliders"são criados. Agora seria interessante obter métricas em uma outra configuração.

Foi notado nos os testes anteriores a câmera sempres esteve apontada para o centro de onde os objetos colidem com o plano virtual. Na vida real, muitas vezes a câmera se mexe de um ponto a outro ou rotaciona um certo ângulo. A nova configuração proposta é formar uma pilha de lixo estática sem a adição de "colliders"com o tempo,ou seja, um número fixo de "colliders"de modo a exaurir a potência gráfica do dispositivo dependendo da posição da câmera.

De modo que os gráficos 4.15 4.16 sobre sacos de lixo e os gráficos 4.17 4.18 sobre latas de sopa demostram o que acontece quando a câmera se move e rotaciona.

Não foi possível fazer os gráficos da caixa de leite uma vez que não conseguimos colocar objetos o suficiente para exaurir o dispositivo.

O que temos aqui é uma relação simples, quando a câmera ponta para o centro do lixo, ou seja, renderiza mais triângulos o FPS cai. Deste modeo vemos um espelhamento entre o número de triângulos e o FPS. Notamos também que a memória permanece estável em todas as situações

4.5 Relação de número de "colliders"com triângulos

Os dois últimos gráficos deste artigo querem fazer uma relação entre o número de triângulos e o número de "colliders". Os gráficos 4.19 4.20 fazem uma relação entre estes dois atributos. Eles mostram que tanto o número de triângulos e o número de "colliders"podem ser o fator de gargalo. O gargalo não necessariamente é definido por

Figura 4.15 – Gráfico mostrando 600 sacos de lixo sem usar a física da "Unity"com "No Shadow"

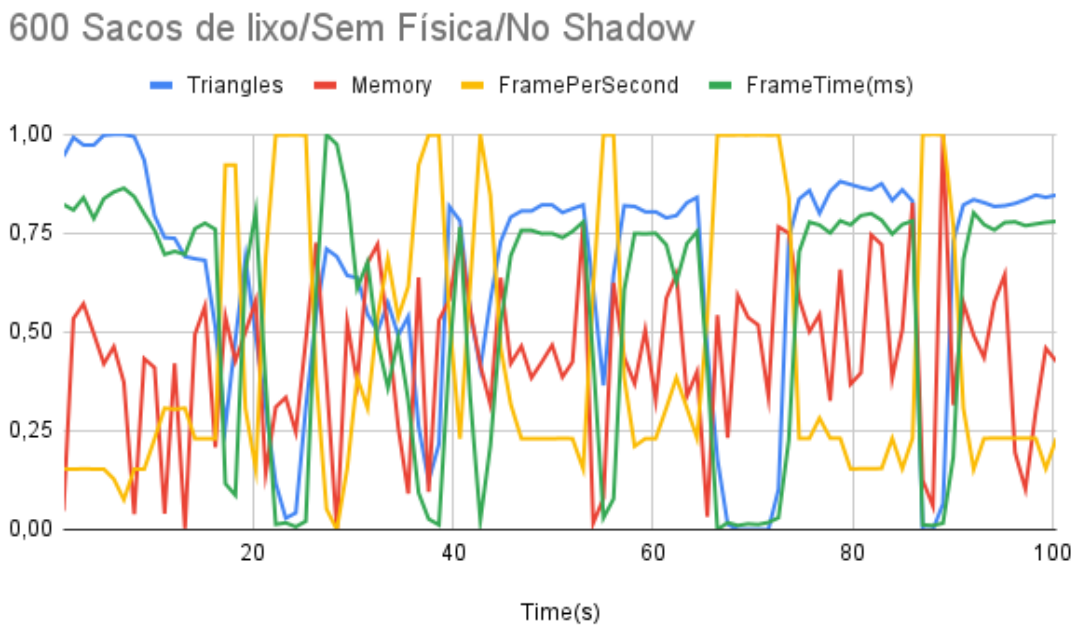


Figura 4.16 – Gráfico mostrando 450 sacos de lixo sem usar a física da "Unity"com "Hard Shadow"

