UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Geração de Sombras em Objetos Modelados por Geometria Sólida Construtiva

por

MARCOS LUÍS CASSAL

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Anatólio Laschuk
Orientador

Porto Alegre, março de 2001

CIP - CATALOGAÇÃO DA PUBLICAÇÃO

Cassal, Marcos Luís

Geração de Sombras em Objetos Modelados por Geometria Sólida Construtiva / Marcos Luís Cassal. – Porto Alegre: PPGC da UFRGS, 2001.

149 p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2001. Orientador: Laschuk, Anatólio.

1. Geometria Sólida Construtiva (CSG). 2. Sombras. 3. Fontes de Luz. 4. Triangularização de Polígonos. 5. Classificação de Vértices e Arestas. 6. OpenGl. 7. Teoria dos Conjuntos. I. Laschuk, Anatólio. II. Título.

Agradecimentos

Ao professor Anatólio Laschuk, pela sua confiança, sugestões, conselhos e críticas que foram fundamentais para a realização deste trabalho.

À CAPES, pelo auxílio financeiro que foi imprescindível para a realização desta dissertação.

Aos amigos Cadinho, Maristela, Cathi, Rogério, Cláudia, Marilton e Júlio pela amizade, apoio, jantares, momentos de lazer e descontração, que também são importantes em nossa vida.

Aos meus pais Luiz e Marlita, aos irmãos Luciano, Ana e Denise, pelo amor, carinho, apoio e incentivo que recebi em todos os momentos de minha vida.

Aos colegas Cadinho, Rogério e Daniela pelas colaborações e trocas de idéias em discussões sobre este trabalho.

Aos colegas, professores e funcionários do Instituto de Informática da UFRGS que direta ou indiretamente ajudaram na realização deste trabalho.

Em especial quero agradecer a minha noiva, Ana Paula, pelo amor, afeto, companheirismo, compreensão, paciência, apoio, incentivo,... que recebi e continuo recebendo em todos os momentos.

A Deus pela alegria de viver e pelos dons recebidos.

Sumário

Lista de Abreviaturas

1D	Unidimensional
2D	Bidimensional
3D	Tridimensional
ARB	*Architecture Review Board*
PPGC	Programa de Pós Graduação em Computação
CSG	Geometria sólida construtiva
OpenGl	*Open Graphics Library*
UFRGS	Universidade Federal do Rio Grande do Sul

Lista de Figuras

Lista de Tabelas

Resumo

Atualmente os sistemas computacionais mais sofisticados são aqueles que apresentam imagens gráficas. Devido às características de alta velocidade de processamento e excelente resultado na geração de imagens o uso da Computação Gráfica se dá em diversas áreas como a indústria, pesquisa, publicidade, entretenimento, medicina, treinamento, dentre outras.

Este trabalho aborda dois assuntos clássicos na Computação Gráfica, Geometria Sólida Construtiva (CSG) e Sombras Projetadas. Ambos são muito importantes para esta linha de pesquisa da Ciência da Computação. A Geometria Sólida Construtiva é utilizada na modelagem de objetos e as sombras projetadas são necessárias para aumentar o realismo das imagens.

Geometria sólida construtiva (CSG) é uma técnica para a modelagem de sólidos, que define sólidos complexos pela composição de sólidos simples (primitivas). Isso inclui também a composição de objetos já combinados, até que se chegue a um objeto mais complexo.

Um fator muito importante e necessário na obtenção de imagens realistas e que deve ser considerado é a utilização de sombras, pois estas são eficazes no realismo e impressão espacial de objetos tridimensionais. As sombras estabelecem diversos níveis de profundidade na imagem, fazem uma pontuação geométrica na cena de modo a evitar que os objetos não pareçam estar flutuando no ar.

Este trabalho consiste em apresentar uma proposta para a geração de sombras em objetos modelados pela Geometria Sólida Construtiva. Para tanto foram estudados os assuntos referentes à modelagem de objetos por CSG, algoritmos para a geração de sombras “bem delimitadas” e formas de gerar sombras na Geometria Sólida Construtiva.

O processo de geração de sombras em cenas modeladas por CSG, através da aplicação das mesmas operações booleanas envolvidas na modelagem dos objetos, sobre as sombras nem sempre apresenta resultados corretos.

Diante disso, foram investigadas outras formas de solucionar o problema. Dentre estas, uma alternativa é a realização de transformações na árvore binária CSG, através de outras operações, envolvendo o uso de complemento com operações de união e interseção, para a modelagem do objeto e geração da sombra correspondente.

Com base nos estudos realizados foram implementados dois protótipos que exibem a sombra projetada de objetos modelados por CSG. Na implementação do protótipo A utilizaram-se as técnicas tradicionais de modelagem de sólidos e sombra projetada. Os resultados obtidos com este protótipo serviram de referência. No protótipo B os resultados foram obtidos através da aplicação da zona ativa das primitivas na modelagem dos objetos e a sombra é projetada durante o processo de avaliação de contornos do sólido. Os resultados obtidos com este protótipo são comparados com os resultados do protótipo A e são apresentados como forma de exibir a aplicação do método proposto.

Palavras-chave: Geometria Sólida Construtiva (CSG), Sombras, Fontes de Luz, Triangularização de Polígonos, Classificação de Vértices e Arestas, OpenGl, Teoria dos Conjuntos.

TITLE: “SHADOWS GENERATION OF OBJECTS MODELED BY THE CONSTRUCTIVE SOLID GEOMETRY”

Abstract

Nowadays, the most sophisticated computational systems are those presenting graphical images. Due to the characteristics of high processing speed and excellent results in the generation of images, Computer Graphics is used in several areas as industry, research, marketing, entertainment, medicine, training, and others.

This work approaches two classical subjects in Computer Graphics, Constructive Solid Geometry (CSG) and Projected Shadows. Both are very important for this research area in Computer Science. Constructive Solid Geometry is used in object modeling and projected shadows are necessary to increase the realism of the images.

Constructive Solid Geometry (CSG) is a technique used for solid modeling, where complex solids are defined by the composition of simple solids (primitives). That also includes the composition of already combined objects, to obtain more complex objects.

A very important and necessary factor in obtaining realistic images and that should be considered is the use of shadows, because these are effective in the realism and space perception of three-dimensional objects. The shadows establish several depth levels in the image and make a geometric punctuation in the scene, avoiding objects not to seem to be floating in the air.

This work presents a proposal for the generation of shadows in objects modeled by Constructive Solid Geometry. For so, subjects referring to CSG modeling of objects were studied, as well as algorithms for the generation of “well defined” shadows and forms of generating shadows in the Constructive Solid Geometry.

The process of shadow generation in CSG scenes using the same boolean operations involved in the objects modeling does not always present correct results.

Other forms of solving the problem were investigated. Among these, an alternative is to transform the binary CSG tree, through other operations, involving the complement with union and intersection, for object modeling and shadow generation.

Based in such studies two prototypes were implemented that exhibit the projected shadows of objects modeled by CSG. In the prototype A implementation traditional techniques of modeling of solids and projected shadows were used. The results obtained with this prototype served as reference. In the prototype B the results were obtained through the application of the active zone of the primitive in the modeling of the objects and the shadows are projected during the boundary evaluation process of the solid. The results obtained with this prototype are compared with the results of the prototype A and they are presented as application of the proposed method.

Keywords: Constructive Solid Geometry (CSG), Shadows, Light Sources, Polygon Triangularization, Vertexes and Edges Classification, OpenGl, Set Theory.

1 Introdução

Geometria sólida construtiva (CSG) é uma técnica para a modelagem de sólidos, que define objetos pela composição de sólidos simples (primitivas) e/ou objetos já combinados, até que se chegue ao objeto final.

“Na geometria sólida construtiva, primitivas simples são combinadas por meio de um conjunto de operações booleanas regularizadas, correspondentes às operações da teoria dos conjuntos (união, interseção e diferença), que são incluídas diretamente na representação” [FOL 97].

Dentre os muitos desafios da Computação Gráfica um dos mais relevantes é a geração de imagens com um alto grau de realismo. Uma das maiores dificuldades na geração de imagens realísticas encontra-se na complexidade do mundo real, que possui uma infinidade de cores, texturas, sombras, reflexões e outras inúmeras características próprias. Criar imagens que simulem tais características é um grande desafio, que passa por várias etapas, desde a modelagem dos objetos e definição de pontos de visualização até a remoção de elementos ocultos e a utilização de efeitos de sombreamento, reflexão, refração, etc [FOL 97].

Um fator importante e necessário na obtenção de imagens realistas e que deve ser considerado é a utilização de sombras, pois estas são eficazes no realismo e na impressão espacial de objetos tridimensionais. As sombras estabelecem diversos níveis de profundidade na imagem, fazem uma pontuação geométrica na cena de modo a evitar que os objetos não pareçam estar flutuando no ar [GOM 90].

Este trabalho consiste em apresentar uma proposta para a geração de sombras em objetos modelados pela Geometria Sólida Construtiva. Para tanto foram estudados: conceitos referentes a modelagem de objetos por CSG, algoritmos para geração de sombras, uma proposta para gerar sombras com as técnicas da Geometria Sólida Construtiva, o desenvolvimento de um protótipo e uma análise dos resultados obtidos.

Dentro do âmbito do Curso de Pós Graduação em Ciência da Computação da UFRGS várias dissertações já foram concluídas na área de Computação Gráfica. Dentre as que estão relacionadas com esse trabalho, cita-se a dissertação: Estudo de Algoritmos Para Geração de Sombras Projetadas na Síntese de Imagens Foto-Realísticas [NAS 92]. O aspecto que diferencia o trabalho aqui proposto do já realizado é a geração de sombras em objetos modelados pela Geometria Sólida Construtiva.

1.1 Motivação

Na literatura encontram-se vários algoritmos geradores de sombra que foram e continuam sendo desenvolvidos [PRI 2000, VER 2001]. Em geral, tais algoritmos são implementados através de *raytracing*, que apresenta ótimos resultados, porém são caros computacionalmente e muitas vezes deixam de ser utilizados em sistemas gráficos.

Além disso, vários dos algoritmos existentes para a geração de sombras não são diretamente aplicáveis para objetos definidos pela Geometria Sólida Construtiva. Objetos CSG são definidos com combinações booleanas de

primitivas. Aplicando as respectivas operações booleanas nas sombras das primitivas individualmente, nem sempre será gerada uma sombra correta para o objeto resultante CSG [JAN 91].

Nesse contexto, busca-se identificar uma forma alternativa para a geração de sombras em objetos modelados por CSG. Foi observada a possibilidade de serem utilizadas operações booleanas para gerar a sombra, e a partir disso, difundido o processo de geração de sombras.

Estas três perguntas representam o eixo principal da realização desta pesquisa.

- A sombra da união dos objetos é igual a união das sombras dos objetos ?
- A sombra da interseção dos objetos é igual a interseção das sombras dos objetos ?
- A sombra da diferença dos objetos é igual a diferença das sombras dos objetos?

1.2 Objetivos

O problema apresentado neste trabalho aborda dois assuntos clássicos da Computação Gráfica: Geometria Sólida Construtiva e Sombras. A Geometria Sólida Construtiva tem se mostrado como uma forma cientificamente rica na modelagem de objetos, sendo empregada em muitos *softwares* comerciais, principalmente em aplicações da engenharia mecânica. O uso das sombras, por sua vez, é fundamental para a apresentação de imagens realísticas.

Desta forma, os principais objetivos deste trabalho são:

1 – Pesquisar e estudar a geração de sombras em objetos modelados pela Geometria Sólida Construtiva.

Nesta etapa do trabalho a meta pretendida foi a de adquirir as bases técnica e teórica, necessárias para um maior aprofundamento no estudo do problema proposto. Estes conhecimentos específicos englobam:

- Modelagem de objetos pelo método da Geometria Sólida Construtiva
- Algoritmos geradores de sombras

2 – Implementação e uma análise de um protótipo para a geração de sombras em objetos modelados pela Geometria Sólida Construtiva.

Baseando-se no levantamento bibliográfico e estudos realizados durante a primeira etapa, desenvolveu-se o protótipo de um avaliador de contornos (boundary evaluator) e também de um algoritmo que gera sombras projetadas “bem delimitadas”, isto é, somente com região de umbra sem a de penumbra, apresentando resultados para o problema proposto.

1.3 Organização do Texto

Esta dissertação é composta de oito capítulos.

No capítulo dois, são descritos os principais conceitos envolvidos na técnica de modelagem CSG: primitivas geométricas, transformações geométricas e as operações boolenas regularizadas. As definições de avaliador de contornos (*Boundary* Evaluator) e a classificação da pertinência a um conjunto (*Set-Membership Classification*) também são assuntos deste capítulo.

O capítulo três apresenta as principais características das fontes de luz e os tipos de controles que podem ser aplicados sobre as mesmas. Trata também das sombras e sua importância na geração de imagens realísticas por computador e descreve brevemente os principais algoritmos para a geração de sombras projetadas. Nesse mesmo capítulo são apresentadas alternativas para a geração de sombras em cenas modeladas por Geometria Sólida Construtiva, bem como a forma proposta para a geração de sombras deste trabalho.

No capítulo quatro discute-se a geração de sombras com os métodos da Geometria Sólida Construtiva. Analisa-se também a possibilidade de gerar sombras com operações booleanas que envolvam o uso do complemento dos objetos juntamente com a operação de união. É demonstrado que com combinações entre estes operadores é possível se chegar a resultados equivalentes aos obtidos nas operações de interseção e diferença.

O capítulo cinco descreve o avaliador de contornos, explicando quais foram as técnicas utilizadas nesta etapa do modelador CSG.

O capítulo seis descreve as técnicas utilizadas na implementação do protótipo.

Já no capítulo sete apresentam-se os resultados obtidos com a realização deste estudo e alguns dos problemas enfrentados.

Finalmente, no capítulo oito, são feitas considerações finais em forma de conclusão e identificados possíveis trabalhos futuros.

2 Geometria sólida construtiva

Sistemas de modelagem e projetos auxiliados por computador são desenvolvidos para ajudar projetistas a criarem modelos. O principal objetivo destes sistemas tem sido prover uma integração e interação de ambientes tridimensionais em que o usuário possa visualizar e analisar o modelo como um todo [KRI 97].

No dia-a-dia pode-se verificar, do ponto de vista geométrico, que existe uma grande quantidade de objetos, que podem ser modelados através da combinação de outros objetos mais simples. Por exemplo, uma mesa pode ser modelada como um retângulo longo e fino com quatro cilindros como pernas, ou então modela-se um sorvete através de um cone com uma esfera e assim sucessivamente.

Baseado na forma em que o ser humano monta seus objetos sólidos, elaborou-se a técnica CSG (*Constructive Solid Geometry*) para a criação e representação de sólidos computacionalmente. Tal técnica representa uma maneira natural de se construir objetos, uma vez que ela permite que os resultados obtidos sejam derivados de outras operações naturais entre conjuntos (união, interseção e diferença) [SIL 97] .

Geometria Sólida Construtiva (CSG) é uma forma de modelagem de sólidos que define sólidos complexos pela composição de sólidos simples (primitivas) e/ou objetos já modelados, até que se chegue a um objeto mais complexo. Na Geometria Sólida Construtiva, primitivas simples são combinadas por meio de um conjunto de operações booleanas regularizadas, correspondentes às operações da teoria dos conjuntos (união, interseção e diferença), que são incluídas diretamente na representação [FOL 97].

Um objeto modelado por CSG é representado através de uma árvore binária, onde as primitivas sofrem alterações geométricas e são combinadas através de operações booleanas até que se chegue ao objeto desejado. Na figura 2.1 (a) apresenta-se uma primitiva do tipo quadrado, esta primitiva foi utilizada na modelagem do objeto R. Após serem aplicadas as transformações geométricas sobre a primitiva, obteve-se os objetos A, B e C da figura 2.1 (b). Na árvore binária a raiz representa resultado pretendido, ou seja, o objeto R.

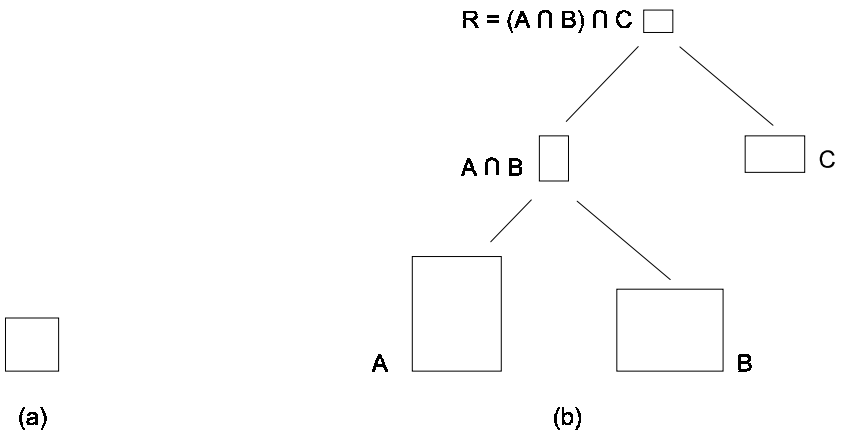


FIGURA 2.1 – Primitiva e árvore binária

A idéia da Geometria Sólida Construtiva está baseada em olhar para um objeto como se este estivesse dividido em partes e com a combinação destas partes construir o objeto como um todo. Esta técnica de modelagem trabalha com três pontos básicos: primitivas geométricas, transformações geométricas e operações booleanas.

2.1 Primitivas Geométricas e Objetos

A Geometria Sólida Construtiva é um tipo de representação que tem por objetivo facilitar o modo de interação para a modelagem de sólidos no computador. A idéia principal é de que as partes envolvidas na modelagem, que irão sofrer modificações através das operações booleanas e transformações geométricas, sejam feitas sobre as primitivas geométricas.

Muitas peças fabricadas podem ser agrupadas dentro de classes ou famílias de formas similares, onde membros individuais da família são distinguidos por alguns parâmetros (dimensões chaves). Uma família de formas simples é uma primitiva genérica e os membros são primitivas que sofreram algum tipo de transformação geométrica [MOR 85].

Objetos simples, elementares e fáceis de serem descritos e representados no computador, que constituem os blocos básicos para a construção dos modelos, são chamados de primitivas geométricas. Esferas, cones, cilindros ou sólidos retangulares são combinados com o uso dos operadores booleanos (união, interseção e diferença) e das transformações geométricas (rotação, translação e mudança de escala) até se chegar ao objeto desejado.

Sistemas de modelagem de sólidos, que utilizam a técnica CSG, geralmente oferecem uma biblioteca de primitivas geométricas, sobre as quais são feitas as transformações necessárias para se chegar a modelagem do objeto final. Cabe lembrar que, apesar de muito importante para o sistema de modelagem, o número de primitivas não é uma indicação do poder descritivo de um sistema de modelagem.

Outra forma de modelagem consiste em sólidos primitivos serem uma combinação, por meio de operações booleanas regularizadas, de superfícies orientadas ou semi-espaços, em grupos pré-arranjados. Uma superfície orientada é uma superfície cuja normal em cada ponto determina o que está de um lado e de outro da superfície. Uma superfície divide o espaço em duas regiões, os semi-espaços. Essa combinação de semi-espaços de forma pré-arranjada deve formar conjuntos regulares limitados. Por exemplo, um sistema com esquema de representação CSG tendo apenas semi-espaços planar e cilíndrico, tipicamente podem oferecer duas primitivas sólidas: um cubo definido pela interseção regularizada de seis semi-espaços planares, e um cilindro definido como a interseção de um semi-espaço cilíndrico e dois semi-espaços planares [REQ 82]. Ver figura 2.2.

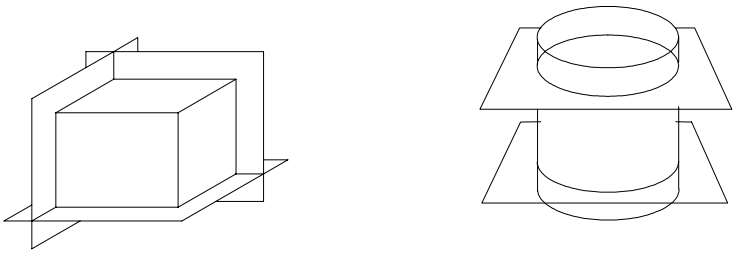


FIGURA 2.2 – Primitivas constituídas por semi-espaços planar e cilíndrico

Em um espaço E n-dimensional, um objeto é uma região R limitada deste espaço ($R \subset E$). Esta região é representada por três partes: a sua fronteira (*boundary*), o seu interior e o seu exterior. Os pontos que compõe uma região deste espaço podem ser classificados como pontos da fronteira, pontos interiores ou pontos exteriores.

Pode-se definir os pontos exteriores (eA) como sendo todos os pontos que pertencem ao espaço E e não pertencem ao objeto A. Os pontos da fronteira (bA) são os pontos que pertencem ao objeto e possuem pelo menos um vizinho exterior. Já os pontos interiores (iA) são todos os pontos que não pertencem à fronteira e também não pertencem ao exterior. Desta forma define-se que os objetos geométricos serão objetos fechados, ou seja, possuirão obrigatoriamente pontos de fronteira e pontos interiores.

Na figura 2.3 pode-se ilustrar mais claramente estes conceitos. No espaço E, o objeto A está dividido em dois subconjuntos, bA e iA, onde bA (*boundary*) representa a fronteira de A e iA representa o conjunto de todos os pontos interiores, desta forma o objeto A é a união dos pontos interiores com os pontos da fronteira ($A = bA \cup iA$). Vale $bA \cap iA = \varnothing$.

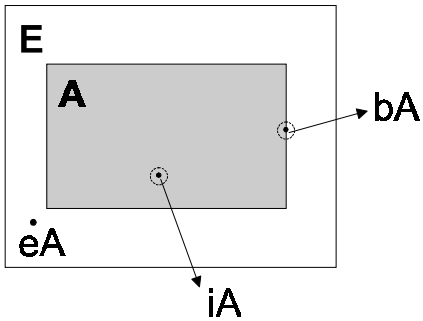


FIGURA 2.3 – Vizinhança

A abordagem mais comum dos sistemas de modelagem é oferecer um conjunto finito de primitivas, onde tamanho, forma, posição e orientação são determinados por um conjunto de parâmetros especificados pelo usuário. Por exemplo, uma primitiva do tipo bloco necessita dos parâmetros comprimento, altura e largura. Uma primitiva do tipo cilindro recebe os parâmetros raio e altura. Ver figura 2.4.

FIGURA 2.4 – Primitivas formadas por parâmetros, especificados pelo usuário

2.2 Transformações Geométricas

As transformações geométricas têm dupla finalidade na CSG: posicionar as primitivas no espaço, e/ou modificar a geometria das primitivas. No posicionamento das primitivas são utilizados os movimentos de rotação e translação. Com o uso das transformações geométricas pode-se dar outras formas para as primitivas, por exemplo, a partir do quadrado, obtêm-se qualquer quadrilátero do plano, ver figura 2.5. O uso de transformações para modificar primitivas permite reduzir o número destas [GOM 98].

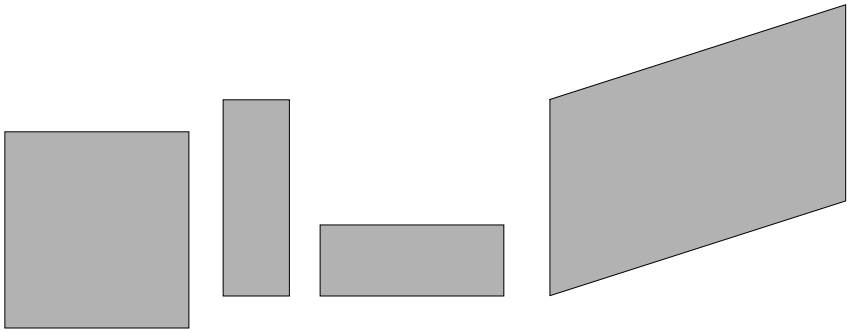


FIGURA 2.5 – Transformações geométricas aplicadas sobre o quadrado

2.3 Operações Booleanas

Na modelagem geométrica, quando se combinam formas simples, chamadas primitivas, para gerar formas mais complexas, a teoria dos conjuntos é muito usada.

Conjunto é um conceito fundamental em todos os ramos da matemática. Um conjunto é uma lista, coleção ou classe de elementos bem definidos. Na modelagem geométrica sólida, o elemento básico é o ponto [LIP 98,MOR 85].

Em aritmética, é possível a realização de operações como: soma, subtração e multiplicação de números. Com conjuntos também se pode realizar operações, estas envolvem operações de união, interseção e diferença.

A união dos conjuntos A e B é um conjunto de todos os elementos que pertencem a A ou a B ou a ambos. A união entre dois conjuntos é representada por $A \cup B$ [LIP 98].

A interseção de A e B é o conjunto dos elementos que são comuns a A e B, isto é, os elementos que pertencem a A e também pertencem a B. Representa-se a interseção por A ∩ B [LIP 98].

A diferença entre dois conjuntos A e B é o conjunto de elementos que pertencem a A mas que não pertencem a B. Indica-se a diferença de A e B por A – B [LIP 98].

A figura 2.6 apresenta os diagramas de Venn, para as operações de união, interseção e diferença entre os conjuntos A e B.

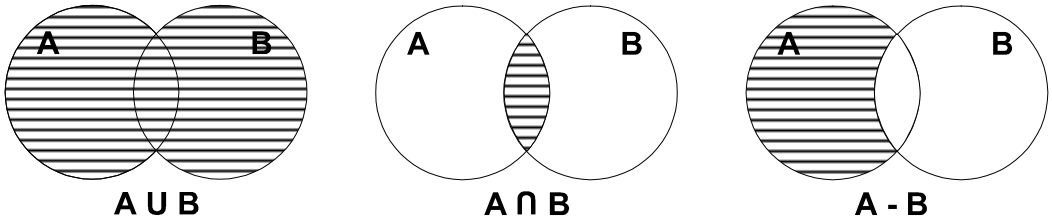


FIGURA 2.6 – Diagramas das operações de união, interseção e diferença

Na modelagem de objetos CSG as operações sobre os objetos baseiam-se na teoria dos conjuntos e segundo [KRI 96] nas seguintes regras, aplicadas a dois objetos A e B:

- União de A com B: Um componente de A é parte do novo sólido se estiver no exterior de B. Um componente de B é parte do novo sólido se estiver no exterior de A.
- Interseção entre A e B: Um componente de A é parte do novo sólido se estiver no interior de B. Um componente de B é parte do novo sólido se estiver no interior de A.
- Diferença A – B: Um componente de A é parte do novo sólido se estiver no exterior de B.

Após as primitivas devidamente posicionadas no espaço, o sistema CSG utiliza as operações booleanas para combinar as diversas primitivas e criar o modelo final. As operações booleanas são a união, interseção e a diferença, as mesmas operações da teoria dos conjuntos [GOM 98].

Objetos geométricos são definidos como conjuntos fechados de pontos, compostos de dois subconjuntos: interior e fronteira. Operações ditas booleanas semelhantes a interseção, união e diferença sobre conjuntos são usadas para combinar objetos simples formando outros mais complexos. Os algoritmos que executam essas operações devem produzir objetos que também sejam conjuntos fechados de pontos, tendo como subconjuntos o interior e a fronteira, preservando a dimensão dos objetos iniciais. Um último requerimento é o de que em qualquer operação booleana, todos os objetos devem estar na mesma dimensão espacial [MOR 85].

Se o sistema estiver trabalhando com um objeto poliédrico, este obrigatoriamente deve ter uma fronteira e um interior. Neste caso, o objeto sólido tem a sua fronteira representada por polígonos, ou seja, pela união de todas as faces, e cada face é limitada por um conjunto de vértices e arestas [REQ 85].

Os operadores convencionais sobre conjuntos, União ($\cup$), Interseção ($\cap$) e diferença (-), são usados para modelar por meio de operações que combinem sólidos. A maior parte deles, entretanto, é inadequada porque destrói a regularidade, violando o princípio de que o domínio dos sólidos modelados a partir de conjuntos regulares seja fechado sob os operadores combinacionais escolhidos [RIP 87].

No sistema CSG usam-se primitivas sólidas com o objetivo de construir sólidos. A idéia é que operações booleanas com sólidos resultem em sólidos.

A ilustração da figura 2.7 mostra dois objetos bidimensionais A e B. Cada um está bem definido e ambos são fechados e homogeneamente dimensionados.

Neste caso os dois objetos foram posicionados de uma maneira que ficassem sobrepostos, para então combiná-los, através de uma operação booleana, a fim de formar o objeto resultante C.

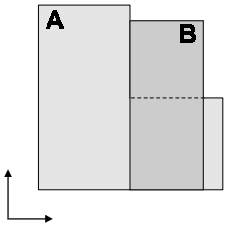


FIGURA 2.7 – Objetos bidimensionais prestes a sofrerem operações booleanas

Aplicam-se as operações desejadas, que podem ser união, interseção ou diferença. Na figura 2.8 pode-se verificar o resultado das operações booleanas aplicadas sobre as primitivas A e B.

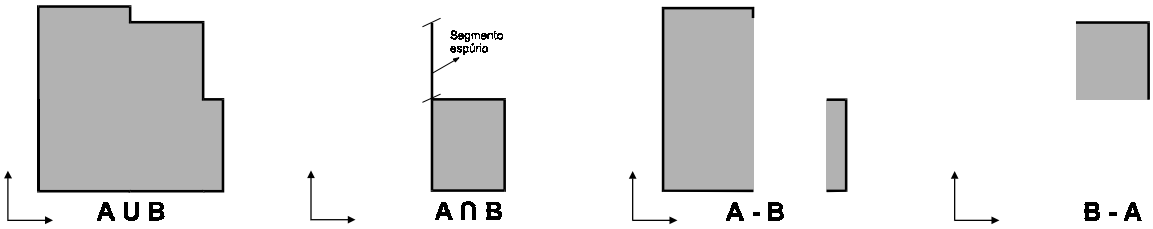


FIGURA 2.8 – Operações booleanas aplicadas sobre as primitivas

Com base na teoria dos conjuntos pode-se afirmar que os resultados apresentados nas figuras acima estão corretos, porém ao ser feita uma análise destas mesmas figuras com fundamentação na Geometria Sólida Construtiva observa-se que na operação de interseção (b) existe um segmento espúrio, que deve ser desprezado. Nas operações de diferença (A–B e B–A), as fronteiras apresentadas não são as esperadas. O resultado correto para a operação de interseção CSG pode ser visto na figura 2.9(b), onde tem-se a chamada interseção regularizada.

(a) – Não regularizada (b) – Regularizada

FIGURA 2.9 – Operação booleana de interseção

Para corrigir tais situações, são necessárias algumas mudanças nas operações booleanas, as chamadas regularizações. As operações booleanas utilizadas na Geometria Sólida Construtiva são então identificadas como operações booleanas regularizadas.

As demonstrações das propriedades algébricas das operações booleanas regularizadas podem ser encontradas em [MOR 85].

2.4 Árvores Binárias

A construção de um modelo CSG possui uma hierarquia naturalmente associada. Essa hierarquia pode ser representada por uma árvore binária que possui dois tipos de nós: nós de operações e nós de objetos geométricos.

Cada nó de operação está associado a uma das operações booleanas (união, interseção ou diferença). Esse nó possui dois nós descendentes, cada um desses nós corresponde a objetos que devem ser combinados pela operação do nó correspondente. A cada nó de um objeto pode estar associada uma transformação do espaço [GOM 98].

Os sólidos geométricos representados pela CSG são ordenados em árvores binárias, onde os nodos folha ou terminais são as primitivas. Os nodos não terminais, representam as operações boolenas regularizadas (união, interseção e/ou diferença) ou então as transformações geométricas (mudança de escala, rotação e/ou translação), que operam sobre os seus dois subnodos. Cada sub árvore de um nodo representa um sólido resultante das operações de transformação e combinação indicadas. A raiz da árvore representa o objeto final.

Na figura 2.10 pode ser visto um exemplo de árvore binária, que representa um objeto R modelado por CSG. Os nodos terminais estão representando os sólidos primitivos com suas respectivas transformações geometricas (mudança de escala, rotação e/ou translação). Os nodos não terminais caracterizam as operações booleanas regularizadas, união ($\cup^*$) e diferença ($-^*$), que são aplicadas sobre os seus subnodos. A raiz da árvore representa o objeto final, o resultado pretendido.

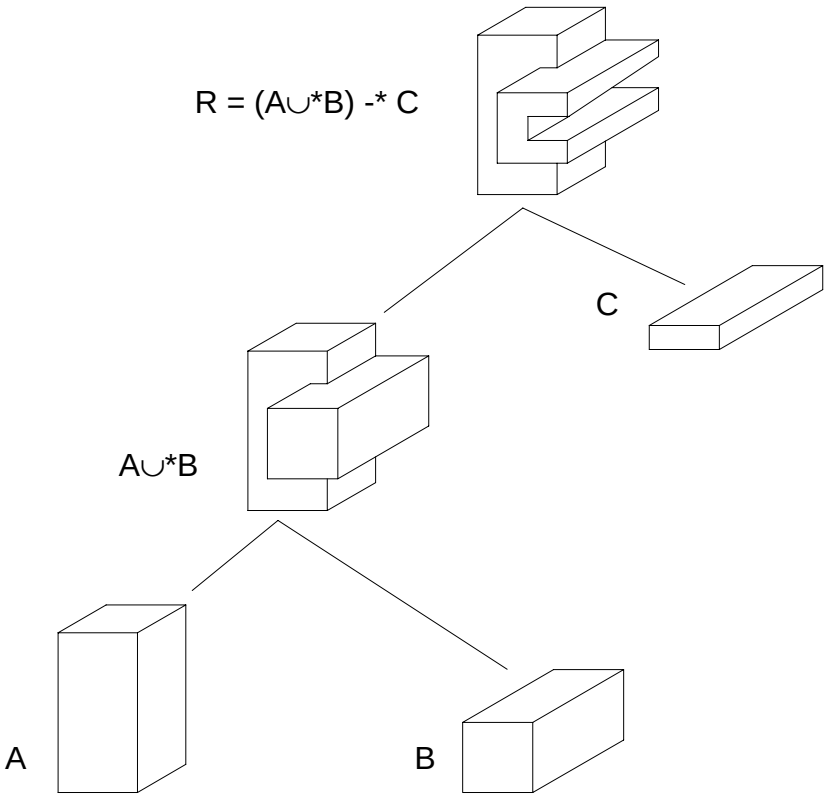


FIGURA 2.10 – Árvore binária CSG para o objeto modelado R

2.5 Avaliador de Fronteiras (*Boundary Evaluator*)

A partir da árvore binária que representa o objeto modelado, o passo seguinte é aplicar as operações booleanas às primitivas, com o objetivo de se chegar a fronteira do novo objeto que está sendo modelado.

A descrição do objeto final modelado pela Geometria Sólida Construtiva é montada por um conjunto de algoritmos, constituindo o chamado avaliador de contornos (*boundary evaluator*). O avaliador de contornos é normalmente a parte mais complexa de um modelador CSG, exigindo tempo e memória do computador.

Segundo [RIP 87], um modelo descrito como R = (A ∪* B) –* C, não informa nada de quantitativo sobre o sólido R (como as coordenadas dos vértices ou alguma informação sobre as arestas e faces), ela apenas especifica a combinação dos sólidos primitivos constituintes. Pode-se ter todas as informações geométricas e topológicas sobre A, B e C, mas tudo que se sabe sobre R é como construí-lo. Assim, esta representação se denomina procedural ou não avaliada (*unevaluated*). Caso se deseje saber mais sobre R, deve-se avaliar o modelo booleano: computar interseções, determinando novos vértices e arestas.

É tarefa do avaliador de contornos analisar a conectividade desses novos elementos, para determinar as características topológicas do modelo, e representar os contornos do objeto. Para a realização deste processo, o

avaliador de contornos faz uso de um algoritmo que realiza a classificação da pertinência a um conjunto (*Set-MemberShip Classfication*), onde os elementos de um objeto são classificados em relação ao outro objeto e vice-versa. A classificação da pertinência a um conjunto é descrita com detalhes na próxima seção.

A representação por superfícies contém, explicitamente, as faces, arestas e vértices que limitam o sólido. Essa descrição possui duas partes: a representação topológica, correspondente à conectividade dos elementos, e a representação geométrica, consistindo dos dados numéricos que descrevem a posição dos elementos.

O avaliador de contornos determina onde as faces são truncadas e onde novos vértices e arestas são criados ou removidos. Sempre que duas faces se interseccionam, uma reta suporte é criada. O avaliador de contornos encontra as interseções desta reta suporte com as arestas que formam as faces e então determina, através da chamada classificação de pertinência a um conjunto (*Set-Membership Classification*), a classificação dos segmentos da reta suporte que irão dar origem às novas arestas.

Na figura 2.11, pode-se ver uma primitiva (a) e duas instâncias, desta primitiva, posicionadas no espaço antes da avaliação dos contornos (b).

FIGURA 2.11 – Primitivas instanciadas e posicionadas no espaço

Na figura 2.12, pode ser observado o objeto resultante, após a avaliação de contornos (*boundary evaluator*). Como é mostrado na figura, a aresta a, não sofreu alterações, permanecendo como na primitiva. A aresta b, por sua vez, foi transformada em um segmento menor e a aresta c, que não existia foi criada, devido a interseção de duas faces. O ponto T permaneceu o mesmo, o qual é elemento das duas primitivas. Os pontos P e S não existiam nas primitivas e são os pontos que originam a aresta c. Os pontos R, Q, U e V compõem uma face que não sofreu alterações. Já a face formada pelos pontos P, Q, R, S sofreu alterações pela existência da interseção entre as faces.

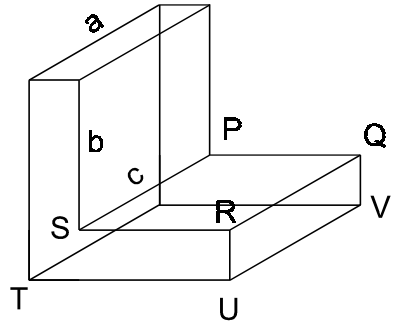


FIGURA 2.12 – Sólido resultante após a avaliação dos contornos

2.6 Classificação da Pertinência a um Conjunto (S*et-Membership Classification*)

A classificação da pertinência a um conjunto opera sobre dois conjuntos de pontos: um conjunto referência (S) e um conjunto de pontos candidato (X). O conjunto candidato X é particionado (classificado), em relação ao conjunto referência S, em três subconjuntos XinS, XonS e XoutS, correspondendo a quais regiões de S, os subconjuntos estão ou não contidos. Se um ponto P $\in$ X pertencer ao interior de S, ele será classificado como IN e será incluído no conjunto XinS. Se um ponto P $\in$ X pertencer ao contorno de S, ele será classificado com ON e será incluído no conjunto XonS e se um ponto P $\in$ X pertencer ao exterior, pontos externos, de S (complemento de S), será classificado como OUT e será incluído no conjunto XoutS.

Para serem feitas as classificações acima primeiramente encontram-se as interseções das fronteiras dos conjuntos referência e candidato.

Para o caso de poliedros após ser feita a determinação de onde cada aresta ou face é interseccionada, o procedimento de classificação (*set membership*) determina a classificação de cada uma das partes seccionadas de um objeto em relação ao outro objeto.

Na figura abaixo apresenta-se um exemplo deste processo de classificação. Neste caso pode-se observar que novos pontos foram gerados a partir das interseções entre os dois objetos A e B. Estas interseções estão representadas pelos pontos 1, F, 3, 4 e R. O próximo passo será fazer as classificações das arestas de um objeto em relação ao outro.

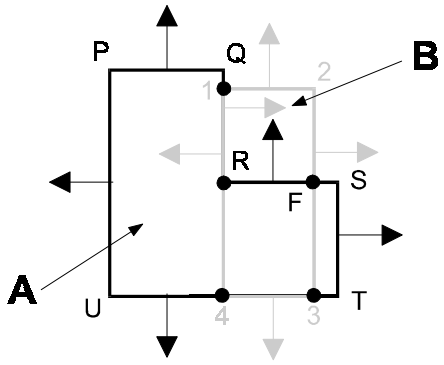


FIGURA 2.13 – Interseções entre dois objetos [MOR 85]

Após as interseções algumas arestas que formam os polígonos sofreram alterações e ficaram organizadas nos seguintes segmentos e pontos:

$\overline{QR} = \overline{Q1}, 1, \overline{1R};$

$\overline{RS} = \overline{RF}, F, \overline{FS};$

$\overline{TU} = \overline{T3}, 3, \overline{34}, 4, \overline{4U};$

$\overline{23} = \overline{2F}, F, \overline{F3}$ *e*

$\overline{41} = \overline{4R}, R, \overline{R1}.$

Nas tabelas abaixo, apresentam-se as classificações dos vértices e arestas da figura 2.13. Os segmentos com a classificação ON, possuem atributos adicionais que serão explicados mais adiante, ainda nesta seção.

TABELA 2.1 – Classificação dos vértices da figura 2.13

Objeto A em relação ao objeto B		**Objeto B em relação ao objeto A**	
Vértice	Classificação	Vértice	Classificação
P	OUT	1	ON
Q	OUT	2	OUT
R	ON	3	ON
S	OUT	4	ON
T	OUT	F	ON
U	OUT		
F	ON		

TABELA 2.2 – Classificação das arestas da figura 2.13

Objeto A em relação ao objeto B			**Objeto B em relação ao objeto A**		
Segmento	Classificação	Orientação	Segmento	Classificação	Orientação
PQ	OUT		12	OUT	
Q1	OUT		2F	OUT	
1R	ON –	Diferente	F3	IN	
RF	IN		34	ON +	Igual
FS	OUT		4R	IN	
ST	OUT		R1	ON –	Diferente
T3	OUT				
34	ON +	Igual			
4U	OUT				
UP	OUT				

Uma aresta, ou um segmento seu, é classificado como ON se todos os seus pontos pertencerem simultaneamente à fronteira dos dois objetos. Na figura 2.13 pode-se observar que os segmentos $\overline{R1} e \overline{34}$ possuem pontos que são comuns aos dois objetos, portanto estão classificados como ON. Para as classificações do tipo ON existe uma subclassificação, elas podem ser ON+ ou ON-.

No caso da figura 2.13, os polígonos possuem vetores normais que apontam para fora. Para verificar a classificação do segmento considera-se a orientação dos vetores normais. Se os vetores estiverem apontando no mesmo sentido a classificação será ON+, caso contrário ON- [RIP 87].

