Universidade do Porto
Faculdade de Engenharia
FEUP

Ana Cláudia de Loureiro e Nogueira

Armazenamento e Busca Eficientes de Raios Luminosos em Síntese de Imagem

Um Protótipo de Avaliação da Técnica Holodeck

Porto, 2001

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores

Armazenamento e Busca Eficientes de Raios Luminosos em Síntese de Imagem

Um Protótipo de Avaliação da Técnica *Holodeck*

Ana Cláudia de Loureiro e Nogueira

Graduada com o
Curso de Estudos Superiores Especializados em Engenharia Informática Industrial pelo
Instituto Superior de Engenharia do Porto do
Instituto Politécnico do Porto

Dissertação submetida para a satisfação parcial dos
requisitos do grau de mestre em
Engenharia Electrotécnica e de Computadores
(Área de especialização de Informática Industrial)

Dissertação realizada sob a supervisão do
Professor Doutor António Augusto de Sousa,
da Faculdade de Engenharia da Universidade do Porto
e do
Professor Doutor António Cardoso Costa,
do Instituto Superior de Engenharia do Porto

Porto, Setembro de 2001

Armazenamento e Busca Eficientes
de Raios Luminosos em Síntese de Imagem

Um Protótipo de Avaliação da Técnica *Holodeck*

Resumo

Os métodos de síntese de imagem baseados em modelos de iluminação sofisticados, com preocupações de produzir resultados fisicamente válidos, tendem a consumir recursos de processamento muito pesados, em particular, tempos de processamento incompatíveis com visualizações interactivas.

Nesta dissertação apresenta-se uma nova abordagem, que combina a representação holográfica de uma cena com o cálculo progressivo de raios. O sistema de síntese de imagem corresponde a uma implementação da técnica *Holodeck* que, com base num motor de radiância fisicamente válido, cria uma estrutura que permite armazenar a radiância dos raios que atravessam a cena para posterior reutilização.

Os raios são calculados e armazenados, fisicamente utilizando uma estrutura adequada de informação. Esta estrutura corresponde a uma grelha espacial que é utilizada para ordenar raios. Estes raios são reutilizados para vistas subsequentes e raios adicionais podem ser gerados e armazenados interactivamente. Os raios são agrupados em feixes para um acesso eficiente, para que nenhum processamento intermédio ou compressão seja necessário entre o cálculo de amostras e a utilização das mesmas. Embora os ficheiros *Holodeck* sejam de grandes dimensões, estas estruturas de informação podem ser gravadas em CD-ROM ou outro meio de armazenamento para acesso rápido ou para outro processo de *rendering*, e não necessitam de ser mantidas em memória.

Sempre que se procede ao cálculo de um novo raio luminoso, consulta-se a estrutura a fim de se obter rapidamente uma resposta adequada, recorrendo-se ao uso do motor de radiância apenas quando não existe a resposta satisfatória.

Os resultados obtidos permitem prever que, através desta técnica, poderão ser realizadas visualizações interactivas em cenas geometricamente complexas, com níveis elevados de realismo, fundamentais em certos tipos de aplicação (arquitectura, *design* de iluminação, etc.).

Abstract

Image rendering methods, which are based on sophisticated physics models, tend to consume very large processing resources, in particular, processing times, that become incompatible with interactive displaying.

In this these it is presented an implementation of the technique Holodeck that, based on a program that computes physically valid radiances, generates a data structure that can store light rays for later reuse. Therefore, every time that a light ray is necessary, the structure is consulted in order to obtain an adequate and fast answer, living the use of the radiance program to calculate rays that do not exist.

The rays are calculated and eventually stored physically using a data structure. This structure represents a space grid used to order rays. These rays are reused for subsequent views and additional rays can be generated interactively. The rays are contained in bunches for an efficient access, so that no intermediate processing or compression is necessary between the calculation of samples and the use of the same ones. Although the great dimensions the Holodeck files, the data structures can be recorded in CD-ROM or other middle of storage for fast access or for another rendering process, and these structures doesn't need to be maintained in memory.

Whenever a light ray is calculated, the structure is consulted in order to obtain a quickly appropriated answer and the radiance motor is used when the answer isn't satisfactory.

The results obtained permit to conclude that, through this technique, interactive high realism displays are possible, even in geometrically complex scenes. This is fundamental for certain types of applications such as architecture, illumination design, etc.

Résumé

Les méthodes de synthèse de l'image basées sur des modèles physiques sophistiqués, avec les inquiétudes de produire des résultats physiquement valides, ont tendance à consommer de très grandes ressources de traitement, en particulier, des temps de traitement incompatibles avec les visualisations interactives.

Dans cette dissertation une nouvelle approche est présentée, qui combine la représentation de l'holografique d'une scène avec le calcul progressif de rayons. Le système de synthèse de l'image présente une mise en oeuvre de la technique Holodeck qui, basé sur un moteur de rayonnement physiquement valide, crée une structure qui permet de garder le rayonnement des rayons qui traversent la scène pour une utilisation postérieure.

Les rayons sont calculés et sont gardés en utilisant une structure d'information. Cette structure représente une grille de l'espace utilisée pour ranger des rayons. Ces rayons sont réutilisés pour les vues subséquentes et des rayons additionnels peuvent être produits interactivement. Les rayons sont contenus dans des tas pour un accès efficace, afin que tout traitement intermédiaire ou compression ne soient pas nécessaires entre le calcul d'échantillons et leur utilisation. Bien que les dossiers Holodeck soient de grandes dimensions, les structures d'information peuvent être enregistrées dans un CD-ROM ou un autre milieu de stockage pour accès rapide ou pour un autre processus de synthèse de l'image, et ces structures n' ont pas besoin d'être maintenu dans mémoire.

Chaque fois qu'on calcule un rayon illuminant, la structure est consultée pour obtenir rapidement une réponse appropriée et on utilise le moteur du rayonnement seulement quand il n'existe pas de réponse satisfaisante.

Les résultats obtenus permettent de prévoir que, à travers cette technique, les visualisations interactives peuvent être accomplies dans des des scènes géométriquement complexes, avec de hauts niveaux de réalisme, fondamentaux dans certains types de logiciels (architecture, dessin de l'illumination, etc.).

Agradecimentos

a Sérgio Amaral;

aos meus pais e à minha irmã;

ao Prof. António Augusto Sousa, orientador;

ao Prof. António Cardoso Costa, co-orientador;

ao Eng.º José Fernando Martins;

ao Prof. Alexandre Sousa;

ao Eng.º Miguel Losa;

ao Eng.º José Moreira;

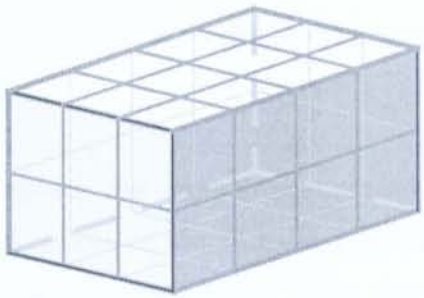
à antiga Unidade de Sistemas de Informação e Computação Gráfica do INESC Porto.

Para aqueles que pretendam dedicar-se à síntese de imagem, espero que possam dispor dos meios necessário para tal, pois esta é uma área que desperta a imaginação e intelecto.

Índice

Índice de Figuras

Índice de Algoritmos



1 Introdução

Quando se observa uma imagem criada através de um processo de síntese de imagem baseado no método de traçagem de raios (*Ray-Tracing* [Whitted80]), apreende-se na realidade, um conjunto de pontos coloridos, cada um deles localizado numa grelha rectangular. Para a obtenção dessas colorações é necessária uma grande quantidade de cálculos: em cada um desses pontos são projectados raios que partem do ponto de observação (câmara) em direcção à cena; cada um desses raios pode atingir vários objectos da cena em diversos pontos; a partir do ponto mais próximo obtido são gerados novos raios que podem atingir outros objectos, dando assim origem a novos raios, etc. Deste modo, quando o número de objectos é muito grande, a geração de novos raios pode atingir níveis muito elevados. No fim deste processo, tipicamente, o tempo de cálculo consumido é muito elevado.

Este problema torna-se mais grave quando se trata de criar imagens dinamicamente, por exemplo, durante a navegação numa cena (*walkthrough*), dado que os tempos de processamento, demasiado altos, são incompatíveis com a cadência necessária para produção de imagens com fins de visualização.

Em [Larson98a] apresenta-se uma nova solução: pré-processar e armazenar alguma informação relacionada com os raios luminosos, a fim de ser utilizada mais tarde. Assumindo-se que o observador se pode deslocar livremente na cena, tem de se procurar garantir que a informação armazenada alberga raios representativos de "todas" as direcções de vista possíveis.

Este tipo de abordagem não tem sido aprofundado, apesar das suas potencialidades parecerem ser grandes. Por exemplo, a informação de raios, depois de armazenada, pode ser transportada, permitindo a reconstituição da cena (visualização) noutro local. Além disso, reduz-se ou mesmo elimina-se o processo de intersecção de novos raios, pelo que o tempo de processamento poderá tornar-se significativamente inferior, sendo esta a principal vantagem.

Nesta dissertação, apresenta-se uma implementação de um protótipo da técnica *Holodeck*, que permite o armazenamento e busca eficientes de raios luminosos com o intuito de possibilitar a navegação interactiva em cenas virtuais numa plataforma computacional comum.

Para além de se propor um faseamento de refinamento da imagem (não explícito no artigo em que se apresenta a técnica *Holodeck* [Larson98a]), analisa-se o desempenho dos passos críticos deste método, assim como o grau de parametrização adequado para navegações interactivas. Para esse efeito, apresenta-se um estudo que visa encontrar a relação ideal entre os parâmetros subjacentes à referida técnica. Visa-se igualmente determinar uma gama adequada para o número de amostras a inserir na estrutura de dados do *Holodeck*.

1.1 Estrutura da Dissertação

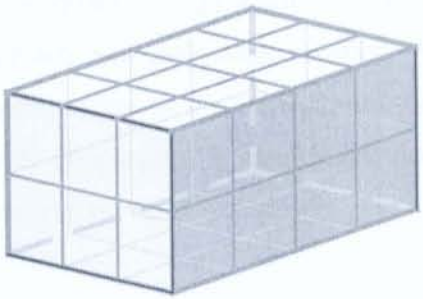
No capítulo 2, descrevem-se os trabalhos realizados na área da síntese de imagem e os que mais se aproximam do âmbito desta dissertação. Faz-se também uma comparação entre eles. Deste conjunto de trabalhos, é analisada em pormenor, no capítulo 3, a técnica de *Holodeck*, descrita em [Larson98a].

No capítulo 4, faz-se a descrição da aplicação protótipo implementada, baseada na técnica *Holodeck*, dando-se especial ênfase à sua estrutura de raios, a qual é criada para utilização

pelo processo de visualização. O protótipo, seguindo as linhas gerais apontadas por [Larson98a], permite avaliar quais os passos mais críticos do método, fornecendo indicadores importantes sobre as suas evoluções mais adequadas.

No capítulo 5 é apresentada uma avaliação dos resultados obtidos a partir da criação de imagens, utilizando a aplicação implementada.

Por último, expõem-se as principais conclusões sobre o trabalho realizado, discutem-se algumas limitações actuais e apresentam-se sugestões para trabalho futuro.