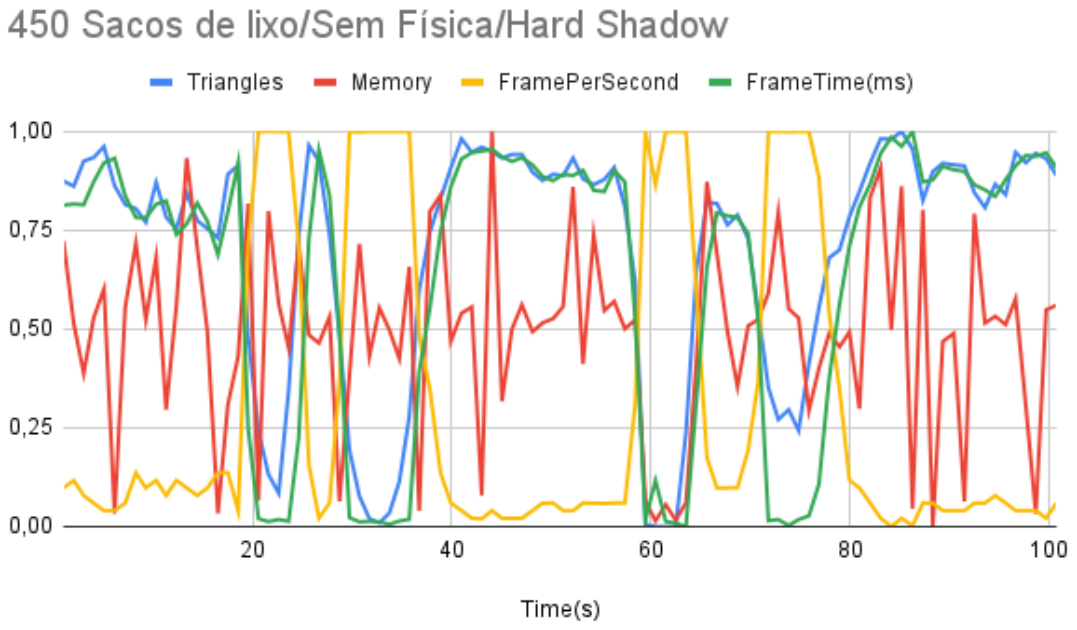


Figura 4.17 – Gráfico mostrando 1650 latas de sopa sem usar a física da "Unity"com "No Shadow"