Segundo [RIP 87], problemas geométricos similares podem aparecer de diferentes formas. Considere os seguintes exemplos:

a) Inclusão de um ponto: dado um sólido S e um ponto P; P está dentro, fora ou na fronteira de S?

b) Recorte reta/polígono: dado um polígono P e um segmento de reta R; que porção de R está dentro de P? Que porção está fora? E que porção está na superfície de P?

c) Interseção de polígonos: dados dois polígonos P e Q; qual é o polígono gerado pela interseção de A e B (A ∩ B)?

Nas figuras 2.14, 2.15 e 2.16 pode-se ver alguns exemplos destas classificações. Provas e detalhes matemáticos relacionados a esta classificação podem ser encontrados em [REQ 78].

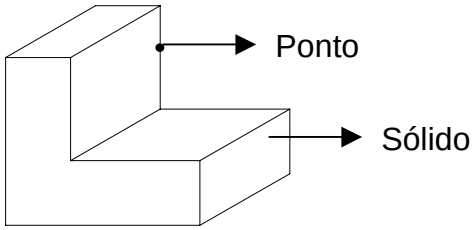


FIGURA 2.14 – Inclusão de um ponto [RIP 87]

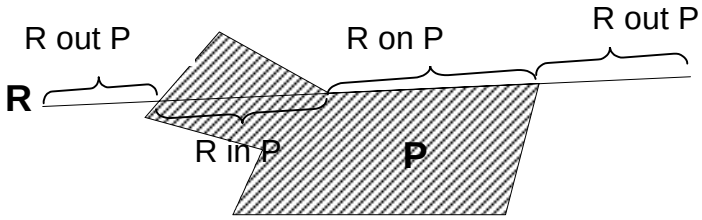


FIGURA 2.15 – Recorte reta/polígono [RIP 87]

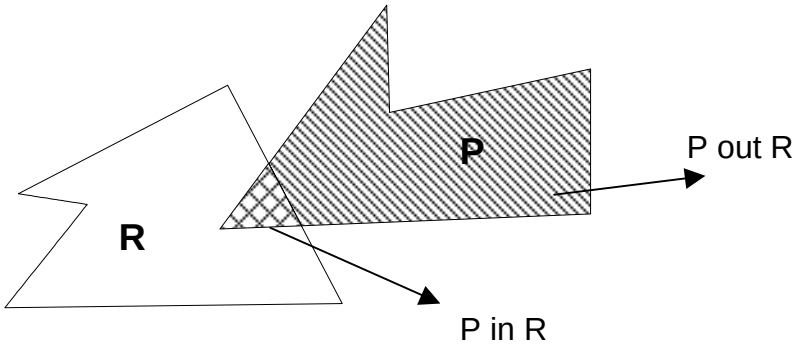


FIGURA 2.16 – Interseção de polígonos [RIP 87]

Após ser concluída a classificação dos vértices e arestas pela classificação da pertinência a um conjunto (*Set-Membership Classification*), o avaliador de contornos faz a seleção dos elementos que farão parte do novo sólido que está sendo modelado, levando em consideração as classificações das arestas e a(s) operação(ões) booleana(s) envolvida(s) na modelagem. O processo de avaliação de contornos, utilizado neste trabalho, está descrito no capítulo cinco.

2.7 Considerações Finais

Neste capítulo foram descritos os conceitos que caracterizam a técnica de modelagem CSG (*Constructive Solid Geometry*). O uso das primitivas geométricas, as transformações geometricas e operações booleanas que podem ser aplicadas sobre tais primitivas, visando a geração dos objetos CSG.

Também foram apresentados a hierarquia dos objetos na forma de árvore binária e os procedimentos para a classificação das arestas dos objetos envolvidos na modelagem (*Set-Membership Classification*). A representação dos contornos dos objetos (*Boundary Evaluator*), por fim, foi abordada.

3 Fontes de Luz e Sombras

Em imagens geradas por computador um dos fatores a ser considerado é a presença de sombras na imagem. Uma sombra somente irá existir se houver, pelo menos, uma fonte de luz na cena. Portanto, a fonte de luz é um fator importante na geração de imagens por computador.

Luz é uma forma de radiação eletromagnética. A energia eletromagnética se propaga como uma onda (ou conjunto de ondas) que pode ser caracterizada por outros comprimentos de onda ou por sua freqüência. O espectro eletromagnético inclui ondas de rádio, infravermelho e uma faixa de freqüência, associadas à visão. O espectro visível, que tem comprimento de onda entre 350 e 780 nanometros (nm), é chamado luz [ANG 2000].

Uma fonte de luz é simplesmente mais um objeto na especificação da cena. O que a distingue dos outros objetos é o fato de emitir luz. Dessa forma, as fontes de luz apresentam suas características físicas sendo necessários os seguintes atributos para a sua modelagem:

- Geometria da fonte de luz
- Distribuição de intensidade luminosa
- Distribuição espectral emitida

3.1 Geometria da Fonte de Luz

A geometria da fonte de luz é necessária para se calcular corretamente as sombras projetadas, pois algumas porções da fonte de luz podem estar parcialmente obstruídas, criando o efeito de penumbra que é típico de ambientes reais [NAS 92]. Quanto à geometria, as fontes de luz podem ser classificadas em quatro tipos:

- Fontes Direcionais – Para este tipo de fonte define-se um vetor unitário, que tem a função de especificar a direção da iluminação. Tais fontes são utilizadas para simular fontes consideradas no infinito e que emitem raios paralelos, o sol é um exemplo de fonte de luz direcional.
- Fontes Puntiformes – São definidas como um ponto do espaço e emitem luz em todas as direções.
- Fontes Lineares – São definidas por dois pontos no espaço que definem um segmento de reta e por um vetor que define a direção da iluminação.
- Fontes de Área – São definidas pelos vértices do polígono que formam uma superfície e por um vetor que define a direção da iluminação.

3.2 Distribuição de Intensidade Luminosa

Segundo [VER 84], a distribuição de intensidade luminosa é a característica mais importante de uma fonte de luz, pois várias fontes de luz não emitem luz igualmente em todas as direções, portanto, uma distribuição de

intensidade que descreva a variação de intensidade direcional sobre a superfície da fonte de luz é necessária para representar esse fenômeno.

Para representar estas distribuições são utilizados os diagramas goniométricos, onde para cada ângulo em torno de um determinado eixo da fonte luminosa é especificada a sua intensidade. A utilização destes diagramas para modelar o comportamento das fontes de luz requer a determinação de um vetor direção, a partir do qual os ângulos serão medidos. Ver figura 3.1.

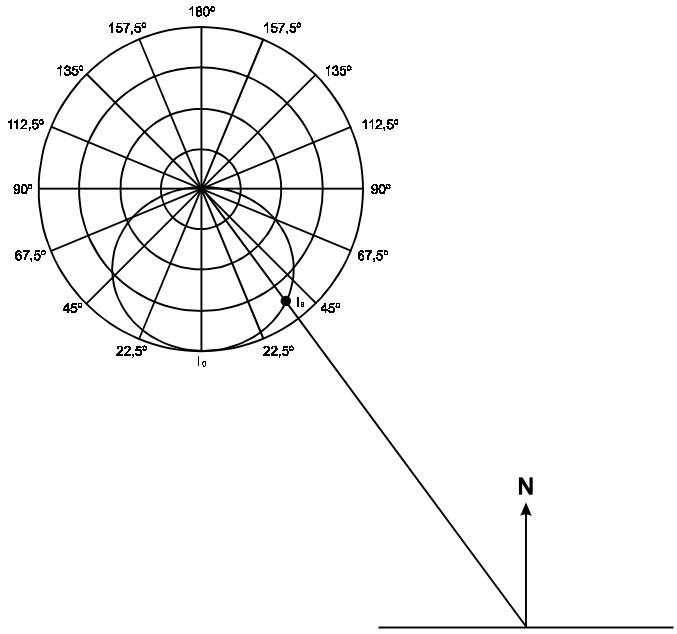


FIGURA 3.1 – Fonte de luz e diagrama goniométrico [FOL 97]

3.3 Distribuição Espectral Emitida

A distribuição espectral é fundamental para a percepção de cores. O olho humano pode detectar ondas eletromagnéticas com comprimento de onda desde 350nm até 780 nm [FOL 97]. Essa faixa de variação é chamada de faixa visível do espectro.

A percepção de cor no ser humano é provocada pela energia radiante que atinge o olho [GOM 94]. Como cada um dos três receptores de cor do olho humano responde diferentemente, são esperadas três respostas, de acordo com o estímulo provocado. Uma resposta para o vermelho (R, *red*), uma para o verde (G, *green*) e outra para o azul (B, *blue*). Na Computação Gráfica, o que habitualmente é feito, é amostrar a intensidade espectral da fonte emissiva de energia radiante somente para o vermelho, verde e o azul [NAS 92].

3.4 Controles Para Fontes de Luz Puntiformes

Direção e concentração são os dois controles básicos que podem ser aplicados sobre as fontes de luz. Esses controles permitem direcionar e ajustar a distribuição da luz sobre uma determinada área [NAS 92].

Para restringir os efeitos da luz em uma determinada área da cena, Warn [WAR 83] implementou abas (*flaps*) e cones, permitindo que se tenham controles sobre as fontes de luz. Com isso pode-se também simular alguns efeitos de estúdios fotográficos.

As abas (*flaps*) são utilizadas para simular os *barn doors* dos estúdios fotográficos e confinam o caminho da luz a um determinado intervalo nas coordenadas x, y e z do universo. Valores mínimo e máximo podem ser especificados para cada coordenada. Cada um desses valores é chamado de aba. Cada aba possui também um *flag* indicando se ela esta ativada ou desativada. Durante o cálculo da função da intensidade de cor, o modelo de iluminação é avaliado para uma determinada fonte de luz, somente se as coordenadas do ponto estiverem dentro do intervalo das abas ativadas [FOL 97]. Ver figura 3.2.

Os cones são usados para simular a luz dos holofotes (*spotlights*). O tamanho de um cone de luz é especificado por um ângulo β em torno da direção da luz. Ver figura 3.3. Se o ângulo γ que o vetor $\overrightarrow{LP}$ faz com o vetor direção da luz ($\overrightarrow{LD}$) for maior que o ângulo β, a luz não terá nenhum efeito no ponto P [FOL 97, NAS 92].

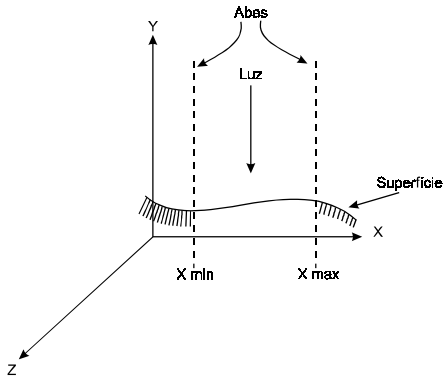


FIGURA 3.2 – Abas em X [FOL 97]

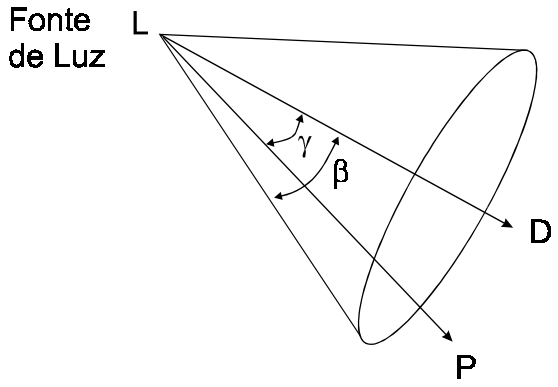


FIGURA 3.3 – Cone de luz [FOL 97]

3.5 Sombras

Sombra é uma região escura de uma cena iluminada. Esta região escura é determinada pela ausência de energia luminosa em certas regiões da cena. Se um objeto impede a passagem de luz, seja total ou parcial, uma região de sombra será formada.

A presença de sombras é uma constante na maioria das cenas iluminadas, e essa presença é fundamental para aumentar o realismo. As sombras estabelecem diversos níveis de profundidade na cena, sendo eficazes em melhorar a impressão espacial na imagem de um objeto tridimensional, pois evitam que os objetos pareçam estar flutuando no ar, fornecem fortes indicações sobre formas, posições e características das superfícies dos objetos.

A forma da sombra depende do tipo de material do objeto que está obstruindo a luz (transparente, opaco, etc) e da geometria da fonte de luz. Porém de um modo geral, em relação a cada fonte de luz, a sombra se caracteriza pela presença de duas regiões. Uma região, chamada região de umbra, cuja intensidade luminosa é nula; e outra região, região de penumbra, ver figura 3.4, na qual a intensidade luminosa varia de zero até a intensidade do ambiente. A presença das regiões de umbra ou penumbra na cena depende da relação entre a geometria do objeto que produz a sombra, e da geometria da fonte de luz [GOM 90].

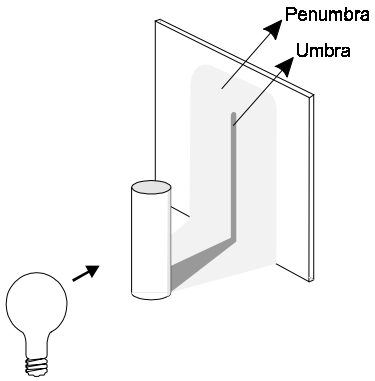


FIGURA 3.4 – Formação das regiões de umbra e penumbra [WAT 2000].

3.6 Tipos de Sombras

Segundo [NAS 92] as sombras podem ser classificadas basicamente em quatro tipos.

a) Sombras “bem delimitadas”

São caracterizadas pela presença de apenas uma região da sombra, a umbra. O domínio das fontes de luz que geram este tipo de sombra se restringe a fontes puntiformes e direcionais. Ver figura 3.5.

b) Sombras “suaves”

As sombras “suaves” são caracterizadas pela presença das duas regiões de sombra: umbra e penumbra. Fontes lineares e superficiais geram este tipo de sombra. Ver figura 3.4.

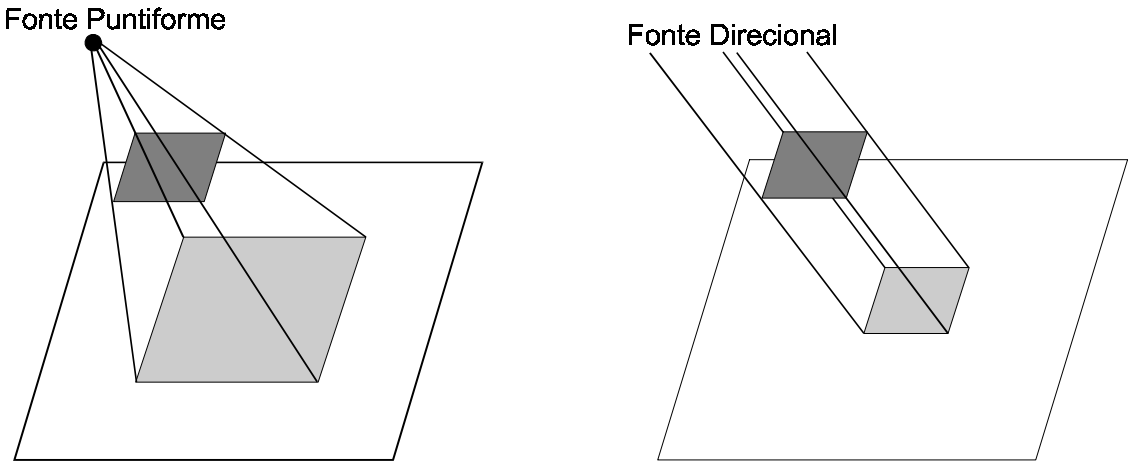


FIGURA 3.5 – Sombra “bem delimitada” [NAS 92]

c) Sombras de objetos transparentes

Como um objeto transparente não obstrui totalmente a luz a determinação de sombras em cenas com objetos transparentes é mais complexa, pois não basta encontrar somente o primeiro objeto que está obstruindo a luz (se este for transparente), mas todos os possíveis objetos que estão na direção da fonte de luz.

As sombras geradas por objetos transparentes podem possuir, além da umbra e penumbra, as regiões de cáusticas. Quando a luz passa por uma superfície curva que é um transmissor especular (por exemplo, um copo de água), esse corpo se comporta como se fosse uma lente, de modo que a luz transmitida é focada para a superfície de sombra formando padrões, chamados de “cáusticas” [GOM 90].

d) Sombras coloridas

Uma sombra não é necessariamente uma região negra da cena. Se forem utilizadas fontes de luz coloridas as sombras geradas podem ser coloridas. Por exemplo, se forem utilizadas três fontes de luz coloridas (vermelha, verde e azul), a região de sombra resultante da obstrução da luz em uma determinada fonte por um objeto será influenciada pela cor das outras duas fontes de luz.

3.7 Algoritmos de Sombra

De um ponto de vista puramente geométrico, o estudo das sombras é equivalente ao estudo da visibilidade do ponto de vista da posição da fonte de luz. Os objetos visíveis a partir desse ponto de vista não estão na sombra, os objetos invisíveis são os que estão em uma região de sombra da cena.

O número de algoritmos de geração de sombras, existente na literatura é bastante grande. Segundo Gomes e Velho [GOM 90], os algoritmos para a geração de sombras podem ser classificados em três categorias: algoritmos geométricos, algoritmos de mapeamento e algoritmos físicos.

Nos algoritmos geométricos as regiões de sombra da cena são modeladas como um objeto em cena, para posteriormente serem processadas de modo adequado por ocasião do cálculo da função de intensidade de cor. Os algoritmos de mapeamento procuram construir uma textura com a informação de sombra de modo a utilizá-la para modular o cálculo da intensidade de cor em cada ponto da cena. Nos algoritmos físicos a presença ou ausência de sombra em uma cena é conseqüência imediata do próprio método de integração da equação de iluminação.

Nas seções abaixo têm-se uma breve descrição dos principais algoritmos geradores de sombras encontrados na literatura [GOM 90, NAS 92, FOL 97, WAT 2000].

3.7.1 Sombras Geradas por Projeção

É um método puramente geométrico de geração de sombras, que consiste em modelar a região de sombra a partir da projeção do objeto do ponto de vista da fonte de luz. É um método eficiente nos casos em que a sombra é projetada em uma superfície plana como piso ou parede [BLI 88].

Este método só funciona para ambientes que utilizam como primitiva de modelagem o polígono e gera somente sombras “bem delimitadas”, podendo utilizar fontes de luz direcionais e puntiformes e tem como característica principal a alta velocidade de execução. Ver figura 3.6.

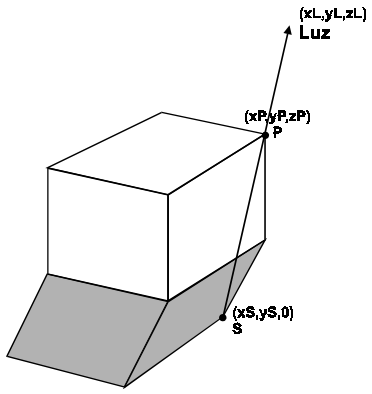


FIGURA 3.6 – Sombra projetada [WAT 2000]

3.7.2 Volumes de Sombra

Neste método faz-se o cálculo da função de iluminação em um ponto P, para tanto se utiliza um algoritmo que determina se o ponto P está dentro do volume de sombra, de modo a fazer a atenuação necessária no cálculo da intensidade [CRO 77].

No caso de poliedros convexos, para cada objeto cria-se um volume de sombra, definido pela fonte de luz e os polígonos que formam as faces do objeto. Este volume é definido pelas faces invisíveis do objeto, olhando-se do ponto de vista da fonte de luz.

O volume da sombra é composto por vários polígonos, onde cada um deles é definido pela fonte de luz e os vértices de cada uma das arestas de contorno do objeto. Estes polígonos são “invisíveis”, portando não são renderizados, são utilizados para determinar se outros objetos estão na sombra ou não. Ver figura 3.7.

Cada polígono do volume da sombra possui um vetor normal que aponta para fora, existindo assim dois tipos de polígonos, os polígonos frontais (*front-facing*) e os traseiros (*back-facing*). Os polígonos frontais são aqueles, cujo ângulo entre o vetor normal e o ponto em questão que chega até o observador, for menor ou igual a 90 graus, caso contrário, ou seja, se o ângulo encontrado for maior que 90 graus, então o polígono será traseiro.

Se for considerada a posição do observador, um polígono de sombra frontal faz com que tudo que esteja atrás dele seja considerado sombra e um polígono de sombra traseiro cancela o efeito de um frontal.

A operação do algoritmo para um *pixel* em particular. Considera-se um raio do ponto de vista do observador até o *pixel* e a relação entre o polígono real e o polígono da sombra ao longo deste vetor. Existe um contador para o *pixel*, e ao percorrer a lista de polígonos o contador é incrementado quando um polígono frontal é encontrado e decrementado quando um polígono traseiro é encontrado. O valor deste contador é informado quando encontra-se um polígono real, verificando-se se o *pixel* está na sombra ou não, se o valor do contador for 0 (zero) então o polígono não está na sombra, se o valor do contador for diferente de zero, então o ponto está na sombra, ou seja, o *pixel* estará na sombra se o vetor interseccionar mais polígonos frontais do que traseiros. Ver figura 3.8.

Este algoritmo aceita como primitivas de modelagem o polígono e só gera sombras “bem delimitadas” de fontes puntiformes.

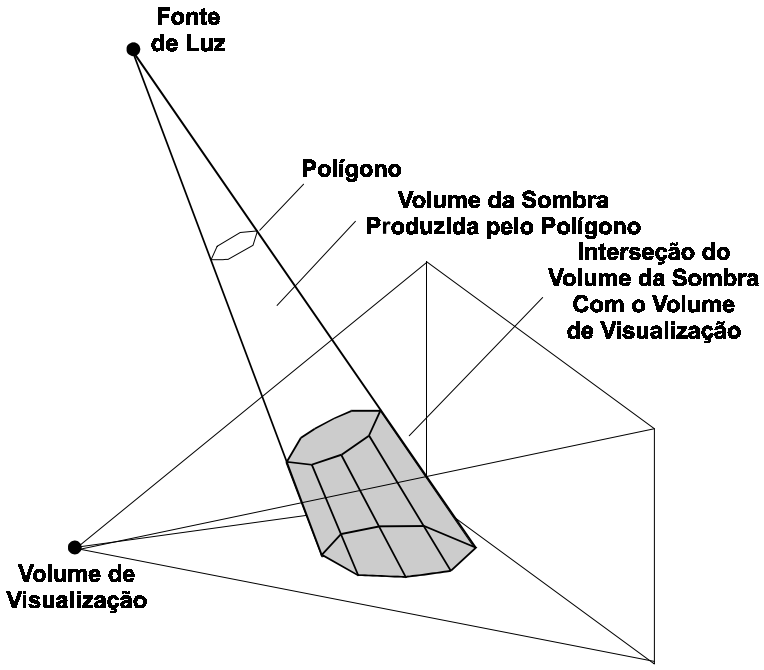


FIGURA 3.7 – Volume da sombra [WAT 2000]

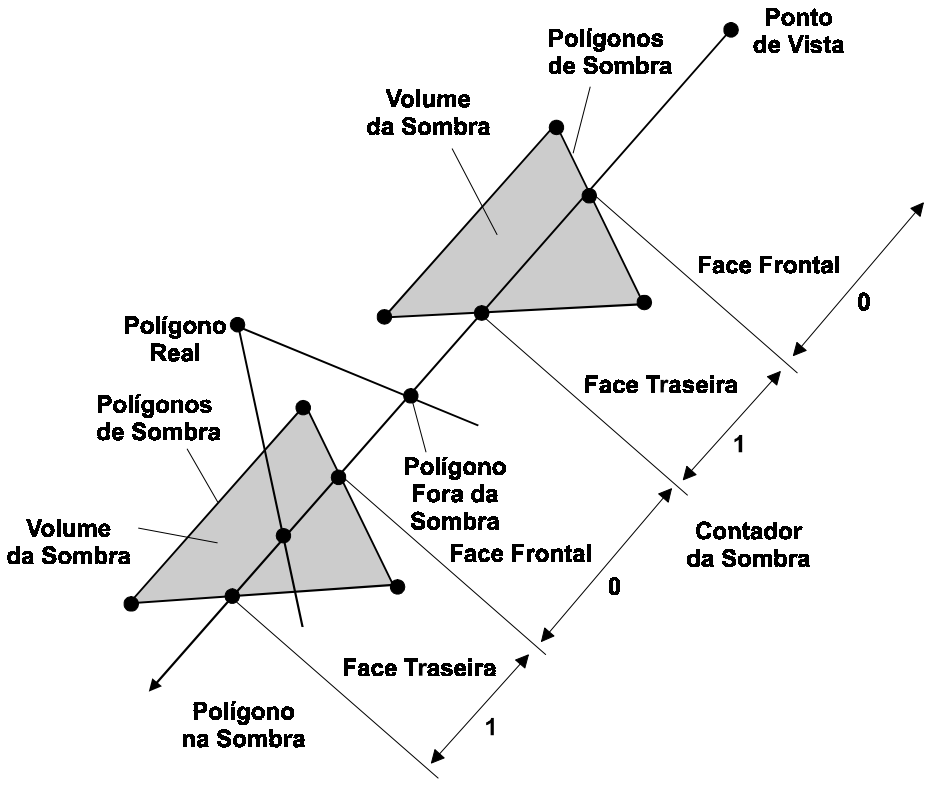


FIGURA 3.8 – Polígonos frontais e traseiros da sombra [WAT 2000]

3.7.3 Algoritmo de Atherton, Weiler e Greenberg

Este algoritmo geométrico explora a sombra como sendo um problema de visibilidade do ponto de vista da fonte de luz. O algoritmo cria um modelo constituído de todos os pontos da cena que não são visíveis a partir da fonte de luz, esse modelo é então transformado e anexado ao modelo original da cena [ATH 88].

A cena é processada utilizando-se duas vezes o mesmo algoritmo, uma para a fonte de luz e outra para o observador. O primeiro passo, que analisa a imagem para o ponto de vista da fonte de luz, através de um algoritmo para remoção de superfícies ocultas, obtêm as partes que estão sendo iluminadas. Essas partes iluminadas são novos polígonos, que uma vez definidos são acrescentados ao ambiente original, sendo identificados por seu polígono pai e tratados como polígonos detalhes das superfícies originais.

O segundo passo envolve a determinação das superfícies visíveis do ponto de vista do observador e o cálculo da intensidade luminosa para cada ponto dos polígonos, levando-se em consideração a informação de sombra obtida no primeiro passo.

Um algoritmo para a renderização da cena é utilizado, onde as superfícies cobertas pelos polígonos detalhe são renderizadas como iluminadas. Por outro lado, as superfícies visíveis e não cobertas pelos polígonos detalhe são renderizadas na sombra.

Este algoritmo pode ser usado tanto para fontes puntiformes como para fontes direcionais e trata somente objetos modelados por primitivas poliédricas. A figura 3.9 apresenta um exemplo simples da aplicação deste algoritmo, onde (a) representa um objeto poligonal no sistema de coordenadas do universo, o objeto (b) representa o plano de visão da posição da fonte de luz, o (c) é a remoção das superfícies ocultas do ponto de vista da fonte de luz, o (d) representa os polígonos visíveis de (c) transformados para o sistema de coordenadas do universo. O objeto (e) mostra partes de (a) e (d) combinadas para produzir as partes que contêm os polígonos da sombra. A partir do (e) pode-se produzir qualquer visão do objeto com sombras, como mostrado na figura 3.9 (f).

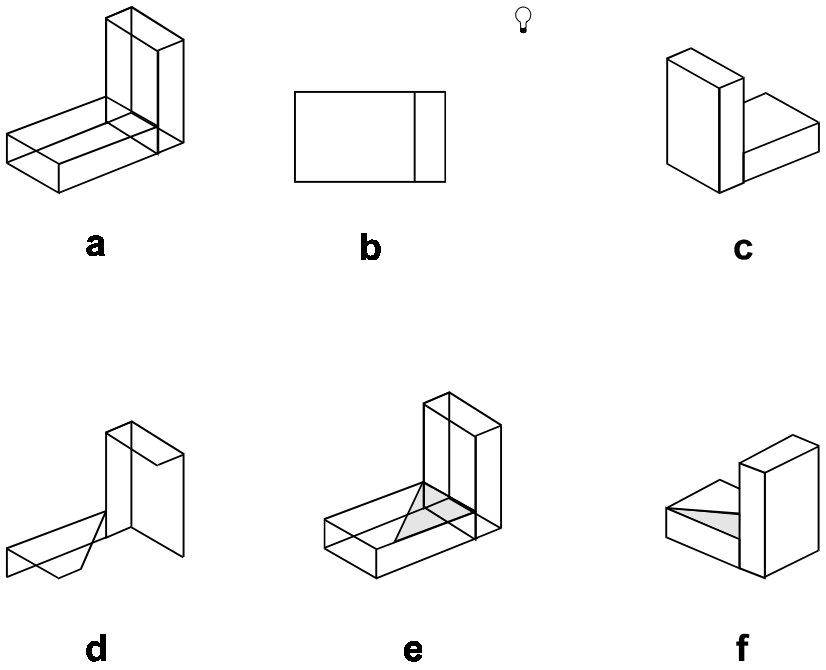


FIGURA 3.9 – Algoritmo de Atherton, Weiler e Greenberg [WAT 2000]

3.7.4 Algoritmo de Willians

Esta técnica requer uma sombra separada para cada fonte de luz. O algoritmo é executado em dois passos, primeiro uma visão da cena é constituída do ponto de vista da fonte de luz, utilizando-se de um algoritmo *z-buffer*, onde somente os valores da coordenada z (que determinam a profundidade) precisam ser computados e armazenados, formando um mapa de profundidade chamado de *light-buffer* [WIL 78].

No segundo passo, a cena é renderizada usando-se um algoritmo *z-buffer*. Se um ponto é visível, então uma transformação de coordenadas é usada para mapear a coordenada do ponto no universo (x,y,z), para as coordenadas do ponto no sistema de referência da fonte de luz (x',y',z'). Os pontos (x', y') são usados para indexar a sombra e o valor da profundidade correspondente é comparado com o z'. Se o z' for maior que o valor de z, então

o ponto está na sombra e sua intensidade de sombra é usada, caso contrário o ponto é renderizado normalmente.

3.7.5 Algoritmo de Appel

Este algoritmo emprega a técnica de invisibilidade quantitativa para determinar a visibilidade do ponto de vista do observador e da fonte de luz. A invisibilidade quantitativa de uma linha é representada pelo número de polígonos frontais que escondem tal linha do observador ou da fonte de luz. Quando uma linha passa por trás de um polígono frontal, sua invisibilidade quantitativa é incrementada de uma unidade, e quando ela sai de trás deste polígono sua invisibilidade quantitativa é decrementada de uma unidade. Uma linha é visível somente quando sua invisibilidade quantitativa for zero [APP 68]. Ver figura 3.10.

A idéia básica deste algoritmo é que as interseções geradas pelos planos de recorte podem ser avaliadas quanto às mudanças na invisibilidade quantitativa. Essas interseções são geradas quando cada plano do corte atravessa as superfícies dos objetos e são projetadas no plano de projeção. Cada uma destas linhas é composta por um ou mais segmentos resultantes das interseções. Os segmentos que são completamente invisíveis para o observador, isto é, cuja invisibilidade quantitativa não for zero, podem ser descartados. Aqueles que estão localizados em superfícies traseiras em relação à fonte de luz estão completamente na sombra e não precisam ser analisados. E aqueles localizados em superfícies frontais em relação à fonte de luz, devem ser analisados quanto à invisibilidade quantitativa do ponto de vista da fonte de luz. Os segmentos invisíveis para a fonte de luz estão na sombra e os visíveis são iluminados pela mesma.

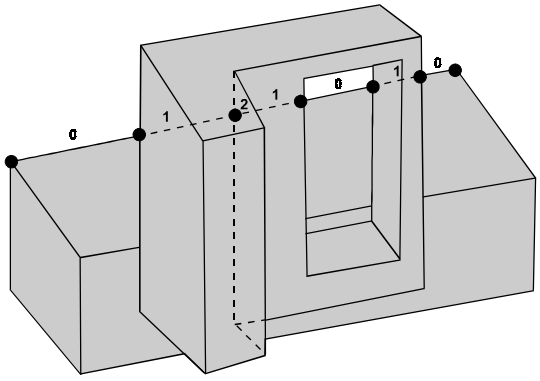


FIGURA 3.10 – Invisibilidade quantitativa de uma linha [NAS 92]

3.7.6 Traçado de Raios (*Ray-tracing*)

Nesta técnica de renderização um centro de projeção, que é o observador, e uma janela como plano de visão arbitrário são selecionados. Esta janela é dividida numa grade regular, onde os elementos correspondem aos *pixeis* da resolução desejada.

Raios são disparados do observador em direção a cada *pixel*, calculando as interseções do raio com as superfícies dos objetos da cena e estabelecendo que a superfície atingida que possuir a menor distância em relação ao observador será a superfície visível.

Para a geração das sombras projetadas “bem delimitadas”, é necessário disparar um raio secundário a partir do ponto de interseção em direção à fonte de luz. Se o raio interceptar qualquer objeto entre a sua origem e a fonte de luz, então o ponto do objeto está na sombra da fonte de luz.

A técnica básica de *Ray-Tracing* não requer nenhuma memória adicional e pré-processamento para a determinação de sombras projetadas e permite utilizar, além de polígonos, superfícies curvas definidas implícita ou parametricamente, contudo, a complexidade de execução deste método para a determinação das sombras é bem alta.

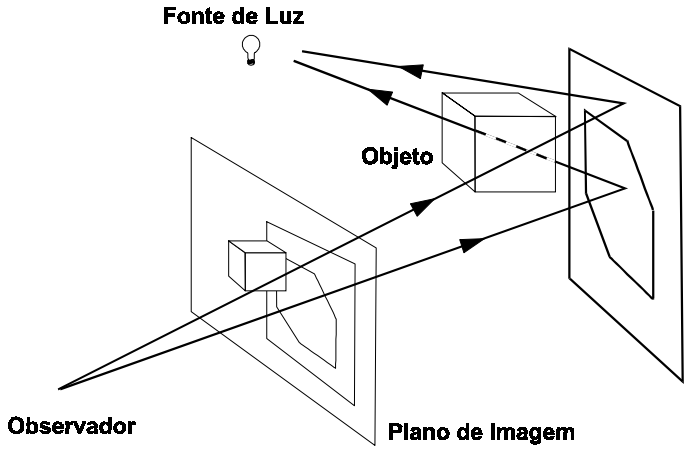


FIGURA 3.11 – *Ray-Tracing* [JAN 91]

3.7.7 Algoritmos Geradores de Sombra Para Objetos CSG

Até este ponto do trabalho, foram descritos algoritmos clássicos para a geração de sombras. O objetivo principal do trabalho é tratar a geração de sombras em objetos modelados pela Geometria Sólida Construtiva. Esta seção concentra-se neste assunto, onde são apresentados e descritos algoritmos que podem ser utilizados na renderização de cenas CSG, com a geração de sombras.

3.7.7.1 Renderizando uma Descrição CSG com *Ray-Tracing*

Segundo [WAT 2000], o fator que distingue uma representação CSG de outras formas é de que esta não é uma representação direta da geometria do objeto, algo mais precisa ser interpretado. Na representação de um objeto CSG a base de dados é uma árvore que relata um conjunto de objetos primitivos que são combinados, através de operações booleanas, até que se chegue a um outro objeto. Este tipo de representação facilita o poder de interatividade. O preço pago por isso é uma renderização expansiva e complexa.

Uma descrição CSG consiste somente das primitivas geométricas e suas dimensões, sua relação espacial e o operador de conjuntos que combina as primitivas. Esta informação não mostra um objeto pronto para a renderização. O objeto composto, que contém fatores geométricos como as interseções, não estão nas primitivas. Esta construção do objeto é a maior dificuldade.

O principal problema envolvido na renderização de objetos CSG é encontrar alguma maneira satisfatória de renderizar o objeto representado pela base de dados CSG.

O contorno de um objeto descrito por CSG pode ser avaliado através de um raio que é lançado para cada *pixel* do plano da imagem. Em uma simples projeção paralela pode-se explorar o espaço do objeto com um conjunto de raios paralelos. O processo divide-se em dois estágios. Primeiro considera-se um simples raio. Toda primitiva instanciada é comparada com este raio para ver se o mesmo intersecciona a primitiva. Este meio soluciona o problema dos testes de interseções. Todas as interseções são ordenadas pela profundidade Z. Tem-se então uma classificação primitiva/raio para cada raio, ver figura 3.12. Pode-se então aplicar qualquer operação booleana entre as primitivas encontradas ao longo do raio.

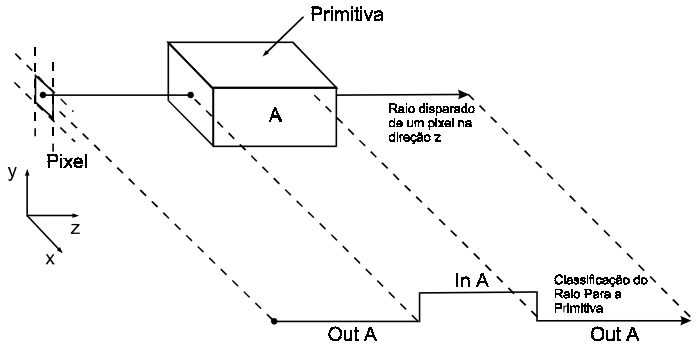


FIGURA 3.12 – Classificação de um raio para a primitiva [WAT 2000]

Na figura 3.13, pode-se ver a avaliação das operações booleanas entre as duas primitivas ao longo do raio.

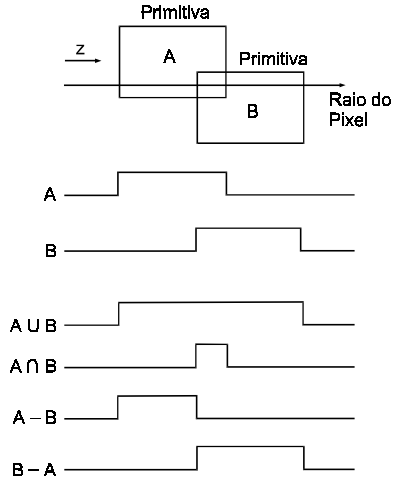


FIGURA 3.13 – Avaliação das operações booleanas ao longo do raio [WAT 2000]

Para a inclusão de sombras em um algoritmo de *Ray-Tracing*, calcula-se um vetor $\vec{L}$ em direção à fonte de luz, e insere-se um teste de interseção dentro do algoritmo, onde $\vec{L}$ é considerado como um outro raio qualquer. Se $\vec{L}$ interseccionar qualquer objeto então o ponto está na sombra e a intensidade da iluminação deste ponto, consequentemente é reduzida. Se $\vec{L}$ não interseccionar nenhum objeto, então o ponto está iluminado, ou seja, fora da região de sombra. Ver figura 3.14.

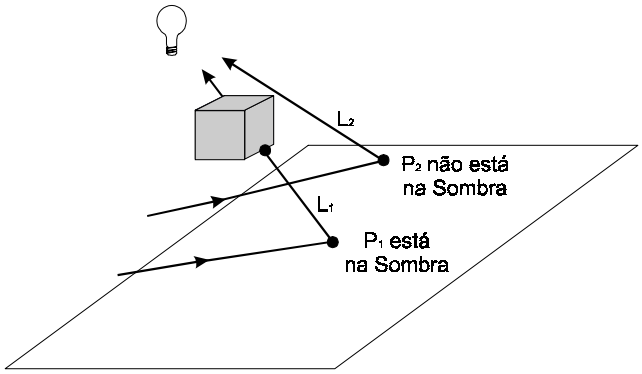


FIGURA 3.14 – Sombra com *Ray-Tracing* [WAT 2000]

3.7.7.2 Renderizando Descrições CSG com Traçamento *Beam* (feixe)

Este algoritmo pode ser considerado como uma alternativa para *Ray-Tracing*. O método apresentado está baseado na subdivisão recursiva do espaço da imagem principal [GHA 98].

Em um primeiro momento, a região correspondente a entrada da tela é associada à visão *beam* e é subdividida recursivamente até uma região limite ser obtida ou um pixel ser alcançado. Uma visão *beam* é uma pirâmide traçada do observador para a cena. Ver figura 3.15. Esta pirâmide é definida pela posição do observador e quatro raios que passam sobre os quatro cantos da região da imagem. Se a região é ambígua, isto é, pode ser subdividida em quatro, quatro novas sub-pirâmides são traçadas, e assim sucessivamente.

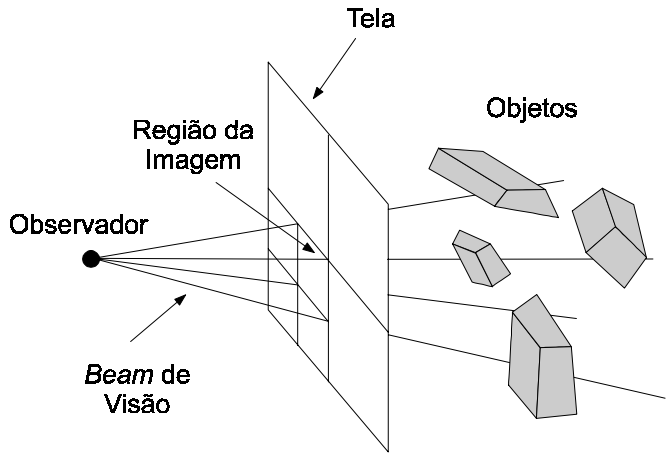


FIGURA 3.15 – Subdivisão da imagem e *beam* de visão [GHA 98]

Para determinar se uma região da imagem é ambígua ou uniforme, adiciona-se para cada lado do *beam*, o primeiro objeto intersecionado, se este objeto existir. Assim, pode-se considerar quatro casos de interseções com o lado do objeto:

1. Existe pelo menos uma interseção com o objeto, mas todas as arestas do *beam* não interseccionam o mesmo objeto. Ver figura 3.16 (a).
2. Todas os arestas interseccionam o mesmo objeto, mas não o mesmo lado do objeto. Figura 3.16 (b).
3. Todos as arestas interseccionam o mesmo lado do mesmo objeto. Figuras 3.16 (c) e 3.16 (d).
4. Não há nenhuma interseção das arestas do *beam* com objetos. Figura 3.16 (e) e 3.16 (f).