2 Trabalhos Relacionados

A produção de imagens realistas tem sido, ao longo do tempo, um dos objectivos da computação gráfica. Actualmente, o realismo e a interacção em tempo real têm que ser balanceados, isto é, tem que ser encontrado um equilíbrio entre ambos. Para se atingir uma cadência elevada de visualização de imagens, tem que se perder em termos de realismo. Por outro lado, quando se aumenta o realismo de imagem pretendido, perde-se em interactividade.

Um sistema de síntese de imagem com iluminação global total pode ser uma ferramenta poderosa para utilizadores que pretendam visualizar um objecto ou espaços complexos. Para se ter uma verdadeira sensação de como vai parecer uma cena criada, o utilizador deve ser capaz de viajar nela e ver o sua geometria de vários ângulos, à medida que se move. Muitos dos sistemas de síntese de imagem existentes não fornecem esta funcionalidade. A traçagem de raios convencional não suporta um observador móvel a cadências interactivas porque os cálculos são dependentes da posição do observador e têm que ser realizados novamente para cada nova posição. Para visualizações correctas de ambientes realistas, é necessário um sistema que seja interactivo e, eventualmente, progressivo, explorando todos os potenciais

da síntese de imagem e capacidade de processamento, a fim de fornecer uma sequência de imagens o mais rapidamente possível.

Vários foram os trabalhos desenvolvidos nesta área, os quais se agrupam em duas abordagens classificadas por [McMillan97] e se apresentam nas próximas secções. A primeira abordagem, designada por visualização aproximada, refere-se ao mapeamento de texturas e reprojecção de imagens ou raios. A segunda abordagem é designada por armazenamento e relaciona-se com o armazenamento de imagens ou raios luminosos.

2.1 Abordagem pela Visualização Aproximada

Imagens mapeadas em geometria simplificada são frequentemente utilizadas como uma representação aproximada de ambientes virtuais. Os mapas de texturas são talvez o exemplo mais óbvio para este caso. Outra aproximação mais subtil envolve a assumpção que todos ou a maioria da geometria que descreve a cena se encontra tão distante do observador que a sua forma se torna pouco importante.

As técnicas de mapeamento de texturas têm sido objecto de muito estudo na área da computação gráfica.

Basicamente, as técnicas de mapeamento de texturas baseiam-se em funções de mapeamento textura-para-modelo ou perspectiva-projecção onde a cena é construída a partir do espaço tridimensional para coordenadas espaciais da imagem desejada. [McMillan97] afirma que actualmente, os sistemas de síntese de imagem realizam um ou ambos os processos de mapeamento das coordenadas da imagem desejada para as coordenadas tridimensionais no espaço dos objectos visíveis e posteriormente, efectuam o mapeamento a partir das coordenadas dos objectos tridimensionais para as coordenadas das texturas no espaço-imagem.

Para melhorar a representação das cenas utilizam-se modelos tridimensionais precisos que especifiquem a forma dos objectos. Ao combinar esta dificuldade de representação com a necessidade de representar a textura de cada ponto da superfície do objecto, o problema torna-se ainda mais complexo.

Em [Catmull75] apresenta-se um algoritmo para síntese de imagem modelada com superfícies aproximadas por polígonos biparametrizados. Enquanto que o algoritmo de Catmull trabalha sobre a definição matemática das superfícies, em trabalhos anteriores a este os objectos eram aproximados por colecções de polígonos planares.

Em [Blinn76] desenvolve-se uma extensão a este trabalho. Para melhorar a qualidade da imagem, este autor utiliza a teoria de processamento de sinal digital e fórmulações matemáticas relacionadas com superfícies curvas.

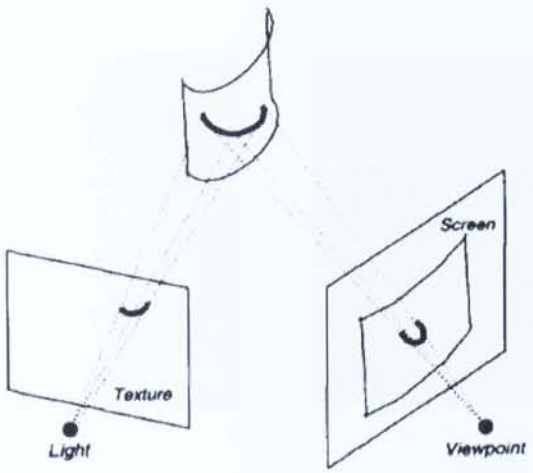


Figura 1: Visualização de uma textura projectada

(figura retirada de [Segal92])

Segal [Segal92] realiza uma extensão simples do mapeamento de texturas com perspectiva para simular vários efeitos de luz (Figura 1). Inclui projecção arbitrária de imagens bidimensionais na geometria, pontos de luz realistas e geração de sombras. Estes efeitos são obtidos utilizando *hardware* que implementa um mapeamento de texturas.

Uma classe de aproximações a cenas 3D resulta quando uma imagem é mapeada para uma figura geométrica específica constituída por um conjunto de pontos. O mapeamento é conseguido da mesma forma que a aplicação dos mapas de texturas a esferas, uma vez que cada ponto da esfera pode ser directamente associado a um ponto localizado a uma distância infinita do centro da esfera. Esta observação também é a base dos mapeamentos referidos

anteriormente. Estas representações caracterizam o trabalho de Regan e Pose [Regan94] e o sistema *QuickTimeVR* [Chen95].

Regan e Pose [Regan94] descrevem um sistema híbrido no qual imagens panorâmicas de referência são geradas utilizando os tradicionais sistemas de síntese de imagem *geometry-based* e é fornecida síntese de imagem interactiva através de um subsistema *image-based*. Desta forma, a interacção é permitida ao observador através da alteração da sua orientação em torno de um ponto fixo de visualização. A Figura 2 apresenta uma vista criada a partir de uma imagem sintetizada na superfície de um cubo.

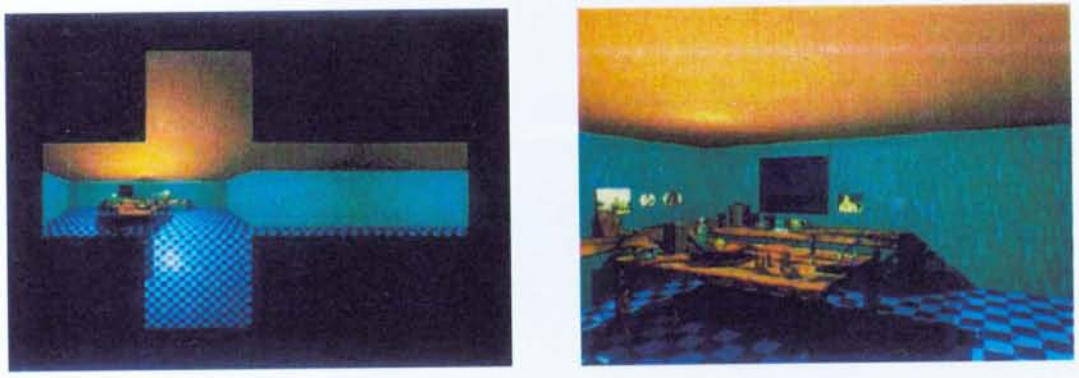


Figura 2: Imagem sintetizada na superfície de um cubo e uma vista criada a partir dessa imagem

(figura retirada de [Regan94])

Outro trabalho [Chen95] com o mesmo princípio, comercializado com o nome *QuickTimeVR*, recorre a um conjunto de imagens cilíndricas. Este trabalho permite que as operações de visualização se efectuem a grande cadência de interacção (mais de 20 fotogramas por segundo), recorrendo apenas a *software*. A posição do observador já não se encontra confinada a um ponto fixo, uma vez que são permitidas translações através da selecção de novas imagens cilíndricas cujo centro de projecção se aproxima da posição corrente do observador. O *QuickTimeVR* introduziu a capacidade de utilizar um conjunto de fotografias como imagens de referência (Figura 3), o que possibilitou novas aplicações na área da computação gráfica.

Mais recentemente, os trabalhos de [Shade96], [Sillion97] e [Schaufler98] substituem objectos complexos por "impostores", tirando partido do *hardware*. A utilização destes "impostores" tem o inconveniente de requerer frequente reprocessamento para evitar artefactos.

Figura 3: Imagem cilíndrica e algumas das fotografias que lhe deram origem (figura retirada de [Chen95])

[Shade96] apresenta um método que utiliza a coerência de polígonos para acelerar *walkthroughs* de cenas estáticas e geometricamente complexas. O método constrói uma árvore (BSP) que divide hierarquicamente as primitivas geométricas da cena. Durante a navegação, imagens de nodos a vários níveis são armazenadas numa *cache* para reutilização em fotogramas subsequentes. As imagens armazenadas são reutilizadas através de mapeamento de texturas num único polígono quadrilateral que e desenhado em vez da geometria contida no nodo correspondente.

Em [Sillion97] o trabalho é focalizado na navegação em ambientes urbanos que apresentam grandes desafios devido à enorme complexidade de informação geométrica e às grandes

variações das condições de visibilidade. O conceito central baseia-se na utilização avançada de "impostores", já conhecidos desde o trabalho de [Maciel95], para representar cenários distantes, conforme se ilustra na Figura 4.

a) Modelo local b) "Impostor" c) Local + "Impostor"

Figura 4: Princípios de segmentação de modelos e utilização de "impostores" (figura retirada de [Sillion97])

A abordagem de [Schaufler98] realiza a reprojecção de informação de uma imagem para novas vistas e é baseado no facto de partes de uma imagem, a uma certa distância do observador, se moverem uniformemente. Esta reprojecção é aproximada pela visualização de vários níveis de mapeamento de texturas sobre polígonos quadrilaterais.

Algumas técnicas oferecem soluções aproximadas à iluminação global para reflexão especular e refracção de superfícies, com cadências interactivas em ambientes simples [Ofek98]. Walter et al. [Walter97] usaram o sombreado e geometria representativa de luz/sombra local para aproximar a aparência de uma solução de iluminação global não difusa através da utilização de *hardware* em síntese de imagem. Apesar de os resultados não serem exactos, especialmente para as sombras, produziram algumas simulações muito eficientes de ambientes simples. Contudo, os ambientes complexos continuam a representar um sério desafio.

No trabalho recente de [Walter99], as amostras de raios são guardadas numa *cache* e visualizadas interactivamente num ciclo contínuo de actualização. Contudo, Walter não mantém um registo permanente dos raios calculados, ao contrário do que se propõe na implementação da técnica *Holodeck*.

2.2 Abordagem por Armazenamento

O sistema *movie-map* [Lippman80] é uma das primeiras tentativas de construção de um sistema puramente *image-based*. Este sistema permite operações de deslocação, rotação e ampliação de uma cena, recorrendo a imagens de referência, previamente armazenadas em discos laser de vídeo interactivos. Para haver liberdade de movimentos é necessário o armazenamento de milhares dessas imagens. Esta abordagem é incapaz de reconstruir todas as vistas possíveis sem sobrecarregar o espaço de armazenamento.

Levoy e Hanrahan [Levoy96] apresentam um método de armazenamento de raios luminosos em vez de imagens. Esta técnica, intitulada *light field*, permite reduzir a parametrização destes raios. Os raios que atravessam um espaço vazio que se encontra entre dois planos, podem ser descritos utilizando apenas quatro parâmetros (uma coordenada bidimensional em cada plano) em vez das cinco dimensões necessárias para a especificação típica de um raio (uma coordenada tridimensional para a origem do raio e dois ângulos para especificar a sua direcção). A representação de raios em "quatro dimensões" apresenta as vantagens de diminuir o espaço necessário para armazenamento e simplifica a reconstrução da função de radiância calculada a partir destes raios [Levoy96].