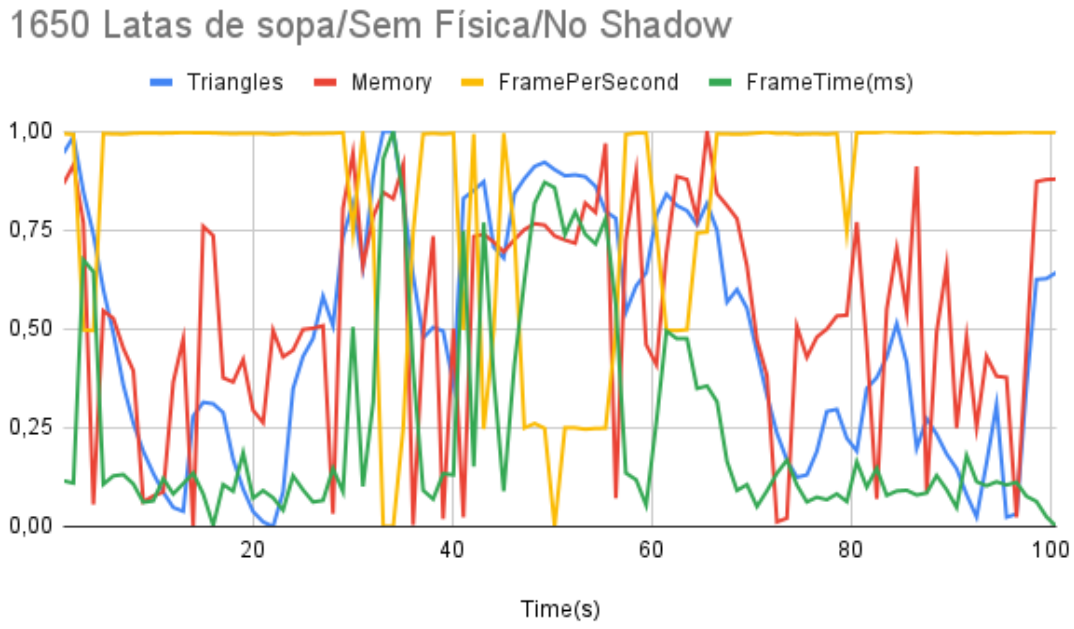


Figura 4.18 – Gráfico mostrando 1500 latas de sopa sem usar a física da "Unity"com "Hard Shadow"