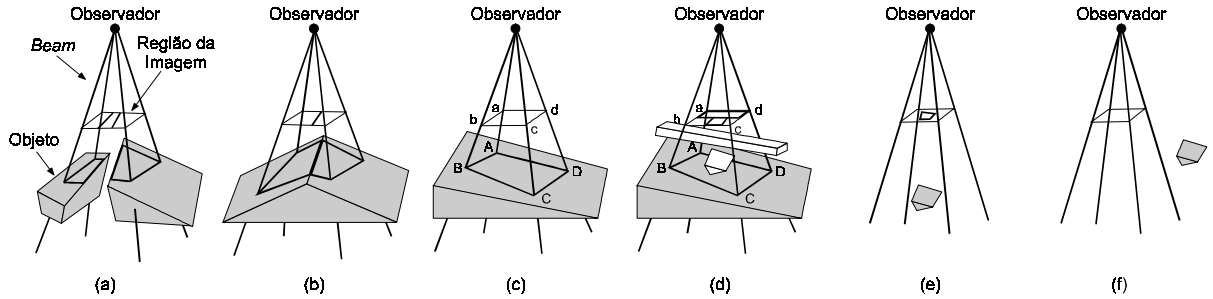


FIGURA 3.16 – Interseções com os lados do objeto [GHA 98]

Nos dois primeiros casos (casos 1 e 2), há pelo menos um lado visível dentro do *beam* (figura 3.16 (a) ou 3. 16 (b)). Assim, a região ambígua é obvia e futuras subdivisões na região da imagem são necessárias.

O caso (3) pode ser subdivido em dois subcasos:

- Não há nenhum lado do poliedro visível dentro do *beam* (figura 3.16 (c)), consequentemente a região é uniforme e subdivisões não são requeridas.
- Há pelo menos um lado do poliedro visível dentro do *beam* (figura 3.16 (d)), a região é ambígua e futuras subdivisões são requeridas.

O problema é verificar se o *beam* truncado abcdABCD contém ou intersecciona um objeto. A solução está em realizar uma subdivisão espacial, sendo que cada objeto possui uma lista de *voxels* (subespaços) com os quais o mesmo se intersecciona. A idéia é procurar quais são os *voxels* que estão totalmente ou parcialmente localizados no interior do *beam* e então associar a estes a sua lista de objetos ordenados pela posição de cada objeto relativo ao *beam*. Ver figura 3.17.

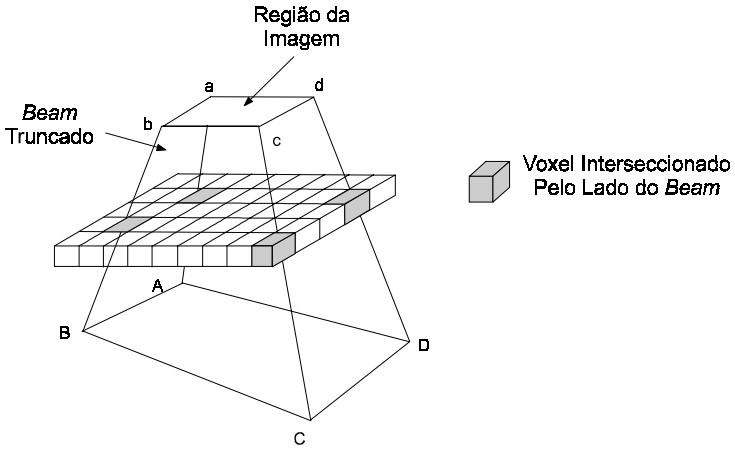


FIGURA 3.17 – *Voxels* interseccionados pelo lado do *beam* [GHA 98]

Cada *voxel* atravessado tem a sua posição associada relativamente ao *beam*. Para determinar se um objeto está do lado de fora de um *beam*, usa-se a seguinte regra: se existe um plano que separa o espaço em um semi-espaço que contém completamente um *beam* e um semi-espaço que contém completamente o objeto, então o objeto está do lado de fora do *beam*. Prudentemente as separações dos planos são:

- O plano da imagem (figura 3.18 (a)),
- O plano que contém ABCD (figura 3.18 (b)),
- Um dos quatro lados do *beam* (figura 3.18(c)),
- Todos os planos tangentes para uma aresta do *beam* que contém $\vec{e}$ e $\vec{v} = \vec{E} \times \vec{e}$ (figura 3.18(d), onde $\vec{E}$ é um vetor paralelo a uma aresta do poliedro.

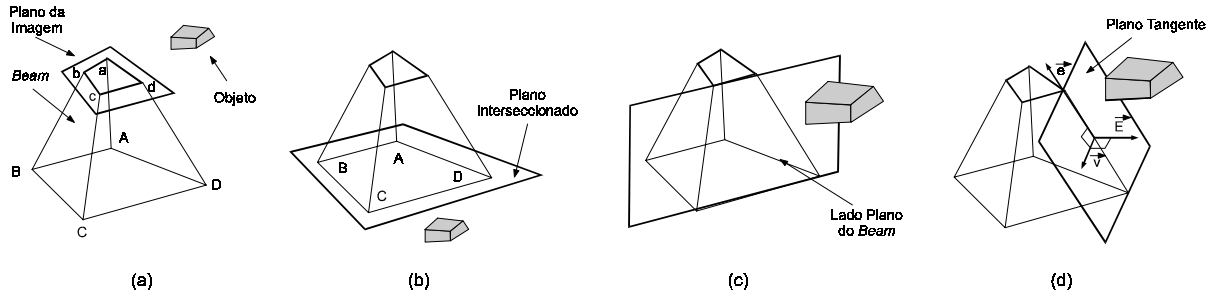


FIGURA 3.18 – Separando planos [GHA 98]

Se um destes planos for igual à regra anterior, o objeto está do lado de fora do *beam* e então se testa o próximo objeto. Se todos os objetos consideráveis estão fora do *beam* (figura 3.16 (c)), a região é uniforme e subdivisões não são necessárias. Em outros casos o *beam* pode conter ou interseccionar objetos, assim a região correspondente para o *beam* é considerada ambígua e subdivisões são necessárias (figura 3.16 (d)).

No caso 4 (figuras 3.16(e) ou 3.16(f)), não há nenhuma interseção entre as arestas do *beam* e o objeto. Portanto utiliza-se a mesma regra de determinação dos objetos que estão fora do *beam*, de forma semelhante ao caso anterior.

3.7.7.2.1 *Beams* de Sombras

Quando os testes anteriores chegam a uma região uniforme, então a visão correspondente ao *beam* não requer subdivisões. Isto significa que somente um lado do poliedro está visível no *beam*. A interseção ABCD deste lado com o *beam* (figura 3.19) é usada para produzir um *beam* secundário, chamado *beam* de sombra. Um *beam* de sombra é uma pirâmide que tem ABCD como sua base e a fonte de luz S com seu ápice (figura 3.19). A visibilidade de ABCD de S é determinada por novos testes. Estes testes são similares aos usados para os *beams* visíveis, onde é necessário distinguir entres os três seguintes casos:

1. ABCD é totalmente visível de S e assim está diretamente iluminado;
2. ABCD está totalmente escondido de S e desta forma está dentro da sombra;
3. ABCD é parcialmente visível de S.

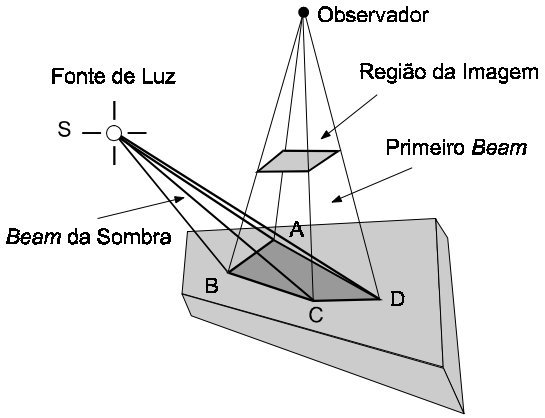


FIGURA 3.19 – *Beam* da sombra [GHA 98]

Nos dois primeiros casos não existe ambigüidade, subdivisões não são necessárias e uma nova fonte de luz, se existir, é testada. No caso 3, a região correspondente da imagem para ABCD é subdividida. Pode-se notar que, neste caso, não é necessário executar o teste de visibilidade, para novos quatro *sub-beams*. Testa-se somente o *sub-beam* produzido da sombra para determinar a visibilidade de suas interseções. Finalmente, uma região é considerada uniforme se neste passo seu *beam* de visão e todos os seus *beams* de sombra são uniformes. Os testes usados para os *beams* de sombra são simples como aqueles usados para os *beams* de visão.

Uma região é considerada uniforme quando todos os *beams* são testados e quando não há visibilidade ambígua dentro deles. A cor computada na região da imagem é executada por *Ray-Tracing*. Durante o processo de interseções com os objetos dois tipos de regiões podem ser encontrados: regiões maiores que um *pixel* e regiões do tamanho de um *pixel*.

No primeiro caso, existe uma região correspondente para alguns *pixels*. Quatro raios que passam nos cantos da região são traçados, a cor dos *pixels* é a média das cores alcançadas para os quatro raios. No segundo caso, a cor do *pixel* é exibida na tela.

3.7.7.3 Renderizando modelos CSG com *ZZ-Buffer*

O *ZZ-buffer* é um algoritmo para aceleração da renderização de cenas onde se aplica a técnica *Ray-Tracing*. Pode ser aplicado a uma grande variedade de cenas, incluindo aquelas que possuem pequenos detalhes, texturas, superfícies transparentes e sombras. Este algoritmo também pode ser utilizado em renderizações de cenas definidas por Geometria Sólida Construtiva [SAL 90].

Nos modelos renderizados por este algoritmo é importante ressaltar a descrição das primitivas. Por exemplo, um objeto A é descrito por uma função característica $C_A(p)$, que associa cada ponto p da primitiva a uma classe do espaço. Este ponto pode ser exterior (E), de superfície (S) ou interior (I), onde os pontos de superfície estão entre os pontos interiores e os pontos exteriores.

Uma operação CSG combina as funções características dos objetos que estão sendo combinados, sendo que a classe do ponto p do objeto resultante dependerá das classes de p em cada operando. $C_{A\cup B}(p)$, é a mais forte entre as classes $C_A(p)$ e $C_B(p)$. Em particular, para a operação de união entre A e B, o interior é mais forte que a superfície e a superfície é mais forte que o exterior, ou seja, se um mesmo ponto p pertence à classe (I) de um objeto e à classe (S) do outro objeto, ele fará parte da classe (I) do objeto resultante. Se um ponto pertencer à classe (S) de um objeto e à classe (E) do outro objeto, no objeto resultante ele fará parte da classe (S).

Com a finalidade de simplificar as expressões CSG, as operações de interseção e diferença, neste algoritmo, são definidas pelas seguintes equações: $A \cap B = (A^C \cup B^C)^C$ e $A - B = (A \cap B^C)$. Mais detalhes sobre estas transformações serão discutidos no capítulo quatro.

Para a renderização cada objeto possui uma função de sombreamento $S_A(p)$, que associa a cada ponto do espaço os parâmetros de cor e transparência.

As operações CSG combinam, além das funções características dos operandos, os seus parâmetros de sombreamento. Por exemplo, para a operação de união de um ponto p qualquer, onde a classe do objeto A é diferente da classe do objeto B ($C_A(p) \neq C_B(p)$), os parâmetros de sombreamento do objeto resultante serão do objeto que possui a classe mais forte de p. Nos casos onde um ponto qualquer p pertence à mesma classe para o objeto A e o objeto B ($C_A(p) = C_B(p)$), é feita uma combinação apropriada sobre a operação de união. O mesmo ocorre para as operações de complemento.

As cenas que serão renderizadas são representadas por árvores CSG, onde a estrutura de dados possui dois tipos de nodos: Primitiva e União. O nodo Primitiva descreve a primitiva básica CSG e contém dois campos: o campo forma, que descreve a partição do espaço, em pontos interiores, exteriores e de superfície e o campo material, que especifica os parâmetros de sombreamento usados para cada classe de ponto, ou seja, para as classes de pontos exteriores (E), pontos interiores (I) e de superfície (S).

O nodo união representa a união de dois modelos CSG e contém ponteiros correspondentes para as subárvores. A operação complemento é

representada na estrutura da árvore CSG através um campo booleano, que indica se o nodo é complementado ou não.

O algoritmo de renderização *ZZ-buffer* consiste de duas fases: *tiling*[1] e renderização. Na fase de *tiling*, é feito um pré-processamento da cena para a estrutura *ZZ-Buffer*. A fase de renderização é executada durante o traçado de raios, onde são computados os sombreamentos dos objetos da cena.

O *ZZ-Buffer* é uma matriz, que representa a divisão da tela em células retangulares. Cada célula possui um bloco de *pixels* (ou um único *pixel*) da imagem final. Cada posição da matriz *ZZ-Buffer* descreve a porção da cena visível para a célula correspondente, onde a entrada do *ZZ-Buffer* é uma lista de *tiles*, que é um fragmento da cena recortado para um determinado intervalo de coordenadas Z da tela. Ver figura 3.20.

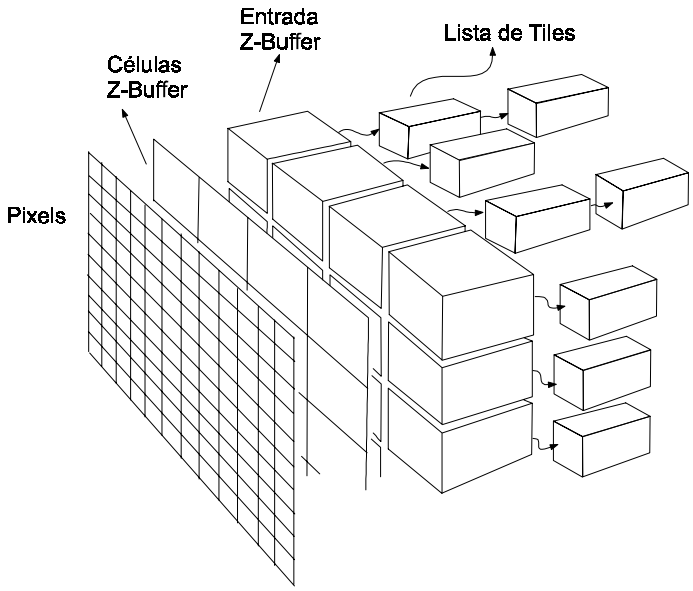


FIGURA 3.20 – *ZZ-Buffer* [SAL 90]

Os *tiles* possuem uma estrutura composta por três campos: o zz, que determina o intervalo de coordenadas z para o *tile*, a expressão CSG que descreve a parte da cena exibida pelo *tile* e o campo *flags* que armazena as características de sombreamento do *tile*.

O número de elementos de uma lista de *tiles*, bem como o intervalo da coordenada z, podem mudar de acordo com a expressão CSG descrita na cena para o *tile* correspondente.

Na fase de *tiling* computa-se o *ZZ-Buffer* para a árvore CSG que representa a cena que está sendo renderizada. Inicia-se computando uma única célula que cobre toda a tela e possui uma lista de *tiles* simples. Posteriormente faz-se um refinamento deste *ZZ-Buffer*, recursivamente dividindo cada célula em células menores, computando novas listas de *tiles*,

[1] *Tiling* aqui está associado à palavra *tile* que, em inglês, refere-se a azulejo, ladrilho. Para o algoritmo este termo, significa dividir a tela em quadros, como se fossem azulejos.

até que todas as expressões presentes na lista de *tiles*, possam ser renderizadas diretamente, ou as células obtenham um tamanho mínimo. A computação da lista de *tiles* de uma expressão é feita recursivamente a partir das listas de *tiles* das primitivas que a compõem.

A figura 3.21 mostra a lista de *tiles* para uma célula particular do *ZZ-Buffer*, na cena descrita pela expressão CSG (A∪B), onde A e B são esferas. Para cada primitiva foi criada uma lista de *tiles*, onde os intervalos de partições são determinados pelas coordenadas Z de cada primitiva. A expressão 0, apresentada nos *tiles*, significa que todos os pontos daquele *tile*, são pontos exteriores. Já os *tiles* com a expressão A ou B, representam os pontos pertencentes à respectiva primitiva.

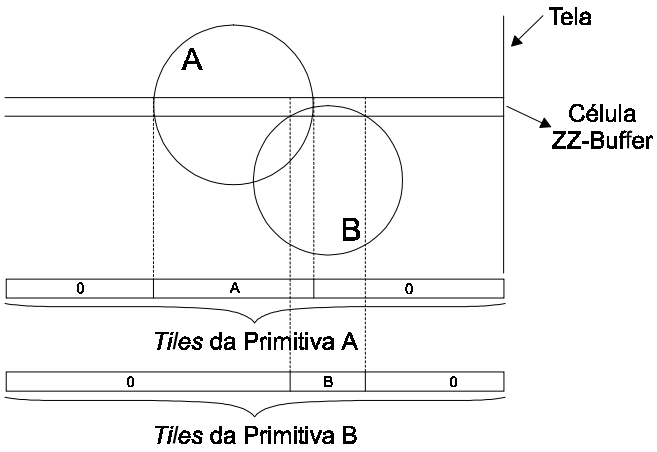


FIGURA 3.21 – Listas de *tiles*

Na união de duas listas de *tiles*, *tiles* são acrescentados ou divididos até que as duas listas tenham o mesmo número de *tiles* e os *tiles* correspondentes nas duas listas possuam o mesmo intervalo de coordenadas Z. O próximo passo é realizar a união das duas listas gerando a lista correspondente ao objeto resultante.

Na figura 3.22, pode-se observar a representação das listas de *tiles* da figura 3.21, após a divisão e a operação de união.

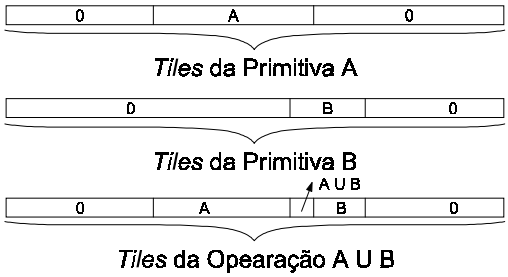


FIGURA 3.22 – Listas de *tiles* da união.

Uma vez gerado o *ZZ-Buffer* inicia-se o processo de renderização, que executa o Ray-tracing sobre os objetos armazenados no *ZZ-Buffer*. Mais precisamente cada imagem é subdividida em uma matriz S x S de *pixels*. Para

cada raio disparado localiza-se a célula do *ZZ-Buffer* que contém um ponto e examina-se o ponto correspondente na lista de *tiles*.

O *ZZ-Buffer* é usado de forma similar para a geração de sombras. Para a implementação das sombras, modifica-se o algoritmo em duas fases. Durante a fase de *tiling*, é construído um *ZZ-Buffer* adicional do ponto de vista da fonte de luz. Na fase de renderização, quando computa-se a iluminação de cada ponto visível da cena, usa-se o *ZZ-Buffer* da fonte de luz para testar se o ponto é sombreado por outro objeto.

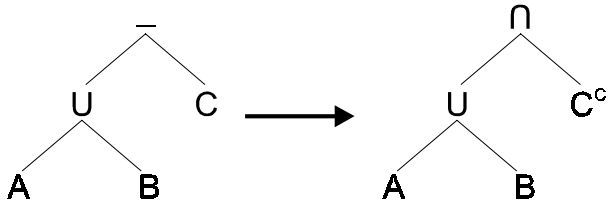
Mais precisamente, para computar a luminosidade de um ponto p para uma determinada fonte de luz, lança-se um raio da fonte em direção a p e localiza-se a célula no *ZZ-Buffer* que este raio interseciona. Extrai-se a lista de *tiles* daquela célula, e para cada *tile* computa-se as interseções, entre o raio e o objeto da expressão. As interseções são computadas pelo mesmo algoritmo usado para a câmera. Se for encontrado um objeto opaco que interseciona o raio conclui-se que o ponto esta na sombra. Caso seja encontrado um objeto semitransparente filtra-se a luz pelos coeficientes de transmissão.

3.7.7.4 Sombra Para Cenas CSG com Algoritmo Volumétrico

A geração de sombras para uma cena CSG proposta pelo autor [JAN 91] faz uso da árvore da expressão CSG convertida para a sua forma positiva. Para cada primitiva, o algoritmo calcula a árvore da sombra correspondente, originada pelo contorno das partes da primitiva. A sombra completa do objeto CSG é a união destas árvores de sombras.

Este algoritmo está baseado na geração de sombras volumétricas, portanto, as superfícies que estão dentro do volume da sombra estão na sombra. Um volume de sombra pode ser calculado primeiramente determinando os contornos do objeto, vistos da posição da fonte de luz. Um lado de contorno é um lado compartilhado por uma face frontal e uma face traseira. Uma face é frontal se o ângulo entre o vetor normal da face e o vetor diretor com a fonte de luz for maior que 90º; caso contrário será uma face traseira. Para cada lado de contorno, um polígono de sombra é gerado com os vértices do lado e a fonte de luz, este assunto já foi discutido na seção 3.7.2.

Neste algoritmo é realizada uma conversão da árvore CSG, de maneira que, cada operador de diferença seja substituindo por um operador de interseção, complementando o lado direito da árvore. Desta maneira obtêm-se uma árvore CSG com operadores de união, interseção, primitivas e primitivas complementadas, não existindo operadores de diferença. Ver figura 3.23.



(a) Árvore CSG (b) Árvore Convertida

FIGURA 3.23 – Conversão da árvore CSG [JAN 91]

A conversão de uma árvore CSG está baseada nas relações abaixo [JAN 91], que são tratadas detalhadamente no capítulo quatro.

$$A - B = A \cap B^c \quad (3.1)$$

$$(A \cup B)^c = A^c \cap B^c \quad (3.2)$$

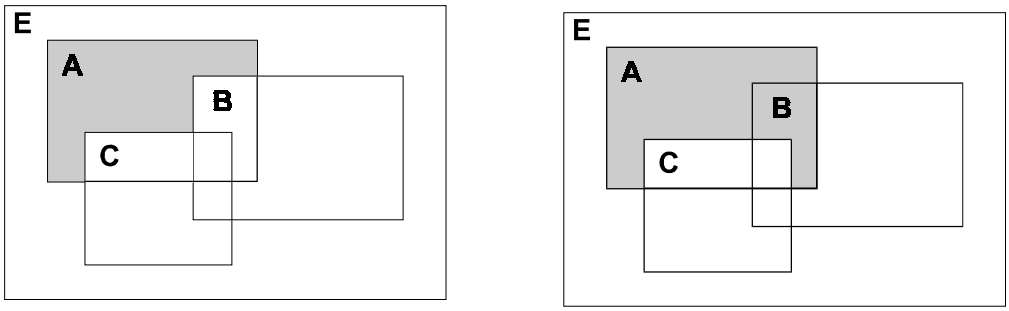
$$(A \cap B)^c = A^c \cup B^c \quad (3.3)$$

$$(A^c)^c = A \quad (3.4)$$

A chamada zona ativa de uma primitiva é a parte da primitiva que faz parte do sólido, mas que não está contida em outras primitivas. Assim, se ocorrerem mudanças sobre esta parte da primitiva, também ocorrerão mudanças no sólido.

A zona interna de uma primitiva é a parte da primitiva que está contida no sólido. A zona interna é geralmente maior do que a zona ativa, porque partes da primitiva que estão dentro de outras primitivas fazem parte desta zona.

Para a expressão CSG (A ∪ B) – C, as respectivas zonas ativa e zona interna, da primitiva A são apresentadas na figura 3.24.



(a) – Zona ativa de A. (b) – Zona interna de A.

FIGURA 3.24 – Zona ativa e zona interna da primitiva A [JAN 91]

Para exibir um sólido S modelado por CSG, mostra-se o contorno deste sólido. O contorno de S é a união das zonas ativas das primitivas. Porém, para exibir S, não é necessário encontrar os contornos das primitivas de suas zonas ativas, porque partes das primitivas que estão dentro do sólido poderão ser escondidas por partes de outras primitivas que estão no contorno do sólido. Por esta razão, é suficiente calcular os contornos das primitivas somente para as zonas Internas.

Este mesmo processo é aplicado para a geração do volume da sombra. O volume da sombra de um sólido será a união dos subvolumes de sombra, gerados pelos contornos das zonas ativas das primitivas. Porém, pode-se gerar o volume da sombra através da união dos subvolumes de sombra dos contornos das zonas internas das primitivas, porque os subvolumes de sombra estarão parcialmente sobrepostos, dando o mesmo resultado.

Assim, para cada primitiva calcula-se uma árvore-I, que representa a zona interna da primitiva.

A sombra do objeto é obtida pela união das subárvores das primitivas. Para cada primitiva, uma árvore para o volume de sombra é criada, que é idêntica a sua árvore-I, ou seja, igual a sua zona interna.

A renderização da cena é executada por um algoritmo *scan-line* para CSG. Os polígonos das sombras são processados ao mesmo tempo que os polígonos dos objetos.

3.8 Considerações Finais

Neste capítulo foram apresentadas as principais características e atributos físicos que devem ser levados em consideração para a modelagem de uma fonte de luz. Alguns controles que podem ser aplicados em fonte de luz, utilizados na simulação de efeitos especiais encontrados principalmente em *stúdios* fotográficos, foram mostrados.

Sombras, foi o outro assunto discutido neste capítulo, onde descreve-se a importância desta técnica na geração de imagens realísticas por computador, a definição de sombra, bem como as partes que a compõe: umbra e penumbra. Também fazem parte do conteúdo do capítulo os diferentes tipos de sombras, suas principais características e uma breve descrição dos principais algoritmos existentes na literatura para a geração de sombras.

A maioria dos algoritmos existentes para a renderização de cenas com objetos modelados por CSG, que apresentem sombra, estão baseados na aplicação de renderização por *Ray-Tracing*. Apesar desta técnica apresentar ótimos resultados é muito cara computacionalmente, sendo necessário encontrar outras alternativas para solucionar este problema. A alternativa encontrada será apresentada na seção 6.1.

Foram apresentados algoritmos para a geração de sombras projetadas, encoontrados na literatura, além desses existem muitos outros, entre os quais destacam-se os algoritmos para geração de sombras “suaves”.

4 Sombras na Geometria Sólida Construtiva

Como já foi referenciado no capítulo 3, de um modo geral os algoritmos de sombra exploram o fato de que a sombra é um problema de visibilidade do ponto de vista da fonte de luz. Neste capítulo apresenta-se a geração de sombras através da Geometria Sólida Construtiva, mais especificamente foi estudada a geração de sombras com o uso de operações de união e complemento dos conjuntos, que combinadas podem produzir os mesmos resultados das operações de interseção e diferença.

No capítulo dois desta dissertação foi apresentada uma forma de modelar sólidos, chamada Geometria Sólida Construtiva. Este método modela objetos através de operações booleanas regularizadas de união ($\cup^*$), interseção ($\cap^*$) e diferença ($-^*$), semelhantes às operações da teoria dos conjuntos, partindo do princípio de que um objeto complexo pode ser modelado através da combinação de outros objetos mais simples.

Neste capítulo será apresentada uma outra operação da teoria dos conjuntos que também pode ser utilizada na modelagem de objetos CSG, é a operação unária denominada complemento. Será demonstrado que combinando operações de união e complemento, pode-se chegar a resultados equivalentes aos obtidos através das operações de interseção e diferença.

No capítulo corrente, todos os conjuntos considerados são subconjuntos de um conjunto universo U.

4.1 Geração de Sombras

Dentre os vários métodos existentes para a geração de sombras, o que será usado neste protótipo é o da sombra projetada. É um método puramente geométrico da geração de sombras, que consiste em modelar a região de sombra da cena a partir da projeção do objeto do ponto de vista da fonte de luz, como pode ser visto na figura 4.1. Esse método é simples e apresenta bons resultados, no caso em que a sombra se projeta em uma única superfície plana (piso, parede, etc) [GOM 90].

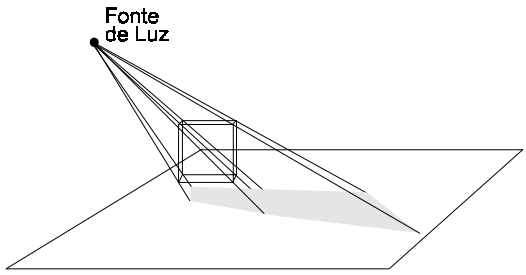


FIGURA 4.1 – Sombra projetada

A caracterização da sombra se dá pela ausência de energia luminosa em determinadas regiões da cena. O estudo da sombra é equivalente ao da

visibilidade, porém do ponto de vista da posição da fonte de luz. Os objetos, ou partes suas, que são visíveis deste ponto de vista não estão na sombra, os objetos invisíveis são os que estão em uma região de sombra da cena.

A sombra pode ser calculada através dos contornos do objeto. Se o ângulo entre o vetor normal da face e a posição da fonte de luz for maior do que 90º então a face encontra-se na sombra, caso contrário, a face será iluminada. A projeção dos vértices das faces iluminadas, sobre o plano de projeção gera um polígono, aqui chamado de polígono da sombra.

Como é conhecida a posição da fonte de luz, os vértices que formam a face e o plano de projeção, basta encontrar os pontos de interseção dos vetores, formados a partir da fonte de luz até os vértices da face, com o plano de projeção, que se obtêm as sombras projetadas de cada face.

Para calcular a interseção de uma reta com um plano, como mostra a figura 4.2, é necessário levar em consideração que os pontos P sobre o plano são denotados por $P(u,w) = A + u\vec{b} + w\vec{c}$, os pontos Q da linha por $Q(t) = D + t\vec{e}$ e o ponto de interseção por P_i. Um segmento de reta pode, ou não, interseccionar o plano. Se ocorrer a interseção, então $P(u,w) = Q(t)$, ou $A + u\vec{b} + w\vec{c} = D + t\vec{e}$. As equações para o cálculo da interseção da reta com o plano de projeção são, apresentadas abaixo [MOR 85].

$$P_i = D + t\vec{e} \tag{5.1}$$

$$t = \frac{(\vec{b}\times\vec{c})\bullet A - (\vec{b}\times\vec{c})\bullet D}{(\vec{b}\times\vec{c})\bullet C} \tag{5.2}$$

Onde:

- A é um dos pontos do plano de projeção;
- $\vec{b}$ e $\vec{c}$ são vetores orientados do plano de projeção, obtidos através da diferença dos vértices A e B ($x_B - x_A$, $u_B - y_A$, $z_B - z_A$) e entre os vértices A e C ($x_C - x_A$, $y_C - y_A$, $z_C - z_A$) respectivamente;
- D é a posição da fonte de luz;
- $\vec{e}$ o vetor diretor entre a fonte de luz (D) e os vértices da face (VF), ($x_{VF} - x_D$, $y_{VF} - y_D$, $z_{VF} - z_D$).

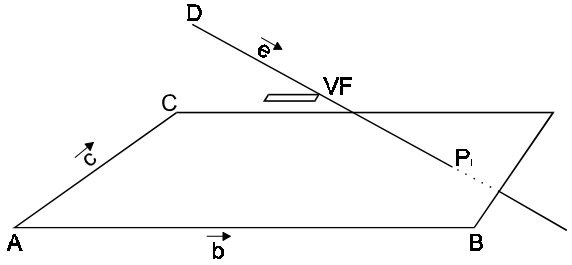


FIGURA 4.2 – Interseção de uma reta e um plano

4.2 A Relação Entre Funções e Sombras

Segundo [ALE 76] um grafo F é um grafo funcional se, para todo x, existe quando muito um elemento correspondente de x por F. Portanto, se F é um grafo funcional, não há dois pares ordenados diferentes em F com o mesmo primeiro elemento, isto é, se os pares ordenados (x, y_1) e (x, y_2) são elementos de F, então $y_1 = y_2$.

Uma relação binária f = (F,A,B) de A em B é uma função de A em B se e somente se as duas condições abaixo são verificadas:

a) O conjunto de partida A de f é igual ao seu domínio D(f), isto é:

$A = D(f)$

Segundo esta condição, para todo $x \in A$ existe $y \in B$, tal que $(x,y) \in F$:

$(\forall x)\ (\exists y)\ (x \in A,\ y \in B \text{ e } (x,y) \in F)$

b) O grafo F de f é um grafo funcional, isto é:

$(x,y_1) \in F \text{ e } (x,y_2) \in F \Rightarrow y_1 = y_2$

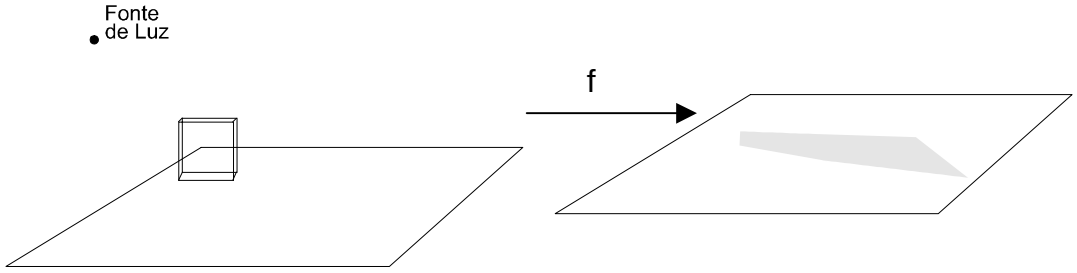
Segundo esta condição, é único o correspondente $y \in B$ de $x \in A$ por f.

As condições a e b da definição referem-se, pois, respectivamente à existência e à unicidade do elemento $y \in B$ cada vez que é dado $x \in A$.

Uma função f de A em B diz-se também função f definida em A e com valores em B ou aplicação f de A em B.

Ao ser feita uma análise de objetos modelados por Geometria Sólida Construtiva e suas respectivas sombras, pode-se relacionar sombras com funções. De forma semelhante às funções, um objeto pode representar o domínio de uma função e a sombra deste objeto, a imagem da função.

Aplicando-se então os conceitos de função, pode-se dizer que o conjunto A é composto por todos os pontos do objeto, representando o domínio da função, e a imagem da função são os pontos do objeto projetados sobre o plano de projeção, ou seja, a sombra do objeto. Ver figura 4.3.



(a) Domínio = Objeto (b) Imagem = Sombra

FIGURA 4.3 – Objeto e sombra

4.3 Operações com Conjuntos

Em [ABE 91, ALE 76 e LIP 98] estão a base teórica e as demonstrações necessárias para o desenvolvimento deste trabalho.

Durante a realização deste estudo, percebeu-se a existência de diferentes maneiras de modelar sólidos na Geometria Sólida Construtiva, através das operações booleanas. Na tentativa de encontrar um maneira de aplicar as operações booleanas na geração de sombras em cenas que possuem objetos CSG, encontrou-se uma forma de representar as operações de interseção e diferença através das operações de união e complemento. Para isso, é necessário realizar algumas conversões, que apresentam resultados equivalentes, como será descrito nesse capítulo.

As conversões estão baseadas nas propriedades das operações dos conjuntos. Abaixo são apresentadas estas propriedades, sendo que as suas demonstrações podem ser encontradas em [ABE 91 e ALE 76].

4.3.1 Interseção

Chama-se interseção de dois conjuntos A e B ao conjunto de todos os elementos que pertencem simultaneamente a A e a B.

Esse conjunto indica-se pela notação: $A \cap B$, que se lê: “A interseção B”. Simbolicamente tem-se:

$$A \cap B = \{x \mid x \in A \text{ e } x \in B\}$$

Isto é:

$$x \in A \cap B \Leftrightarrow x \in A \text{ e } x \in B$$

Na figura 4.4, apresenta-se o diagrama de Venn, para a operação de interseção entre os conjuntos A e B.

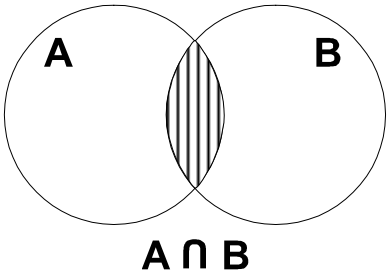


FIGURA 4.4 – Diagrama de Venn da interseção

A tabela 4.1 apresenta as propriedades da interseção. Algumas destas propriedades serão utilizadas no processo de reordenação das operações em combinações com união e complemento, descritas na seção 4.7, deste capítulo.

TABELA 4.1 - Propriedades da interseção.

Propriedades da interseção	
1	$A \cap B = B \cap A$
2	$(A \cap B) \cap C = A \cap (B \cap C)$
3	$A \cap A = A$
4	$A \cap \varnothing = \varnothing$
5	$A \cap U = A$

Dois conjuntos A e B se dizem disjuntos, se e somente se, não possuem elementos em comum. Portanto, A e B são disjuntos se e somente se, a interseção entre eles é o conjunto vazio:

$$A \cap B = \varnothing$$

4.3.2 União

Chama-se união de dois conjuntos A e B ao conjunto de todos os elementos que pertencem a A ou a B.

Esse conjunto indica-se pela notação: $A \cup B$, que se lê: “A união B”. Simbolicamente tem-se:

$$A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$$

Isto é:

$$x \in A \cup B \Leftrightarrow \begin{cases} \text{seja } x \in A \text{ e } x \notin B \\ \text{seja } x \notin A \text{ e } x \in B \\ \text{seja } x \in A \text{ e } x \in B \end{cases}$$

Na figura 4.5, apresenta-se o diagrama de Venn, para a operação de união entre os conjuntos A e B.

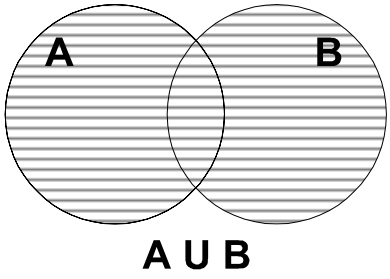


FIGURA 4.5 – Diagrama de Venn da união

Na tabela 4.2, estão as propriedades da união. No processo de reordenação das operações em combinações com união e complemento, descritas na seção 4.7, essas propriedades serão utilizadas.

TABELA 4.2 - Propriedades da união

	Propriedades da união
1	$A \cup B = B \cup A$
2	$(A \cup B) \cup C = A \cup (B \cup C)$
3	$A \cup A = A$
4	$A \cup \varnothing = A$
5	$A \cup U = U$

Se os conjuntos A e B não forem disjuntos, os seus elementos comuns figuram uma só vez em $A \cup B$.

4.3.3 Diferença

Chama-se diferença entre os conjuntos A e B ao conjunto de todos os elementos de A que não pertencem a B.

Esse conjunto indica-se pela notação: A – B, que se lê: “A menos B”. Simbolicamente tem-se:

$$A - B = \{x \mid x \in A \text{ e } x \notin B\}$$

Isto é:

$$x \in A - B \Leftrightarrow x \in A \text{ e } x \notin B$$

Observa-se que a diferença entre A e B é um subconjunto de A.

Na figura 4.6, apresenta-se o diagrama de Venn, para a operação de diferença (A – B).

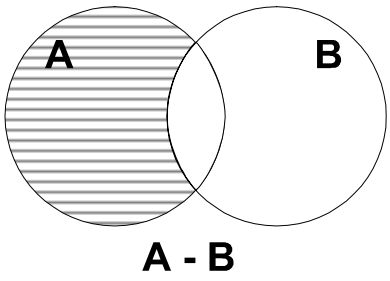


FIGURA 4.6 – Diagrama de Venn da diferença

Na tabela 4.3, estão as propriedades da diferença. Assim como algumas propriedades da união e interseção serão utilizadas no processo de reordenação das operações em combinações com união e complemento, essas também serão.

TABELA 4.3 – Propriedades da diferença

	Propriedades da diferença
1	$A - B = A \cap B^c$
2	$A - \varnothing = A$
3	$\varnothing - A = \varnothing$
4	$A - U = \varnothing$
5	$U - A = A^c$
6	$A - A = \varnothing$
7	$A - A^c = A$
8	$(A - B)^c = A^c \cup B$
9	$A - B = B^c - A^c$
10	$(A - B) - C = A - (B \cup C)$
11	$A - (B - C) = (A - B) \cup (A \cap C)$
12	$A \cup (B - C) = (A \cup B) - (C - A)$
13	$A \cap (B - C) = (A \cap B) - (A \cap C)$
14	$A - (B \cup C) = (A - B) \cup (A - C)$
15	$A - (B \cap C) = (A - B) \cup (A - C)$
16	$(A \cup B) - C = (A - C) \cup (B - C)$
17	$(A \cap B) - C = (A - C) \cap (B - C)$
18	$A - (A - B) = A \cap B$
19	$(A - B) - B = A - B$

Se A e B são disjuntos, então a diferença entre A e B será o próprio conjunto A.

4.3.4 Complemento

O complemento de um conjunto A, em um universo E é o conjunto de todos os elementos do universo que não fazem parte de A.

Esse conjunto indica-se pela notação: A^c, que se lê: “Complemento de A”. Simbolicamente tem-se:

$$A^c = \{x \mid x \in E \text{ e } x \notin A\}$$

Isto é:

$$x \in A^c \Leftrightarrow x \in E \text{ e } x \notin A$$

O conjunto E, em relação ao qual se determina o complementar, chama-se conjunto de referência ou conjunto referencial. Ver figura 4.7.

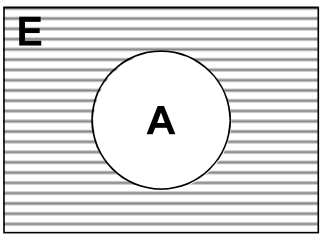


FIGURA 4.7 – Diagrama de Venn do complemento

A tabela 4.4 apresenta as propriedades do complemento, das quais a propriedade 1, será utilizada no processo de reordenação das operações.

TABELA 4.4 – Propriedades do complemento

Propriedades do Complemento	
1	$(A^c)^c = A$
2	$U^c = \varnothing$
3	$\varnothing^c = U$

4.3.5 – Propriedades Interrelacionando Complemento, Interseção e União

Os conjuntos nas operações de união, interseção e complemento, satisfazem várias propriedades, dentre estas estão as leis de De Morgan (propriedades 8 e 9 da tabela 4.5). A tabela 4.5 relaciona estas propriedades e as suas demonstrações são encontradas na literatura [ABE 91, ALE 76 e LIP 98].

A tabela 4.5 apresenta as propriedades de Interrelacionamento de Complemento, Interseção e União. Essas possibilitarão a utilização de união e complemento como forma de reordenar as demais operações booleanas, em especial as propriedades 8 e 9.

TABELA 4.5 – Propriedades interrelacionando complemento, interseção e união

Propriedades interrelacionando complemento, interseção e união	
1	$A \cup (A \cap B) = A$
2	$A \cap (A \cup B) = A$
3	$A \cup A^c = U$
4	$A \cap A^c = \varnothing$
5	Se $A = B^c$, então $B = A^c$
6	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
7	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
8	$(A \cup B)^c = A^c \cap B^c$
9	$(A \cap B)^c = A^c \cup B^c$

Objetos geométricos são definidos como conjuntos de pontos, compostos de dois subconjuntos: interior e fronteira. As operações booleanas de interseção, união e diferença sobre conjuntos são usadas para combinar objetos simples formando outros mais complexos. Os algoritmos que executam essas operações devem produzir objetos que também sejam conjuntos de pontos, tendo como subconjunto o interior e a fronteira, preservando a dimensão dos objetos iniciais [MOR 85].