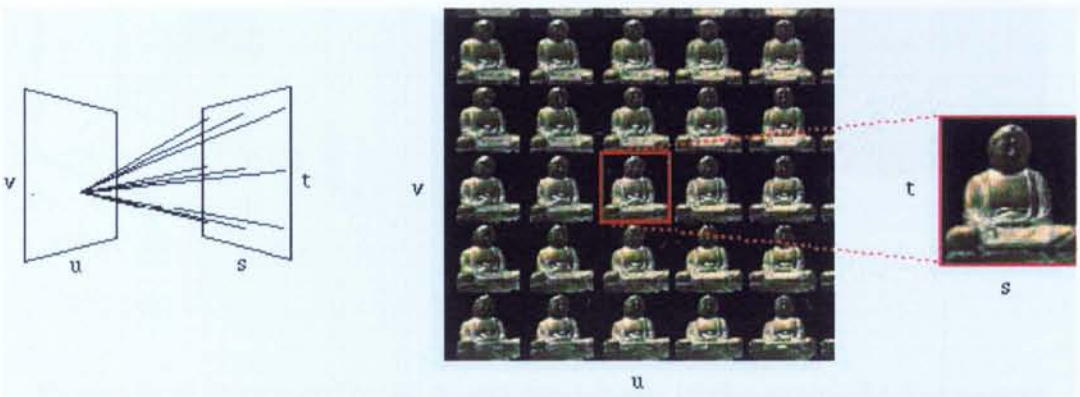


Figura 5: Visualização de um *light field*
(figura retirada de [Levoy96])

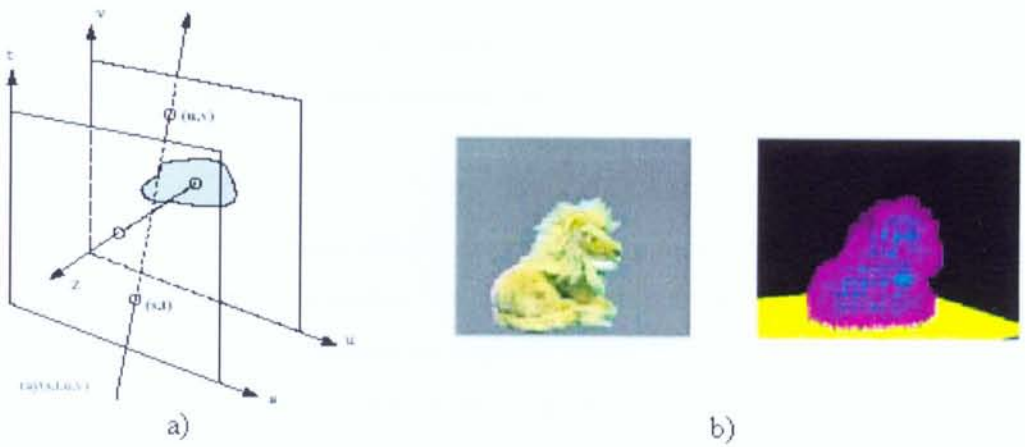
Este método parte do princípio que todos os raios são definidos por dois pontos de intersecção com dois planos. O plano de entrada corresponde a um espaço bidimensional onde todos os raios passam para intersectar o plano de saída. Portanto, o subconjunto de

todos os raios que têm origem num único ponto do plano de entrada vai dar origem à imagem observada a partir daquele ponto, como pode ser observado na Figura 5. Desta forma, obtém-se uma família de subconjuntos de raios organizados segundo o ponto no plano de entrada, que corresponde a um conjunto de imagens de referência.

As imagens de referência são adquiridas posicionando uma câmara ao longo de um plano, usando-se uma plataforma de movimento.

A imagem final é construída através da determinação das coordenadas de um determinado *pixel*, projectadas no campo de visualização, a partir dos planos de entrada e de saída. O raio desejado pode então ser armazenado na base de dados, utilizando os quatro parâmetros referidos.

Uma vez que os requisitos de armazenamento de raios numa base de dados pode ser muito grande, Levoy e Hanrahan apresentam um método de compressão com perdas que permite minimizar algumas redundâncias na representação de *light fields*.



a) b)

Figura 6: a) Parametrização de um *lumigraph*. b) Segmentação de imagens (figuras retiradas de [Gortler96])

Outro algoritmo que utiliza uma parametrização quadridimensional de raios que atravessam um par de planos com orientação fixa é a técnica *lumigraph* [Gortler96]. Esta é uma técnica de pesquisa numa base de dados de raios, muito semelhante ao *light field*. [McMillan97] diferencia estes dois algoritmos pelos métodos de aquisição utilizados e o processo final de

reconstrução. Ao contrário do *light field*, o *lumigraph* considera a geometria do modelo no processo de reconstrução das vistas desejadas, recorrendo a segmentações de imagens baseadas nas silhuetas dos objectos da cena, como pode se apresenta na Figura 6.

Outra diferença com a técnica *light field* reside no facto de ser arbitrária a posição da câmara que captura as imagens de referência a armazenar na base de dados, obrigando a um pré-processamento considerável.

A projecção de cada uma das imagens de referência caracteriza o processo de reconstrução do *lumigraph*. Quando a imagem de abertura no plano de entrada e a imagem de referência no plano de saída são projectadas como seriam observadas a partir da vista desejada, a região da imagem de referência visível da abertura pode ser visualizada na imagem desejada.

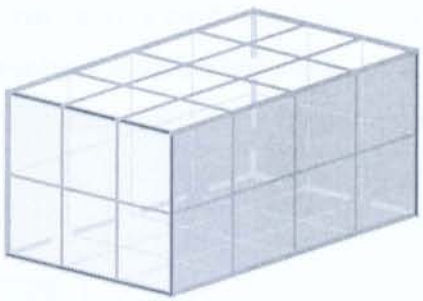
Das duas abordagens referidas, visualização aproximada e armazenamento, a primeira actualiza a visualização à medida que os raios são calculados para uma dada vista. A segunda abordagem faz um pré-processamento da cena para um conjunto pré-definido de vistas, sendo outras vistas obtidas por interpolação das vistas iniciais. Surgiu recentemente uma nova abordagem, designada por *Holodeck* [Larson98a], que combina o armazenamento e pesquisa de raios luminosos anteriormente calculados com o cálculo interactivo de novos raios.

A primeira abordagem tem como grande desvantagem o facto da informação relativa a uma vista ser perdida a partir do momento em que o observador se desloca na cena (cálculo da nova vista). A principal desvantagem da segunda abordagem reside no facto de não permitir uma actualização dinâmica dos dados. A abordagem do *Holodeck* combina as vantagens das duas anteriores, sem apresentar as suas desvantagens, na medida em que visa atingir cadências interactivas de visualização através da reutilização de raios previamente calculados, complementada com o cálculo de novos raios.

2.3 O Conceito de Holodeck

Na área da computação gráfica, cada vez mais se exige a criação de técnicas para realidade virtual que facilitem a circulação na cena (*walkthrough*), sendo crucial o tempo de

visualização de cada imagem. O facto de se navegar na cena obriga a que cada imagem seja novamente calculada, apesar de alguma informação poder ser algo coerente ou mesmo constante. Surgiu então a ideia de se armazenar a informação relacionada com os raios, por forma a não ser necessário o cálculo repetido dos mesmos, sempre que tal seja solicitado na visualização da cena. Para se atingir este objectivo, [Larson98a] desenvolveu uma técnica que designou por *Holodeck* e que se baseia numa estrutura de informação que armazena raios de luz, a fim de serem utilizados posteriormente.



3 A Técnica *Holodeck*

A técnica *Holodeck* para síntese de imagem assenta no pressuposto de que a radiância de um raio de luz é constante ao longo de todo o seu percurso[1] [Glassner95]. Dentro de uma certa tolerância em termos de direcção de um qualquer raio, será possível aproximá-lo por um outro que seja conhecido de cálculos anteriores, independentemente do facto de este último ter diferentes pontos de origem e de destino. Assim sendo, pode envolver-se parte da cena num volume que é atravessado pelos diversos raios necessários para representar uma imagem. Este volume serve de suporte ao armazenamento dos valores de radiância dos raios dessa parte da cena.

Assim, a ideia base da técnica *Holodeck* assenta na representação desse volume através um paralelepípedo imaginário, integrado na cena, através de uma estrutura de dados adequada.

---

[1] Na ausência de meios participantes.

3.1 Análise do Holodeck segundo Larson

A estrutura de informação do *Holodeck* destina-se a armazenar raios, à medida que estes vão sendo processados. Configura-se num paralelepípedo que envolve a cena, ou parte dela, sendo cada uma das suas faces subdivididas, de acordo com uma malha de quadrículas [Larson98a].

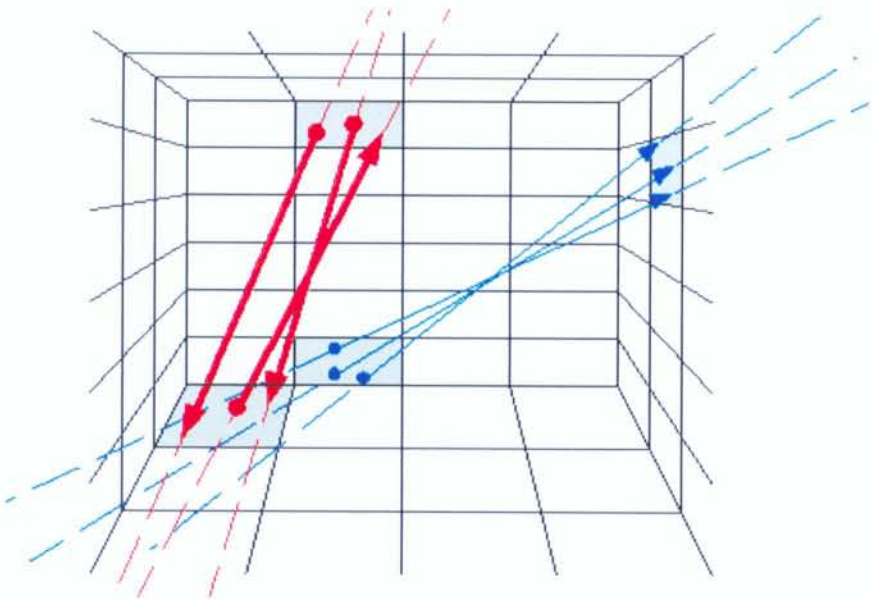


Figura 7: Vista interior do *Holodeck*

Um raio que atravesse o paralelepípedo, intersecta-o em dois pontos situados em duas faces diferentes do *Holodeck*.

A operação fundamental sobre a estrutura de informação consiste em, face a um raio novo em processamento, pesquisar o raio conhecido, que mais se lhe aproxime. Em caso de sucesso, a radiância do novo raio pode então ser aproximada pela radiância do raio encontrado na estrutura. No caso de não ser encontrado qualquer raio naquelas condições, não há informação recuperável e o novo raio deverá ser processado pelo método normal.

A vantagem do método capitaliza-se então no facto de se poderem evitar cálculos complexos de intersecção de raios com objectos, reaproveitando, tanto quanto possível, cálculos de radiância anteriormente efectuados com raios semelhantes.