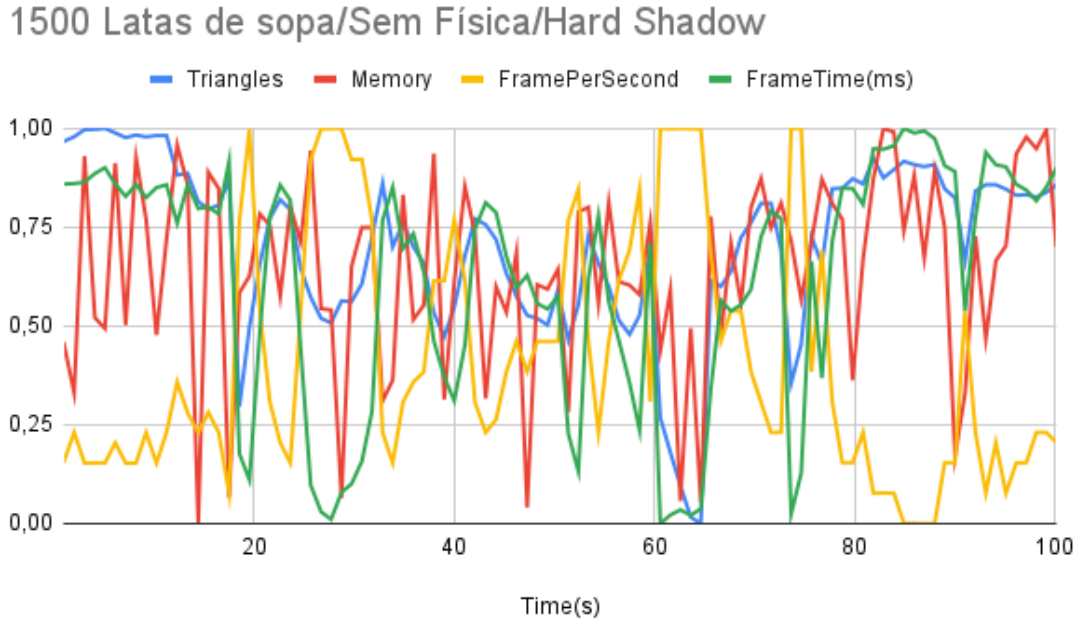
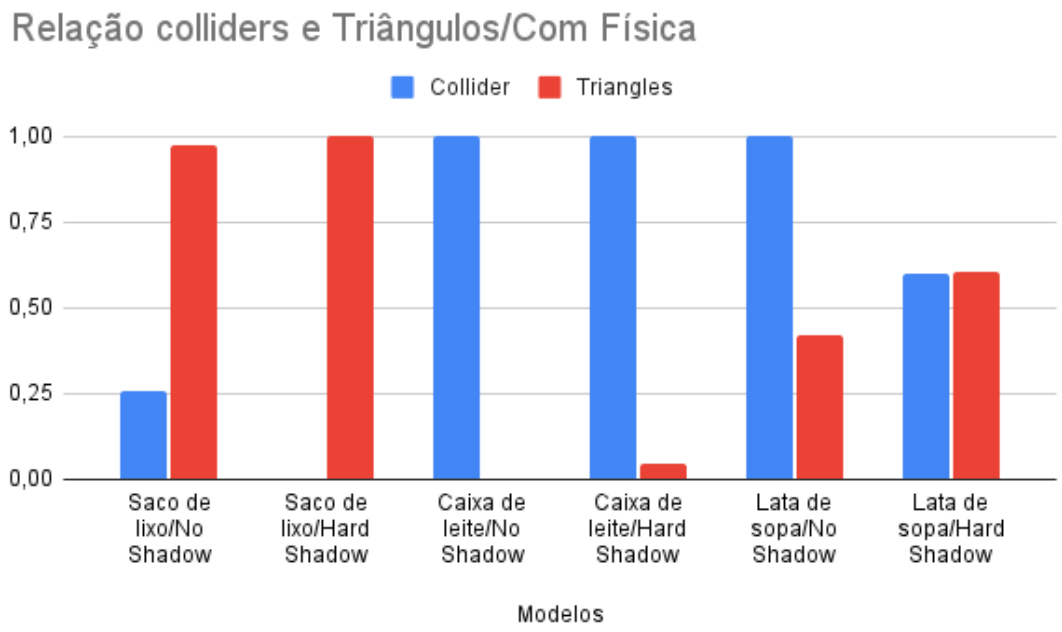
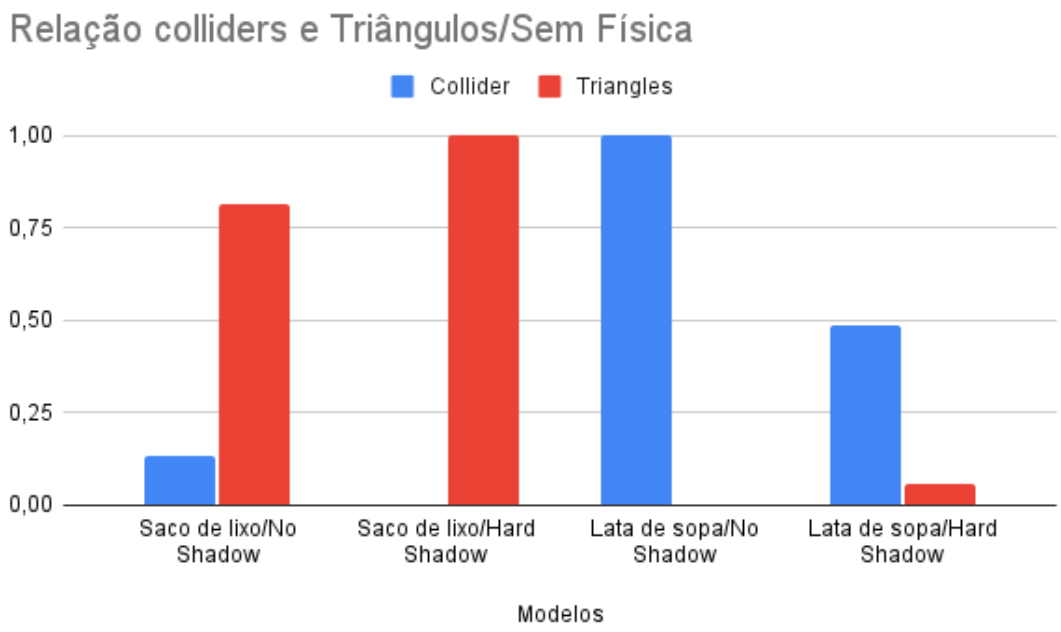


Figura 4.19 – Gráfico mostrando a relação Triângulos e "colliders"com física



apenas um deles, mas as vezes sim por um balanço entre eles.