Seguindo estes conceitos aplicam-se as operações booleanas entre dois conjuntos, onde o conjunto A, formado pelos subconjuntos iA (pontos interiores) e bA (pontos do contorno), representa um objeto e o conjunto B, formado pelos subconjuntos iB (pontos interiores) e bB (pontos do contorno), o outro objeto. Ou seja:

$$A = (iA \cup bA) \quad (4.3)$$

$$(iA \cap bA) = \varnothing \quad (4.4)$$

$$B = (iB \cup bB) \quad (4.5)$$

$$(iB \cap bB) = \varnothing \quad (4.6)$$

4.4 União na Geometria Sólida Construtiva

Serão pontos válidos para o objeto resultante da operação de união, todos os pontos que pertencerem ao conjunto A ou pertencerem ao conjunto B, assim como A e B são definidos pelos subconjuntos de pontos interiores e pontos de contorno. A representação para o resultado R da operação de união entre os conjuntos é:

$R = A \cup B$
Usando as equações (4.3) e (4.5), temos:
$R = A \cup B = (iA \cup bA) \cup (iB \cup bB)$
$= \{(iA \cup bA) \cup (iB \cup bB)\}$

Neste caso, pode-se verificar que qualquer elemento, que pertencer a um dos dois conjuntos, será elemento do conjunto final.

4.5 Interseção na Geometria Sólida Construtiva

Na operação de interseção serão válidos os pontos que pertencerem ao conjunto A e ao conjunto B. Seguindo a definição de que A e B são definidos por subconjuntos de pontos interiores e pontos de fronteira, a representação para o resultado R da interseção entre os dois conjuntos é:

$R = A \cap B$
Usando as equações (4.3) e (4.5), temos:
$R = A \cap B = (iA \cup bA) \cap (iB \cup bB)$
$= \{(iA \cup bA) \cap (iB \cup bB)\}$

Assim, pode-se verificar que um ponto pertence a interseção dos conjuntos, se o mesmo pertencer aos conjuntos A e B.

4.6 Diferença na Geometria Sólida Construtiva

Na operação da diferença somente serão válidos os pontos que pertencerem ao conjunto A e não pertencerem ao conjunto B, considerando que A e B são definidos pelos subconjuntos de pontos interiores e pontos de fronteira. A representação para o resultado (R) da operação da diferença entre os dois conjuntos é:

$R = A - B$
Usando as equações (4.3) e (4.5), temos:
$R = A - B = (iA \cup bA) - (iB \cup bB)$
$= \{(iA \cup bA) - (iB \cup bB)\}$

Deste modo, um ponto pertence ao conjunto da operação A – B, se o mesmo pertencer ao conjunto A e não pertencer ao conjunto B.

4.7 Interseção e Diferença como Combinação de União e Complemento

Por razões que serão mostradas formalmente, as operações de interseção e diferença, serão substituídas por formas equivalentes com a combinação de união e complemento.

4.7.1 Interseção

Neste caso, realiza-se uma conversão da operação de interseção para uma expressão que represente os mesmos pontos da interseção, porém com o uso de operações de união e complemento, $R = A \cap B$.

Pode-se afirmar que:

$R = A \cap B = ((A \cap B)^c)^c$ (propriedade 1 da tabela 4.4)

Segundo as Leis de De Morgan (propriedade 8 e 9, da tabela 4.5):

$((A \cap B)^c)^c = (A^c \cup B^c)^c = ((A^c)^c \cap (B^c)^c)$

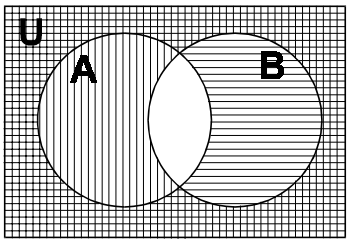
Segundo a propriedade do complemento (propriedade 1, da tabela 4.4):

$(A^c)^c = A$ e $(B^c)^c = B$, então

$((A^c)^c \cap (B^c)^c) = (A \cap B)$, logo

$R = (A^c \cup B^c)^c = A \cap B$

A figura 4.8 representa a união dos complementos, ou seja, o complemento de A unido com o complemento de B.



$(A^C \cup B^C)$

FIGURA 4.8 – Diagrama de Venn da união dos complementos de A e B

Na figura 4.9 pode-se ver a aplicação do complemento sobre a união dos complementos de A e B $((A^c \cup B^c)^c)$, esta operação resulta em um conjunto equivalente a interseção dos conjuntos A e B.

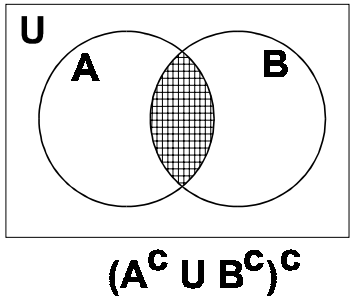


FIGURA 4.9 – Diagrama de venn, Interseção $(A^c \cup B^c)^c$

Tanto nas demonstrações acima, como nos diagramas de Venn (figuras 4.4 e 4.9), é possível verificar que $R = (A^c \cup B^c)^c = A \cap B$, pois os resultados são equivalentes nas duas situações.

4.7.2 Diferença

Aplicando sobre os conjuntos A e B as propriedades das operações dos conjuntos, foi possível converter uma operação de diferença em operações que envolvam somente união e complemento, onde o resultado apresentado seja equivalente a operação, R = A – B.

Pode-se afirmar que:

$$\mathbf{R = A - B = (A^c \cup B)^c}$$

Segundo as Leis de De Morgan (propriedade 8, da tabela 4.5):

$$(A^c \cup B)^c = ((A^c)^c \cap B^c)$$

Segundo a propriedade do complemento (propriedade 1, da tabela 4.4):

$$((A^c)^c \cup B^c) = A \cap B^c$$

$A \cap B^c = \{x \mid x \in A \text{ e } x \notin B\}$ e

$A - B = \{x \mid x \in A \text{ e } x \notin B\}$, então

$$\mathbf{R = A - B = (A^c \cup B)^c}$$

No diagrama de Venn da figura 4.10, pode-se ver a união do complemento de A com o conjunto B.

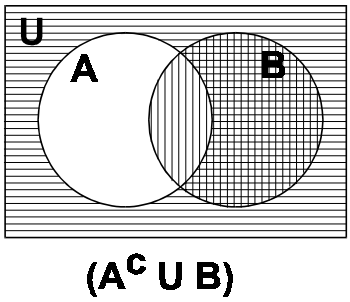
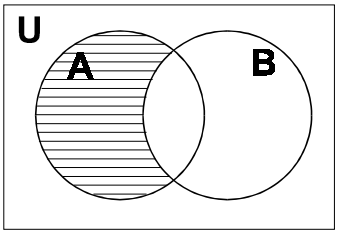


FIGURA 4.10 – Diagrama de Venn, união do complemento de A com B

A figura 4.11 representa a aplicação do complemento sobre a operação de união entre o complemento de A e o conjunto B ($A^c \cup B$). Como pode ser visto o resultado apresentado é equivalente a operação A – B.



$(A^C \cup B)^C$

FIGURA 4.11 – Diagrama de Venn, diferença $(A^c \cup B)^c$.

Assim verifica-se, pelas demonstrações e pelos diagramas de Veen (figuras 4.6 e 4.11), que $R = (A^c \cup B)^c = A - B$, pois os resultados das duas expressões são equivalentes.

4.8 Exemplos

Nesta seção, serão apresentados alguns exemplos da aplicação das transformações demonstradas acima. Salienta-se que o objetivo da utilização destes exemplos é o de enriquecer o assunto discutido e mostrar a sua funcionalidade.

A representação da modelagem de objetos na Geometria Sólida Construtiva se dá através de árvores binárias, como já foi comentado no capítulo dois, seção 2.4. As conversões realizadas sobre as operações de interseção e diferença, para uma combinação de união e complemento, discutidas na seção 4.7, também podem ser realizadas sobre as árvores CSG. Nesta seção serão apresentadas as árvores convertidas em função de união e complemento.

Na seção 2.4 foi comentado que os nodos não terminais de uma árvore CSG, representam as operações boolenas regularizadas (união, interseção e/ou diferença) ou então as transformações geométricas (mudança de escala, rotação e/ou translação), que operam sobre os seus dois subnodos. Nesta seção serão apresentadas as árvores após as conversões para operadores de união e complemento. É importante esclarecer que nos casos em que aparecem os complementos, primeiramente são realizadas as operações de união e posteriormente sobre estes resultados aplica-se o complemento.

Todas as operações booleanas dos exemplos abaixo serão aplicadas sobre as primitivas que estão na figura 4.12.

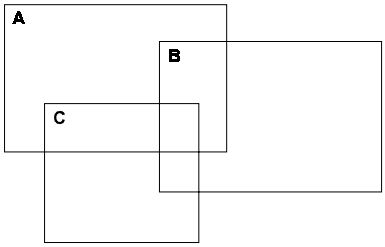


FIGURA 4.12 – Primitivas dos exemplos

Exemplo 1 (A ∪ B) – C

Seja R = (A ∪ B) – C.

Neste caso tem-se uma operação de diferença, assim esta expressão pode ser convertida com o uso das propriedades demonstradas na seção 4.7.2. Portanto:

$R = ((A \cup B)^c \cup C)^c$.

Na figura 4.13 são apresentadas as árvores CSG do exemplo 1. No caso (a) mostra-se a árvore com uma operação de união entre os objetos A e B, deste resultado subtrai-se o objeto C. No caso (b) a árvore apresenta um resultado equivalente ao caso (a), porém utilizam-se somente operações de união e complemento. Para isso, realizou-se uma operação de união entre os objetos A e B, a este resultado aplicou-se o complemento, que foi unido ao objeto C, posteriormente aplicou-se o complemento na raiz da árvore, que é o resultado final.

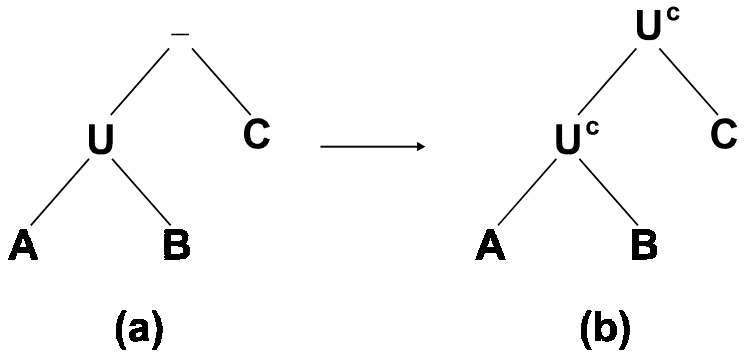


FIGURA 4.13 – Conversão da árvore CSG do exemplo 1.

No diagrama de Venn, da figura 4.14, pode-se ver o complemento da operação de união entre os objetos A e B, unido ao objeto C.

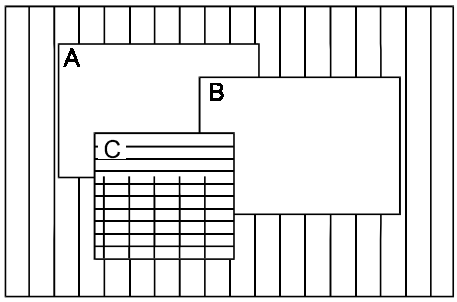


FIGURA 4.14 – Diagrama de Venn, $(A \cup B)^c \cup C$

Na figura 4.15, apresenta-se o diagrama de Venn da aplicação do complemento sobre a operação $(A \cup B)^c \cup C$, que é o resultado pretendido, ou seja, $(A \cup B) – C$.

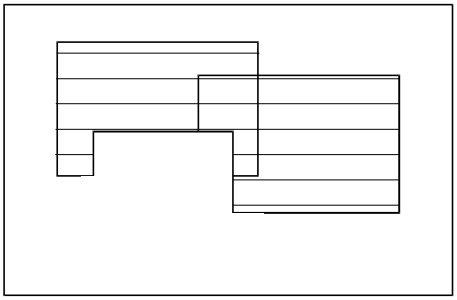


FIGURA 4.15 – Diagrama de Venn, $((A \cup B)^c \cup C)^c$

Exemplo 2 $(A \cup B) \cap C$

Seja $R = (A \cup B) \cap C$.

Neste exemplo encontra-se uma operação de interseção. Para converter a expressão usam-se as propriedades demonstradas na seção 4.7.1, portanto:

$R = ((A \cup B)^c \cup C^c)^c$.

Na figura 4.16 são apresentadas as árvores CSG do exemplo 2. No caso (a) mostra-se a árvore com uma operação de união entre os objetos A e B, com este resultado aplica-se uma interseção com o objeto C. No caso (b) a árvore apresenta um resultado equivalente ao caso (a), porém utilizam-se somente operações de união e complemento, onde realizou-se uma operação de união entre os objetos A e B, a este resultado aplicou-se o complemento, que foi unido ao complemento do objeto C. Posteriormente aplicou-se o complemento na raiz da árvore, que é o resultado final.

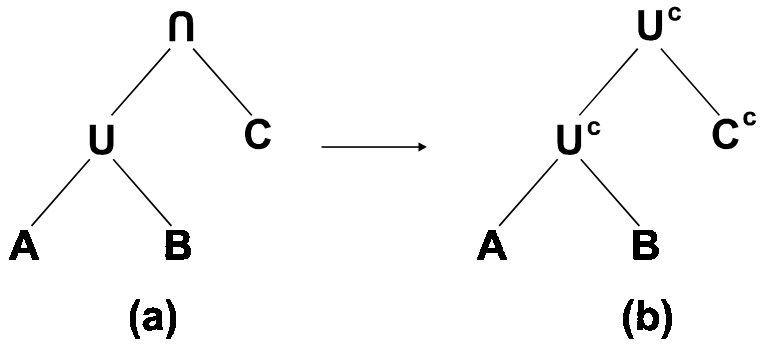


FIGURA 4.16 – Conversão da árvore CSG do exemplo 2

No diagrama de Venn, da figura 4.17, pode-se ver o complemento da operação de união entre os objetos A e B, unido ao complemento do objeto C.

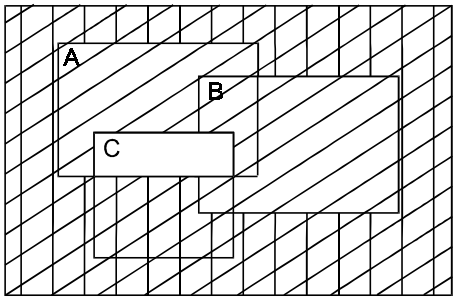


FIGURA 4.17 – Diagrama de Venn, $(A \cup B)^c \cup C^c$

Na figura 4.18, apresenta-se o diagrama de Venn da aplicação do complemento sobre a operação $(A \cup B)^c \cup C^c$, que é o resultado pretendido, ou seja, $(A \cup B) \cap C$.

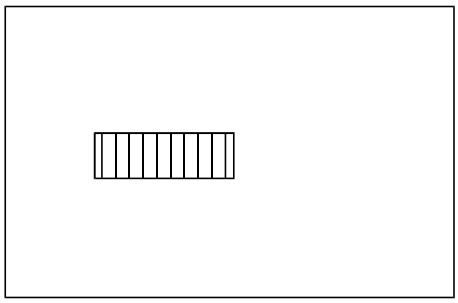


FIGURA 4.18 – Diagrama de Venn, $((A \cup B)^c \cup C^c)^c$

Exemplo 3 (A – B) – C

Seja R = (A – B) – C.

Nesta expressão encontram-se duas operações de diferença. Para converter a expressão usam-se as propriedades demonstradas na seção 4.7.2 e também a propriedade 1 da tabela 4.4, assim:

$R = (A^c \cup B \cup C)^c$.

Na figura 4.19 são apresentadas as árvores CSG do exemplo 3. No caso (a) tem-se a árvore com uma operação de diferença entre os objetos A e B, deste resultado subtrai-se o objeto C. No caso (b), a árvore apresenta um resultado equivalente ao caso (a), porém utilizam-se somente operações de união e complemento, onde, realizou-se uma operação de união entre o complemento do objeto A e o objeto B, a este resultado uniu-se o objeto C. Posteriormente aplicou-se o complemento na raiz da árvore, que é o resultado final.

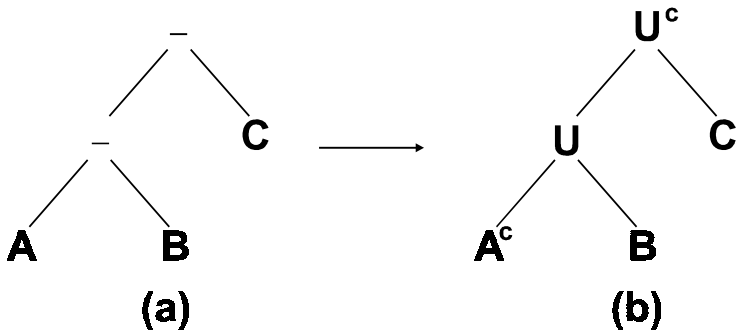


FIGURA 4.19 – Conversão da árvore CSG do exemplo 3

No diagrama de Venn, da figura 4.20, pode-se ver o complemento do objeto A unido ao objeto B, que estão unidos ao objeto C.

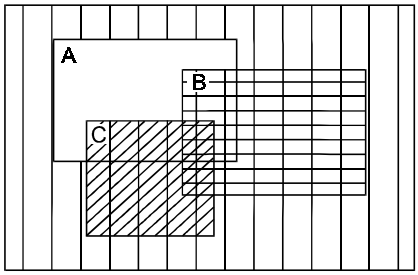


FIGURA 4.20 – Diagrama de Venn, $(A^c \cup B) \cup C$

Na figura 4.21, apresenta-se o diagrama de Venn da aplicação do complemento sobre a operação $(A^c \cup B) \cup C$, que é o resultado pretendido, ou seja, (A – B) – C.

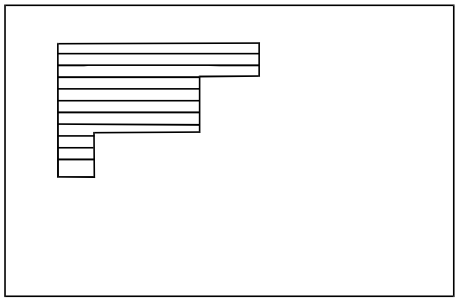


FIGURA 4.21 – Diagrama de Venn, $((A^c \cup B) \cup C)^c$

Exemplo 4 $(A \cap B) \cap C$

Seja $R = (A \cap B) \cap C$.

Neste exemplo existem duas operações de interseção. A conversão da expressão está baseada nas demonstrações da seção 4.7.1 e também no uso da propriedade 1 da tabela 4.4, assim:

$R = (A^c \cup B^c \cup C^c)^c$.

Na figura 4.22 são apresentadas as árvores CSG do exemplo 4. No caso (a) mostra-se a árvore com uma operação de interseção entre os objetos A e B, este resultado é interseccionado com o objeto C. No caso (b), a árvore apresenta um resultado equivalente ao caso (a), porém utilizam-se somente operações de união e complemento: realizou-se uma operação de união entre os complementos dos objetos A e B, a este resultado uniu-se o complemento do objeto C, posteriormente aplicou-se o complemento na raiz da árvore, que é o resultado final.

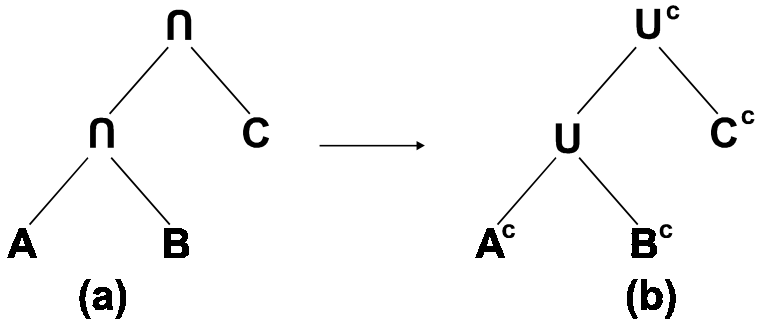


FIGURA 4.22 – Conversão da árvore CSG do exemplo 4

No diagrama de Venn, da figura 4.23, pode-se ver o complemento do objeto A unido ao complemento do objeto B. Este resultado foi unido ao complemento do objeto C.

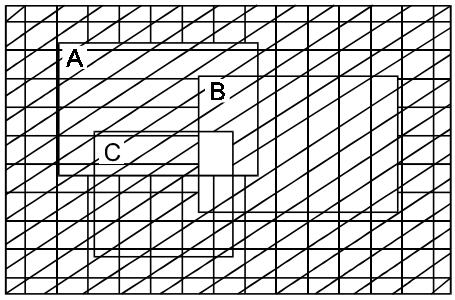


FIGURA 4.23 – Diagrama de Venn, $(A^c \cup B^c) \cup C^c$

Na figura 4.24, apresenta-se o diagrama de Venn da aplicação do complemento sobre a operação $(A^c \cup B^c) \cup C^c$, que é o resultado pretendido, ou seja, $(A \cap B) \cap C$.

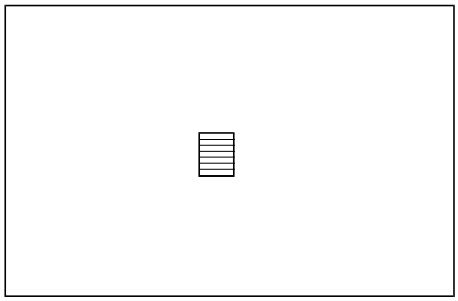


FIGURA 4.24 – Diagrama de Venn, $((A^c \cup B^c) \cup C^c)^c$

4.9 Operações de União, Interseção e Diferença nas Sombras

A aplicação das operações booleanas de união, interseção e diferença, sobre as sombras projetadas, nem sempre apresenta o resultado correto. Como já foi discutido no capítulo três, a sombra é uma região escura de uma cena iluminada. Somente existirá sombra na cena se houver, no mínimo, uma fonte de luz e um objeto, que impeça a passagem, total ou parcial da luz.

O posicionamento dos objetos em relação a fonte de luz é um fator importante na análise da sombra, pois devido a isso, podem existir situações em que as sombras possuam pontos em comum, devido a sobreposição das sombras projetadas. Nestas situações a aplicação das operações booleanas sobre as sombras pode gerar um resultado que não corresponde a sombra dos objetos modelados por CSG.

Logo abaixo, são apresentadas as três situações que foram consideradas para este estudo. A demonstração gráfica dos resultados das operações será através dos diagramas de Venn.

Os casos estão baseados nos objetos da figura 4.25. Sobre esses serão aplicadas operações de união, interseção e diferença, sendo também feita uma análise de cada situação em particular.

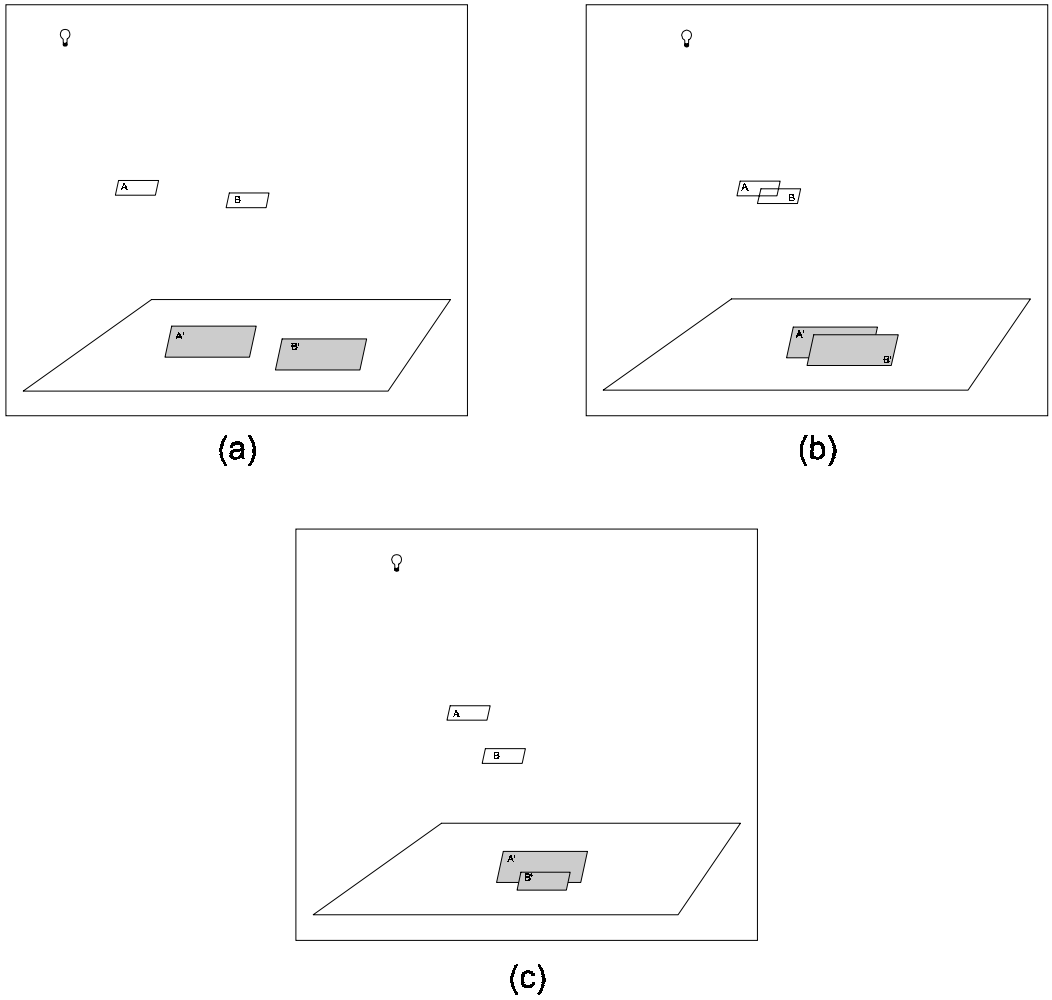


FIGURA 4.25 – Casos de sombras projetadas

Como pode ser visto na figura 4.25, para cada objeto é feita a projeção das faces iluminadas. Esta operação gera o polígono da sombra de cada objeto, sobre estes polígonos serão aplicadas as operações para a geração da sombra final da cena.

No exemplo da figura 4.25, pode-se verificar que existem duas situações para as sombras. Devido ao posicionamento dos objetos, em relação a fonte de luz, elas podem estar sobrepostas, o que caracteriza a existência de pontos em comum (casos (b) e (c) da figura 4.25) ou então, disjuntas (caso (a) da figura 4.25). Nas operações abaixo serão tratadas as operações booleanas nestas diferentes situações.

4.9.1 Operação de União

A figura 4.26 está apresentando a sombra projetada da união para cada um dos casos da figura 4.25. Isto é, primeiramente aplicou-se uma operação de união sobre duas primitivas (A e B), que consequentemente, originou um novo

objeto, após esta operação de união foram projetadas as faces iluminadas do objeto resultante, dando origem a sombra da união, S(A ∪ B).

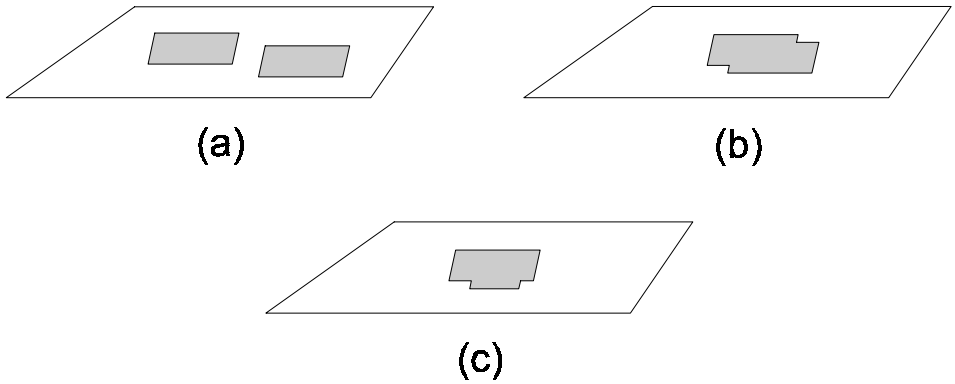


FIGURA 4.26 – Sombra projetada da união (S(A ∪ B))

Na figura 4.27, a situação é outra. Primeiro foram projetadas as faces iluminadas de cada objeto, dando origem aos polígonos da sombra, S(A) e S(B), posteriormente aplicou-se sobre estes polígonos uma operação de união, S(A) ∪ S(B).

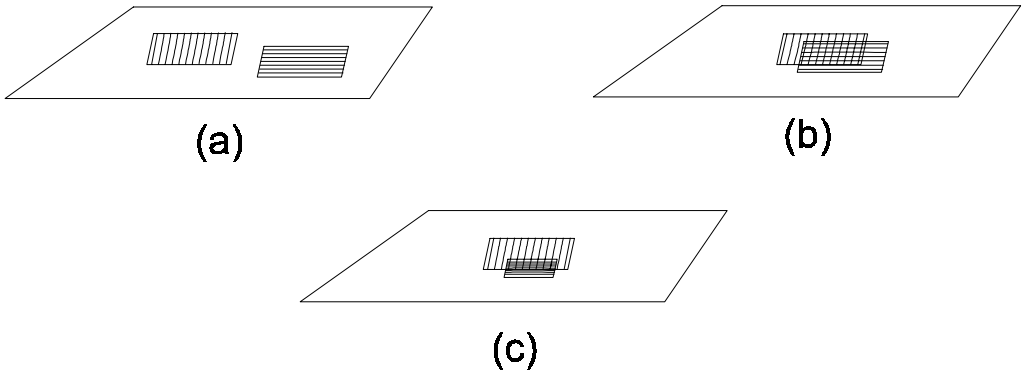


FIGURA 4.27 – União das sombras projetadas (S(A) ∪ S(B))

Ao ser feita a análise dos diagramas de Venn, nas figuras 4.26 e 4.27, pode-se verificar que os resultados são os mesmos. Isto significa que a sombra projetada da união (S(A∪B)) é equivalente a união das sombras projetadas (S(A)∪S(B)), pois a união de dois conjuntos contém todos os elementos que pertencem a A ou a B, como já foi discutido na seção 4.5.

4.9.2 Operação de Interseção

A figura 4.28 está apresentando a sombra projetada da interseção, para cada um dos casos da figura 4.25. Um novo objeto foi gerado a partir da interseção entre as primitivas A e B. O passo seguinte foi projetar as faces iluminadas do novo objeto, S(A ∩ B), dando origem a sombra da interseção.

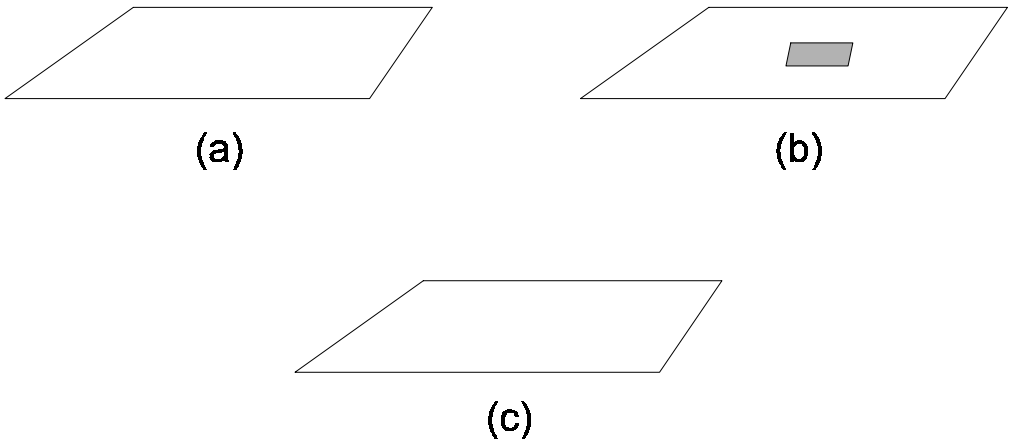


FIGURA 4.28 – Sombra projetada da interseção (S(A ∩ B))

As sombras apresentadas na figura 4.29 foram obtidas de outra forma. Primeiramente projetaram-se as faces iluminadas de cada objeto, S(A) e S(B), que originaram aos polígonos da sombra, sobre estes polígonos aplicou-se a operação de interseção, S(A) ∩ S(B).

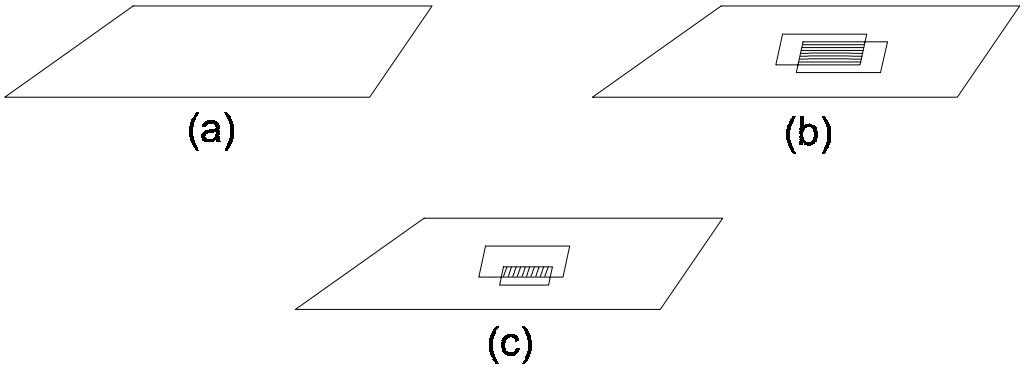


FIGURA 4.29 – Interseção das sombras projetadas (S(A) ∩ S(B))

Como já foi discutido na seção 4.3.1, a interseção entre dois conjuntos são todos os pontos que pertencem simultaneamente a A e a B. Nos casos (a) e (c) da figura 4.25, a interseção entre os objetos é igual ao conjunto vazio, pois os objetos são disjuntos.

Ao ser feita a análise da figura 4.28, pode-se verificar que nos casos (a) e (c) não temos sombra e o caso (b) apresenta a sombra da interseção. Todas as sombras apresentadas nesta figura estão corretas, uma vez que nos casos (a) e (c) têm-se o conjunto vazio como resultado da interseção entre os objetos A e B (ver figura 4.25), consequentemente a sombra de um conjunto vazio não existe. No caso (b) a sombra também está correta, pois corresponde a sombra do objeto gerado pela interseção entre A e B, da figura 4.25.

Na figura 4.29 (a), os polígonos da sombra dos objetos (S(A) e S(B)), também são disjuntos, eqüivalendo ao resultado esperado (não há sombra). No caso 4.29 (c) também não deveria haver sombra.

Devido ao posicionamento dos objetos A e B da figura 4.25 (c) em relação a fonte de luz, ocorre uma sobreposição dos polígonos da sombra

destes dois objetos, como mostra a figura 4.29 (c). Nesta situação temos um problema na interseção entre a sombra do objeto A (S(A)) e a sombra do objeto B (S(B)). Como não existe interseção entre objetos disjuntos, caso (c) da figura 4.25, não pode existir sombra, mas como se pode ver a figura 4.29 (c) apresenta sombra.

Assim, pode-se dizer que a sombra projetada da interseção nem sempre é igual a interseção das sombras.

4.9.3 Operação de Diferença

A figura 4.30 está apresentando a sombra projetada da diferença, para os casos da figura 4.25. Um novo objeto foi gerado a partir da diferença entre as primitivas A e B da figura 4.25, projetando as faces iluminadas do novo objeto, originou-se a sombra da diferença, S(A – B).

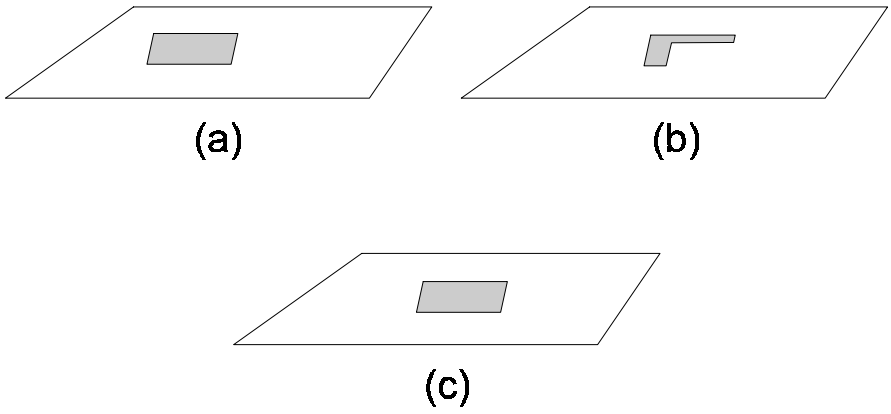


FIGURA 4.30 – Sombra projetada da diferença (S(A – B))

Na figura 4.31 são exibidas as sombras correspondentes à operação de diferença entre as sombras projetadas, S(A) – S(B). O processo de obtenção para os resultados apresentados foi a partir da projeção das faces iluminadas dos objetos A e B da figura 4.25.

Na figura 4.31, primeiramente projetaram-se as faces iluminadas de cada objeto, originando as sombras correspondentes a cada um dos objetos S(A) e S(B), sobre estes polígonos aplicou-se a operação da diferença, S(A) – S(B).

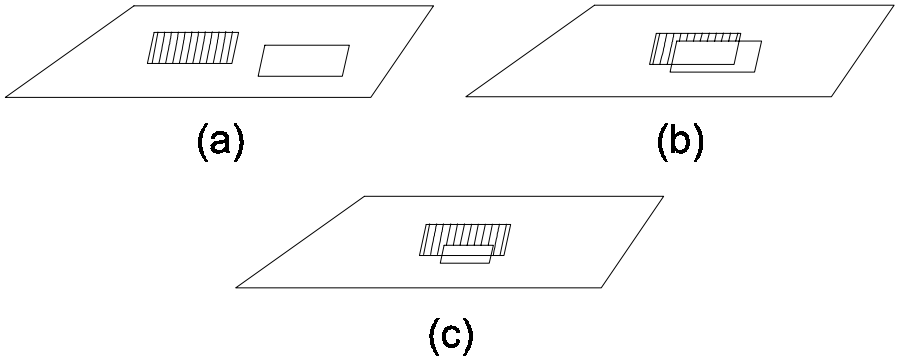


FIGURA 4.31 – Diferença das sombras projetadas (S(A) – S(B))

Na figura 4.30, pode-se verificar que nos casos (a) e (c) as sombras apresentadas correspondem a sombra do objeto A. Como as sombras estão baseadas na disposição dos objetos da figura 4.25, pode-se facilmente perceber que esta sombra é correta, pois segundo a teoria apresentada na seção 4.3.1, a diferença entre os conjuntos A e B é composta por todos os elementos que pertencem ao conjunto A e não pertencem a B e quando A e B são disjuntos A – B = A. Os objeto A e B, nos casos (a) e (c), são disjuntos, portanto S(A – B) = S(A).

Devido ao posicionamento dos objetos A e B, nos casos (b) e (c) da figura 4.25, em relação a fonte de luz, têm-se uma sobreposição dos polígonos da sombra, como mostra a figura 4.31 (b) e (c). A sombra do caso (b) é correta, pois é composta por todos os pontos que pertencem a sombra de A e não pertencem a sombra de B, que tem o resultado equivalente ao da figura 4.30 (b).

Comparando-se as figuras 4.30(c) e 4.31(c), pode-se perceber a não equivalência entre a sombra projetada da diferença S(A – B) e a diferença das sombras projetadas S(A) – S(B). Os objetos são disjuntos e a operação A – B, corresponde neste caso, ao objeto A, portanto a sombra correta deve ser a sombra do próprio objeto A.

Com isso, verifica-se que a sombra projetada da diferença nem sempre é igual a diferença das sombras.

4.9.4 Complemento

A figura 4.32 apresenta a sombra projetada do complemento de A, $S(A^c)$. Essa é a sombra de todos os elementos do universo, ao qual A pertence, que não pertencem ao conjunto A.

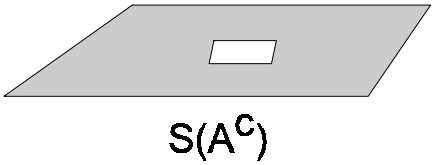


FIGURA 4.32 – Sombra do complemento

Na figura 4.33, é apresentado o complemento da sombra projetada de A, $(S(A))^C$. Projetou-se a face iluminada do objeto A e posteriormente sobre esta projeção aplicou-se o complemento, que são todos os pontos pertencentes ao universo (plano de projeção), e que não pertencem a sombra projetada de A.

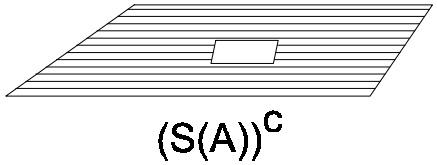


FIGURA 4.33 – Complemento da sombra

Ao serem analisadas as figura 4.32 e 4.33, pode-se constatar que a sombra do complemento de A, $S(A^C)$, é igual ao complemento da sombra de A, $(S(A))^C$.

Após a verificação dos diagramas de Venn, aplicados nas operações de união, interseção, diferença e complemento sobre as sombras projetadas, pode-se afirmar que nas operações de união e complemento as sombras são equivalentes, ou seja, $S(A \cup B) = S(A) \cup S(B)$ e $S(A^c) = (S(A))^c$. Nas operações de interseção e diferença, nem sempre os resultado são equivalentes, nos casos em que existem pontos em comum, ou seja, quando as sombras se sobrepõem, a sombra da interseção é diferente da interseção das sombras, $S(A \cap B) \neq S(A) \cap S(B)$ e a sombra da diferença é diferente da diferença das sombras, $S(A - B) \neq S(A) - S(B)$.

Devido a esses motivos estudou-se a possibilidade de transformar operações de interseção e diferença em operações que apresentassem resultados equivalentes a estas operações. Procurando encontrar uma possível forma de obter resultados equivalentes com a sombra de uma operação e com a mesma operação aplicada à sombra.

4.10 Reordenação das Operações Usando União e Complemento

Como já foi descrito na seção 4.7 deste capítulo, as operações de interseção e diferença podem ser convertidas em operações que envolvam somente união e complemento. Esta conversão foi aplicada nas operações que envolvem sombras, ou seja, as operações de interseção e diferença, foram convertidas para operações que possuem somente união e complemento. Os resultados serão apresentados nos diagramas de Venn a seguir.