Na publicação [Larson98a], que serviu de base para este trabalho, não é clara a existência de um melhoramento progressivo da imagem.

O artigo também não apresenta um estudo dos valores ideais para o número de divisões do *Holodeck* e o número de pares de quadrículas. O autor apenas refere que o número de divisões deve estar compreendido entre 4 e 24 e que o número de raios a inserir se aproxime do valor 10^6.

As respostas a estas e outras questões, assim como a obtenção de um protótipo com capacidade de evolução futura são objectivos dessa dissertação.

3.2 Um Protótipo do Holodeck

Nesta secção efectua-se a descrição de um protótipo da técnica *Holodeck*, apresentando o seu diagrama de processos e descrevendo a utilização da estrutura de dados do *Holodeck* e do seu preenchimento.

3.2.1 Diagrama de Processos

Com base no trabalho *Holodeck* proposto em [Larson98a], o sistema implementado constitui-se de quatro processos principais: visualização da imagem, gestor do cálculo de raios, gestor de *Holodeck* e motor de radiância (Figura 8).

O processo gestor do cálculo de raios encarrega-se de coordenar as operações de chamada ao motor de radiância e as operações de consulta ao processo gestor de *Holodeck*.

O processo do motor de radiância recorre à aplicação *Radiance*, a partir dos dados enviados pelo gestor de *Holodeck* (pontos de partida e direcções dos raios a processar) e implementa o método de *Ray-Tracing* de forma fisicamente válida. O resultado dos cálculos efectuados é armazenado para ser analisado pelo gestor de *Holodeck*.

O gestor de *Holodeck* organiza a pesquisa e/ou o armazenamento da informação dos raios calculados pelo motor de radiância, em memória ou em ficheiro físico. Sempre que seja possível encontrar uma resposta no *Holodeck* a um pedido de processamento de um raio, o motor de radiância não é utilizado.

As operações de visualização da imagem são comandadas por um processo de navegação, que altera os parâmetros de visualização de acordo com a actuação de dispositivos de interacção.

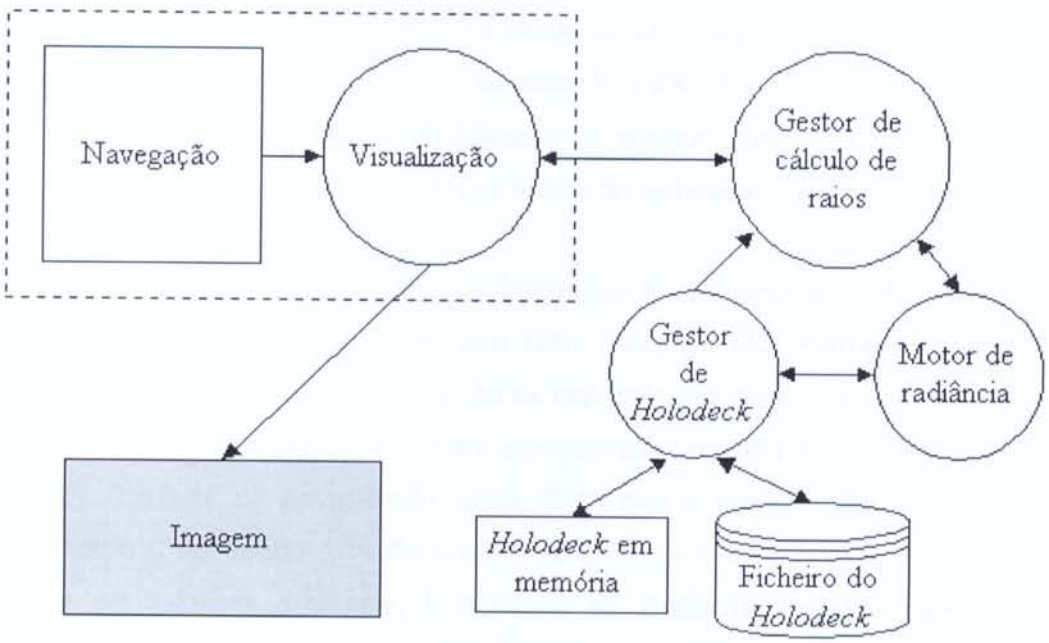


Figura 8: Diagrama de processos do sistema de *rendering* do *Holodeck*

O processo de visualização da imagem cria uma janela virtual a partir da qual pode ser visualizada a cena. Para o efeito, cria um conjunto de raios, passando cada um deles por um *pixel* e com origem na câmara. Estes raios são enviados para o gestor de cálculo de raios que lhe devolve os respectivos valores de radiância, utilizados na representação da imagem.

O sistema assim concebido é muito dependente da eficiência da pesquisa realizada sobre a estrutura do *Holodeck*. A estrutura de dados definida para armazenar a informação do *Holodeck* foi assim concebida para permitir a máxima eficácia nessa pesquisa interactiva numa cena virtual.

Um dos objectivos desta implementação é a avaliação da utilização de um motor de radiância fisicamente válido numa plataforma computacional comum (PC), em ambiente de navegação.

Nestas plataformas de capacidade de processamento modesta, tem sido comum a utilização de métodos simplificados de síntese de imagem. A utilização de técnicas de aceleração, nomeadamente do *Holodeck*, pode vir a permitir a maior divulgação de técnicas mais sofisticadas.

No entanto, entende-se que a produção de imagens por estas técnicas, à cadência de uma animação, pode ainda ser difícil. Assim, este trabalho inclui, no método de visualização, um esquema de faseamento explícito de refinamento da imagem. Este refinamento divide-se em três fases, activadas automaticamente ou por ordem do utilizador.

O método de cálculo de radiância de raios luminosos em iluminação global utilizado neste trabalho é o método de traçagem de raios (*Ray-Tracing*). Dos vários métodos de *Ray-Tracing* existentes para se realizar o cálculo da radiância dos raios luminosos, foi escolhido um dos que possui validade física mais comprovada: o *software Radiance* [Ward94], [Larson98b]. Trata-se de um método lento, dado que o número de raios luminosos a processar tende a ser muito elevado e envolve cálculos complexos. A aplicação *rtrace*, pertencente ao *software Radiance*, é o motor de radiância utilizado, que tem como parâmetros de entrada possíveis: os pontos de origem e os versores de direcção de cada raio a calcular, assim como o grau de iluminação difusa da cena. Esta aplicação produz os valores de radiância do raio calculado, o ponto de intersecção com o objecto mais próximo, a distância a esse objecto, a identificação do objecto, da sua superfície, entre outros.

3.2.2 Estrutura de Dados do *Holodeck*

As faces do paralelepípedo do protótipo do *Holodeck* são definidas por forma a serem perpendiculares aos eixos principais da cena, o que simplifica o cálculo de intersecções de raios. As seis faces do paralelepípedo são subdivididas em rectângulos ou quadrículas, sendo que o número de quadrículas pode não ser igual nas três dimensões, como se observa na Figura 9.

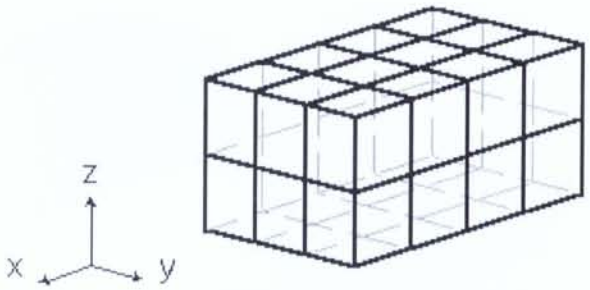


Figura 9: Vista exterior do *Holodeck*

Consideram-se face e quadrícula de entrada, aquelas que contêm o primeiro ponto de intersecção e face e quadrícula de saída, aquelas que contêm o segundo ponto de intersecção do raio em causa, segundo a direcção e sentido do mesmo.

O interesse da determinação das quadrículas de entrada e de saída dos raios reside no facto de todos os raios que partilham as mesmas quadrículas, de entrada e de saída, poderem ser agrupados e classificados de forma semelhante. O armazenamento destes raios "quase paralelos" é, assim efectuado de forma a facilitar a sua pesquisa.

De acordo com [Larson98a], os raios são catalogados pelo par de quadrículas, de entrada e de saída, segundo a sua direcção, mas não segundo o seu sentido. Isto significa que dois raios, quase paralelos mas com sentidos contrários, que partilhem as mesmas quadrículas de entrada e de saída, são classificados da mesma forma. Como exemplo, apresenta-se na Figura 7, um conjunto de raios assinalados a traço grosso que partilha o mesmo par de quadrículas, mas o sentido de um dos raios é inverso dos restantes. A distinção do sentido de cada raio é conseguida através do armazenamento em duas listas associadas a cada par de quadrículas. Numa das listas são armazenados os raios no sentido directo e na outra são armazenados os raios no sentido inverso.

Conforme descrito na secção 3.1, o *Holodeck* é representado por um paralelepípedo. Na aplicação implementada, cada uma das suas faces é identificada através de uma numeração sequencial de 1 a 6, de acordo com a Figura 10. Nesta figura pode visualizar-se, em a) o *Holodeck* visto do exterior, sendo visíveis as faces 1, 2 e 3. Em b) estão representadas apenas as faces restantes, o que corresponde a observar o *Holodeck* a partir do seu interior.

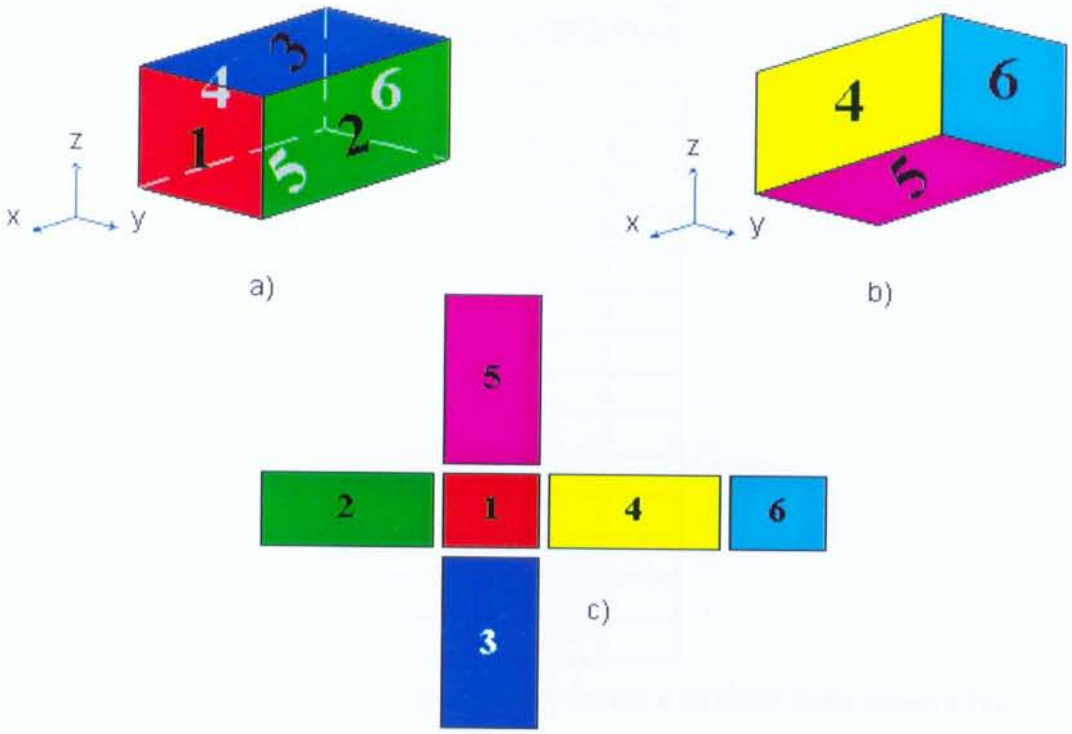


Figura 10: Identificação de cada uma das faces do *Holodeck* a) Vista frontal do *Holodeck* b) Vista traseira do *Holodeck* c) Método europeu de projecção

Para se armazenar um raio, é necessária a determinação do par de faces, de entrada e de saída, que são intersectadas pelo raio[2]. Na concepção da estrutura de dados adequada à representação do *Holodeck*, consideram-se assim todas as combinações possíveis para "faces de entrada" e "faces de saída". Cada uma das seis faces pode combinar-se com as restantes cinco, pelo que existem trinta combinações possíveis de pares "face de entrada"/"face de saída". Dado que o sentido dos raios não é tido em conta, o número de combinações reduz-se para metade.