Figura 4.20 – Gráfico mostrando a relação Triângulos e "colliders"sem física



5 CONCLUSÃO

O aplicativo se mostrou muito interessante no que tange à simulação de AR que, certamente, aumenta a imersão do usuário no aplicativo. A capacidade de instanciar modelos 3D de um mundo virtual para um mundo real, em tempo real, gera uma capacidade de interação com o usuário mais intensa do que com métodos mais convencionais como, por exemplo, a interação bidimencional mais tradicional em computadores e dispositivos "mobile". É notável dizer que a AR amplifica a experiência de aprendizado(HANTONO;*et al*,2018)

O uso de uma "Game Engine", neste caso a "Unity", também se demonstrou uma escolha certa, pois facilitou várias tarefas necessárias ao aplicativo, como simulação de física, interpretação de variados modelos 3D e capacidade de importar para "smartphones android".

A captura das métricas e, posteriormente, a análise sobre elas nos repassam uma grande quantidade de informações e um esclarecimento sobre o quanto um dispositivo consegue processar.

6 REFERÊNCIAS

ALMEIDA JR, R. de A.; AMARAL, Sérgio Pinto. Lixo urbano, um velho problema atual. **XIII Simpósio Internacional de Administração**, p. 1-7, 2006. Disponível em:https://simpep.feb.unesp.br/anais/anais_13/artigos/78.pdf Acesso em: 22/09/2022

DALIM, Che Samihah Che *et al*. Factors influencing the acceptance of augmented reality in education: A review of the literature. **Journal of computer science**, v. 13, n. 11, p. 581-589, 2017. Disponível em:http://researchonline.ljmu.ac.uk/id/eprint/8068/ Acesso em: 5/09/2022

HANTONO, Bimo Sunarfri; NUGROHO, Lukito Edi; SANTOSA, P. Insap. Meta-review of augmented reality in education. In: 2018 **10th international conference on information technology and electrical engineering (ICITEE)**. IEEE, 2018. p. 312-315. Disponível em: https://ieeexplore.ieee.org/abstract/document/8534888 Acesso em: 3/09/2022

KESIM, Mehmet; OZARSLAN, Yasin. Augmented reality in education: current technologies and the potential for education. **Procedia-social and behavioral sciences**, v. 47, p. 297-302, 2012. Disponível em:
https://www.sciencedirect.com/science/article/pii/S1877042812023907
Acesso em: 10/09/2022

LEE, Kangdon. Augmented reality in education and training. **TechTrends**, v. 56, n. 2, p. 13-21, 2012.Disponível em:https://link.springer.com/article/10.1007/s11528-012-0559-3 Acesso em: 13/09/2022

MONTERO, Alvaro *et al*. Designing and implementing interactive and realistic augmented reality experiences. **Universal Access in the Information Society**, v. 18, n. 1, p. 49-61, 2019.Disponível em:https://link.springer.com/article/10.1007/s10209-017-0584-2 Acesso em: 5/09/2022

NOH, Zakiah; SUNAR, Mohd Shahrizal. A review of shadow techniques in augmented reality. In: 2009 **Second International Conference on Machine Vision**. IEEE, 2009. p. 320-324.Disponível em:https://ieeexplore.ieee.org/abstract/document/5381137 Acesso em: 5/09/2022

SUGANO, Natsuki; KATO, Hirokazu; TACHIBANA, Keihachiro. The effects of shadow representation of virtual objects in augmented reality. In: **The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings**. IEEE, 2003. p. 76-83.

Disponível em:https://ieeexplore.ieee.org/abstract/document/1240690
Acesso em: 5/09/2022

VAN KREVELEN, D. W. F.; POELMAN, Ronald. A survey of augmented reality technologies, applications and limitations. **International journal of virtual reality**, v. 9, n. 2, p. 1-20, 2010.Disponível em:https://ijvr.eu/article/view/2767/8825 Acesso em: 26/09/2022