4.10.1 Operação de Interseção $(A^c \cup B^c)^c$

Considerando-se que $A \cap B = (A^c \cup B^c)^c$, os casos (a), (b) e (c) da figura 4.34 ilustram as operações de união entre os complementos das sombras. Na figura 4.35, o complemento da operação realizada na figura 4.34, ou seja, o complemento da união dos complementos é apresentado.

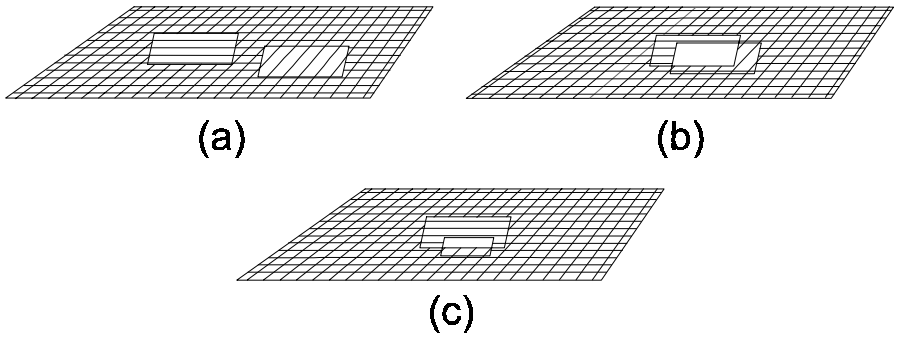


FIGURA 4.34 – Operação $(A^c \cup B^c)$

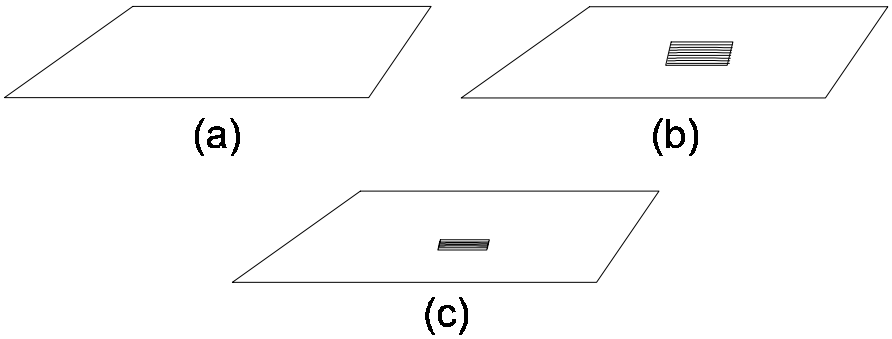


FIGURA 4.35 – Operação $(A^c \cup B^c)^c$

Comparando os resultados da figura 4.35, com os resultados apresentados na figura 4.28, pode-se verificar que nos casos (a) e (b), as sombras são corretas. No caso (a) apresenta-se uma solução correta, pois como não existe interseção entre os objetos, na figura 4.25, consequentemente a sombra não existe, com o uso com os complementos pode-se ver que o complemento do universo é igual ao conjunto vazio ($U^c = \varnothing$, propriedades do complemento), e a sombra do conjunto vazio não existe. No caso (b) o resultado exibido é equivalente ao caso (b) da figura 4.28, onde apresenta-se a sombra da interseção, porém usando somente operações de união e complemento.

O caso (c) continua apresentando sombra. Nessa situação permanece o erro, pois entre os objetos, da figura 4.25 (que está sendo usada como exemplo modelo) não existe interseção entre os objetos, uma vez que estes são disjuntos, portanto não existe sombra, para a interseção.

4.10.2 Operação de Diferença $(A^c \cup B)^c$

Segundo a equivalência $A - B = (A^c \cup B)^c$, nos casos (a), (b) e (c) da figura 4.36, apresentam-se as operações de união entre o complemento da primeira sombra com a segunda sombra. Na figura 4.37 o complemento desta união.

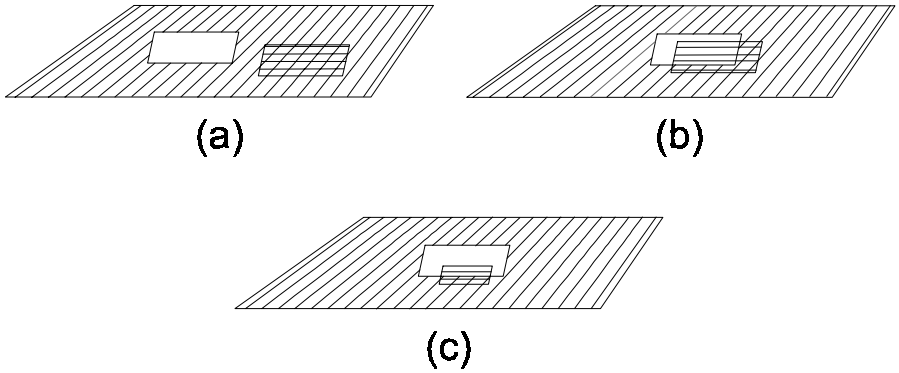


FIGURA 4.36 – Operação $(A^c \cup B)$

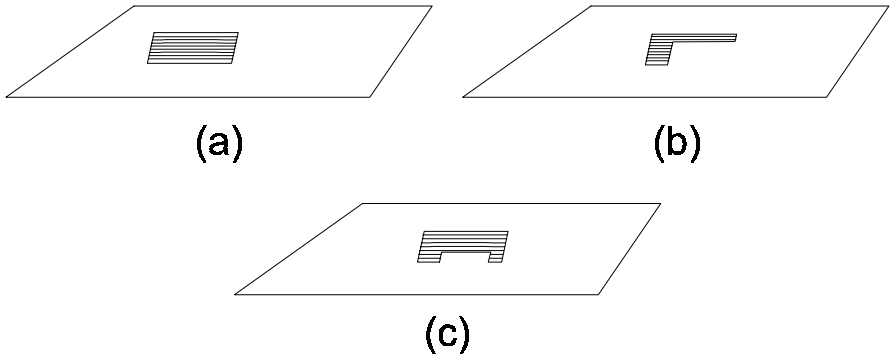


FIGURA 4.37 – Operação $(A^c \cup B)^c$

Comparando os resultados da figura 4.37, com os resultados apresentados na figura 4.30, que são os pretendidos, pode-se verificar que nos casos (a) e (b), as sombras são corretas. No caso (a) a sombra apresentada corresponde a sombra do objeto A, neste caso, A – B = A, pois os objeto A e B são disjuntos. No caso (b) temos uma sobreposição nas sombras, mas os objetos A e B, da figura 4.25, também possuem pontos em comum, neste caso a sombra apresentada na figura 4.37 (b), corresponde aos pontos projetados que pertencem ao objeto A e não pertencem ao objeto B. Este resultado é equivalente ao resultado da figura 4.30 (b).

Já no caso (c) da figura 4.37 isso não acontece. Os objetos da figura 4.25 (c) (que está sendo usada como exemplo modelo) não estão sobrepostos, então a operação A – B é igual a A, e nesse caso a sombra gerada é diferente da sombra projetada de A.

4.11 Considerações Finais

Percebe-se que apesar de ser feita a reordenação das operações de interseção e diferença, com uso de união e complemento, não se encontrou uma solução genérica para o problema da aplicação de CSG em sombras projetadas.

A aplicação das operações booleanas, utilizadas na modelagem do objeto, sobre as sombras dos mesmos objetos não apresenta resultados equivalentes quando duas sombras estão sobrepostas, isto é, possuem pontos em comum e os objetos são disjuntos. Isso também como pode ser visto nas figuras 4.29 (c), 4.31 (c).

A tentativa de solucionar este problema com o uso das conversões das operações de interseção e diferença em operações com união e complemento, que apresentem os mesmos resultados finais, não apresentou resultados satisfatórios, ver figuras 4.35 (c) e 4.37(c).

Neste ponto do trabalho pode-se responder as três perguntas do capítulo um, que representam o eixo principal desta pesquisa:

- A união das sombras é igual a sombra da união ?
- A interseção das sombras é igual a sombra da interseção ?
- A diferença das sombras é igual a sombra da diferença ?

Como pode ser visto, no decorrer deste capítulo somente a união das sombras é igual a sombra da união. O mesmo não ocorre com as operações de interseção e diferença.

Após a realização deste estudo e demonstração através dos casos apresentados, pode-se concluir que a sombra de um objeto, modelado através da Geometria Sólida Construtiva, será exibida de forma correta se essa for obtida através da projeção das faces iluminadas do objeto resultante da operação envolvida, isto é, após a computação do seu contorno.

Neste capítulo foram apresentadas formas alternativas de obtenção de sombras projetadas, mais especificamente, estudou-se a geração de sombras com base em operações de união e complemento, tais operações foram utilizadas também nas operações de interseção e diferença.

5 Avaliador de Contornos (*Boundary Evaluator*)

A descrição gráfica do objeto final modelado pela Geometria Sólida Construtiva é montada por um conjunto de algoritmos, constituindo o chamado avaliador de contornos (*boundary evaluator*). O avaliador de contornos é normalmente a parte mais complexa de um modelador CSG. Este capítulo tem por finalidade descrever os passos executados por um avaliador de contornos.

Como já foi relatado no capítulo dois, o avaliador de contornos é responsável pela descrição gráfica do objeto final. Dentre os algoritmos básicos de um sistema CSG, o avaliador de contornos é o que exige mais tempo e memória do computador.

Um objeto é formado pelo conjunto de todos os pontos que o definem. Este conjunto de pontos é subdividido em dois subconjuntos, o conjunto de pontos interiores e o conjunto de pontos da fronteira do objeto.

Os algoritmos de avaliação de contornos baseiam-se no conceito de classificação de inclusão. O avaliador de contornos determina onde as faces são truncadas e onde novos vértices e arestas são criados ou removidos. Novas arestas são criadas onde duas faces se interseccionam. O avaliador de contornos encontra estas interseções e então determina, através da classificação da pertinência a um conjunto, a classificação dos vértices que irão dar origem às novas arestas, gerando assim um novo sólido.

O avaliador de contornos analisa as relações entre as primitivas, levando em consideração, as suas bordas e seus interiores. Isso tem por finalidade garantir a consistência dos resultados, pois um objeto só é válido para a Geometria Sólida Construtiva se o mesmo possuir pontos interiores e pontos de fronteira, formando assim um objeto regular.

Nesse capítulo é apresentada uma visão global desta parte do sistema, pois a avaliação dos contornos de um objeto CSG passa por várias etapas. Estas etapas foram distribuídas em módulos, os quais são descritos a seguir.

5.1 Processo de Avaliação dos Contornos

O processo de avaliação dos contornos de um objeto baseia-se na determinação de onde novas arestas e vértices devem ser criados e onde arestas e vértices podem ser destruídos.

Novas arestas são criadas onde as superfícies dos dois sólidos, sobre os quais está sendo aplicada a operação booleana, se interceptam. O avaliador calcula estas interseções e, então, determina através da classificação por inclusão, quais são os novos vértices e arestas do novo sólido. Cabe salientar que as faces do novo sólido podem ser modificadas ou destruídas, porém nunca serão criadas.

Para a realização desta tarefa primeiramente intercepta-se as faces do objeto A contra as faces do objeto B. Sempre que duas faces se interceptarem uma aresta será criada, e devem ser colocadas em uma lista de arestas, neste trabalho chamadas de arestas-tentiva.

O passo seguinte será o de interceptar todas as arestas-tentativa com as faces, gerando assim pontos que interceptam tais arestas. Posteriormente classificam-se os segmentos das arestas-tentativa em relação as faces, que mais tarde serão selecionados, ou não, para fazerem parte do novo sólido.

5.2 Envelopes

Quando dois ou mais objetos interagem em um sistema, estes objetos podem entrar em contato interpenetrando-se. A forma como estes objetos entram em contato pode gerar uma interseção.

Detecção de colisões envolve determinar quando um objeto penetra em outro. Esta é uma proposição custosa, especialmente quando um grande número de objetos está envolvido e os objetos têm formas complexas [MAC 98].

Na Geometria Sólida Construtiva é muito importante saber se existe ou não a interseção entre duas partes. Caso haja uma interseção é necessário encontrar os pontos de interseção das mesmas.

Se não existe interseção não existem interseções, consequentemente não há necessidade de se fazer uma avaliação dos contornos. Por outro lado, não são todas as faces de uma primitiva que vão colidir com as faces da(s) outra(s) primitiva(s). Com o objetivo de ganhar tempo de processamento, faz-se o uso de envelopes nos testes de colisões.

Se os envelopes de duas primitivas são disjuntos, então as suas faces não se interceptam. Se os envelopes de duas primitivas não são disjuntos pode ser que as primitivas se interceptem. Neste caso existe a necessidade de se fazer uma análise mais aprofundada, e assim verificar face a face se ocorre interseção, em caso afirmativo, calculam-se os pontos de interseção.

Cada sólido possui um envelope que o engloba. Para determinar o envelope de um objeto, basta encontrar os pontos x_{min}, y_{min}, z_{min}, x_{max}, y_{max} e z_{max}, que definem o envelope do mesmo. Ver figura 5.1.

Entre dois objetos A e B existem seis condições em que não ocorre uma interseção:

$x_{min\text{-}A} > x_{max\text{-}B}$ $\quad$ $y_{min\text{-}A} > y_{max\text{-}B}$ $\quad$ $z_{min\text{-}A} > z_{max\text{-}B}$

$x_{min\text{-}B} > x_{max\text{-}A}$ $\quad$ $y_{min\text{-}B} > y_{max\text{-}A}$ $\quad$ $z_{min\text{-}B} > z_{max\text{-}A}$

Se o teste der um resultado falso então os objetos não colidem, caso contrário, faz-se uma análise mais detalhada, ou seja, cada face do objeto A será testada com cada face do objeto B.

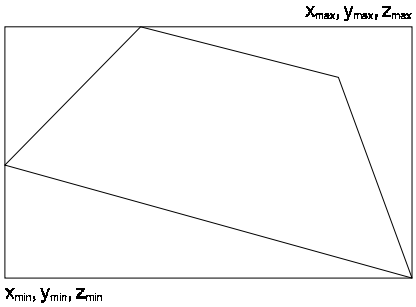


FIGURA 5.1 – Envelope com os pontos x_{min}, y_{min}, z_{min}, x_{max}, y_{max} e z_{max}

5.3 Triangularização

Depois de serem verificadas as ocorrências de colisões, calculam-se as interseções entre as faces, pois sempre que ocorrer uma interseção entre duas faces de um objeto, uma nova aresta irá surgir. Uma das formas de verificar se duas faces estão se interseccionando é descrita abaixo.

Para realizar tal tarefa é necessário fazer a triangularização das faces. O processo de triangularização aqui apresentado é genérico e pode ser aplicado em qualquer polígono, independente do número de lados.

A idéia principal está baseada em calcular o ângulo entre duas arestas que formam um canto do polígono. Estas arestas podem ser representadas através de dois vetores, como na equação (5.1), extraída da geometria, que possibilita encontrar o ângulo entre dois vetores. Se o ângulo encontrado for menor que 180º e todos os outros vértices do polígono estiverem fora do triângulo, então um triângulo pode ser gerado.

$$\alpha = \arccos\left(\frac{\vec{u}.\vec{v}}{|\vec{u}|.|\vec{v}|}\right) \qquad (5.1)$$

Cada face possui uma lista de vértices e o cálculo dos ângulos será feito sempre a partir de três vértices consecutivos, como mostra a figura 5.2.

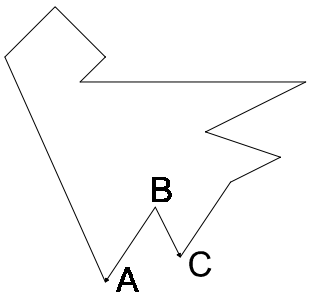


FIGURA 5.2 – Polígono que será triangularizado

Na figura acima os vértices, A, B e C, formam os vetores $\overrightarrow{BA}$ e $\overrightarrow{BC}$. Para fazer a tringularização calcula-se o vetor normal a partir dos vetores $\overrightarrow{BA}$ e $\overrightarrow{BC}$. Se este vetor resultar em um sentido diferente do vetor normal do polígono,

que aponta para o lado de fora, é necessário subtrair o ângulo encontrado entre os dois vetores de 360, caso contrário, ou seja, se o sentido do vetor normal encontrado for igual ao sentido do vetor normal do polígono, simplesmente calcula-se o ângulo entre os vetores.

Por exemplo, na figura 5.2, entre os vetores $\overrightarrow{BA}$ e $\overrightarrow{BC}$ existe um ângulo de 60º, porém o sentido do vetor normal entre eles é diferente do sentido do vetor normal da face, assim o ângulo de 60º será subtraído de 360º e será de 300º.

Quando o ângulo encontrado entre dois vetores for maior que 180º, que é o caso em questão, ou um dos vértices do polígono estiver dentro do triângulo, deve-se armazenar o primeiro vértice (vértice A) em uma lista auxiliar e dar continuidade ao procedimento, ou seja, calcular os vetores e o ângulo entre eles, levando-se em consideração os próximos três vértices da lista inicial, ver figura 5.3.

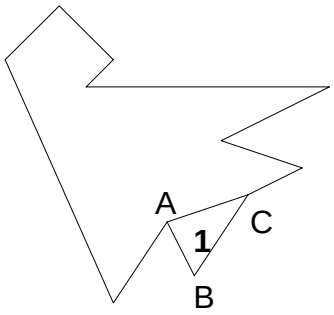


FIGURA 5.3 – Polígono com o primeiro triângulo da triangularização

Na figura 5.3, pode ser visto o primeiro triângulo da triangularização do polígono, pois o ângulo entre os vetores $\overrightarrow{BA}$ e $\overrightarrow{BC}$ é de 60º, o sentido do vetor normal entre eles é igual ao sentido do vetor normal do polígono e os três pontos do triângulo não estão dentro de nenhum outro triângulo do polígono. Sempre que a operação resultar em um triângulo, deve-se então armazená-lo em uma lista de triângulos, que ao final irão representar a lista de triângulos da face em questão.

Executando este procedimento para toda a lista de vértices obtêm-se cinco triângulos, e também uma lista auxiliar com quatro vértices (A', B', C' e D'), conforme a figura 5.4. Aplica-se o mesmo raciocínio, sobre a lista auxiliar, até que esta tenha um número de vértices menor que três, assim o polígono estará triangularizado, ver figura 5.5.

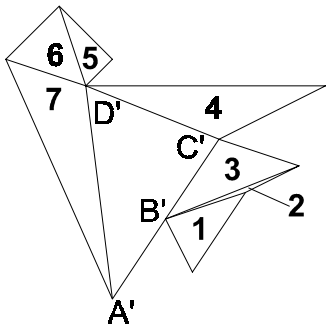


FIGURA 5.4 – Triângulos gerados com a lista de vértices inicial

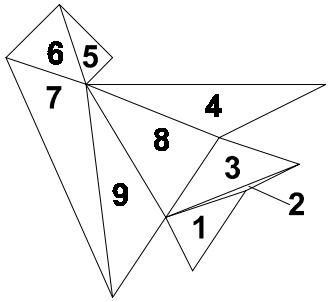


FIGURA 5.5 – Polígono triangularizado

5.4 Arestas-tentativa

Arestas-tentativa correspondem às arestas originadas pela interseção de duas faces de sólidos primitivos distintos. É necessário determinar todas as arestas originadas pela interseção de duas faces de primitivas distintas.

Cada sólido possui um envelope que o engloba, se os envelopes das primitivas em questão forem disjuntos, então as suas faces não se interceptam. Se os envelopes das primitivas não são disjuntos, pode ser que as faces se interceptam, neste caso existe a necessidade de se fazer uma análise mais detalhada, verificando as colisões entre as faces das primitivas, são estas interseções que irão originar as arestas-tentativa.

Para encontrar os vértices que formam uma aresta tentativa faz-se uso dos triângulos das faces, deve-se calcular as distâncias dos três vértices de uma face, F1, em relação ao plano suporte da outra face, F2. Calculam-se estas distâncias pela seguinte equação, extraída da Geometria Analítica:

$$Dist_i = \frac{Ax_i + By_i + Cz_i + D}{\sqrt{A^2 + B^2 + C^2}} \qquad (5.2)$$

Onde x_i, y_i e z_i são as coordenadas de um vértice de F1, e A, B, C e D são os coeficientes da equação do plano suporte de F2.

Durante o processo de análise, o valor encontrado por $Dist_i$ não é relevante, mas sim se o mesmo é positivo, negativo ou nulo, pois as distâncias dos três vértices de um triângulo ao plano da outra face podem ser positivas, negativas ou nulas. Para verificar se uma aresta tentativa é válida ou não, deve-se levar em consideração os seguintes casos:

A) Três distâncias nulas – Neste caso todos os vértices de F1 (A, B e C) estão sobre o plano suporte de F2. As duas faces são coplaneres e podem se interceptar ou não. Se as faces se interceptarem, as arestas geradas são segmentos das arestas das faces, assim esses segmentos não devem ser incluídos na lista de arestas-tentativa. Ver figura 5.6.

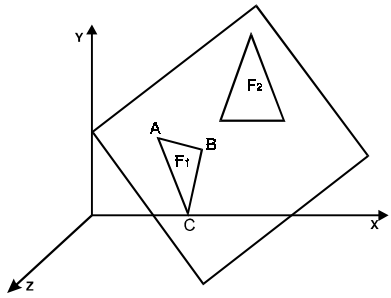


FIGURA 5.6 – Caso A

B) Duas distâncias nulas e uma positiva – Dois vértices de F1 (A e B) estão sobre o plano suporte de F2. Logo uma aresta de F1 está sobre o plano suporte de F2. Se essa aresta interceptar a face F2, a aresta é um segmento da mesma. Assim, esse segmento não deve ser incluído na lista de arestas-tentativa. Ver figura 5.7.

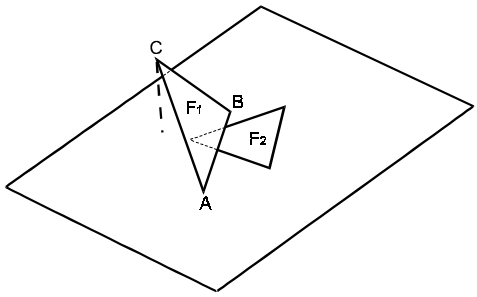


FIGURA 5.7 – Caso B

C) Uma distância é nula e duas são positivas ou negativas – Apenas um vértice de F1 (A) está sobre o plano suporte de F2. Os outros dois vértices de F1 (B e C) estão no mesmo semi-espaço, definido pelo suporte de F2. Não existe a possibilidade de F1 e F2 terem uma aresta como sua interseção. Ver figura 5.8.

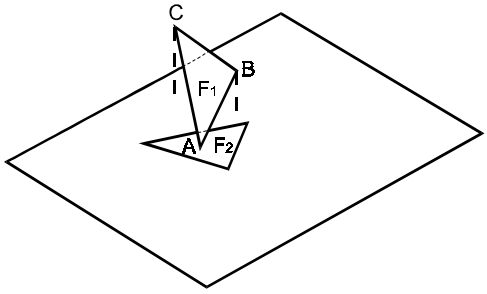


FIGURA 5.8 – Caso C

D) Três distâncias positivas ou negativas – Todos os vértices de F1 (A, B e C) estão em um mesmo semi-espaço definido pelo plano suporte de F2. F1 e F2 não se interceptam. Ver figura 5.9.

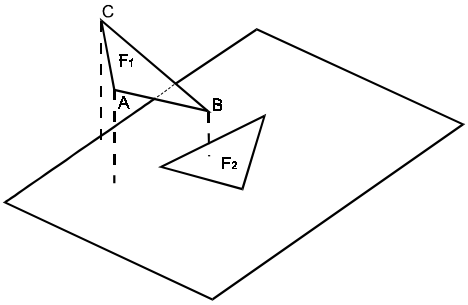


FIGURA 5.9 – Caso D

E) Duas distâncias são positivas e uma é negativa, ou duas são negativas e uma é positiva – A face F1 “cruza” o plano suporte da face F2. Neste caso as faces F1 e F2 têm potencial de se interceptarem gerando uma aresta como sua interseção. Ver figura 5.10.

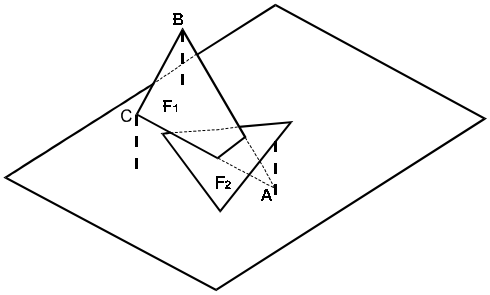


FIGURA 5.10 – Caso E

F) Uma distância é nula, uma é positiva e uma é negativa – Um vértice de F1 está no plano suporte de F2, e os dois outros vértices estão em semi-espaços distintos. A face F1 “cruza” o plano suporte de F2. As faces têm potencial de se interceptarem gerando uma aresta como sua interseção. Ver figura 5.11

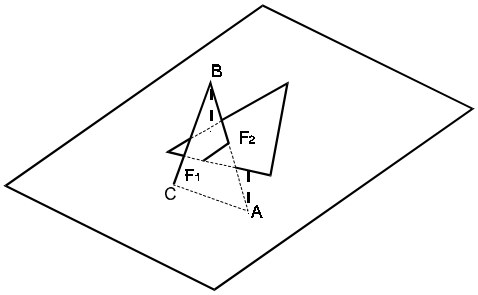


FIGURA 5.11 – Caso F

Como pode-se perceber, somente nos casos E e F existe a possibilidade de uma aresta ser gerada como interseção de duas faces (dois planos). No caso de ocorrem situações como os casos E e F deve-se aplicar o mesmo procedimento, porém agora calculando as distâncias dos vértices da face F2 em relação ao plano suporte de F1. Somente se recaírem novamente nos casos E e F, é que existirá uma aresta como interseção entre as duas faces.

Deve-se então calcular a reta suporte da possível aresta tentativa, que é a reta gerada pela interseção dos dois planos. Para encontrar esta reta basta determinar os pontos de interseção das arestas de uma face com o plano suporte da outra. O ponto de interseção P_i é dado por [RIP 87]:

$$P_i = (x_{int}, y_{int}, z_{int}) \tag{5.3}$$

$$x_{int} = x_1 + \frac{Dist_1(x_2 - x_1)}{(Dist_1 - Dist_2)} \tag{5.4}$$

$$y_{int} = y_1 + \frac{Dist_1(y_2 - y_1)}{(Dist_1 - Dist_2)} \tag{5.5}$$

$$z_{int} = z_1 + \frac{Dist_1(z_2 - z_1)}{(Dist_1 - Dist_2)} \tag{5.6}$$

Onde:

- (x_1, y_1, z_1) e (x_2, y_2, z_2) são os vértices da aresta;
- $Dist_1$ e $Dist_2$ são as distâncias desses vértices ao plano suporte da outra face.

Encontrados os pontos de interseção, que são dois para cada face triangular, pode-se determinar a reta suporte, pois conhecendo dois pontos $A(x_1, y_1, z_1)$ e $B(x_2, y_2, z_2)$ e o vetor direção $\overrightarrow{AB} = (x_3, y_3, z_3)$, é possível encontrar a reta através das equações [RIP 87]:

$$x = x_1 + x_3 \bullet t \tag{5.7}$$

$$y = y_1 + y_3 \bullet t \tag{5.8}$$

$$z = z_1 + z_3 \bullet t \tag{5.9}$$

Onde:

- x_1, y_1 e z_1 são coordenadas do ponto A;
- x_3, y_3 e z_3 são os valores do vetor $\overrightarrow{AB}$, ou seja, $(x_2 - x_1, y_2 - y_1, z_2 - z_1)$;
- variável t deve ter um intervalo de variação suficientemente grande para gerar uma reta maior que o tamanho das faces, neste protótipo a variação é em t, onde $-50 \leq t \leq 50$.

O próximo passo consiste em calcular as interseções entre as arestas que formam as faces triangulares com a reta suporte encontrada. Duas retas somente irão ter um ponto de interseção comum entre elas se as mesmas não forem paralelas.

Para verificar o paralelismo entre duas retas pode-se usar os vetores diretores das mesmas. Sabe-se das propriedades do produto vetorial. O produto vetorial entre os dois vetores, $\vec{a}\times\vec{b}$, origina um outro vetor, ortogonal simultaneamente aos vetores $\vec{a}$ e $\vec{b}$. Calcula-se o módulo deste vetor. Se o valor resultante for igual a 0 (zero) então as retas são paralelas, caso contrário elas são ortogonais.

Segundo [MOR 85], duas arestas, que não são paralelas, podem ter um ponto de interseção comum entre elas. Dadas duas arestas, uma pertence a uma reta suporte $\vec{p}(u) = \vec{a}+u\vec{b}$ e outra pertence a reta suporte $\vec{q}(w) = \vec{c}+w\vec{d}$, onde $\vec{a},\vec{b},\vec{c},\vec{d} \in U \subset \Re^3$; u e w são escalares, tais que u e w $\in$ [0,1]. O seu ponto de interseção é p(u) = q(w) = R, ou seja, $\vec{a}+u\vec{b} = \vec{c}+w\vec{d}$, ver figura 5.12.

A reta $\vec{p}(u) = \vec{a}+u\vec{b}$ possui um ponto inicial A, num sistema de referência com origem O, tendo-se A= O + $\vec{a}$, onde:

(1) $\vec{a}$ é o vetor posição do ponto A e

(2) $\vec{b}$ é o vetor diretor da reta suporte

Da mesma forma, a reta $\vec{q}(w) = \vec{c}+w\vec{d}$, possui um ponto inicial C, no mesmo sistema de referência com origem O. Tendo-se então C= O + $\vec{c}$:

(1) $\vec{c}$ é o vetor posição do ponto C

(2) $\vec{d}$ é o vetor diretor da reta suporte

Usando as propriedades dos vetores, pode-se calcular u e w pelas seguintes equações [MOR 85]:

$$u = -\frac{(\vec{c}\times\vec{d})\bullet\vec{a}}{(\vec{c}\times\vec{d})\bullet\vec{b}} \qquad (5.10)$$

$$w = -\frac{(\vec{a}\times\vec{b})\bullet\vec{c}}{(\vec{a}\times\vec{b})\bullet\vec{d}} \qquad (5.11)$$

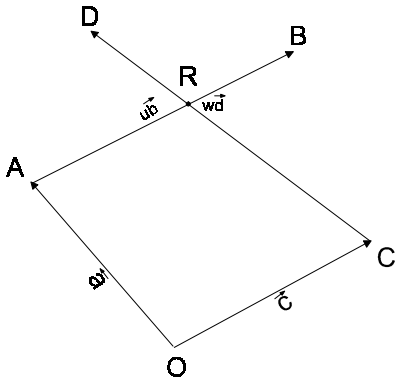


FIGURA 5.12 – Interseção de duas retas

Apesar desta equação calcular o ponto de interseção entre duas arestas para a maioria dos casos, em algumas situações, quando as coordenadas x ou y dos pontos A e B forem iguais, ela apresenta um erro de divisão por zero.

Diante desta situação foi necessário encontrar uma solução para este problema. Considerando a notação $x_{\overrightarrow{AB}}$, para designar a coordenada x do vetor $\overrightarrow{AB}$ e assim para as demais coordenadas dos vetores. A solução encontrada está baseada nas seguintes condições:

Se $((z_{\overrightarrow{AB}} \cdot x_{\overrightarrow{CD}}) - (x_{\overrightarrow{AB}} \cdot z_{\overrightarrow{CD}})) \neq 0$, então

$$u = \frac{((x_{\overrightarrow{CD}} \cdot z_C) - (z_A \cdot x_{\overrightarrow{CD}}) + (x_A \cdot z_{\overrightarrow{CD}}) - (x_C \cdot z_{\overrightarrow{CD}}))}{((z_{\overrightarrow{AB}} \cdot x_{\overrightarrow{CD}}) - (x_{\overrightarrow{AB}} \cdot z_{\overrightarrow{CD}})}$$

Senão Se $((y_{\overrightarrow{AB}} \cdot x_{\overrightarrow{CD}}) - (x_{\overrightarrow{AB}} \cdot y_{\overrightarrow{CD}})) \neq 0$, então

$$u = \frac{((y_{\overrightarrow{CD}} \cdot z_C) - (z_A \cdot y_{\overrightarrow{CD}}) + (y_A \cdot z_{\overrightarrow{CD}}) - (y_C \cdot z_{\overrightarrow{CD}}))}{((z_{\overrightarrow{AB}} \cdot y_{\overrightarrow{CD}}) - (y_{\overrightarrow{AB}} \cdot z_{\overrightarrow{CD}}))}$$

Senão $u = \frac{((x_{\overrightarrow{CD}} \cdot y_C) - (y_A \cdot x_{\overrightarrow{CD}}) + (x_A \cdot y_{\overrightarrow{CD}}) - (x_C \cdot y_{\overrightarrow{CD}}))}{((y_{\overrightarrow{AB}} \cdot x_{\overrightarrow{CD}}) - (x_{\overrightarrow{AB}} \cdot y_{\overrightarrow{CD}}))}$

Para calcular o valor de w, também existem algumas condições a serem analisadas, que são:

Se $x_{\overline{CD}} \neq 0$, então

$$w = \frac{(x_A + u \cdot y_{\overline{AB}} - x_C)}{x_{\overline{CD}}}$$

Senão Se $y_{\overline{CD}} \neq 0$, então

$$w = \frac{(y_A + u \cdot y_{\overline{AB}} - y_C)}{y_{\overline{CD}}}$$

$$\text{Senão} \quad w = \frac{(z_A + u \cdot z_{\overline{AB}} - z_C)}{z_{\overline{CD}}}$$

Com os valores de u e w calculados pode-se calcular o ponto de interseção entre as duas retas. Se u ou w $\notin$ [0,1], então não ocorre interseção entre as retas, caso contrário, ou seja, u e w $\in$ [0,1], então o ponto de interseção é encontrado por:

$$P_i = \vec{a} + u\,\overrightarrow{AB}$$

O passo seguinte é a classificação dos vértices da possível aresta tentativa, em relação às faces triangulares. Estes vértices poderão ser classificados como IN, OUT ou ON, conforme será caracterizado em 5.3.7. A partir da classificação dos vértices é possível determinar se um segmento da aresta tentativa é IN, OUT ou ON. Pois, para que uma reta intercepte ambas as faces simultaneamente, e gere uma aresta tentativa, deve haver um segmento da mesma que seja IN em relação a ambas as faces. As arestas que forem classificadas como IN são arestas-tentativa válidas, sendo incluídas na lista de arestas-tentativa.

Os vértices das arestas-tentativa devem ser classificados em relação as arestas das faces, buscando verificar se estes são do tipo ON, quando estes forem assim classificados devem ser incluídos na lista de vértices que formam a face, pois são vértices novos e passam a fazer parte da face.

5.5 Classificação dos Vértices

A tarefa de classificação dos vértices, como já foi dito no capítulo dois, é tarefa do algoritmo de classificação *Set-Membership Classification*. Este procedimento deve ser aplicado quando deseja-se classificar um dado conjunto X em relação a um conjunto S, onde XinS, XonS e XoutS são os segmentos do conjunto X, respectivamente no interior, contorno e exterior do conjunto referência S.

A classificação dos vértices pode ser IN (vértice no interior do objeto), OUT (vértice fora do objeto) ou ON (vértice está sobre o contorno do objeto). Esta classificação é feita sob as duas faces que estão colidindo e têm a possibilidade de originarem uma aresta tentativa.

Também é necessário fazer a classificação dos vértices de um objeto em relação ao outro e classificar as arestas. Pois é com base na classificação

das arestas que formam o objeto, juntamente com as arestas-tentativa, que o contorno do objeto resultante será traçado.

5.5.1 Classificação ON

Para verificar se um vértice é ON, é necessário saber se o mesmo encontra-se sobre o contorno do objeto. O vértice deve ser classificado em relação às arestas que formam a face em questão.

A classificação aqui usada leva em consideração a distância do vértice em relação a aresta que está sendo testada e os vetores normais de ambas as faces. Para calcular a distância entre um ponto e uma aresta, deve-se obter o vetor direção desta aresta e usar a seguinte equação [GEO 99]:

$$\text{Distância} = \sqrt{\frac{\begin{vmatrix} y_0 - y_1 & z_0 - z_1 \\ b & c \end{vmatrix}^2 + \begin{vmatrix} z_0 - z_1 & x_0 - x_1 \\ c & a \end{vmatrix}^2 + \begin{vmatrix} x_0 - x_1 & y_0 - y_1 \\ a & b \end{vmatrix}^2}{a^2 + b^2 + c^2}} \qquad (5.12)$$

Onde:

- x_0, y_0 e z_0 são as coordenadas do ponto;
- x_1, y_1 e z_1 são as coordenadas iniciais da aresta;
- a, b e c são os valores correspondentes ao vetor direção.

Se o valor da distância for igual a 0 (zero), então o ponto está localizado sobre a reta. Agora basta testar se ele está localizado dentro do intervalo determinado pelos pontos inicial e final da aresta, em caso afirmativo este vértice é classificado como ON.

Na figura 5.13 apresenta-se um exemplo para este caso. O ponto P tem classificação ON, pois o mesmo encontra-se dentro do intervalo dos pontos A e B, que formam uma aresta da face, e a distância deste ponto em relação a aresta AB é igual a 0 (zero).

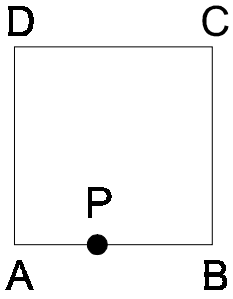


FIGURA 5.13 – Classificação ON de um ponto

5.5.2 Classificação IN ou OUT

Para classificar os vértices como IN ou OUT é necessário utilizar a triangularização e os vetores normais das faces. Os triângulos que estão sendo considerados pertencem a faces, portanto, o vetor normal destes triângulos é igual ao vetor normal da face, ou seja, estão apontando para fora.

Um vértice será classificado como IN se ele estiver dentro de um dos triângulos. Para fazer esta classificação, basta calcular o vetor diretor de uma das arestas do triângulo e também o vetor formado pelo ponto que está sendo testado e um dos vértices da aresta. Desta maneira teremos dois vetores e a partir desses calcula-se o produto vetorial, que irá originar um novo vetor normal. Esse vetor normal é comparado com o vetor normal da face, se eles apontarem para o mesmo sentido então o vértice é classificado com IN. Repetem-se os mesmos passos para as outras duas arestas que formam o triângulo, se nos três casos o vértice for classificado com IN então ele está localizado no interior do triângulo, se um dos vetores normais encontrados pelos produtos vetoriais apontar para uma direção diferente do vetor normal da face então o vértice será classificado como OUT.

Na figura 5.14 pode-se verificar o exemplo de um ponto que é classificado como IN. Calculando o produto vetorial entre os vetores $\overrightarrow{AP}$ e $\overrightarrow{AB}$, $\overrightarrow{BP}$ e $\overrightarrow{BC}$ e $\overrightarrow{CP}$ e $\overrightarrow{BC}$ teremos três vetores normais, um para cada operação realizada. Se os três vetores encontrados apontarem no mesmo sentido do vetor normal da face, que é este caso, então o ponto está localizado dentro do triângulo, sendo classificado como IN.

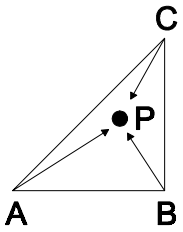


FIGURA 5.14 – Classificação IN de um ponto

5.6 Classificação das Arestas

Após ser feita a classificação dos vértices é possível classificar as arestas. Uma aresta é formada por dois vértices, para classificá-la é necessário analisar os vértices que a formam, juntamente com o tipo de operação booleana envolvida na modelagem do objeto.

Para encontrar as arestas que farão parte do objeto resultante, inicialmente é necessário realizar uma classificação de todos os vértices do objeto A em relação ao objeto B e vice-versa. Cada um dos vértices é classificado como ON, IN ou OUT. Depois de classificar todos os vértices é necessário selecionar as arestas que irão representar o objeto resultante.

As arestas são classificadas pelos dois vértices que a formam e o tipo de operação envolvida na modelagem do objeto. Nas tabelas abaixo apresentam-se as classificações das arestas de acordo com a operação booleana envolvida [RIP 87], chama-se a atenção para as arestas com a classificação ON*, que será explicada mais adiante.

Na tabela 5.1 são apresentadas as classificações das arestas na operação de interseção.

TABELA 5.1 – Interseção

Vértice 1	**Vértice 2**	**Classificação da aresta**
IN	IN	IN
IN	ON	ON*
ON	IN	ON*
ON	ON	ON*/OUT
ON	OUT	OUT
OUT	ON	OUT
OUT	OUT	OUT

Na operação de interseção quando a classificação de um dos vértices for ON e o outro for IN, é necessária uma análise mais detalhada. Nesta situação, se os vetores normais das faces apontarem para o mesmo sentido então a aresta será classificada como ON+ senão será ON-. O mesmo vale para as classificações onde os dois vértices são ON, se os vetores normais apontarem para o mesmo sentido, então a classificação da aresta será ON+, senão será OUT.

Na tabela abaixo são apresentadas as classificações das arestas na operação de união.

TABELA 5.2 – União

Vértice 1	**Vértice 2**	**Classificação da aresta**
IN	IN	IN
IN	ON	IN
ON	IN	IN
ON	ON	IN/ON*
ON	OUT	ON*
OUT	ON	ON*
OUT	OUT	OUT

Na união, sempre que um dos vértices for classificado como ON e o outro como OUT, faz-se uma análise mais detalhada. Quando os vetores normais das faces apontarem para o mesmo sentido então a aresta é classificada como ON+, senão será ON-. Aplica-se o mesmo para as classificações ON e ON, se os vetores normais apontarem para o mesmo sentido, então a classificação da aresta será ON+, senão será IN.

Na tabela 5.3, apresentam-se as classificações das arestas para a operação de diferença (A – B).

TABELA 5.3 – Diferença

Vértice 1	Vértice 2	Classificação da aresta
IN	IN	OUT
IN	ON	ON*
ON	IN	OUT
ON	ON	ON*/OUT
ON	OUT	ON*
OUT	ON	OUT
OUT	OUT	OUT

Na diferença, sempre que os dois vértices forem classificados como ON, faz-se uma análise mais detalhada. Se os vetores normais das faces apontarem para o mesmo sentido então a aresta é classificada como OUT, senão será ON-. Nos casos em que as classificações dos vértices forem IN e ON ou ON e OUT, realiza-se o teste dos vetores normais, se eles apontarem para o mesmo sentido a classificação da aresta será ON+, senão será ON-.