Assim, a estrutura de dados do *Holodeck* baseia-se num vector com quinze posições (Figura 11). Cada elemento deste vector contém apontadores para uma estrutura que representa o par de quadrículas em que o raio intersecta as faces do *Holodeck*.

---

[2] Dado que as faces são planas, não é possível que um raio entre e saia na mesma face.

	Faces Intersectadas		...
1	1	2	...
2	1	3	...
3	1	4	...
4	1	5	...
5	1	6	...
6	2	3	...
7	2	4	...
8	2	5	...
9	2	6	...
10	3	4	...
11	3	5	...
12	3	6	...
13	4	5	...
14	4	6	...
15	5	6	...

Figura 11: Relação entre a posição no vector e as duas faces associadas

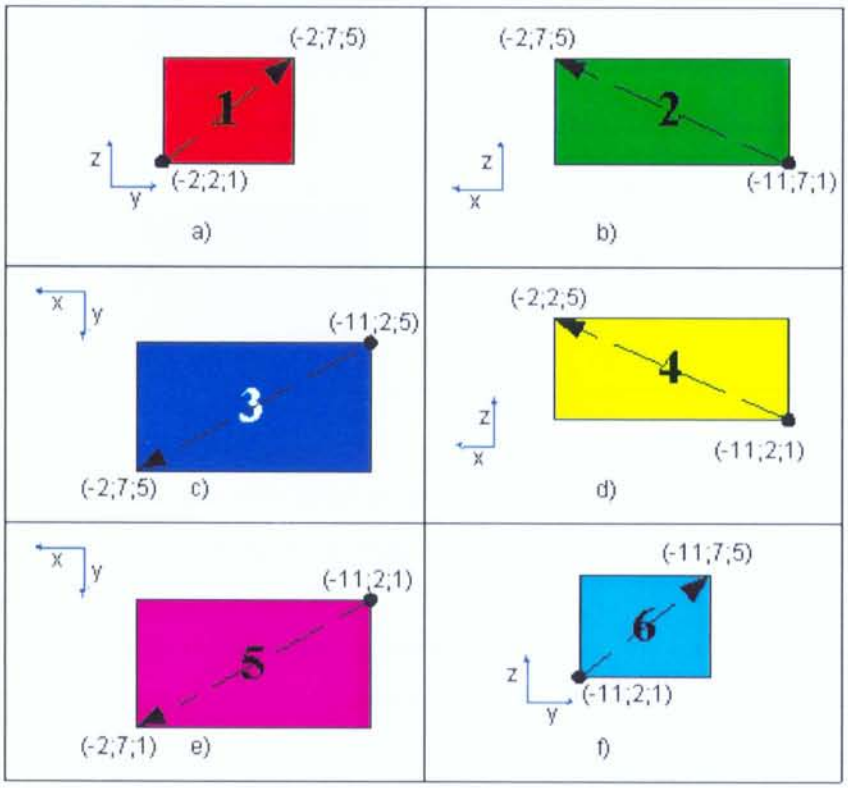


Figura 12: Projecção das faces do *Holodeck*

a) Face 1 b) Face 2 c) Face 3 d) Face 4 e) Face 5 f) Face 6

A conversão da representação de um paralelepípedo em três dimensões para a representação das suas faces em duas dimensões apresenta-se na Figura 12 e, em cada face, apresenta-se o plano correspondente do seu sistema de coordenadas. O sentido crescente das coordenadas (a flecha apresentada em cada uma das faces da Figura 12) é definido a partir da localização do menor ponto do *Holodeck*. As faces 1 e 6 são paralelas ao plano *yz*, as faces 2 e 4 são paralelas ao plano *xz* e as faces 3 e 5 são paralelas ao plano *xy*.

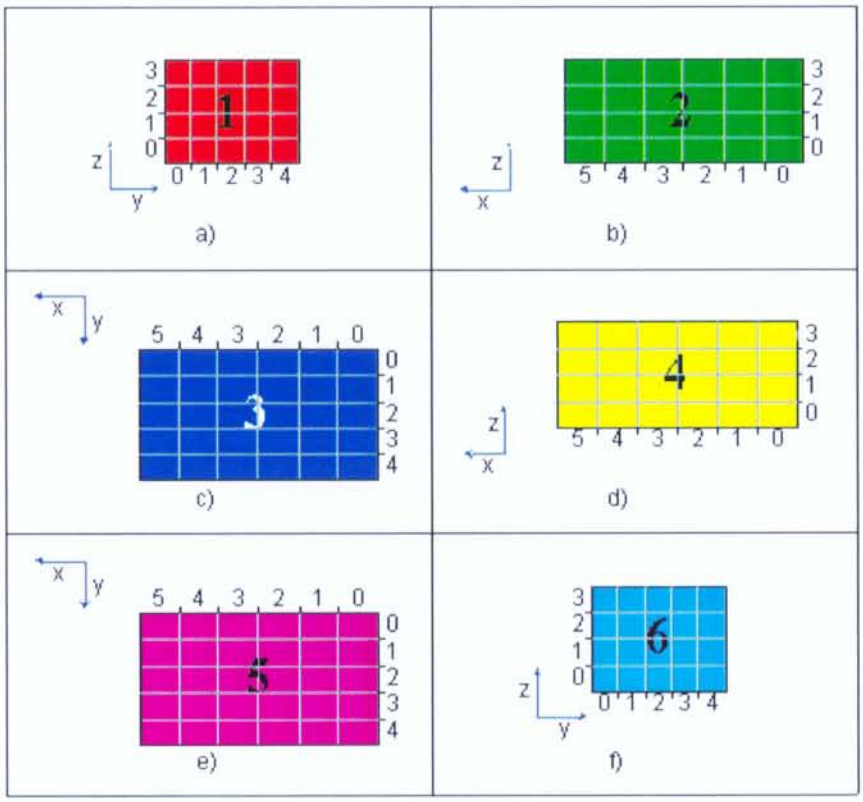


Figura 13: Projecção das faces do *Holodeck* com as respectivas divisões
a) Face 1 b) Face 2 c) Face 3 d) Face 4 e) Face 5 f) Face 6

Na Figura 13, o número de quadrículas segundo as direcções dos eixos coordenados são: 6, 5 e 4, respectivamente para *x*, *y* e *z*. Cada quadrícula é numerada sequencialmente, seguindo o sentido crescente das coordenadas dos respectivos eixos, ou seja, é identificada pelos valores representativos da coluna e da linha em que se encontram na respectiva face. Dado que, numa quadrícula, podem passar vários raios, torna-se ainda necessário caracterizar a posição destes dentro dessa quadrícula, isto é, determinar as coordenadas locais do ponto de

intersecção. Estas coordenadas são normalizadas para o intervalo [0, 1[, a fim de facilitar a redução de espaço de memória a ocupar pelas referidas coordenadas.

Assim, tendo também em atenção a informação de radiância que caracteriza o raio, a informação que é necessário armazenar por raio, na estrutura anterior é:

- Componente de radiância vermelha, verde e azul;
- Distância em x ao canto superior esquerdo da quadrícula de entrada;
- Distância em y ao canto superior esquerdo da quadrícula de entrada;
- Distância em *x* ao canto superior esquerdo da quadrícula de saída;
- Distância em *y* ao canto superior esquerdo da quadrícula de saída.

A informação contida no *Holodeck* é pesquisada pelo processo gestor de *Holodeck* sempre que é necessária a determinação da radiância de um raio que o atravessa. Tenta-se encontrar, na estrutura de dados anterior, um raio previamente calculado que seja suficientemente semelhante, do ponto de vista geométrico, ao raio a calcular.

Assim, o primeiro passo a dar no processo de pesquisa consiste na determinação da intersecção do raio a calcular com o volume do *Holodeck*. Se a intersecção não existe, o raio é enviado pelo processo gestor de cálculo de raios, para processamento no motor de radiância, dado que a estrutura do *Holodeck* não tem, garantidamente, nenhuma informação relevante para fornecer.

Um raio é também enviado para o motor de radiância quando intersecte o *Holodeck* num só ponto (caso das arestas e dos vértices) ou quando as duas quadrículas, de entrada e de saída, partilhem uma aresta do *Holodeck*, (quadrículas contíguas mas de faces distintas). Desta forma evitam-se, no primeiro caso, erros de representação numérica e, no segundo, perdas de tempo de pesquisa em situações que, na maioria das vezes, não encontram raios semelhantes (dado que a grande variabilidade de direcção dos raios armazenados naquelas circunstâncias raramente torna possível a adopção de um deles como aproximação ao raio em questão).

Quando um raio a processar intersecta o *Holodeck*, consulta-se a estrutura, com base nas faces e quadrículas correspondentes, para determinar quais os raios calculados aí existentes. Estes são analisados para se determinar se existe e qual é o raio geometricamente mais próximo do raio em processamento. Este último recebe o valor da radiância do raio

encontrado na pesquisa ou, na sua ausência, é passado para o motor de radiância, para o seu cálculo exaustivo por *Ray-Tracing*.

Partindo do princípio que o tempo de pesquisa de um raio na estrutura do *Holodeck* é significativamente inferior ao tempo de processamento pelo motor de radiância e que é grande a probabilidade de um raio ser encontrado, conclui-se que o método apresenta grandes potencialidades em termos de velocidade de geração de imagens.

Ficam assim por analisar os efeitos das diferentes parametrizações da estrutura de dados (nº de quadrículas, pré-preenchimento, etc.), quer na velocidade de processamento, quer na qualidade das imagens finais.

3.2.3 Pré-preenchimento do *Holodeck*

O sistema será tanto mais rápido quantos mais raios estiverem armazenados na estrutura de dados do *Holodeck*, uma vez que, não havendo sucesso na consulta ao *Holodeck* para avaliação de um raio, torna-se necessário recorrer ao motor de radiância.

Assim, à medida que o motor de radiância fornece novos valores de radiância pedidos pelo processo de visualização, estes são inseridos na estrutura do *Holodeck*, para posterior utilização.

Numa situação de utilização em *walkthrough*, o utilizador desloca-se na cena virtual por pequenas distâncias entre duas imagens consecutivas. Desta forma, se uma percentagem significativa de radiâncias calculadas na imagem anterior tiverem sido armazenadas, existe uma grande probabilidade de estas serem encontradas na estrutura de dados do *Holodeck*.

Extrapolando o exposto, conclui-se da utilidade do preenchimento do *Holodeck*, previamente ao processo de construção da imagem, ou seja, como pré-processamento.

O processo de preenchimento do *Holodeck* em pré-processamento baseia-se na geração, à *priori*, de um certo número de raios que intersectem o *Holodeck*. Para o efeito, definem-se dois pontos aleatórios que pertençam ao *Holodeck* e, em seguida, faz-se passar um raio por

esses pontos. A informação de radiância correspondente é calculada pelo motor de radiância e é armazenada no *Holodeck*.

Em seguida descrevem-se vários métodos para realizar o pré-preenchimento do *Holodeck*.

3.2.3.1 Método Completamente Aleatório

Para pré-preencher o *Holodeck* através de um método completamente aleatório, a geração de cada raio processa-se da seguinte forma:

- Selecciona-se aleatoriamente um par de faces (Figura 14 a);
- Determina-se qual das duas faces corresponde à "face de entrada" e à "face de saída", respectivamente;
- Determina-se a qual dos planos do sistema de eixos (xy, xz ou yz) a "face de entrada" é paralela (Figura 14 b);
- Idem para a "face de saída";
- Atendendo ao número de divisões de cada face, gera-se aleatoriamente uma quadrícula da "face de entrada" (Figura 14 c);
- Idem para a "face de saída";
- Determinam-se dois valores aleatórios, pertencentes ao intervalo [0;1[, para cada uma das duas quadrículas anteriores. Estes valores servem como coordenadas normalizadas locais, de um ponto situado em cada uma das quadrículas;
- Traça-se um raio entre os dois pontos assim definidos, no sentido da quadrícula de entrada para a de saída (Figura 14 d);
- Uma vez processado o raio assim obtido, pelo motor de radiância, envia-se a informação correspondente para o gestor de *Holodeck*, que a armazena de acordo com as faces e quadrículas correspondentes, juntamente com a informação de coordenadas locais normalizadas e de radiância.

3.2.3.2 Método Sequencial Total

Um outro método de preenchimento, sequencial, pode ser utilizado quando se pretende o preenchimento total do *Holodeck*. Neste caso, não se seleccionam as quadrículas do *Holodeck* de forma aleatória, pois quando é ultrapassada uma certa percentagem de

preenchimento, a probabilidade de acertar nos pares de quadrículas ainda não preenchidas torna-se muito baixa. Assim sendo, percorre-se sequencialmente toda a estrutura, para garantir que todas as quadrículas são visitadas. Por cada par de quadrículas geram-se todos os raios pretendidos.

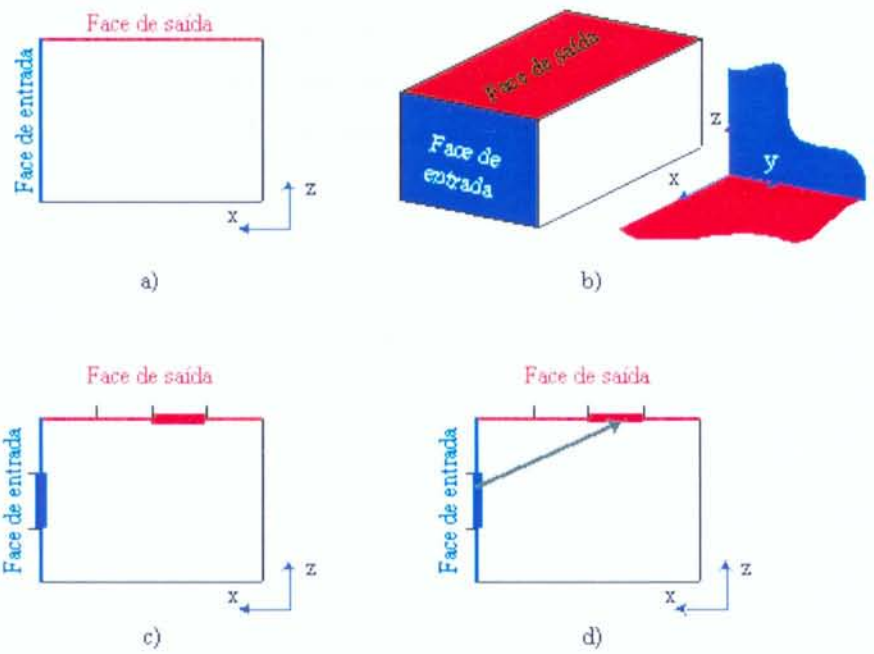


Figura 14: Preenchimento do *Holodeck* em pré-processamento
a) Selecção aleatória de duas faces. b) Determinação de paralelismo entre faces e planos principais. c) Determinação aleatória da divisão para cada dimensão das faces. d) Traçar raio.

3.2.3.3 Método Sequencial Parcial Controlado

Uma variante deste método permite o preenchimento parcial do *Holodeck*, baseando-se na estratégia de enchimento aleatório controlado. Quando se selecciona um par de quadrículas, pela mesma sequência do método anterior, gera-se um valor aleatório entre 0 e 100. Este é comparado com um valor de limiar (igual à percentagem de *Holodeck* a preencher) e decide-se se é ou não gerado o raio correspondente.

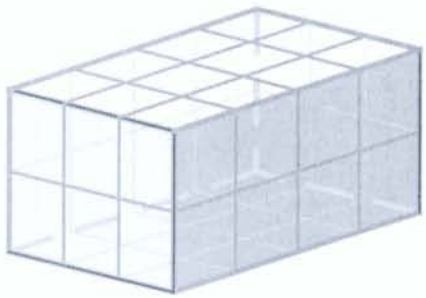
3.2.3.4 Método Direccional Parcial Controlado

Outro método de preenchimento do *Holodeck* permite armazenar apenas os raios que atravessam os pares de faces que se encontram na direcção do observador, isto é, pares de faces visíveis/invisíveis. A principal vantagem deste método é a produção de um *Holodeck* que contém amostras significativas para as posições iniciais de observação da cena, permitindo navegações iniciais rápidas. No entanto, este método apresenta a desvantagem de não ser eficiente na visualização de imagens cuja direcção de observação difira muito da direcção pré-definida.

Qualquer dos métodos anteriores pode ser utilizado para preencher o *Holodeck* em pré-processamento. No entanto, a estrutura do *Holodeck* pode ser actualizada durante o processo de visualização, à medida que se navega na cena virtual (secção 4.4).

3.3 Síntese do Capítulo

Neste capítulo descreve-se a técnica *Holodeck* em termos do seu diagrama de processos, de uma estrutura de dados subjacente e dos mecanismos de determinação da intersecção de um raio com o *Holodeck*. Finaliza-se este capítulo com a descrição de vários métodos alternativos de pré-preenchimento de um *Holodeck*.



4 Implementação do Protótipo

O protótipo desenvolvido permite visualizar imagens resultantes da navegação na cena. Neste capítulo será descrita uma implementação deste protótipo formado por vários blocos funcionais, que se organizam de acordo com o fluxograma da Figura 15.

A execução do protótipo inicia-se com a leitura do ficheiro de parametrização definido pelo utilizador. Em seguida existe a possibilidade de se proceder ao preenchimento inicial do *Holodeck* através da leitura de um ficheiro que contém uma estrutura de dados de um *Holodeck* preenchido numa sessão anterior do protótipo e/ou através do pré-preenchimento do *Holodeck* por uma das técnicas referidas na secção 3.2.3. Na fase seguinte, geram-se raios que partem da posição do observador e passam pelos *pixels* da imagem a produzir. Sobre a grelha de radiâncias resultante, efectua-se uma interpolação dos *pixels* sem informação e aplica-se um operador de mapeamento de tons, resultando uma imagem da cena. A partir desse momento, o utilizador tem a possibilidade de refinar a imagem visualizada ou navegar interactivamente na cena, com a consequente produção de novas imagens.