Em todas as operações são encontradas classificações ON*, tais classificações precisam de mais detalhes para serem classificadas, pois estas arestas são segmentos tanto do objeto A como do objeto B. Para decidir se esta aresta irá fazer parte do objeto resultante observa-se o vetor normal das faces, se eles apontarem para o mesmo sentido significa que os polígonos possuem pontos interiores em comum, neste caso a aresta é classificada como ON+, caso contrário, significa que os polígonos possuem pontos interiores diferentes, assim a aresta é classificada como ON-.

5.7 Traçado do Objeto Resultante

Após as arestas-tentativa serem geradas e ser feita a classificação de vértices e arestas, pode-se traçar o contorno do objeto resultante da operação booleana envolvida na modelagem deste. Como já foi mencionado anteriormente, o contorno do objeto resultante será originado com base na classificação das arestas [RIP 87,KRI 96].

Na tabela 5.4, pode-se ver os critérios de classificação das arestas que são levados em consideração para cada uma das operações booleanas regularizadas. Dependendo do tipo de operação booleanas as arestas com a classificação correspondente são traçadas, originando o objeto final.

TABELA 5.4 – Operações booleanas e respectivas arestas que serão traçadas

Operação booleana	Arestas classificadas
União	Todas arestas OUT e ON
Interseção	Todas arestas IN e ON+
Diferença	Todas arestas OUT do primeiro objeto Todas arestas IN do segundo objeto Todas arestas ON-

Cabe salientar que as arestas-tentativa sempre são traçadas, independente do tipo de operação booleana que está sendo considerado.

5.8 Considerações Finais

Este capítulo descreve a estrutura de um avaliador de contornos (*Boundary Evaluator*) em um sistema de modelagem CSG. São exploradas as etapas executadas por este módulo e a maneira como estas foram tratadas neste trabalho.

6 Protótipo

Este capítulo tem por finalidade apresentar as implementações propostas neste trabalho, onde se colocaram em prática os conceitos estudados e discutidos nos capítulos anteriores.

Foram implementados dois protótipos, com a aplicação de métodos diferentes para a geração de sombras. Um deles está baseado no algoritmo descrito na seção 3.7.1, ou seja, faz-se a projeção do contorno das faces iluminadas (silhueta) do objeto gerado por CSG, os resultados obtidos, com este método, serviram como referência para comparação com o segundo método. No outro método aplicaram-se as zonas ativas das primitivas, como será descrito na seção 6.1.

Como existem sistemas de modelagem, de domínio público, que oferecem os recursos de integração e interação de ambientes tridimensionais, e com opções de se trabalhar com CSG, optou-se por fazer uso de um modelador já existente. O POVCAD 4.0 foi o modelador escolhido, pois este sistema gera um arquivo texto como saída, possibilitando a utilização deste arquivo em outros *softwares*. O protótipo desenvolvido importa o arquivo gerado pelo POVCAD. Com a especificação dos objetos, gera o objeto CSG e sua respectiva sombra.

No Anexo 1 desta dissertação são apresentados mais detalhes sobre o modelador POVCAD.

O protótipo foi implementado em Delphi 3.0. Usou-se também a biblioteca gráfica OpenGl, disponível para *download* no *site* da *Silicon Graphics* http://www.sgi.com e os componentes gráficos para OpenGl, *SignSoft VisIt Components 1.2* em versão demo, desenvolvidos pela empresa alemã *Sign Soft*, disponível para *download* no *site* http://www.signsoft.com.

6.1 Sombras Geradas por Projeção Para CSG

A partir do algoritmo proposto por [JAN 91] foi elaborado um novo algoritmo para a geração de sombras em cenas CSG, este algoritmo encontra-se descrito abaixo nesta seção e os detalhes da implementação serão expostos no capítulo seis.

A essência deste algoritmo está em gerar sombras para objetos CSG a partir da projeção das faces iluminadas. Para tanto é necessário computar as zonas ativas de cada primitiva.

A zona ativa de uma primitiva A em uma representação CSG de um sólido S é definida como uma região em que mudanças em A afetam o sólido S [ROS 89].

Neste algoritmo, assim como no descrito na seção 3.7.7.4, faz uso da conversão da árvore binária CSG, para que a árvore tenha somente operadores de união e interseção, representando rigorosamente o mesmo sólido da árvore original.

Um nodo interno F de uma árvore S é uma combinação booleana de duas expressões, representadas pelas subárvores esquerda e direita. O

caminho da raiz S até a primitiva A é definido por um conjunto de nodos que atravessam a movimentação de S para A seguindo os *links* “pai-filho” da árvore, incluindo S e A [ROS 89].

Nodos, que no caminho para a raiz, contém um número ímpar de filhos à direita de um nodo que representa uma operação de diferença (–), são chamados de negativos. Por sua vez, nodos que não são negativos são positivos.

Para ser encontrada a zona ativa de uma primitiva é necessário converter uma árvore CSG para a sua forma positiva, ou seja, gerar uma árvore onde existam somente operadores de união e interseção. Qualquer árvore CSG pode ser convertida para a sua forma positiva e para isso aplicam-se as relações abaixo, quando apropriadas, para cada nodo [ROS 89].

1. Trocar os operadores de $\cap$ por $\cup$, e vice-versa nos nodos internos negativos;
2. Substituir os operadores – por $\cap$, se o nodo correspondente é positivo e por $\cup$ em outros casos;
3. Complementar as primitivas negativas.

A relação 2 origina-se da propriedade ($A - B = A \cap B^C$) da teoria dos conjuntos (apresentada na seção 4.3.3). As relações 1 e 3 são decorrentes da aplicação das Leis de De Morgan. Consequentemente o resultado desta conversão eqüivale a aplicação das propriedades da teoria dos conjuntos, como é mostrado no exemplo abaixo.

O resultado desta operação é a árvore CSG na forma positiva, contendo somente operadores de união ($\cup$) e interseção ($\cap$), mas representando o mesmo sólido da árvore CSG original.

A conversão da árvore CSG para a sua forma positiva será detalhada no exemplo para a expressão CSG: $(A \cup B) - ((C - D) \cap (E \cup F))$. Onde são aplicadas as relações definidas anteriormente. Na figura 6.1 apresenta-se a árvore CSG da expressão original.

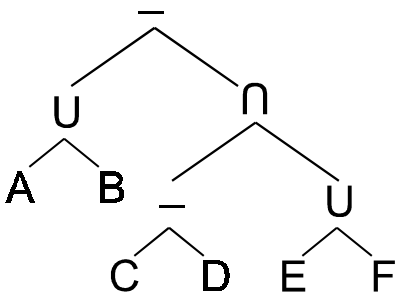


FIGURA 6.1 – Árvore CSG original

Inicialmente é necessário determinar quais são os nodos negativos e quais são os nodos positivos. Os nodos A, B e a operação de união entre eles são nodos positivos, pois se localizam do lado esquerdo da operação de diferença existente na raiz da árvore. Os nodos que na figura 6.2, aparecem circulados são todos nodos negativos, pois a quantidade de nodos à direita das

expressões de diferença, no caminho até a raiz, para estes casos resulta em números ímpares.

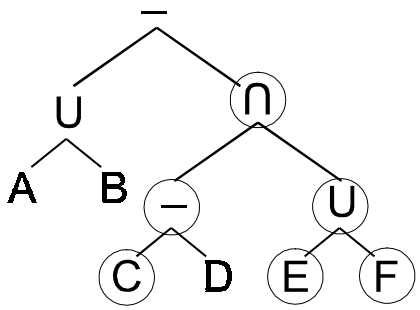


FIGURA 6.2 – Nodos negativos da árvore CSG

O próximo passo consiste em aplicar as relações de conversão para os casos apropriados. Trocar os operadores ∩ e ∪ nos nodos internos negativos (relação 1). Ver figura 6.3.

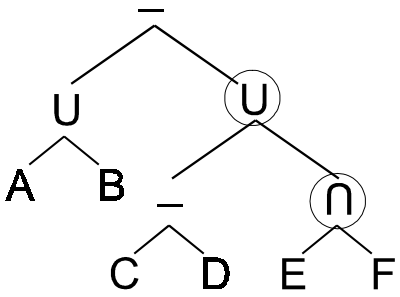


FIGURA 6.3 – Aplicação da relação de conversão 1 sobre os nodos

A etapa seguinte é substituir os operadores – por ∩ nos nodos positivos e por ∪ nos negativos (relação 2). Ver figura 6.4.

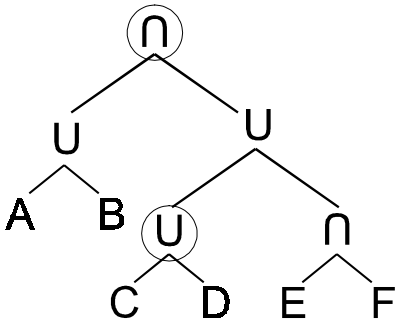


FIGURA 6.4 – Aplicação da relação de conversão 2 sobre os nodos.

Na figura 6.5, apresenta-se a aplicação da relação 3, que é complementar as primitivas dos nodos negativos. Desta forma obtêm-se a árvore positiva para a expressão CSG apresentada.

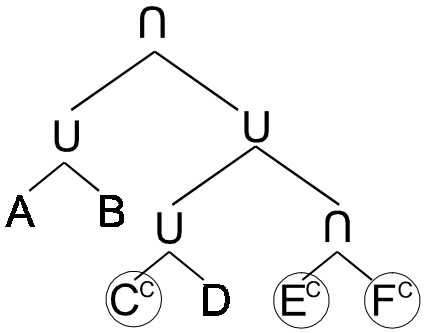


FIGURA 6.5 – Árvore positiva

Ao analisar as regras empregadas na conversão de uma árvore CSG, para a sua forma positiva, pode-se perceber que as conversões estão baseadas nas seguintes propriedades da teoria dos conjuntos [JAN 91, ROS 89].

$$A - B = A \cap B^c \qquad (6.1)$$

$$(A \cup B)^c = A^c \cap B^c \qquad (6.2)$$

$$(A \cap B)^c = A^c \cup B^c \qquad (6.3)$$

$$(A^c)^c = A \qquad (6.4)$$

Mais abaixo se apresentam as aplicações destas propriedades para a expressão CSG: $(A \cup B) - ((C - D) \cap (E \cup F))$, que foi utilizada no exemplo anterior. Pode-se perceber que o resultado obtido é o mesmo.

$(A \cup B) \cap ((C - D) \cap (E \cup F))^C$, propriedade (6.1)

$(A \cup B) \cap ((C - D)^C \cup (E \cup F)^C)$, propriedade (6.3)

$(A \cup B) \cap ((C \cap D^C)^C \cup (E \cup F)^C)$, propriedade (6.1)

$(A \cup B) \cap ((C \cap D^C)^C \cup (E^C \cap F^C))$, propriedade (6.2)

$(A \cup B) \cap ((C^C \cup (D^C)^C) \cup (E^C \cap F^C))$, propriedade (6.3)

$(A \cup B) \cap (C^C \cup D) \cup (E^C \cap F^C))$, propriedade (6.4)

Na prática a árvore positiva não precisa ser calculada, pois o algoritmo atua diretamente sobre a árvore original. Primeiramente substituem-se todos os operadores de diferença por interseção e complementa-se o lado direito da árvore (propriedade 6.1). Se o complemento for aplicado sobre uma subárvore, substituem-se os operadores de interseção por operadores de união e vice-versa, além de complementar as primitivas.

Em uma árvore na forma positiva, todos os nodos que são argumentos de um operador de interseção são chamados de nodos-i e os nodos que são argumentos um operador de união são chamados de nodos-u. Seja X um nodo de A em S, define-se o conjunto que representa o nodo X por X, se este for um nodo-i, e pelo seu complemento, se o nodo X for um nodo-u.

Segundo [ROS 89], zona-i, zona-u e zona ativa de uma primitiva, que pertence ao sólido S, em uma árvore CSG positiva, são definidas por:

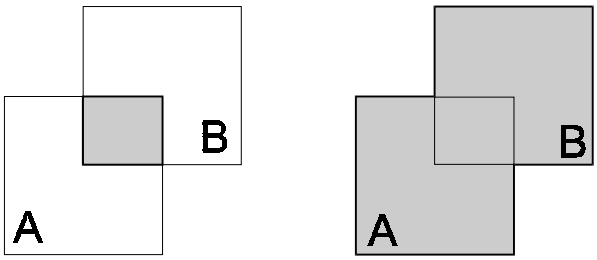
I (zona-i): É a interseção do conjunto universo W com todos os nodos-i de A em S.

U (zona-u): É a união de todos os nodos-u de A em S.

Z (zona-ativa): I – U.

Se não existirem nodos-i, então I = W, e se não existirem nodos-u, então U = ∅. Por exemplo, S é reduzida a uma simples primitiva A, então Z = W - ∅ = W, e no caso A ∪ B, Z = W – B = B^C.

A zona ativa de uma primitiva é a região onde mudanças nessa região afetam o objeto resultante. Como exemplo, na expressão A ∩ B, a zona ativa de A é B, e na expressão A ∪ B, a zona ativa de A é B^C. Na figura 6.6 (a), a zona ativa de A em S = A ∩ B é B, pois modificações em A do lado de fora de B não afetam S e modificações em A no lado de dentro de B afetam S. Similarmente, em 6.6 (b), ou seja, S = A ∪ B, a zona ativa de A é B^C.

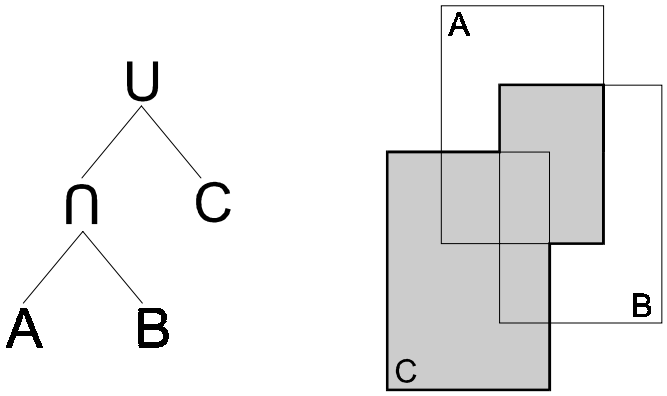


(a) (b)

FIGURA 6.6 – Zona ativa simples [ROS 89]

Em árvores mais complexas, a zona ativa é a interseção das zonas ativas da árvore. Por exemplo, em S = F ∪ C, onde F = A ∩ B, a zona ativa Z de A em S é a interseção da zona ativa C^C de F em S, com a zona ativa de A em F, porque mudanças para A irão afetar S somente se elas mudarem F na sua zona ativa.

Na figura 6.7 apresenta-se a árvore positiva da expressão (A ∩ B) ∪ C e o objeto gerado por esta expressão.



(a) (b)

FIGURA 6.7 – Árvore positiva e objeto S [ROS 89]

Ao ser feita uma análise da árvore positiva da figura 6.7, verifica-se que o nodo-i para a primitiva A é B e o nodo-u é C, como Z = I – U, para este caso, Z = B – C, pela definição da operação de diferença, a zona ativa de A é $B \cap C^C$. Ver figura 6.8 (a). Para a primitiva B o nodo-i é A e o nodo-u C, portanto a zona ativa de B é A – C, que corresponde a $A \cap C^c$. Ver figura 6.8 (b).

Para a primitiva C, pode-se perceber que não tem nodos-i na árvore, neste caso, como já foi discutido o I = W. O nodo-u da primitiva C é $A \cap B$, portanto a zona ativa para a primitiva Z é $W – (A \cap B)$, que corresponde a $W \cap (A \cap B)^c$. Ver figura 6.8 (c).

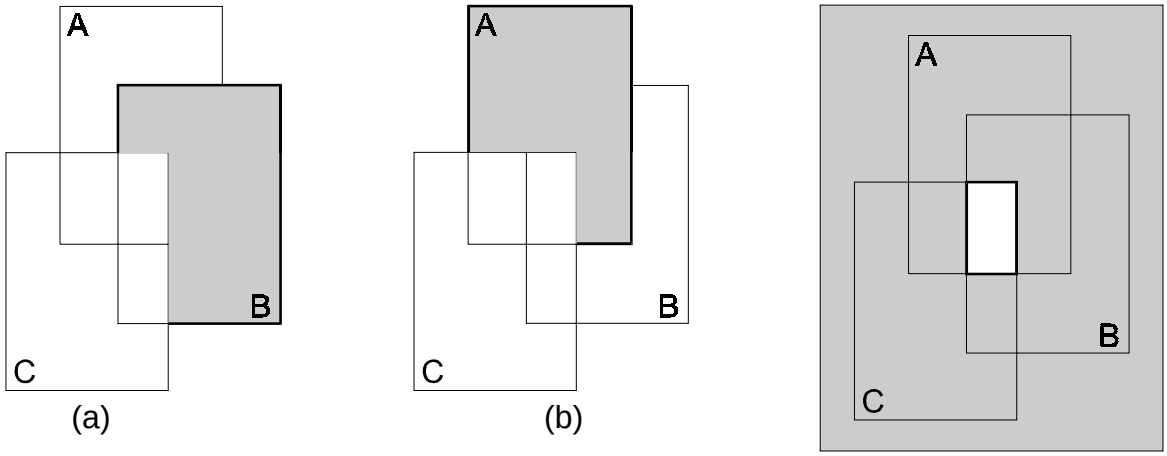


FIGURA 6.8 – Zonas ativas das primitivas A, B e C

As partes das primitivas que farão parte do sólido S são obtidas através da interseção da primitiva com sua zona ativa. Na figura 6.9, são apresentadas as partes das primitivas que irão compor do sólido S.

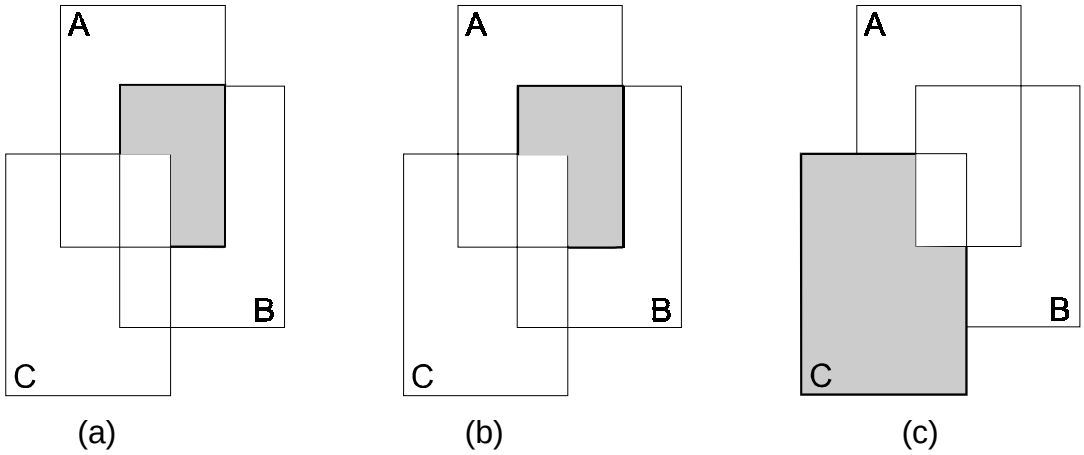


FIGURA 6.9 – Partes das primitivas que farão parte do objeto S

Como já se sabe o contorno de S é obtido pela união dos contornos das partes das primitivas que formam o objeto S. Para tanto, basta unir os contornos das figuras acima, que temos o objeto S correspondente. Ver figura 6.10.

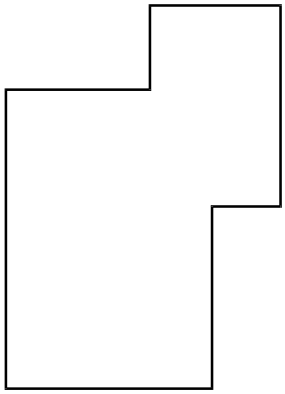


FIGURA 6.10 – Contorno do objeto S

Para a geração da sombra projetada verificam-se quais são as faces iluminadas das primitivas. Cada face possui um vetor normal que aponta para fora, se o ângulo entre o vetor normal da face e a posição da fonte de luz for menor ou igual a noventa graus, então a face é iluminada e consequentemente será projetada sobre o plano de projeção, caso contrário, quando o ângulo encontrado for maior que noventa graus, a face não é projetada, pois se encontra em uma região de sombra da cena.

Assim, cada primitiva possui um conjunto de faces iluminadas, a união destas faces representa a sombra da primitiva e a união das sombras das primitivas representa a sombra para toda cena.

6.2 Sistema Gráfico para Visualização de Sombras em Objetos Modelados por CSG

O Sistema gráfico para visualização de sombras em objetos modelados por CSG foi desenvolvido com o objetivo de apresentar as sombras projetadas de objetos pertencentes a uma cena que foi modelada através da Geometria Sólida Construtiva. A interface do protótipo pode ser vista na figura 6.11.

Legenda da Interface:

A – Menu utilizado para importação do arquivo gerado pelo POVCAD.

B – Área de visualização da câmera.

C – Controle dos deslocamentos da câmera.

D – Controle das rotações em torno dos eixos x, y e z.

E – Grade para exibição dos valores dos parâmetros da câmera.

F – Valores dos incrementos para os parâmetros câmera.

G – Botão que restaura os parâmetros iniciais da câmera.

H – Botão que atualiza os parâmetros da câmera, após a entrada pela grade identificada por F.

I – Exibe as coordenadas do posicionamento da fonte de luz.

J – Caixas de entradas para as coordenadas do plano de projeção.

K – Escolha das operações booleas.

L – Escolha da forma de visualização.

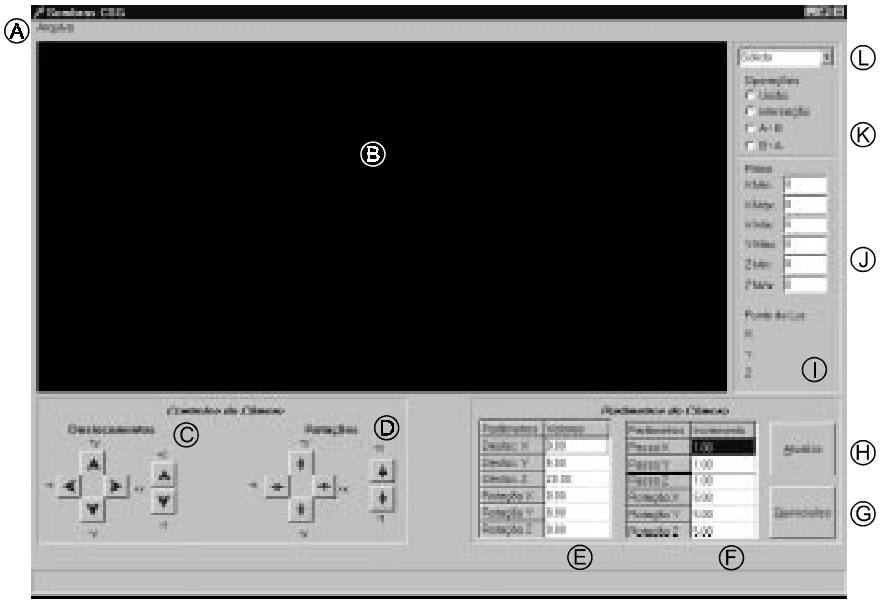


FIGURA 6.11 – Interface do protótipo

6.3 O Ambiente do Protótipo

O protótipo apresenta uma câmera sintética. A movimentação no espaço tridimensional está baseada na chamada esfera de movimentação, onde os objetos que estão sendo visualizados localizam-se sempre na mesma posição e a câmera sintética movimenta-se sobre a superfície de uma esfera imaginária, apontando sempre para o centro da esfera. O raio desta esfera é definido pelo usuário em tempo de execução, permitindo uma aproximação ou afastamento da câmera em relação aos objetos da cena.

A câmera sintética pode ser movimentada através de deslocamentos nos eixos x, y e z, o afastamento angular, que é o giro (esquerda, direita) realizado em torno do eixo y e a altura angular, que descreve o giro vertical (para cima ou para baixo) em torno do eixo x.

Também foram implementados alguns recursos para apresentar diferentes formas de visualização da cena, que são:

1 – Visualização aramada: com este modo de visualização, podem-se ver as arestas do objeto resultante, bem como as arestas que formam a sombra da cena. Este modo de visualização é o mais rápido.

2 – Visualização Sólida: neste modo de visualização, o objeto resultante da operação booleana é preenchido com o modelo de sombreamento *Smooth* da biblioteca OpenGl.

Segundo [WRI 2000], uma aresta tem dois vértices e cada um pode ter uma cor diferente. A cor da linha depende do modelo de sombreamento. O

sombreamento é definido simplesmente como uma transição suave entre uma cor e outra. Quaisquer dois pontos no espaço RGB podem ser conectados por uma linha reta, conforme ilustrado na figura 6.12.

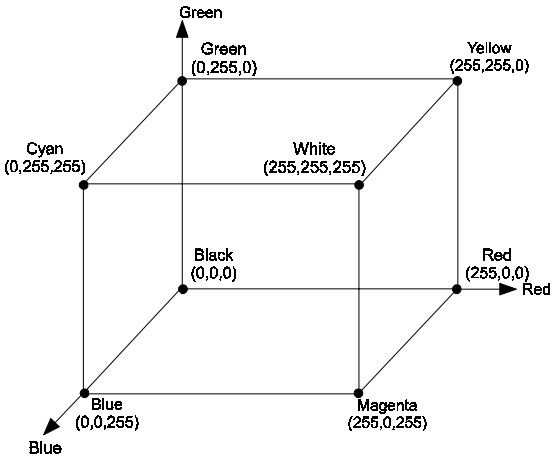


FIGURA 6.12 – Espaço de cor RGB [WRI 2000]

O sombreamento *Smooth* da OpenGl gera as cores ao longo de uma linha fazendo uma variação, através das cores do cubo do espaço RGB, de um ponto para outro. Na figura 6.13, a cor do cubo é mostrada com os pontos preto e branco, abaixo está uma linha com dois vértices, um preto e outro branco.

As cores selecionadas ao longo do comprimento da linha são iguais às cores ao longo do comprimento de uma linha do cubo, do canto preto para o canto branco. Este resultado é uma linha de progressão do preto até o branco.

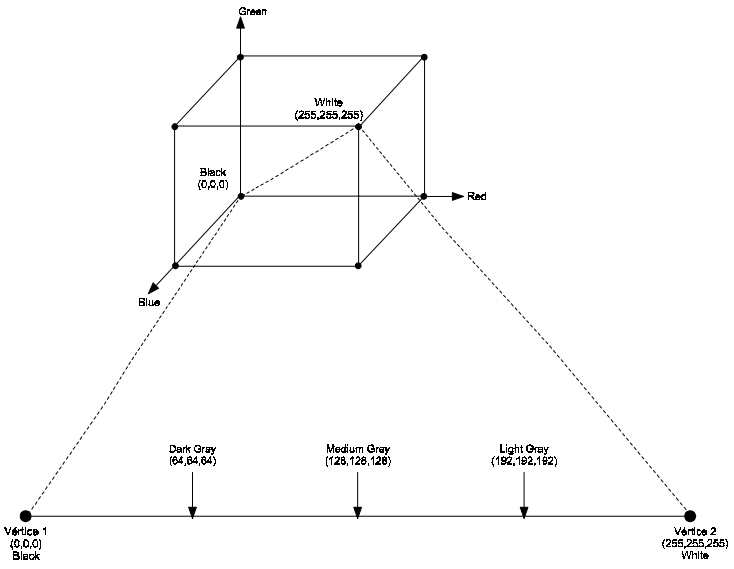


FIGURA 6.13 – Linha preenchida do preto para o branco [WRI 2000]

A fonte de luz no protótipo é puntiforme e localiza-se na cena em uma posição determinada pelo arquivo gerado pelo POVCAD.

6.4 Modularização do Protótipo

Este programa gráfico foi implementado com a finalidade de apresentar as sombras projetadas de objetos modelados a partir da técnica de modelagem de sólidos CSG (Geometria Sólida Construtiva).

O programa é basicamente composto de três partes:

a) Importação do arquivo gerado pelo POVCAD;

b) Obtenção do objeto resultante da operação booleana aplicadas sobre duas primitivas, e

c) Geração e exibição da sombra projetada da cena. O protótipo está dividido em módulos os quais se relacionam entre si, para que desta forma se chegue aos resultados pretendidos. Na figura 6.14 pode-se ver os módulo que compõe este protótipo.

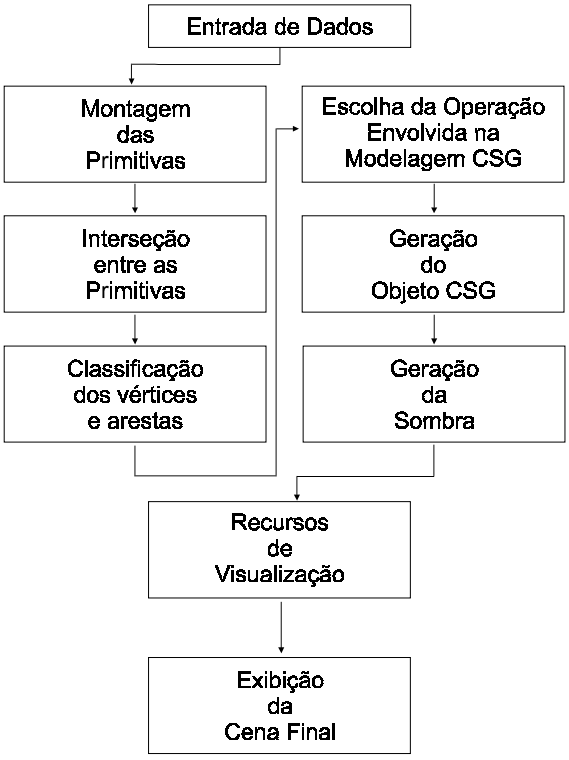


FIGURA 6.14 – Módulos do protótipo

Cabe salientar que a interface, o ambiente e os módulos apresentados acima, foram utilizados pelos dois protótipos. A diferença está no processo de seleção das arestas que farão parte do novo sólido e na geração das sombras.

6.5 Descrição das Primitivas e Estrutura de Dados

As primitivas utilizadas no protótipo são do tipo paralelepípedo, sendo descritas através de uma representação por contornos. O arquivo gerado pelo POVCAD fornece para cada primitiva as coordenadas do canto inferior esquerdo e canto superior direito. Com estas informações encontram-se todos os vértices do objeto.

Na figura 6.15, pode-se verificar todos os vértices do objeto, dentre estes os vértices 1 e 7 são os que representam as coordenadas fornecidas pelo arquivo do POVCAD.

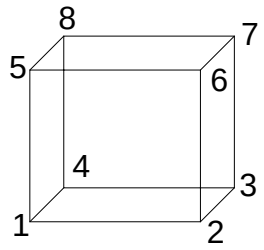


FIGURA 6.15 – Vértices inicias da primitiva

Com base na distribuição dos vértices dos objetos, conforme na figura acima, obtêm-se a seguinte descrição para as faces que formam a primitiva:

- Face inferior: vértices 4, 3, 2 e 1
- Face superior: vértices 5, 6, 7 e 8
- Face lateral esquerda: vértices 1, 4, 8 e 5
- Face lateral direita: vértices 2, 3, 7 e 6
- Face frontal: vértices 1, 2, 6 e 5
- Face traseira: vértices 4, 3, 7 e 8.

Como pode ser observado, usou-se o sentido anti-horário para a geração das faces.

6.5.1 Transformações Geométricas

Após os objetos estarem com os vértices armazenados, cada um em sua respectiva face, aplicam-se sobre os mesmos as transformações geométricas de translação, rotação e mudança de escala, parâmetros estes que também são obtidos do arquivo POVCAD.

Desta forma têm-se as primitivas estruturadas e prontas para serem tratadas pelo avaliador de contornos.

6.5.2 Estrutura de Dados

Como já foi comentado, nos capítulos dois e cinco desta dissertação, a partir das primitivas, um objeto CSG não pode criar novas faces, pode eliminar faces ou então fazer modificações nas faces existentes. Durante o processo de modelagem do objeto CSG, dependendo da situação, pode ocorrer a necessidade de inclusão de novos vértices ou a remoção de vértices

existentes. Diante desta situação optou-se por fazer uso de uma estrutura de dados dinâmica, mais especificamente, com listas encadeadas, as quais permitem o trabalho com um volume de dados indeterminado e possibilitam a manipulação dos elementos da lista de forma segura e rápida, evitando a necessidade de uma reordenação. Logo abaixo se apresentam as descrições das listas utilizadas.

Cada objeto possui uma lista encadeada, onde se armazenam as faces do mesmo. As faces possuem uma fila circular duplamente encadeada, responsável pelo armazenamento dos vértices de cada face.

Lista de objetos: esta lista possui um registro para cada objeto da cena. Cada registro armazena as seguintes informações:

- Um índice, para identificar o objeto;
- Valores máximo e mínimo das coordenadas x,y,z do objeto, identificando o envelope do mesmo;
- Os valores para mudança de escala, rotação e translação do objeto, informações estas que são obtidas do arquivo gerado pelo POVCAD.
- Lista de vértices do objeto;
- Uma lista de faces;
- Ponteiro para o próximo objeto da lista;

Lista de vértices: as listas de vértices que formam os objetos são geradas a partir das informações obtidas do arquivo gerado pelo POVCAD. Durante a montagem das mesmas, é feita uma ordenação dos vértices por face, obedecendo o sentido anti-horário.

Cada registro da lista de vértices armazena os seguintes dados:

- Um índice de identificação do vértice;
- As coordenadas x,y,z do vértice no sistema de referência do universo;
- Um campo para a classificação do vértice, que pode ser IN, ON ou OUT;
- Um ponteiro para o próximo vértice;
- Um ponteiro que aponta para o vértice anterior;

Lista de faces: nesta lista estão armazenadas as informações referentes as faces de cada objeto envolvido na cena. As seguintes informações fazem parte desta estrutura:

- Um índice para identificar a face;
- Uma lista vértices, ou seja, a lista dos vértices que formam a face;
- Uma lista das arestas-tentativa, que fazem parte da face;
- Pontos máximo e mínimo para as coordenadas x, y e z, ou seja, o envelope da face;
- O vetor normal da face;

- Um *flag* para identificar se a face é visível ou não do ponto de vista da fonte de luz;
- Lista de sombras, onde armazenam-se as coordenadas dos vértices que foram projetados sobre o plano de projeção e identificam a sombra da face;
- Lista de triângulos, que é obtida após o processo de triangularização da face;

A figura 6.16 representa graficamente a estrutura de dados utilizada nos protótipos.

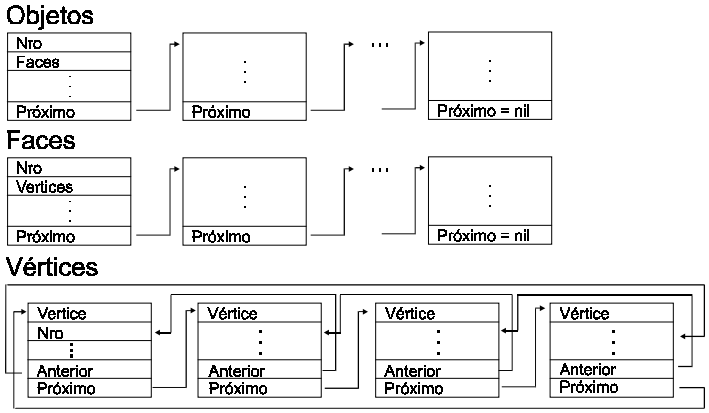


FIGURA 6.16 – Representação da estrutura de dados

6.6 Avaliação de Contornos

Como já foi mencionado nos capítulos dois e cinco, o processo de avaliação de contornos (*boundary evaluator*) é o processo que exige mais tempo e memória do computador.

As faces que formam os objetos podem ser modificadas ou destruídas, mas não criadas. Em cada interseção que ocorrer entre duas faces, uma nova aresta poderá ser criada. O processo de avaliação de contornos é quem vai determinar onde novas arestas e vértices devem ser criados e onde arestas e vértices devem ser destruídos.

Novas arestas são criadas onde as superfícies dos dois objetos que se combinam, se interceptam. É necessário calcular estas interseções e então determinar, através da classificação por inclusão, quais são os segmentos que farão parte do novo objeto.

Para a realização deste processo é necessário a execução dos seguintes passos:

1 – Detectar a interseção entre os objetos

Como pode ser visto na estrutura que descreve a lista de objetos na seção 6.5.2, cada objeto possui um envelope que o engloba. Com o uso destas informações é possível verificar se existe a interseção entre os objetos, testando o envelope de uma primitiva contra o envelope da outra primitiva.

Os testes para esta verificação já foram comentados na seção 5.3. Se ocorrer a interseção entre os envelopes, pode ser que os objetos se

interseccionam ou não. Caso seja verificada a existência de interseção, torna-se necessária uma análise mais detalhada, ou seja, verifica-se a possibilidade de interseções entre as faces dos objetos, também fazendo uso de envelopes, porém agora em nível de faces.

2 – Calcular a interseção entre as faces dos objetos

Ao ser encontrada uma interseção entre os envelopes das faces, é necessário computar a interseção entre as faces através da análise das distâncias dos vértices de uma face em relação ao plano suporte da outra face, como está descrito no capítulo cinco na seção 5.4. Este procedimento originará a reta suporte da interseção entre as faces.

3 – Calcular a interseção entre as arestas

Esta etapa baseia-se no que foi descrito na seção 5.4 e consiste em testar as interseções entre a reta suporte gerada no passo anterior (passo 2), e as arestas que formam as faces em questão, gerando assim a segmentação da reta suporte e originando a aresta-tentativa.

Cada ponto de interseção que for calculado entre as arestas que formam a face e a reta suporte, origina um novo vértice, que segmenta a aresta da face. Este vértice deve ser incluído na lista de vértices da face, sendo classificado como ON.

Após a aresta-tentativa ser encontrada, esta é armazenada na lista de arestas-tentativa das duas faces que a originaram.

4 – Classificar os vértices

A classificação dos vértices é realizada primeiramente, com a classificação de todos os vértices de uma face em relação a outra e vice-versa. Cada vértice é classificado como IN (quando se encontra dentro da outra face), ON (se o vértice se encontra sobre um dos segmentos da outra face) ou OUT (quando o vértice se encontra fora da outra face). O processo de classificação dos vértices utilizados no protótipo encontra-se descrito com detalhes na seção 5.5, do capítulo cinco desta dissertação.

5 – Selecionar os segmentos das arestas que farão parte do novo sólido

Após ser realizada a classificação de todos os vértices é preciso fazer a seleção dos segmentos que farão parte das faces do novo sólido.

Neste ponto existe uma diferença entre os dois protótipos como se descreve a seguir.

No protótipo A, que representa a forma clássica de geração de sombras, a escolha dos segmentos de aresta que farão parte do novo sólido está baseada nas classificações das arestas e na operação envolvida na modelagem desse novo sólido, conforme foi discutido na seção 5.7.

Para a escolha das arestas do novo sólido leva-se em consideração a operação booleana usada na modelagem do objeto e selecionam-se as arestas correspondentes a esta, como é mostrado abaixo.

União: Todas as arestas com classificação ON e OUT

Interseção: Todas as arestas com classificação IN e ON+

Diferença: Todas arestas OUT do primeiro objeto e todas arestas IN do segundo objeto

O processo de seleção das arestas na prática está baseado em percorrer a lista de objetos. Como cada objeto possui uma lista de faces, percorre-se esta lista, por sua vez, a lista de faces possui uma lista de vértices, que também é percorrida. Neste momento são feitas as seleções das arestas que farão parte do novo sólido, com base na classificação dos vértices.

As arestas selecionadas e as arestas-tentativa de cada face, originam as novas faces do novo sólido, estas faces são armazenadas em um novo objeto, que será utilizado para a geração da sombra e renderização da cena.

Como foi descrito nos capítulos três e quatro, existe a possibilidade de se realizar conversões em operações booleanas através do Interrelacionamento, com de operações de união, interseção e complemento.

No protótipo B, a determinação do contorno do objeto final foi realizada através das zonas ativas de cada primitiva da árvore CSG, como discutido no capítulo três, mais especificamente com base nos conceitos apresentados na seção 3.7.11.

Após serem encontradas as zonas ativas das primitivas, é necessário realizar a união dos contornos destas zonas para obtenção do contorno do sólido pretendido.

O processo de determinação do contorno do objeto CSG pelas zonas ativas, envolve somente operações de união e interseção, sendo necessário realizar a conversão da árvore CSG para a sua forma positiva. As operações de diferença são transformadas em combinações de interseção e complemento, de forma que continuem mantendo o mesmo resultado da operação de diferença.

Como esta técnica necessita da utilização do complemento, na seção seguinte descreve-se como foi tratado este problema na implementação do protótipo.

6.6.1 Complemento

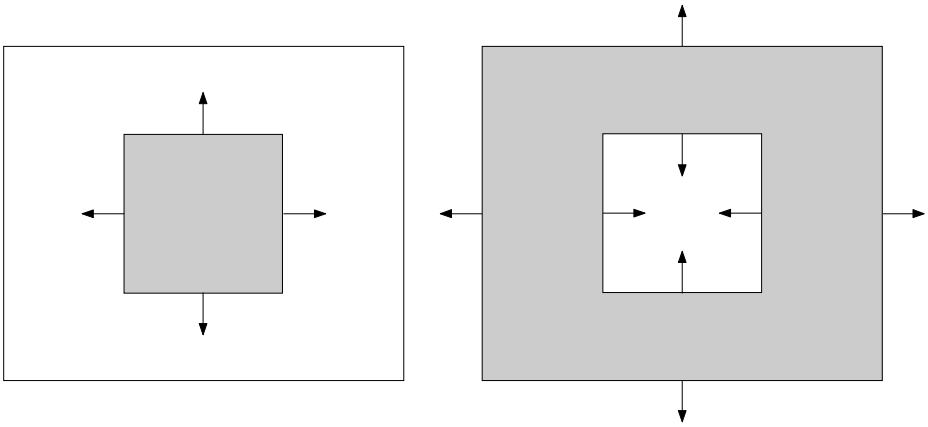
O complemento de um objeto A, é o conjunto de todos os elementos (pontos) do universo que não fazem parte de A.

Para a implementação do complemento dos objetos modelados na cena, o critério de maior importância é o vetor normal de cada face, que neste caso aponta para fora.

A determinação do complemento de um objeto pode ser obtida através da inversão dos vetores normais das faces, pois se o vetor normal a cada face aponta para fora do objeto, a inversão do sentido do vetor normal faz com que o exterior do objeto passe a ser interior no complemento do objeto.