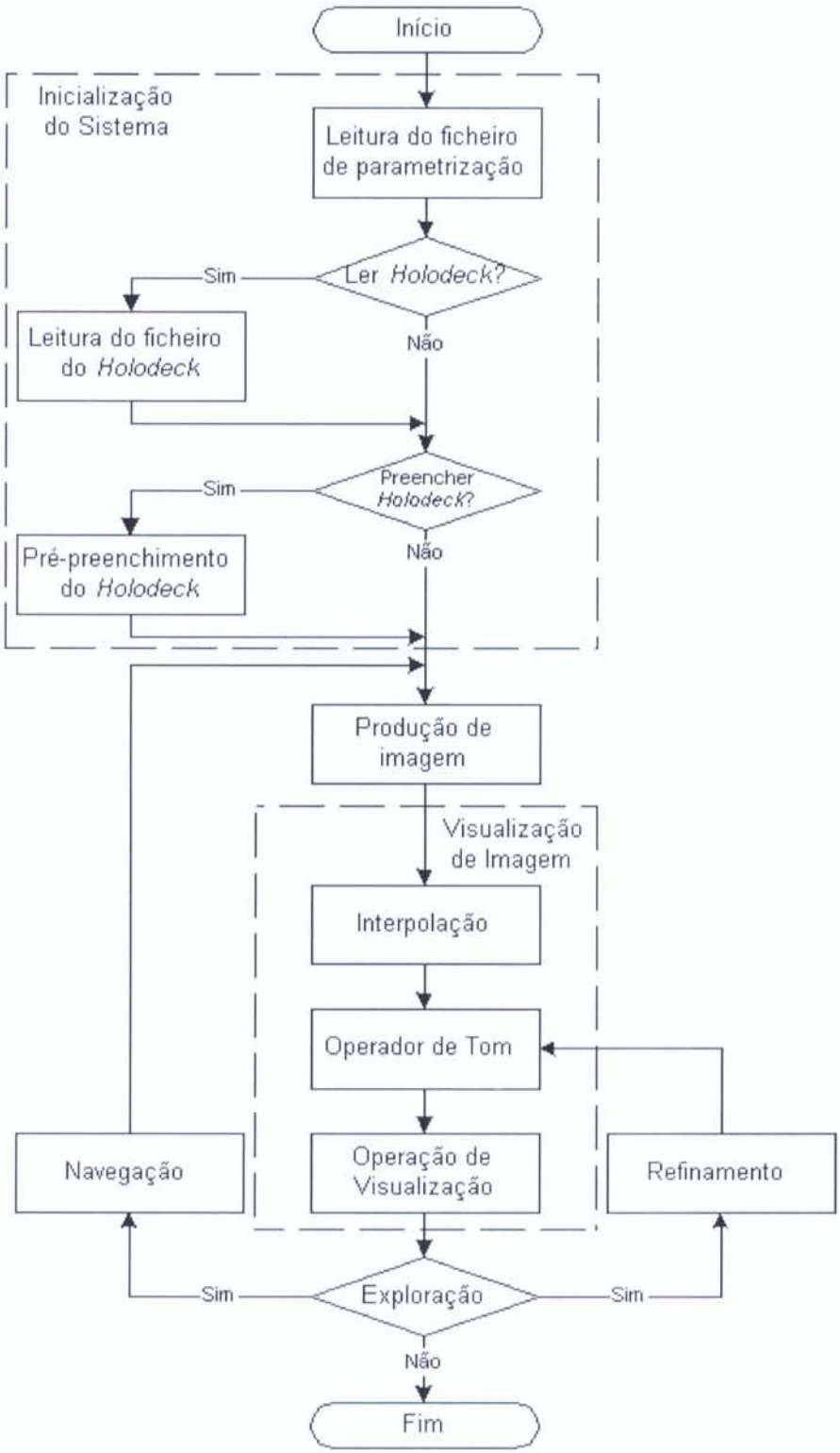


Figura 15: Fluxograma do funcionamento do protótipo

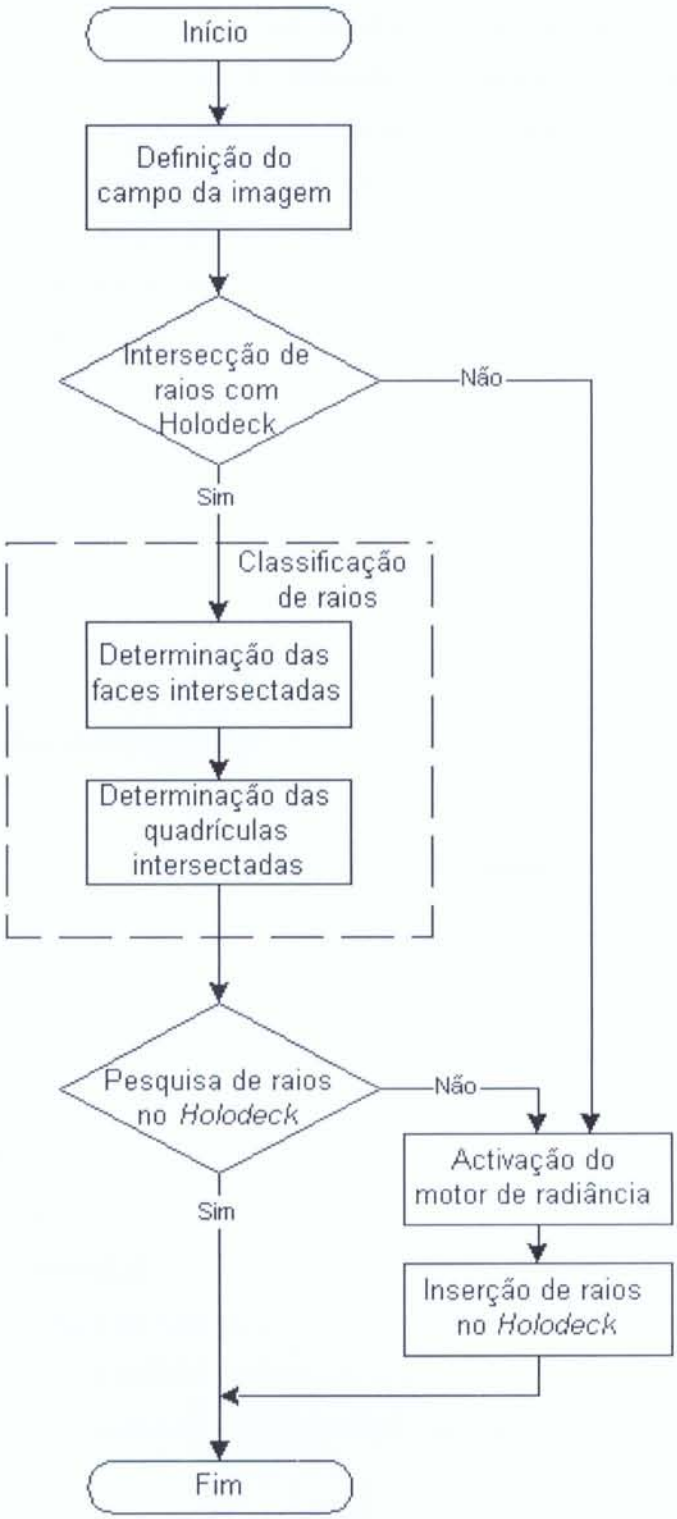


Figura 16: Fluxograma do processo de produção da imagem

O fluxograma da Figura 16 apresenta, em detalhe, o processo de produção da imagem do fluxograma anterior. Inicia-se com a definição do campo da imagem, seguindo-se a classificação dos raios que atravessam o campo de visualização e que intersectam o volume do *Holodeck*. Esta classificação é efectuada pela determinação das faces intersectadas do *Holodeck* e das respectivas quadrículas. Seguidamente, realiza-se a pesquisa dos raios da imagem na estrutura de dados do *Holodeck* e, no caso de insucesso, na pesquisa ou na intersecção dos raios com o *Holodeck*, activa-se o motor de radiância com posterior inserção dos raios na estrutura de dados do *Holodeck*.

Nas secções seguintes descrevem-se os principais componentes do fluxograma da Figura 15, tais como: Inicialização do Sistema, Produção de Imagem, Visualização da Imagem e Exploração Interactiva.

4.1 Inicialização do Sistema

O primeiro passo a realizar na inicialização do sistema é a leitura do ficheiro de parametrizações.

4.1.1 Leitura do Ficheiro de Parametrização

As parametrizações que estão contidas neste ficheiro podem ser divididas em vários blocos:

- Definição do modelo da cena;
- Definição de *Holodeck*[3];
- Definição do campo de visualização;
- Número máximo de pedidos a enviar ao motor de radiância;
- Opções disponibilizadas ao utilizador para realização de algumas acções contidas na aplicação;

---

[3] Apesar do estudo realizado nesta dissertação ter incidido sobre o desempenho de um *Holodeck*, a implementação foi orientada para suportar vários *Holodecks* em simultâneo.

- Parametrização dos valores de deslocação na cena;
- Parametrização do cálculo de proximidade do raio a pesquisar com os raios armazenados na estrutura de dados do *Holodeck*.

A designação da cena (`cena`) a ser processada pelo protótipo de implementação constitui o bloco inicial de informação definida pelo utilizador.

A definição de *Holodeck* é conseguida pelas coordenadas dos dois pontos que definem o volume tridimensional do *Holodeck* (`menorPontoHolodeck` e `maiorPontoHolodeck`), pelo número de quadrículas do *Holodeck* em cada uma das três direcções ortogonais (`nº de divisões em x, y e z`) e pelo número de raios a armazenar por cada par de quadrícula (`nRaiosParQuadrícula`). Neste bloco também é inserida a localização da directoria onde é lido e/ou gravado o ficheiro do *Holodeck* e onde serão colocados todos os ficheiros de saída da aplicação.

O bloco seguinte é constituído por um conjunto de parâmetros que permitem definir o campo de visualização. Estes parâmetros são: como usualmente [Glassner95], a posição do observador (`observador`); a posição do alvo (`alvo`); o vector de verticalidade (`vectorVerticalidade`); o número de *pixels* em x e em y do campo de visualização; o ângulo de abertura em x (`ângulo de abertura em x`). O ângulo de abertura em y é calculado, visto que existe uma relação directa entre o número de *pixels* em x e o número de *pixels* em y que tem de ser mantida.

O número máximo de pedidos (secção 4.2.4) a colocar no pacote de pedido de raios a enviar ao motor de radiância (`nMáximoPedidoRaios`).

O utilizador pode, em seguida, especificar se deseja que a aplicação accione ou não o *Holodeck* (`accionarHolodeck`). Quando o utilizador opta por não inserir o *Holodeck*, não são realizados os cálculos de intersecção de raios com o volume do *Holodeck* e ignora-se a pesquisa à estrutura de dados. Os processos de leitura e de gravação do *Holodeck* também são opcionais, assim como o processo de preenchimento prévio do *Holodeck*, que é realizado até uma percentagem de preenchimento atribuída pelo utilizador (`percentagemNRaios`).

O próximo bloco relaciona-se com interacção e permite ao utilizador definir a quantidade de deslocamento e o ângulo de rotação na cena a cada pressão das teclas de movimento. O deslocamento é efectuado simultaneamente sobre os pontos da posição do observador e do alvo, para a frente, para trás, para a direita ou para a esquerda. A rotação é efectuada sobre o alvo em torno da posição do observador, ou em torno do alvo.

Durante a produção de uma imagem, nos casos em que nem todos os *pixels* têm valores de radiância atribuídos, é realizada uma interpolação de radiâncias para mais tarde se obter uma aproximação do valor da radiância desse *pixel*. O número de *pixels* vizinhos necessários para realizar esta interpolação é parametrizado no último bloco do ficheiro de configurações.

Este bloco permite também parametrizar o processo de refinamento da imagem. São disponibilizadas três fases de refinamento (descritas na secção 4.4), associadas ao cálculo de proximidade de raios, que neste ficheiro de configuração se associam ou não, automaticamente. É possível definir limites mínimos (`limiteMinProximidadeFase1` e `2`) para o valor resultante do cálculo de proximidade do raio a pesquisar com os raios armazenados, para finalizar o automatismo das duas fases, caso tenham sido activadas. A fase 2 é processada iterativamente, sendo cada iteração calculada em função de uma percentagem definida pelo utilizador.

Todos os valores de parametrização anteriores podem ser alterados durante a execução da aplicação através da opção de menu "Configurações".

4.1.2 Criação da Estrutura de Dados

Conhecidas as parametrizações do utilizador, procede-se às inicializações necessárias, nomeadamente a constituição do *Holodeck* e seu preenchimento com os raios adequados (conhecidos anteriormente ou gerados no momento).

Como referido na secção 3.2.3 anteriormente, cada uma das faces do *Holodeck* é classificada através de um índice de 1 a 6. Cada raio inserido no *Holodeck* identifica-se pelo par de faces com menor e maior índice. De salientar que a face de entrada pode não ser a face com menor índice comparativamente com o da face de saída.

Em seguida, relaciona-se o par de faces com cada um dos quinze índices primários da estrutura de dados do *Holodeck* (Figura 11).

A cada índice primário estão relacionadas células que representam os pares de quadrículas intersectadas. Cada célula contém duas listas de raios: uma no sentido directo (da face com menor identificação para a face com maior identificação) e outra no sentido inverso.

Uma vez que a quantidade de informação contida na estrutura de dados do *Holodeck* pode atingir grandes proporções, torna-se absolutamente necessário que essa estrutura seja de acesso rápido, de forma a minimizar as consequências temporais das acções de pesquisa a efectuar.

Na secção seguinte propõe-se uma estrutura de dados a utilizar neste protótipo.