Na ilustração da figura 6.17 pode ser observado um objeto exemplo (a), com os vetores normais apontando para fora e na figura (b), o complemento deste objeto (os vetores normais das faces estão apontando para o lado de dentro).

O complemento de um objeto possui duas fronteiras, a fronteira do universo e a fronteira do complemento. Para os casos estudados, o contorno do universo não foi considerado.



(a) – Objeto (b) – Complemento do objeto

FIGURA 6.17 – Objeto Sólido e o Complemento do Objeto.

Logo abaixo se apresenta o algoritmo que foi utilizado na obtenção dos complementos.

```
Enquanto Faces_Objeto <> nil Faça
    Início
      Face_Objeto.Normal.x := (Face_Objeto.Normal.x * -1);
      Face_Objeto.Normal.y := (Face_Objeto.Normal.y * -1);
      Face_Objeto.Normal.z := (Face_Objeto.Normal.z * -1);
      Repetir
        Se Lista_Vértices.Vértice.Classificação = 'OUT' Então
            Lista_Vértices.Vértice.Classificação = := 'IN'
        Senão Se Lista_Vértices.Vértice.Classificação = 'IN' then
                    Lista_Vértices.Vértice.Classificação := 'OUT';
      Lista_Vértices := Lista_Vértices.Proximo;
      Até Lista_Vértices.Inicio = Verdadeiro;
      Faces_Objeto := Faces_Objeto.Proximo;
    Fim
```

Como pode ser visto no algoritmo acima a inversão do vetor normal da face é simples, basta multiplicar os valores atuais por –1. No mesmo algoritmo realizam-se as modificações nas classificações dos vértices.

Durante a obtenção do complemento, a execução ocorre sobre os vértices do objeto, neste ponto os mesmos já foram classificados pelo avaliador de contorno. Para fazer esta modificação leva-se em consideração os pontos OUT e IN, pois os pontos ON, que são pontos de fronteira do objeto, continuarão sendo pontos de fronteira no complemento. Os pontos classificados como OUT devem ser alterados para IN, pois se o vértice está fora do objeto, logicamente estará dentro do complemento. Esta mesma regra é aplicada para os vértices IN, que são transformados em vértices OUT, pois se estão dentro do objeto, logicamente estarão fora do complemento.

6.6.2 Escolha dos Elementos que Farão Parte do Objeto Final

Após a execução do complemento, o próximo passo é a seleção dos elementos que farão parte do objeto final. Esta seleção de arestas está baseada nas operações booleanas envolvidas na modelagem do objeto. Ressalta-se que serão utilizadas somente operações de união, interseção e complemento, pois como já foi dito anteriormente as operações de diferença são convertidas.

Se a operação escolhida for união, são selecionadas todas as arestas que estão classificadas como ON ou OUT. Caso a operação seja de interseção as arestas selecionadas serão aquelas classificadas como IN ou ON+. Conforme já foi discutido na seção 5.7.

Se a operação for a diferença, primeiramente aplica-se a operação do complemento sobre o segundo objeto e depois a operação de interseção, sobre os dois objetos.

Abaixo se apresentam os algoritmos que realizam as operações de união e interseção.

Operação de união:

```
Enquanto Faces_Objeto <> nil Faça
  Início
    Repetir
       Se ((Classificação_Aresta = 'OUT') ou
            (Classificação_Aresta = 'ON+') ou
            (Classificação_Aresta = 'ON-')) então
                Seleciona_Aresta;
       Lista_Vértices.Próximo;
    Até Lista_Vértices.Inicio = Verdadeiro;
    Se (Face_Iluminada) Então
        Gera_Sombra_Face;
    Faces_Objeto.Próximo;
  Fim
```

Operação de Interseção:

```
Enquanto Faces_Objeto <> nil Faça
  Início
    Repetir
       Se ((Classificação_Aresta = 'IN') ou
             (Classificação_Aresta = 'ON+')) então
                Seleciona_Aresta;
       Lista_Vértices.Próximo;
    Até Lista_Vértices.Inicio = Verdadeiro;
    Se (Face_Iluminada) Então
        Gera_Sombra_Face;
    Faces_Objeto.Próximo;
  Fim
```

Como descrito no capítulo três, a zona ativa de uma primitiva é a região que ao sofrer mudanças afeta o objeto final resultante. Para a obtenção desta

zona são usadas as mesmas operações booleanas caracterizadas anteriormente. A parte da primitiva que irá compor o sólido é obtida pela interseção da primitiva com a sua zona ativa.

O protótipo pode ser estendido para a modelagem de sólidos mais complexos. Primeiramente encontra-se a árvore positiva para a expressão CSG. A transformação da árvore original para a sua forma positiva pode ser executada por uma função que percorre toda árvore, obedecendo aos critérios estabelecidos na seção 3.7.11.

Sempre que um operador de diferença for encontrado, substituí-se o mesmo por um operador de interseção, sendo complementado o lado direito da árvore. Se a complementação cair sobre um nodo operação, então todos os nodos que estão abaixo deste serão afetados, isto é, os nodos operadores de união são substituídos por operadores de interseção e os de interseção são substituídos por união. Se os nodos são primitivas, então estes devem ser complementados.

A determinação da zona ativa para primitivas em árvores mais complexas é obtida através da interseção da primitiva com a interseção das zonas ativas da árvore.

6.7 Geração da Sombra

A última etapa do protótipo é a geração da sombra da cena. Tanto para o protótipo A como para o protótipo B aplicou-se o mesmo algoritmo para a geração de sombras, ou seja, a projeção dos vértices das faces iluminadas.

Para verificar se uma face é iluminada ou não, calcula-se o ângulo entre o vetor normal da face e a posição da fonte de luz, utilizando a equação apresentada na seção 4.2. Se este ângulo for menor ou igual a 90 graus, então a face é iluminada e seus vértices são projetados, conforme descrito na seção 4.1, caso contrário ela não é iluminada, e consequentemente não será projetada.

A seguir apresenta-se o algoritmo que realiza esta operação.

```
Enquanto Faces_Objeto <> nil Faça
  Início 1
    Se Faces_Objeto.Visível = Verdadeiro Então
      Início 2
        Repetir
            Novo_Vértice := Projeta_Vértice_Plano;
            Armazena_Novo_Vértice_Lista_Sombra_Face;
            Lista_Vértices.Proximo;
        Até Lista_Vértices.Inicio = Verdadeiro
      Fim 2
    Faces_Objeto.Próximo;
  Fim 1
```

Para a execução deste procedimento, basta verificar quais são as faces iluminadas de cada objeto. Todas as faces iluminadas de um objeto têm seus vértices projetados. A união destas projeções origina a sombra do objeto e a união das sombras dos objetos representa a sombra da cena.

A projeção dos vértices é feita com base nas equações 4.1 e 4.2, apresentadas na seção 4.1.

Estes novos vértices geram polígonos e são armazenados em uma lista, esta lista representa a sombra da face. Ao ser concluída a execução deste procedimento para todas as faces do objeto, obtêm-se a sombra do objeto.

O ponto que diferencia os dois protótipos é que no protótipo A, o objeto resultante da expressão é modelado e posteriormente verificam-se as faces iluminadas, que consequentemente serão projetadas (silhueta). No protótipo B a verificação das faces iluminadas é realizada durante a avaliação dos contornos do objeto, caso a face seja iluminada, são projetados os vértices obtidos da interseção da primitiva com sua zona ativa, desta forma ao ser concluída a avaliação de contornos também se têm as sombras de cada face.

Ao ser realizada a renderização do objeto resultante realiza-se também a renderização das sombras, originando a exibição da cena. A renderização pode ser aramada ou sólida. Na representação aramada são exibidas as arestas do sólido e as arestas da sombra e na renderização sólida, como foi comentado na seção 6.3 os polígonos que representam as faces do objeto e as sombras são exibidos de forma preenchida, da forma *Smooth*.

6.8 Considerações Finais

Neste capítulo foram discutidas as técnicas empregadas na implementação do protótipo. Foram apresentados a modularização do sistema, a interface do protótipo, descrição das primitivas, a estrutura de dados, os passos executados pelo avaliador de contornos e a geração das sombras.

No próximo capítulo serão mostrados e analisados os resultados obtidos.

7 Resultados Obtidos e Análise

Este capítulo tem por finalidade apresentar os resultados obtidos através da implementação dos protótipos, com base nos algoritmos descritos no capítulo três desta dissertação, além de uma análise desses. Os testes foram realizados através das operações booleanas de união (A ∪ B), interseção (A ∩ B) e diferença (A – B e B – A), aplicadas sobre duas primitivas do tipo paralelepípedo. Apresentam-se também as sombras correspondentes a cada uma das cenas.

Além desses resultados, são feitas algumas considerações sobre a complexidade dos algoritmos. Posteriormente, são descritas algumas limitações deste estudo bem como os problemas encontrados durante a implementação.

7.1 Apresentação de Resultados

Um dos motivos que despertou o interesse por este estudo foi o de encontrar poucas publicações sobre a aplicação de CSG na geração de sombras. Durante o levantamento bibliográfico encontrou-se somente uma publicação específica sobre o assunto, o artigo de Frederik W. Jansen [JAN 91], publicado na *Eurographics '90*.

Nos resultados apresentados pelos protótipos foram utilizados os arquivos gerados pelo POVCAD, sempre levando em consideração dois objetos com formato de paralelepípedos. Sobre estes foram aplicadas as operações booleanas regularizadas de união, interseção e diferença.

No início do trabalho investigou-se a possibilidade de gerar a sombra da cena com base nas operações booleanas, aplicadas sobre a modelagem do objeto CSG. Para cada primitiva, seria criado um polígono da sombra e, posteriormente sobre estes polígonos aplicar-se-iam as mesmas operações envolvidas na modelagem do objeto CSG. Depois de investigar e fazer alguns testes pode-se constatar que a geração de sombras, se realizada desta forma, nem sempre apresenta os resultados corretos. Especialmente para os casos das primitivas estarem disjuntas e devido à posição da fonte de luz as suas sombras estarem sobrepostas, com esse método não se obtém a sombra correta da cena. Os motivos pelos quais se chegou a esta conclusão estão detalhados no capítulo quatro.

Com base nestes resultados procurou-se uma outra forma de solucionar o problema. Buscou-se utilizar somente operações de união e complemento, que seguindo algumas propriedades da teoria dos conjuntos, apresentam os mesmos resultados das operações de interseção e diferença. Este método também não pôde ser usado para os mesmos casos e pelos mesmos motivos que na alternativa anterior.

A solução do problema está baseada na geração da sombra com a projeção das faces iluminadas, isto é, projetar os vértices das faces visíveis do ponto de vista da fonte de luz, após a modelagem do objeto. Ao ser feita a projeção dos vértices das faces iluminadas, sobre o plano de projeção, geram-

se os polígonos de sombra de cada uma das primitivas. A sombra da cena é obtida pela união das sombras das várias primitivas que compõem o sólido.

Com o intuito de enriquecer o trabalho desenvolveu-se um exemplo real de uma cena modelada com operações CSG e suas sombras. A utilização deste exemplo não tem por objetivo validar os resultados da pesquisa, mas sim procurar encontrar uma forma mais clara de ilustrar os resultados obtidos com a aplicação desta técnica.

Nas figuras 7.1, 7.2, 7.3 e 7.4 podem-se ver algumas fotografias que foram tiradas da cena real. As operações booleanas regularizadas são aplicadas sobre a modelagem de um objeto CSG e envolvem operações de união, interseção e diferença sobre dois cubos.

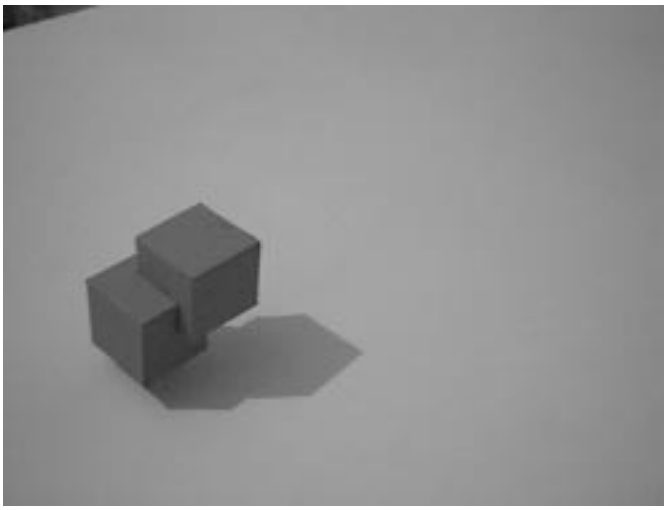


FIGURA 7.1 – Cena real – Operação de união

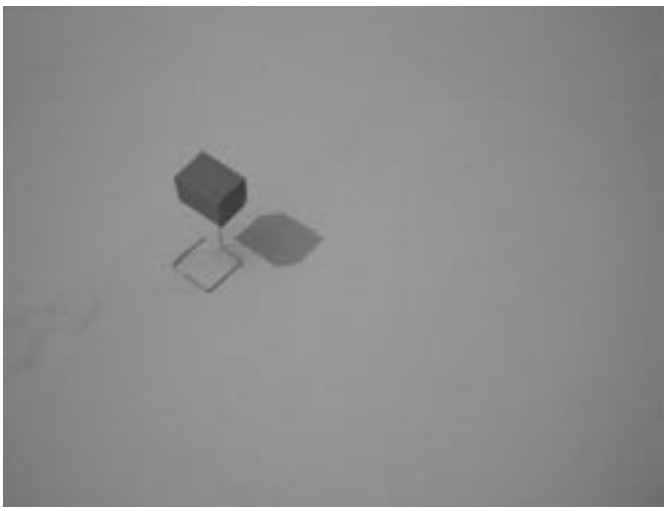


FIGURA 7.2 – Cena real – Operação de interseção

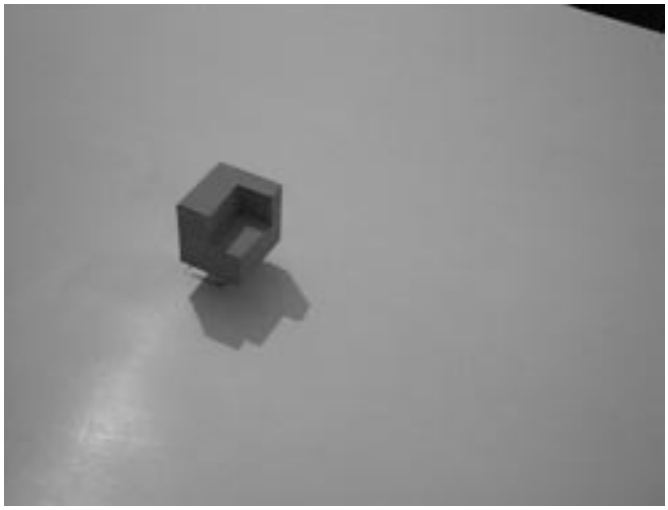


FIGURA 7.3 – Cena real – Operação de diferença (A – B)

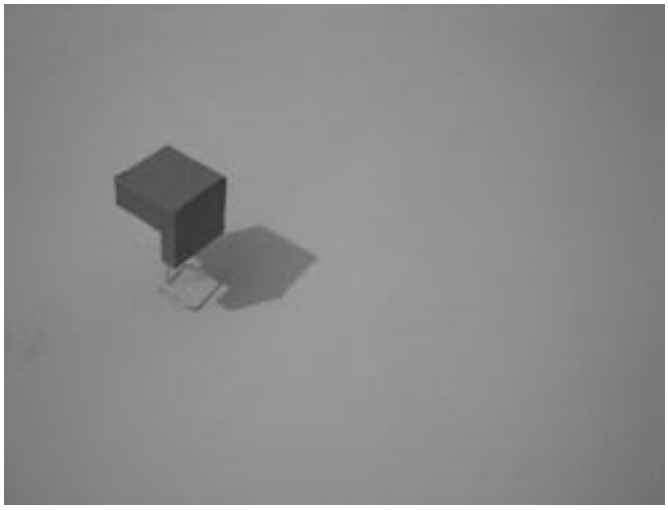


FIGURA 7.4 – Cena real – Operação de diferença (B – A)

Nas figuras abaixo são apresentadas as imagens que representam a mesma cena, modelada através dos protótipos. O protótipo A, que usa a projeção clássica das faces, apresenta os resultados da exibição sólida nas figuras 7.5, 7.9, 7.13 e 7.17 e resultados na forma aramada nas figuras 7.6, 7.10, 7.14 e 7.18.

Nas figuras 7.7, 7.11, 7.15 e 7.19 são apresentadas imagens das representações sólidas e as figuras 7.8, 7.12, 7.16 e 7.20 apresentam representações aramadas, todas as cenas foram modeladas pelo protótipo B, que faz uso das zonas ativas das primitivas.

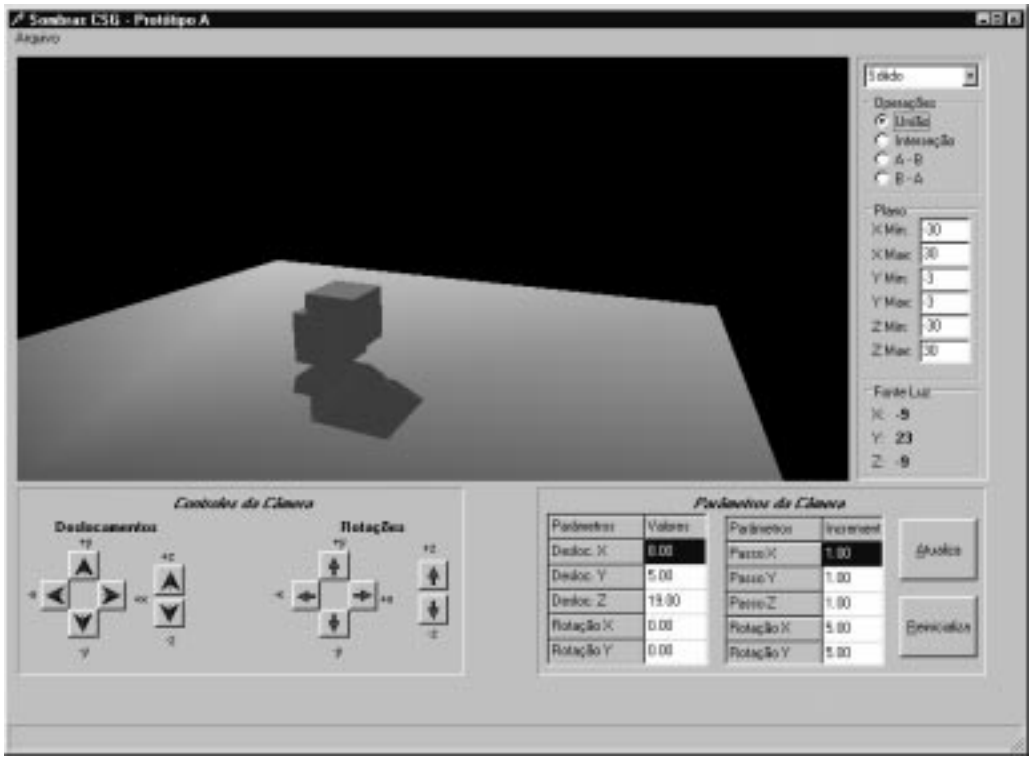


FIGURA 7.5 – Protótipo A – operação de união – representação sólida

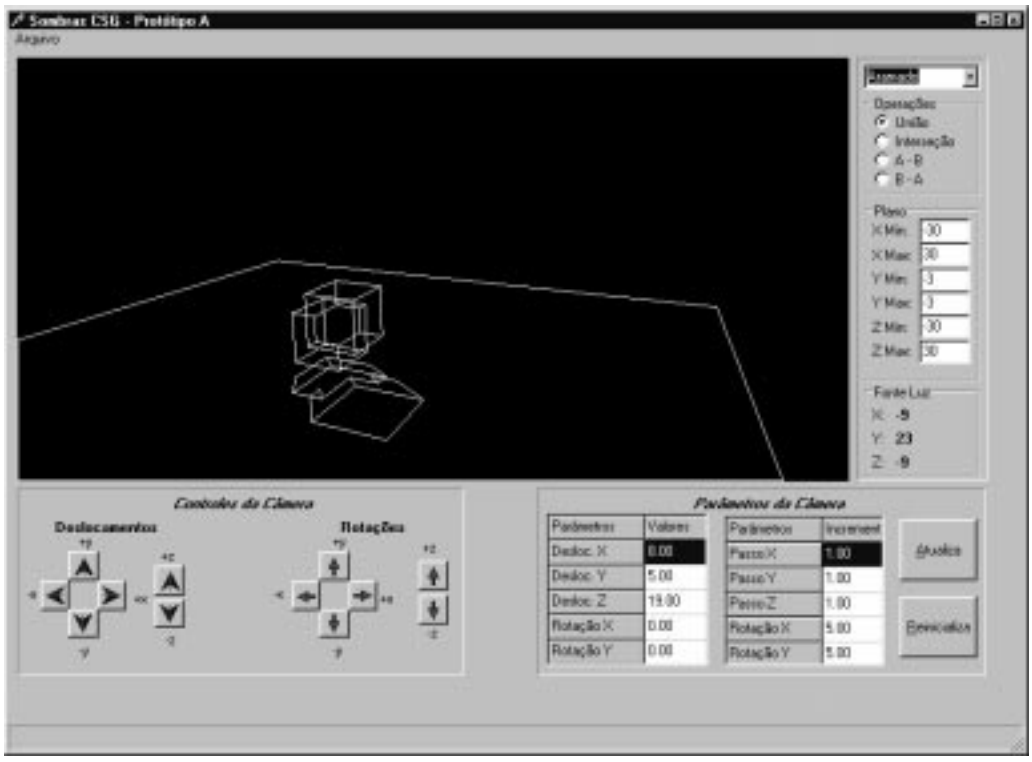


FIGURA 7.6 – Protótipo A – operação de união – representação aramada

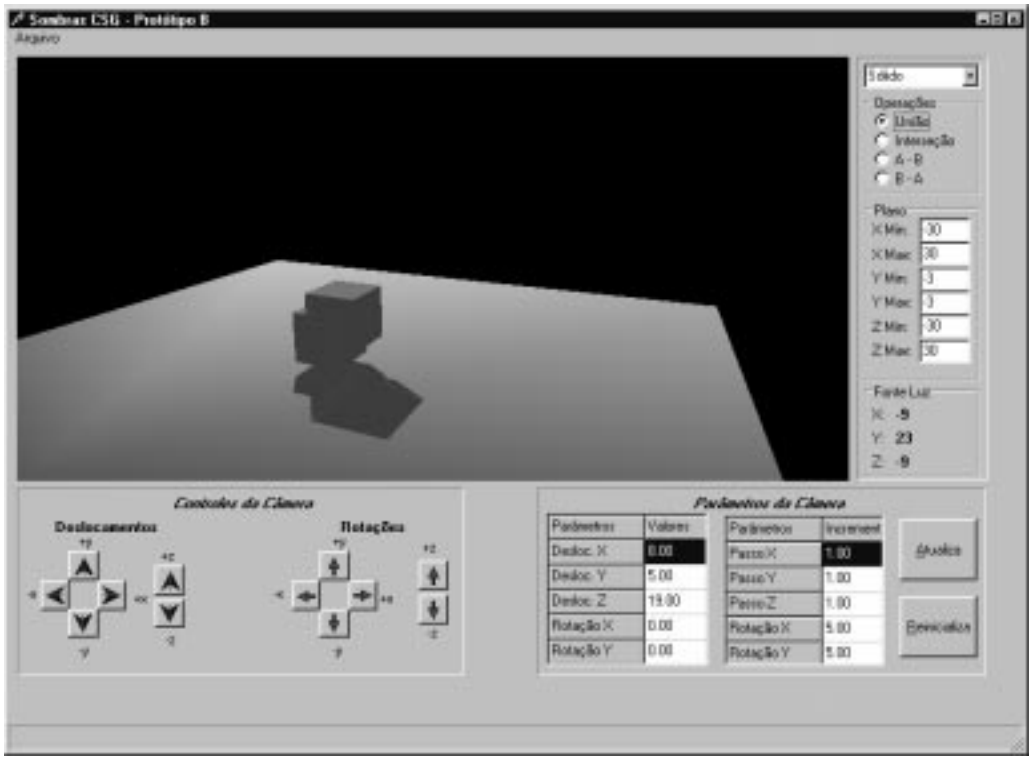


FIGURA 7.7 – Protótipo B – operação de união – representação sólida

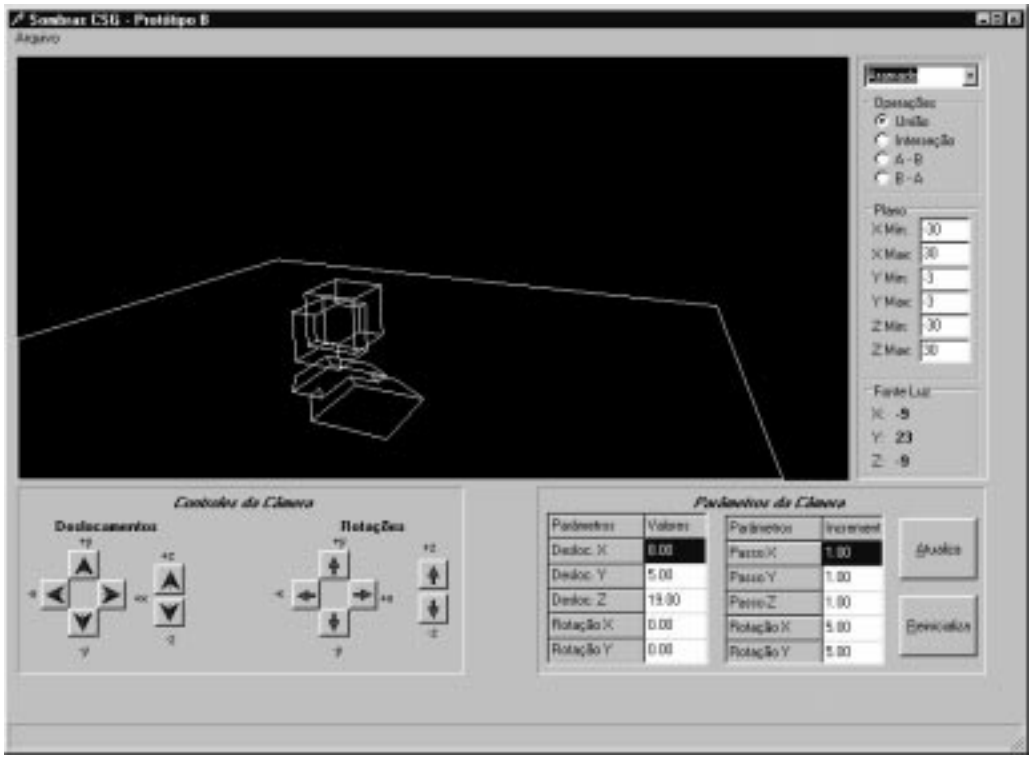


FIGURA 7.8 – Protótipo B – operação de união – representação aramada

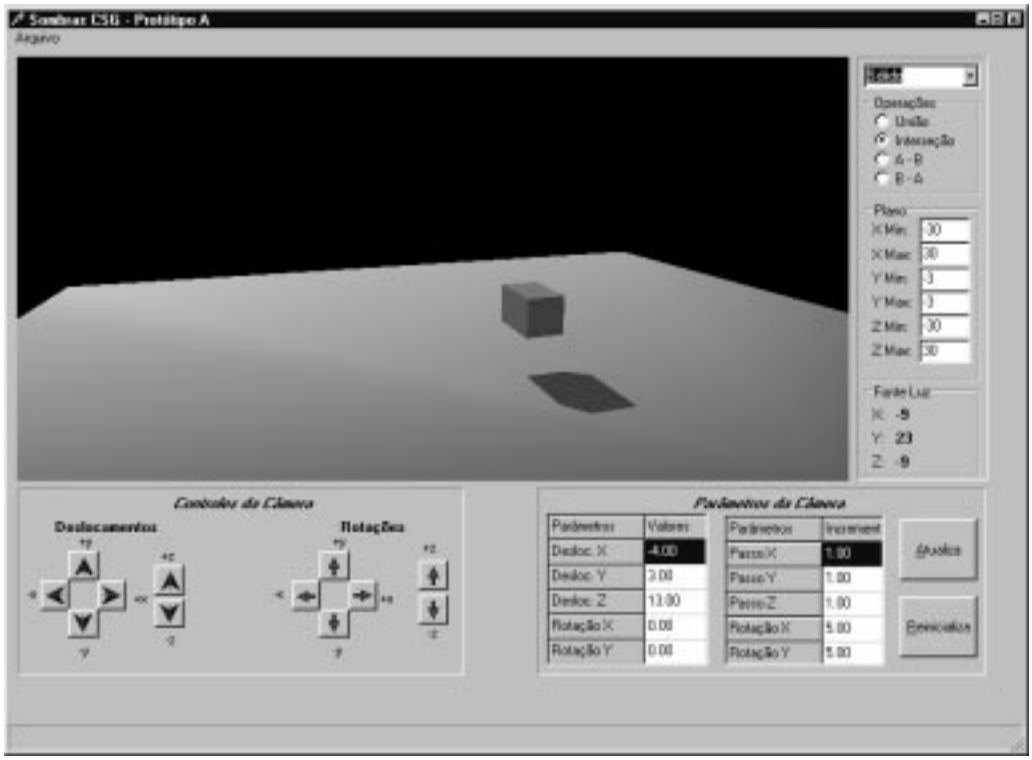


FIGURA 7.9 – Protótipo A – operação de interseção – representação sólida

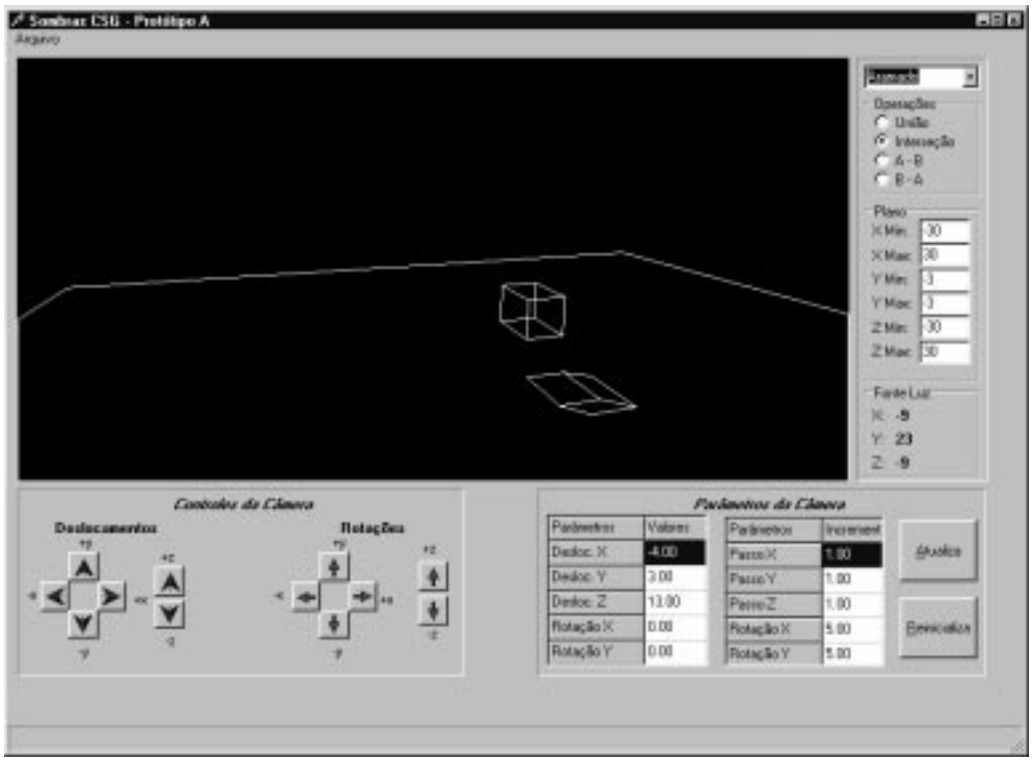


FIGURA 7.10 – Protótipo A – operação de interseção – representação aramada

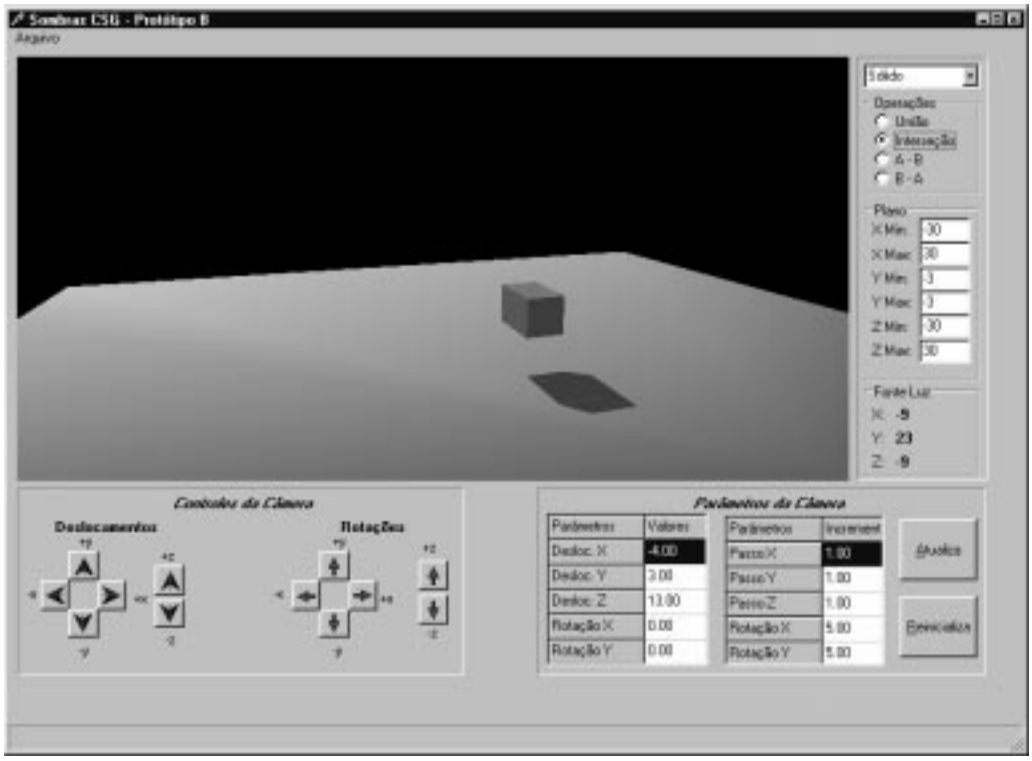


FIGURA 7.11 – Protótipo B – operação de interseção – representação sólida

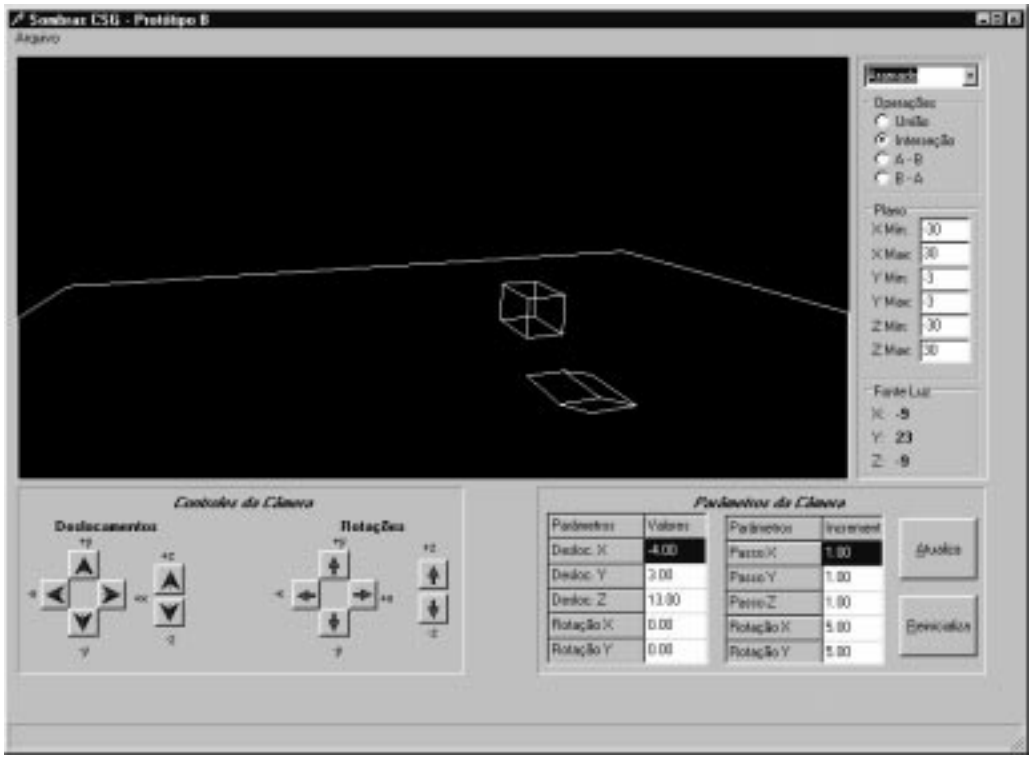


FIGURA 7.12 – Protótipo B – operação de interseção – representação aramada

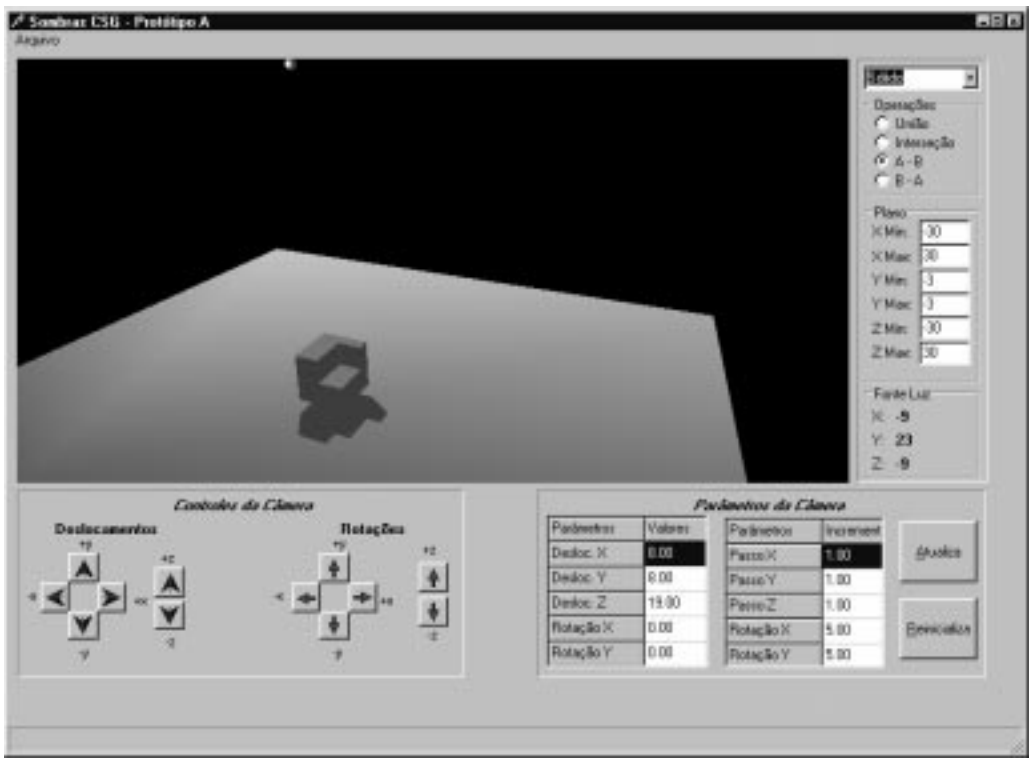


FIGURA 7.13 – Protótipo A – operação de diferença (A – B)
representação sólida

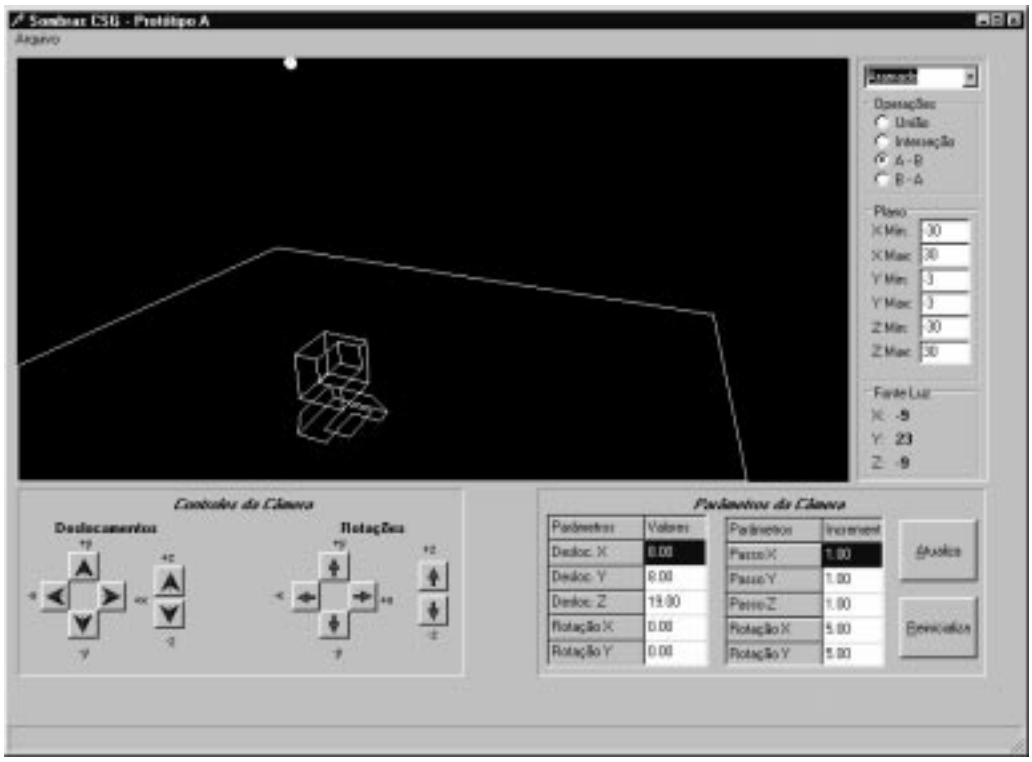


FIGURA 7.14 – Protótipo A – operação de diferença (A – B)
representação aramada.

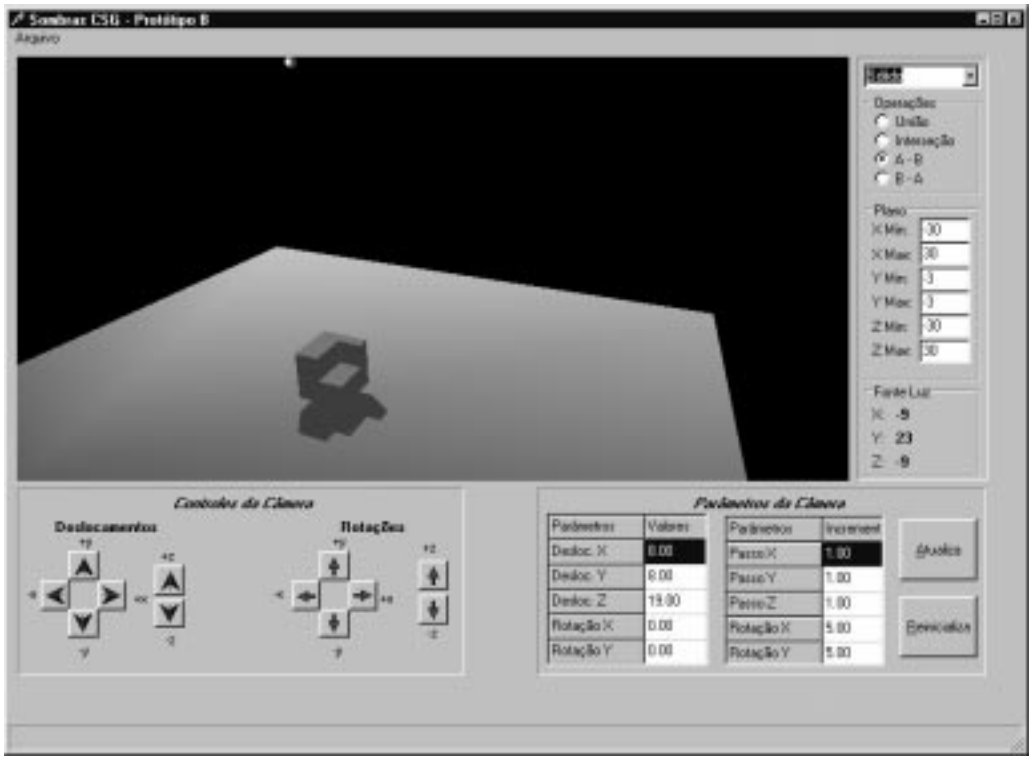


FIGURA 7.15 – Protótipo B – operação de diferença (A – B)
representação sólida

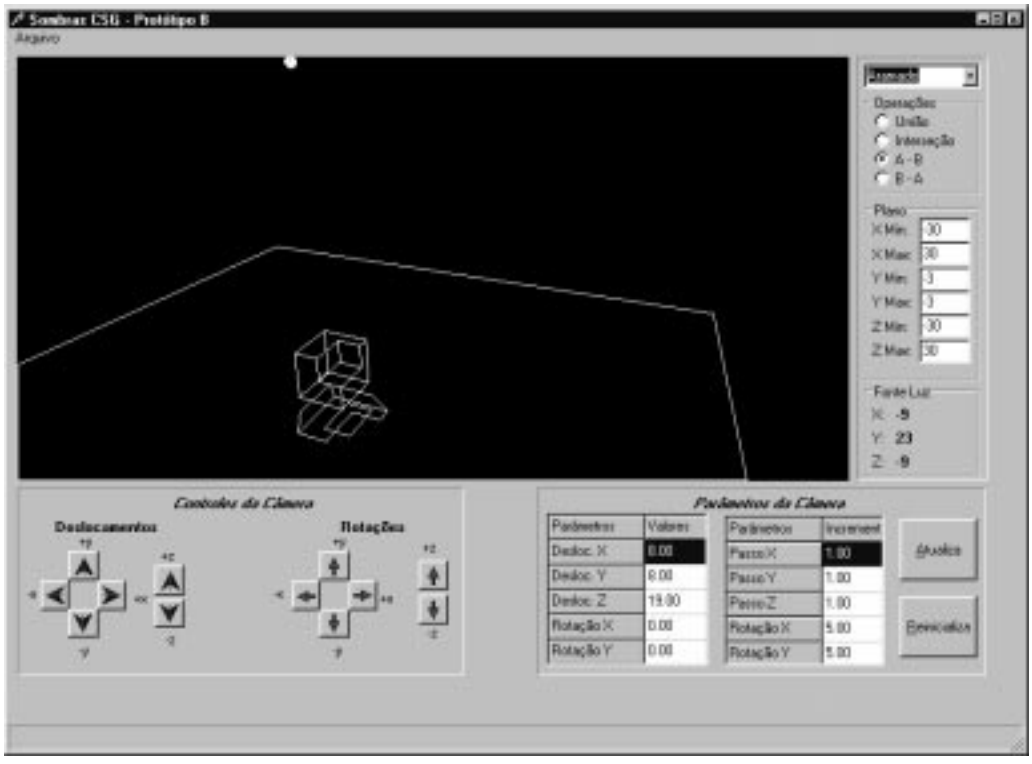


FIGURA 7.16 – Protótipo B – operação de diferença (A – B)
representação aramada

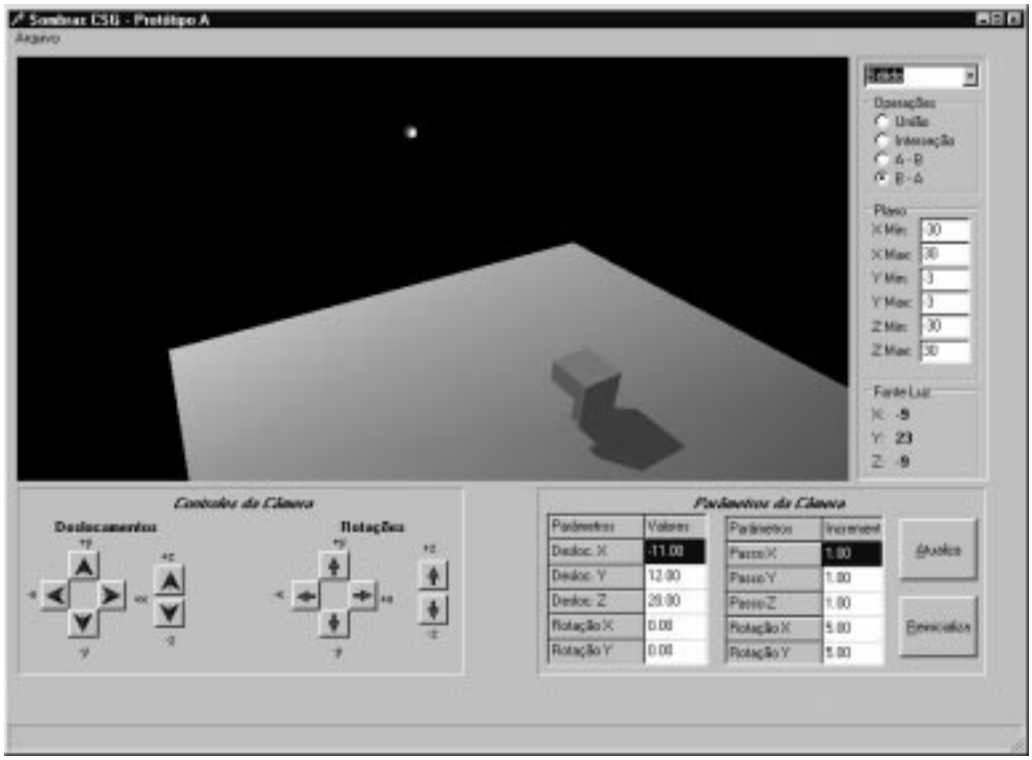


FIGURA 7.17 – Protótipo A – operação de diferença (B – A)
representação sólida

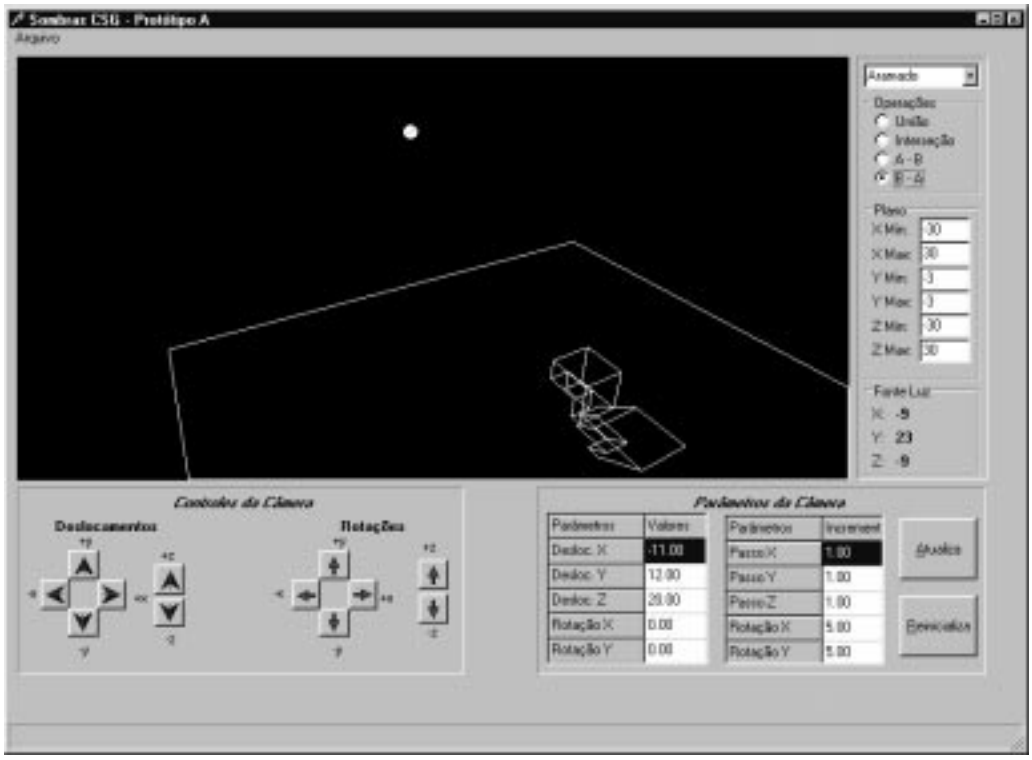


FIGURA 7.18 – Protótipo A – operação de diferença (B – A)
representação aramada

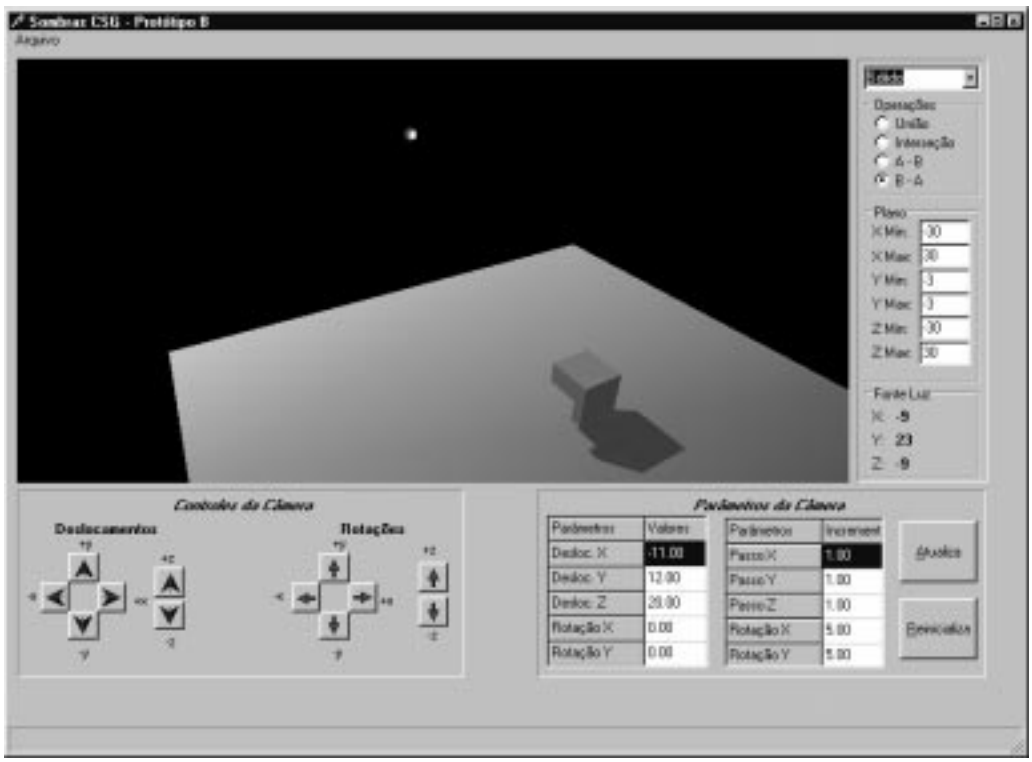


FIGURA 7.19 – Protótipo B – operação de diferença (B – A)
representação sólida

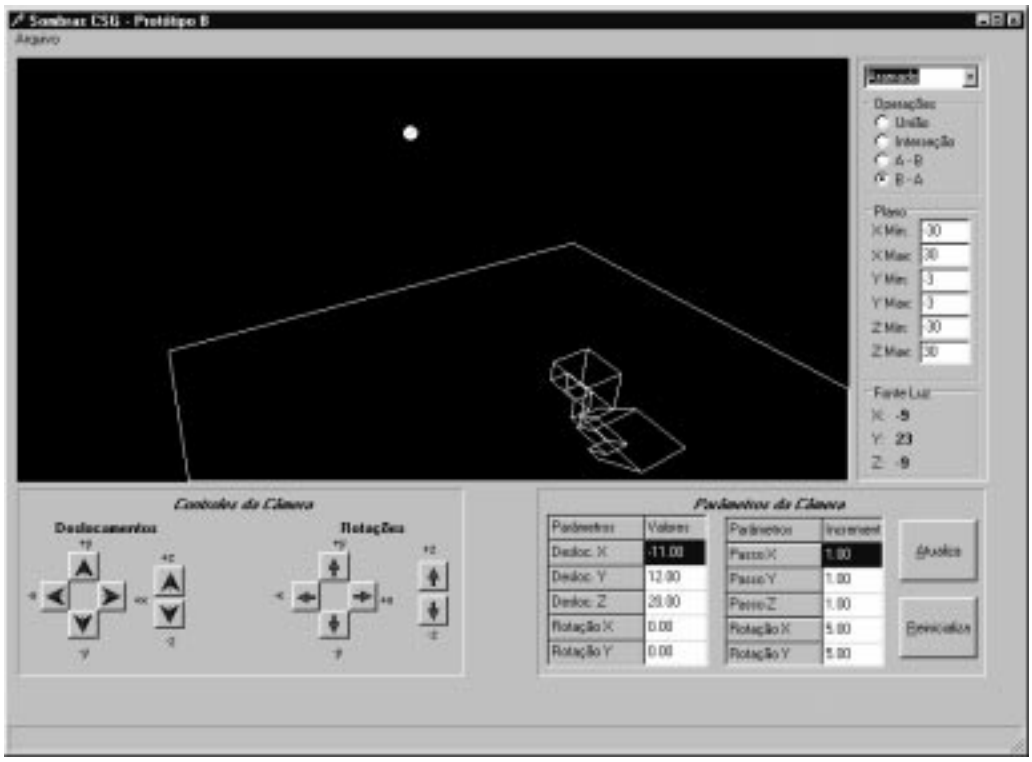


FIGURA 7.20 – Protótipo B – operação de diferença (B – A)
representação aramada

Os resultados obtidos com os protótipos foram satisfatórios, pois como pode ser observado, o resultados apresentados são semelhantes aos casos reais, apresentados nas fotografias.

Pode-se verificar nas figuras acima que os dois protótipos apresentam resultados similares. Chama-se atenção para os resultados do protótipo B, onde propositadamente escolheu-se a exibição aramada para que o leitor pudesse identificar as zonas ativas de cada uma das faces das primitivas.

É importante ressaltar que com a aplicação deste método é possível obter resultados e sombras corretas para cenas modeladas por CSG. A maioria dos algoritmos existentes para a renderização de cenas CSG e consequentemente geração de sombras, faz uso de técnicas que apresentam bons resultados, mas que por outro lado tem um alto custo de processamento. Um dos pontos mais importantes e que merece destaque, além da sombra correta do objeto CSG, é o baixo custo computacional da renderização das cenas com sombras geradas por projeção.

7.2 Limitações e Problemas Encontrados

Os objetos utilizados neste protótipo são constituídos de faces planas do tipo paralelepípedos. No princípio tinha-se como objetivo estudar a possibilidade de aplicar a técnica estudada em objetos com outras formas, e com expressões CSG mais complexas, mas simplificações fizeram-se necessárias para se chegar a conclusão do trabalho em tempo hábil.

Uma das principais dificuldades enfrentada durante a implementação do protótipo foi com relação à estrutura de dados, que necessita ser adequada e consistente para o armazenamento dos vértices que formam o objeto. No início do trabalho pensou-se em utilizar vetores para o armazenamento dos vértices que formam os objetos, pela facilidade de implementação, entretanto, no processo de interseção entre as faces dos objetos, ocorrem alterações nos vértices que formam as faces, pois existem situações em que vértices precisam ser removidos e outras vezes novos vértices são inseridos nas faces, promovendo uma alteração no vetor, o que certamente iria prejudicar o desempenho do processamento.

A solução encontrada para este problema foi a de utilizar uma estrutura de dados dinâmica, utilizando uma lista circular duplamente encadeada, através do uso de ponteiros, possibilitando a manipulação dos vértices. Isso permitiu a inserção ou remoção dos vértices na lista de forma mais eficiente.

Com a estrutura de dados adotada, cada objeto possui uma lista de faces e cada face possui uma lista de vértices. De certa forma, isso introduz uma redundância, pois um mesmo vértice pode pertencer a mais de uma face. Entretanto, descrevendo as faces através de seus vértices evita-se a necessidade de guardar a orientação das arestas que formam a face, mostrando aqui uma economia de recursos.

Um outro problema encontrado durante a implementação foi no uso da biblioteca OpenGl, no modo de visualização sólida dos objetos. Quando a imagem inicial era exibida o objeto apresentava-se correto, porém com a

movimentação da câmera vários erros ocorreram na renderização dos objetos da cena.

Inicialmente estava-se usando a lista de vértices dos polígonos das faces com exibição preenchida, como tal forma apresentou problemas buscou-se uma solução. A solução encontrada foi a de triangularizar os polígonos e preencher os triângulos das faces, tal aplicação foi eficiente e apresentou resultados satisfatórios.

7.3 Complexidade

Através da complexidade de tempo de um algoritmo pode-se determinar como será a execução desse algoritmo, ou seja, quão eficiente ele será. A complexidade associa ao tamanho de entrada do algoritmo uma medida do número de operações que são executadas [TER 90].

Para avaliar a complexidade dos algoritmos desse estudo, convencionou-se:

no: número de objetos

nf: número de faces

nv: número de vértices

Foram consideradas as complexidades da geração dos objetos, geração das sombras, renderização da cena aramada e renderização da cena com visualização sólida para a comparação dos dois protótipos.

Os dois algoritmos para a geração dos objetos, a partir de uma expressão CSG, possuem a mesma ordem de complexidade: O(no nf nv). Para a geração da sombra os dois algoritmos também apresentam a mesma complexidade, que é O(nf nv).

Devido estar sendo considerada a complexidade no pior caso de execução, na avaliação do algoritmo como um todo, a rotina com ordem de complexidade mais significativa determina a complexidade do algoritmo.

Como a geração do objeto CSG possui complexidade O(no nf nv) e a complexidade da geração da sombra é O(nf nv), a complexidade final do algoritmo é O(no nf nv).

Nos protótipos existem duas formas de visualização da cena, a visualização aramada e a representação sólida. As complexidades são O(nf nv) para a visualização aramada e O(nf nv^2) para a visualização sólida. A ordem nv^2 da visualização sólida é devido ao processo de triangularização dos polígonos.

Após a determinação da complexidade dos algoritmos apresentados, pode-se afirmar que o protótipo A é tão eficiente quanto o protótipo B.

7.4 Considerações Finais

Neste capítulo foram apresentados os resultados obtidos com as implementações dos protótipos, bem como uma análise destes resultados.

Dentre os aspectos estudados, cabe salientar que o processamento das interseções entre as faces das primitivas e a classificação dos vértices e arestas são os processamentos mais custosos do protótipo. Lembre-se também que a orientação das arestas é um dos fatores de grande importância dentro deste trabalho, pois se os vértices das faces não estiverem orientados de forma correta, vários equívocos podem acontecer.

Um outro aspecto muito importante é com relação a estrutura de dados dinâmica utilizada no armazenamento das informações, pois a estrutura adotada proporciona um bom desempenho e flexibilidade, neste sentido, possibilitando o uso de novos tipos de primitivas, em extensões futuras do sistema. Isso é fator favorável a utilização de objetos com mais faces e faces com um número maior de vértices.

8 Conclusões

O objetivo deste trabalho é analisar a geração de sombras em objetos modelados por Geometria Sólida Construtiva (CSG). Isso resultou na implementação de dois protótipos que tem por finalidade aplicar e verificar os estudos realizados.

Para a realização desta pesquisa foram desenvolvidas várias etapas, entre elas: a procura de um sistema que oferecesse a possibilidade de gerar sólidos através da Geometria Sólida Construtiva, POVCAD; a interface gráfica do protótipo, ambiente 3D onde são visualizados os objetos resultantes da operação booleana envolvida na cena; procedimentos baseados na técnica de modelagem CSG, como as interseções entre os objetos, faces e arestas; classificação dos vértices e arestas; utilização de uma estrutura de dados dinâmica para atender as necessidades de implementação dos protótipos; o estudo da teoria dos conjuntos; demonstrar que a aplicação das operações booleanas sobre as sombras projetadas nem sempre apresenta os resultados pretendidos; a possibilidade de realizar operações de união, interseção e diferença, fazendo uso de operações de união e interseção juntamente com o complemento; a obtenção da árvore positiva e das zonas ativas das primitivas; o estudo da biblioteca OpenGl e por fim, os procedimentos de geração das sombras, que é o ponto mais importante desta dissertação.

Como já foi mencionado nos capítulos dois e cinco, o avaliador de contornos (*boundary evaluator)* é normalmente a parte mais complexa de um modelador CSG, exigindo tempo e memória do computador. É através deste procedimento que se determina onde as faces são truncadas e onde novos vértices e arestas são criados ou deletados. O *boundary evaluator* encontra estas interseções e então determina, através do *Set-Membership Classification*, a classificação dos vértices, que irão dar origem às novas arestas, gerando assim um novo sólido.

No desenvolvimento do *boundary evaluator* destaca-se o método utilizado para a classificação dos vértices, no sentido de verificar se o vértice localiza-se dentro ou fora da face. Este procedimento utiliza apenas o produto vetorial entre as arestas das faces, o que reduz a complexidade dos cálculos durante esta operação.

Um outro aspecto que não pode ficar fora desta conclusão é o de que pode-se gerar objetos CSG usando-se a combinação de operações com o complemento, como já foi discutido no capítulo cinco.

Qualquer tipo de sólido modelado por CSG pode ser representado, e ter sua sombra gerada, por esse método. Isso devido ao processo de conversão de uma árvore binária CSG, para a sua forma positiva, ou seja, a árvore somente com operadores de união, interseção e complemento poder representar rigorosamente o mesmo sólido da árvore original.

Além da utilização da forma positiva, a zona ativa é outro ponto fundamental. A zona ativa de uma primitiva A, em uma representação CSG de um sólido S, é definida como uma região em que mudanças em A afetam o sólido S. Este é um dos pontos mais importantes estudados neste trabalho, pois através disto apresentou-se uma outra alternativa para a geração de

objetos CSG, onde para cada primitiva ou nodo de uma representação CSG associam-se os conjuntos de nodos-i e nodos-u. Estes conjuntos não dependem da forma da primitiva, mas da sua posição em particular na árvore CSG que representa o sólido e das formas das outras primitivas que fazem parte da representação.

A utilização de listas circulares duplamente encadeadas é outro aspecto a ser considerado nas implementações, pois este tipo de estrutura de dados além de permitir o acesso rápido aos dados possibilita também a inclusão e exclusão de vértices sem a necessidade de alterar a estrutura de dados existente.

Cabe ressaltar que além dos resultados satisfatórios e a velocidade de geração de sombra, uma outra grande vantagem em utilizar a técnica de geração de sombras, apresentada neste trabalho, é que, quando for concluída a modelagem do sólido CSG, tem-se também as informações necessárias para a renderização da sombra projetada da cena. Todas as faces visíveis do ponto de vista da fonte de luz devem ter seus vértices projetados. Desta maneira obtêm-se o polígono da sombra desta face e a união destes polígonos irá originar a sombra final da cena.

8.1 Propostas Para Trabalhos Futuros

Fica como sugestão para trabalhos futuros a modelagem de novos objetos fazendo-se uso de expressões booleanas mais complexas, com a possibilidade de utilizar primitivas de outros tipos. Isso aumentará a complexidade do objeto resultante e será possível a geração de cenas mais complexas, sendo esta uma extensão do algoritmo proposto.

Nesta pesquisa foram aplicadas as técnicas de modelagem CSG somente em primitivas do tipo paralelepípedo. Em objetos mais que podem ser modelados através de uma aproximação poligonal (*Polygon Meshes*), pode-se aplicar os mesmos métodos e técnicas aqui discutidas.

Optou-se por gerar objetos com faces planas neste protótipo. A opção por outros semi-espaços, além do planar, implicaria no estudo de vários outros tópicos, como por exemplo, interseção de superfícies curvas. Pode-se também estender os métodos apresentados nesta dissertação para a realização de interseções e modelagem de objetos CSG, constituídos de faces curvas, possibilitando assim aumentar a variedade de objetos a serem tratados pelo protótipo.

Com relação ao universo que envolve os objetos que estão sendo modelados, se for considerado como universo a união de todos os objetos: será que esta suposição pode apresentar resultados mais satisfatórios? Este é um outro assunto que deve ser investigado.

As três perguntas que representam o eixo principal da realização desta pesquisa, são:

- A sombra da união dos objetos é igual a união das sombras dos objetos ?

- A sombra da interseção dos objetos é igual a interseção das sombras dos objetos ?
- A sombra da diferença dos objetos é igual a diferença das sombras dos objetos?

Como se pode perceber com a realização deste estudo, a aplicação da operação booleana envolvida na modelagem do sólido sobre as sombras projetadas nem sempre apresenta resultados satisfatórios. Esta é a conclusão encontrada com o estudo realizado até o momento, porém isto não quer dizer que esta solução não seja possível. Este assunto é interessante e deve ser tratado com um maior aprofundamento matemático.

Existe a possibilidade de estender os métodos estudados, para outros tipos de objetos, inclusive objetos com faces de superfícies curvas, pois se sabe que objetos deste tipo podem ser modelados através da técnica de *sweep*, gerando uma malha de polígonos, possibilitando assim o uso do método apresentado.

Algoritmos para a geração de sombras projetadas suaves, com as regiões de umbra e penumbra, também podem ser aplicados como extensão desse estudo. Nesta pesquisa apresentaram-se apenas sombras projetadas “bem definidas”. A aplicação de tais tipos de sombra possibilita um maior realismo à cena.

Uma sombra não precisa ser necessariamente uma região escura. Se forem utilizadas fontes de luz coloridas, as sombras geradas podem ser coloridas. Além de se pensar na existência de várias fontes de luz. Estes são outros aspectos que podem ser trabalhados a partir dessa pesquisa.

Anexo POVCAD

Para a execução do trabalho proposto foi necessária a utilização de um sistema para modelagem de sólidos com os recursos da Geometria Sólida Construtiva.

O POVCAD é um sistema que permite a modelagem de objetos em um ambiente visual oferecendo dentre vários recursos a possibilidade se trabalhar com Geometria Sólida Construtiva, é *free* e está disponível para *download* em http://www.povcad.com.

A figura A.1, apresenta a interface do POVCAD.

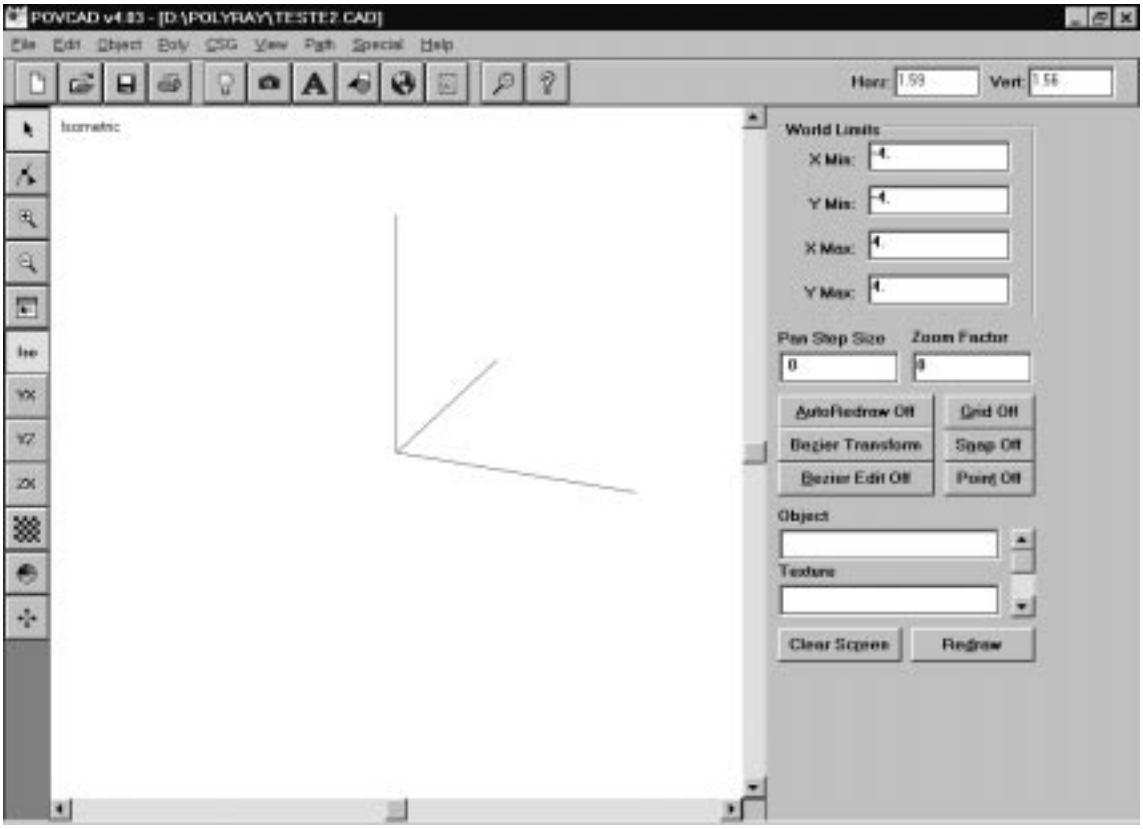


FIGURA A.1 – Sistema para modelagem de sólidos POVCAD 4.0

Logo abaixo se apresenta um exemplo de um arquivo gerado pelo PovCad e que pode ser importado pelo protótipo desenvolvido, apresentando como resultado o objeto modelado e sua respectiva sombra projetada.

```
// Polyray Scene File: CUBOS.PI
//
// Generated With POVCAD 4 (c) Alfonso Hermida 1993 - 1995
//                   (c) Alfonso Hermida 1993 - 1995
//
// Created on 06/05/99 at 11:50 am
include "d:\poly\colors.inc"
include "d:\poly\texture.inc"
include "d:\poly\stones.inc"
```

```
include "d:\povcad4\metals.inc"
include "d:\povcad4\polycolr.inc"
// SET UP THE CAMERA
viewpoint {
from        <2, 5, -12>
at          <0, 0, 0>
up          <0, 1, 0>
angle       45
hither            1.0e-3
resolution  1024, 768
aspect            4/3
yon         1.0e5
max_trace_depth   5
aperture    0
}

// START UNION
object {
     // BOX
     object {
  box <-2.0, -2.0, -2.0>, <2.0, 2.0, 2.0>
  translate <0.0, 0.0, 0.0>
  rotate <0.0, 0.0, 0.0>
  texture {
    surface {
            color White
    diffuse 0.5
    }
  }
}

-

// BOX
     object {
  box <-2.0, -2.0, -2.0>, <2.0, 2.0, 2.0>
  rotate <0.0, 0.0, 0.0>
  translate <2.0, 2.0, 1.0>
  texture {
    surface {
      color White
      diffuse 0.5
    }
  }
}
} // END UNION
// LIGHT_SOURCE
light White, <-9.0, 8.0, -5.0>
```

1 2 3 4

FIGURA A.2 – Arquivo de saída do POVCAD

Na figura A.2, pode-se ver o exemplo de um arquivo gerado pelo POVCAD. Analisando este arquivo verificar-se que ele monta a árvore CSG do objeto que está sendo modelado.

A chave 1, está representando uma primitiva do tipo cubo, com a determinação de tamanho variando do ponto -2, -2, -2 (canto inferior esquerdo

do cubo) até o ponto 2, 2, 2 (canto superior direito do cubo). Esta primitiva não sofreu nenhuma transformação geométrica.

Na chave 2, pode-se observar que a operação booleana envolvida neste caso é uma operação de diferença, representada pelo sinal da subtração –. As operações de interseção e soma são representadas no POVCAD respectivamente pelos símbolos * e +.

A chave 3, representa uma outra primitiva do tipo cubo, com o mesmo tamanho da primitiva anterior, porém com uma transformação geométrica do tipo translação. A primitiva sofreu uma translação de duas unidades no eixo x, duas unidades no eixo y e uma unidade no eixo z.

O objeto resultante da árvore é a raiz, que está representada na figura pela chave 4, ou seja, objeto A – objeto B.

Na figura A.3, pode-se verificar a árvore binária do mesmo objeto, sob a forma de apresentação gráfica.

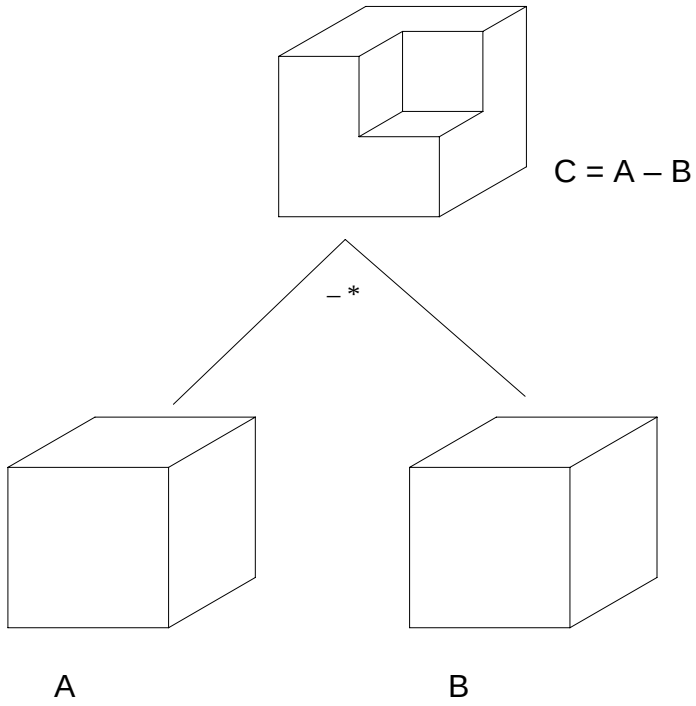


FIGURA A.3 – Árvore binária de um objeto modelado pelo POVCAD

Bibliografia

[ABE 91] ABE, Jair Minoro; PAPAVERO, Nelson. **Teoria Intuitiva dos Conjuntos**. São Paulo: McGraw-Hill, 1991.

[ALE 76] ALENCAR FILHO, Edgard. **Teoria Elementar dos Conjuntos**. São Paulo: Nobel, 1976.

[ANG 2000] ANGEL, Edward. **Interactive Computer Graphics:** A Top-Down Approach With OpenGl. Reading: Addison-Wesley, 2000.

[APP 68] APPEL, A. **Some Techniques for Shading Machine Renderings of Solids**. In: Spring Joint Computer Conference, 1968, Atlantic City. **Proceedings**... Washington: Thompson Books, 1968. 534p. p.37-45.

[ATH 78] ATHERTON, P.; WEILER, K.; GREENBERG, D. Polygon Shadow Generation. **Computer Graphics**, New York, v.12, n.3, p.275-281, Aug. 1978. Trabalho apresentado no SIGGRAPH, 1978, Atlanta.

[BLI 88] BLINN, J. F. Jim Blinn's Corner: me and my (fake) shadow. **IEEE Computer Graphics & Application,** Los Alamos, v.8, n.1, p.82-86, jan. 1988.

[CRO 77] CROW, F. C. Shadow Algoritms for Computer Graphics. **Computer Graphics**, New York, v.11, n.2, p.242-248, July 1977. Trabalho apresentado no SIGGRAPH, 1977, San Jose.

[FOL 97] FOLEY, D. J. et al. **Computer Graphics Principles and Practice**. 2nd ed. Reading: Addison-Wesley, 1997.

[GEO 99] GEOMETRY Formulas and Facts. Disponível em: <http://www.geom.umn.edu/docs/reference/CRC-formulas/>. Acesso em: ago. 1999.

[GHA 98] GHAZANFARPOUR, Djamchid; HANSENFRATZ, Jean-Marc. A Beam Tracing Method With Precise Antialiasing For Polyhedral Scenes. **Computer & Graphics**, New York, v. 22, n.1, p.103-115, 1998.

[GOM 90] GOMES, Jonas; VELHO, Luiz. **Conceitos Básicos de Computação Gráfica**. São Paulo: IME/USP, 1990. Trabalho apresentado na Escola de Computação, 7., 1990, São Paulo.

[GOM 94] GOMES, Jonas; VELHO, Luiz. **Computação Gráfica:** imagem. Rio de Janeiro: IMPA, 1994.

[GOM 98] GOMES, Jonas; VELHO, Luiz. **Computação Gráfica**. Rio de Janeiro: IMPA, 1998.

[JAN 91] JANSEN, Frederik W.; ZALM Arno N. T. V. D. A Shadow Algorithm For CSG. **Computer & Graphics**, New York, v.25, n.2, p.237 – 247, 1991. Trabalho apresentado no Eurographics, 1990.

[KRI 96] KRISHNAN, Shankar; MANOCHA, Dinesh. **BOOLE:** A System to Compute Boolean Combinations of Sculptured Solids. Chapel Hill: Department of Computer Science, University of N. Carolina, 1996. Disponível em: <http://www.cs.unc.edu/~geom.html>. Acesso em: set. 1999.

[KRI 97] KRISHNAN, S. et al. **Interactive Boundary Computation of Boolean Combinations of Sculptured Solids**. Chapel Hill: Department of Computer Science, University of N. Carolina, 1997. Disponível em: <http://www.cs.unc.edu/~geom.html>. Acesso em: set. 1999.

[LIP 98] LIPSCHUTZ, Seymour. **Teoria dos Conjuntos**. São Paulo: McGraw-Hill, 1998.

[MAC 98] MACIEL, Anderson. **Detecção de Colisões Entre Pares de Poliedros Rígidos Aplicada ao Projeto Asimov**. Caxias do Sul: Universidade de Caxias do Sul, 1998. Trabalho de conclusão.

[MOR 85] MORTENSON, M. E. **Geometric Modeling**. New York: John Wiley, 1985.

[NAS 92] NASCIMENTO, Marcos Eduardo. **Estudo de Algoritmos para Geração de Sombras Projetadas na Síntese de Imagens Foto-realísticas**. Porto Alegre: PGCC da UFRGS, 1992.

[PRI 2000] PRITIKIN, Alice; WEED; Dylan. **Extending the Raytracer:** CSG, Soft Shadows, Spot Lights and Three New Shapes. Winter 2000. Disponível em: <http://www.people.fas.harvard.edu/~pritikin/graphics>. Acesso em: fev. 2001.

[REQ 78] REQUICHA, A. A. G.; TILOVE, R. B. **Mathematical Foundations of Constructive Solid Geometry :** General Topology of Closed Regular Sets. Rochester: University of Rochester, 1978.

[REQ 82] REQUICHA, A. A. G.; VOELCKER, H. B. **Constructive Solid Geometry**. Rochester: University of Rochester, 1982.

[REQ 85] REQUICHA, A. A. G.; VOELCKER, H. B. Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms. **Proceedings of the IEEE**, New York, v. 73, n.1, p.30-44, Jan. 1985.

[ROS 89] Rossignac, J. R. and Voelcker, H. B. Active Zones in Constructive Solid Geometry for Redundancy and Interference Detection. **ACM Transaction on Graphics**, New York, v.8, n.1, p.51-87, Jan. 1989.

[RIP 87] RIPOLL, Maria T. S. **Análise de um Avaliador de Superfícies de Forjados Representados por Geometria Sólida Construtiva**. Porto Alegre: PGCC da UFRGS, 1987.

[SAL 90] SALESIN, David; STOLFI, Jorge. Rendering CSG Models With a ZZ-Buffer. **Computer & Graphics**, New York, v. 24, n.4, p.67-76, 1990.

[SIL 97] SILVEIRA, Ismar F. **Implementação de Operações Booleanas Regularizadas Entre Sólidos CSG em VRML**. São José dos Campos: Instituto Tecnológico de Aeronáutica, 1997. Dissertação de mestrado.

[TER 90] TERADA, Routo. **Introdução a Complexidade de Algoritmos**. São Paulo: IME/USP, 1990.

[VER 84] VERBECK, C. P.; GREENBERG, D. P. A Comprehensive Light-Source Description for Computer Graphics. **IEEE Computer Graphics & Applications**, Los Alamos: v.4, n.7, p.66-75, july 1984.

[VER 2001] VERNAL, Michael. **Extending the Raytracer:** Bidirectional Refletance Distributions Functions. 2001. Disponível em: <http://www.fas.harvard.edu/~lib175/projects_fall_2000/vernal>. Acesso em: fev. 2001.

[WAT 2000] WATT, Allan. **3D Computer Graphics**. 3rd ed. Reading: Addison-Wesley, 2000.

[WAR 83] WARN, D. Lighting Controls For Synthic Images. **Computer Graphics**, New York, v.17, n.3, p.13-21, July 1983.

[WIL 78] WILLIAMS, L. Casting Curved Shadows on Curved Surfaces. **Computer Graphics**, New York, v.12, n.3, p.270-274, aug. 1978. Trabalho apresentado no SIGGRAPH, Atlanta, 1978.

[WRI 2000] WRIGHT Jr., Richard S.; SWEET Michael. **OpenGl SuperBible**. 2nd ed. Indianapolis: Waite Group, 2000.