4.1.2.1 Estrutura de Dados Baseada num Vector

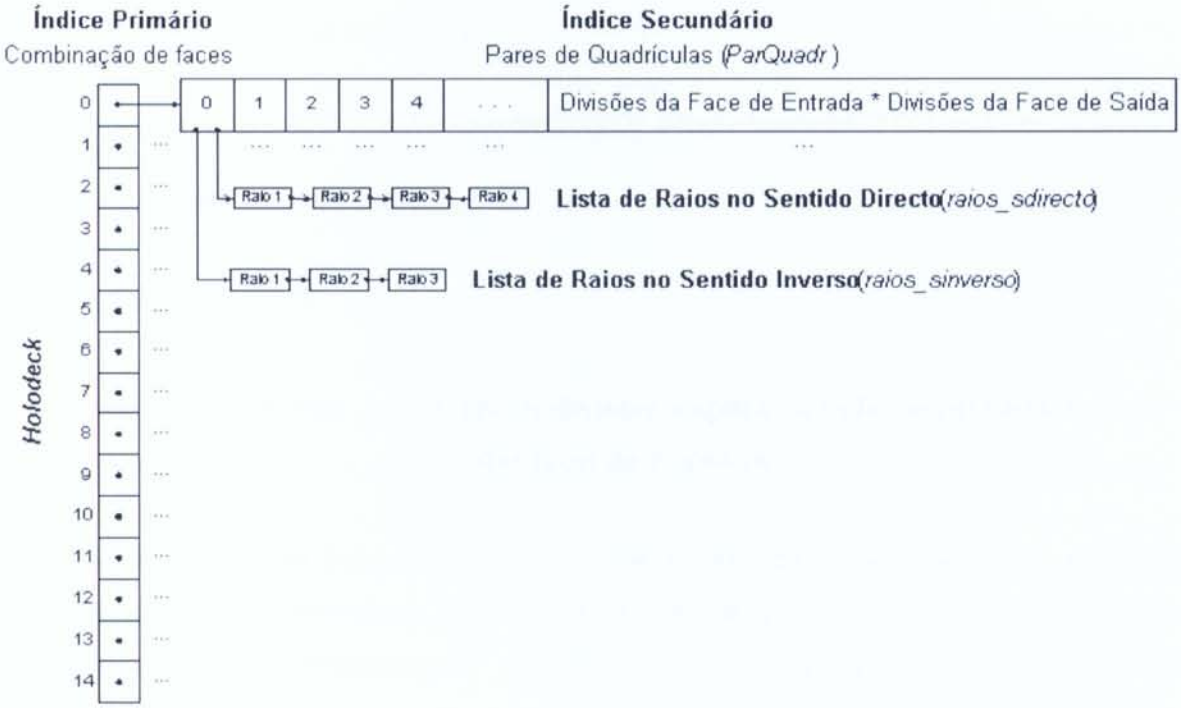


Figura 17: Estrutura de dados do *Holodeck*

A Figura 17 ilustra a estrutura de dados adoptada para o *Holodeck*. Como se pode ver, a cada célula do vector primário, corresponde um vector secundário que contém apontadores para duas listas de raios. Conforme referido na secção 3.2.2, estas listas permitem armazenar os raios no sentido directo e inverso. O tipo de listas adoptado é *FIFO*[4], o que permite uma actualização permanente de raios através da troca de raios antigos pelos mais recentes.

Dado que cada face é convertida para um sistema de duas coordenadas locais (designadas por coordenada 1 e 2), utiliza-se uma função que determina o número de divisões a aplicar a uma das coordenadas de uma dada face do *Holodeck*. Apresentou-se um exemplo de divisão das faces na Figura 13 e descreve-se em seguida a função que efectua essa divisão.

Função ***NDivisões*** *(face, coordenada)*

Se face=2 e coordenada=1 Ou face=3 e coordenada=1 Ou face=4 e coordenada=1 Ou face=5 e coordenada=1 Então

nº de divisões ← nº de divisões em x

Fim Se

Se face=1 e coordenada=1 Ou face=3 e coordenada=2 Ou face=5 e coordenada=2 Ou face=6 e coordenada=1 Então

nº de divisões ← nº de divisões em y

Fim Se

Se face=1 e coordenada=2 Ou face=2 e coordenada=2 Ou face=4 e coordenada=2 Ou face=6 e coordenada=2 Então

nº de divisões ← nº de divisões em z

Fim Se

Devolver (nº de divisões)

Fim Função

Algoritmo 1: Atribuição do número de divisões a aplicar a cada coordenada de cada uma das faces do *Holodeck*

Aquando da intersecção de um raio com o *Holodeck*, determinam-se a face intersectada e o ponto de intersecção. Sabendo-se o número de divisões a aplicar a cada coordenada da face intersectada (função anteriormente descrita), é necessário determinar a quadrícula

[4] O tipo de lista *FIFO* (*First In First Out*) caracteriza-se por o primeiro elemento a entrar ser o primeiro a sair.

intersectada nessa face. Esta quadrícula é definida por dois índices de divisão da face, que são calculados pelo valor inteiro da razão entre a distância do ponto de intersecção ao menor ponto que define o *Holodeck* e a dimensão da quadrícula (para a coordenada correspondente). Este procedimento traduz-se no algoritmo seguinte.

```
Função IDDivisão (face, pontoInt)
    /* determinação do sistema local de duas coordenadas da face intersectada */
    3Dpara2D ← { {y,z}, {x,z}, {x,y}, {x,z}, {x,y}, {y,z} }
    coord1 ← 3Dpara2D[face][1]
    idDivisãoX ← inteiro( (pontoInt.coord1 - menorPontoHolodeck.coord1) * (maiorPontoHolodeck.coord1 -
      menorPontoHolodeck.coord1) ) / NDivisões(face, 1) )
    coord2 ← 3Dpara2D[face][2]
    idDivisãoY ← inteiro( (pontoInt.coord2 - menorPontoHolodeck.coord2) * (maiorPontoHolodeck.coord2 -
      menorPontoHolodeck.coord2) ) / NDivisões(face, 2) )
    Devolver (idDivisãoX, idDivisãoY)
Fim Função
```

Algoritmo 2: Determinação do número da divisão que contém um ponto de intersecção

O índice secundário é sequencial para cada par de quadrícula e é determinado através do número de divisões em cada coordenada e do número da divisão do ponto de intersecção (Algoritmo 1 e Algoritmo 2). Para cada combinação de faces do *Holodeck*, a numeração a atribuir ao último par de quadrículas equivale ao produto do número de divisões na face de entrada pelo número de divisões da face de saída, conforme se ilustra na Figura 17.

4.1.2.2 Número Total de Raios Suportado pela Estrutura de Dados

Seja um *Holodeck* definido por um número de divisões em largura (l), profundidade (p) e altura (a). Para determinar o número total de pares de quadrículas, calcula-se o somatório do número de quadrículas por par de faces. Este resultado deve então ser multiplicado por dois, visto que os raios podem ser armazenados nos dois sentidos no *Holodeck*.

Considerando-se o exemplo das faces 1 e 2 e atendendo a que a face 1 é divida em la quadrículas e a face 2 é divida em pa quadrículas, o número de pares de quadrículas é $lapa$. Extrapolando este cálculo para os restantes 14 pares de faces, obtém-se o seguinte resultado para o número total de pares de quadrículas:

$$NParesQuadriculas = 2\,(lapa + lalp + lapa + lalp + lala + palp + papa + palp + pala + lppa + lplp + lpla + palp + pala + lpla)$$
$$= 2l^2p^2 + 2l^2a^2 + 2p^2a^2 + 8l^2pa + 8lp^2a + 8lpa^2 \quad (3)$$

Multiplicando o valor obtido na equação anterior pelo número de raios por par de quadrículas (r) parametrizado pelo utilizador, obtém-se a seguinte equação:

$$NTotalRaios = r(2l^2p^2 + 2l^2a^2 + 2p^2a^2 + 8l^2pa + 8lp^2a + 8lpa^2) \quad (4)$$

O número de raios cresce exponencialmente com a subdivisão das faces, pelo que se justifica encontrar um compromisso entre o grau de subdivisão de faces em quadrículas e o número de raios a guardar por par de quadrículas, a fim de evitar requisitos muito elevados de memória.

4.1.3 Obtenção Inicial do *Holodeck*

Os raios inicialmente introduzidos na estrutura de dados do *Holodeck*, podem ser obtidos pela leitura de um *Holodeck* anteriormente preenchido. Esta possibilidade não é única, uma vez que estes raios também podem ser adquiridos pelo preenchimento prévio do *Holodeck*. As duas possibilidades podem ser complementares, o que resulta no acréscimo de raios armazenados na estrutura do *Holodeck* inicial, para além dos raios pertencentes ao *Holodeck* lido.

O preenchimento recorre a uma percentagem da totalidade do número de raios armazenáveis no *Holodeck* ou a uma percentagem do número de raios que intersectam todas as faces visíveis do *Holodeck* para contabilizar o número máximo de raios a inserir na estrutura. Estas percentagens são definidas pelo utilizador, como atrás referido, e os métodos da geração de raios são descritos na secção 3.2.3.

Para efeito de teste, o sistema permite a navegação na cena consoante desígnio do utilizador. A aplicação implementada permite calcular os raios necessários para visualizar o percurso em pré-processamento com consequente inserção dos raios calculados na estrutura do *Holodeck*. Desta forma, no processo de navegação, o utilizador tem a oportunidade de viajar

na cena segundo o percurso por si definido, sem ter que recorrer ao moroso processo de cálculo de radiância de raios.

4.2 Produção de Imagem

A produção da imagem compõe-se, de acordo com a Figura 16, de definição do campo de visualização e de classificação dos raios que atravessam o campo de visualização e que intersectam o *Holodeck*. Os raios, depois de classificados são pesquisados na estrutura de dados do *Holodeck* e, caso não se encontrarem nesta estrutura, torna-se necessário activar o motor de radiância e inserir os raios calculados na estrutura de dados do *Holodeck*, para posterior reutilização.

4.2.1 Definição do Campo de Visualização

A partir da posição do observador e do alvo (definidos pelo utilizador), é normalizado o vector da direcção de observação. A direcção em x do plano de visualização é calculada através do produto vectorial resultante dos dois vectores normalizados da direcção da observação e de verticalidade (também definido pelo utilizador). Aplicando a mesma operação ao vector resultante normalizado e ao versor de direcção de observação, obtém-se a direcção em y do plano de visualização.

O ângulo de abertura em y não é definido pelo utilizador, uma vez que pode ser calculado em função da proporcionalidade entre o número de *pixels* em x e o número de *pixels* em y e o ângulo de abertura em x. Assim, o dobro da tangente do ângulo de abertura em x define a medida horizontal do campo de visualização que, dividido pelo número de *pixels* em x (definido pelo utilizador) resulta na distância normalizada entre dois *pixels* horizontais consecutivos no campo de visualização. O mesmo cálculo é realizado para a distância normalizada entre dois *pixels* verticais consecutivos. Todas estas operações estão representadas no Algoritmo 3.

Conhecida a distância normalizada entre *pixels* em x e em y, é possível determinar as coordenadas de qualquer *pixel* no campo de visualização pela função seguinte.

```
Procedimento DistânciaEntrePixels ()
    direcção observação ← alvo – observador
    distância observador ← distância do alvo ao observador
    normalizar (direcção observação)
    normalizar (vectorVerticalidade)
    direcção em x ← produtoVectorial (direcção observação, vectorVerticalidade)
    normalizar (direcção em x)
    direcção em y ← produtoVectorial (direcção observação, direcção em x)
    normalizar (direcção em y)
    ângulo de abertura em y ← arcotangente[(nº pixels em y/nº pixels em x)*tangente(ângulo de abertura em x)]
    distância normalizada entre pixels em x ← 2 * tangente(ângulo de abertura em x) / nº pixels em x
    distância normalizada entre pixels em y ← 2 * tangente(ângulo de abertura em y) / nº pixels em y
Fim de procedimento
```

Algoritmo 3: Cálculo da distância entre *pixels* no campo de visualização

```
Função CálculoPixel (x, y)
    kx ← x * distância normalizada entre pixels em x – tangente (ângulo de abertura em x)
    ky ← y * distância normalizada entre pixels em y – tangente (ângulo de abertura em y)
    ponto ← observador + distância observador * direcção observação + kx * direcção em x + ky * direcção em y
    Devolver (ponto)
Fim Função
```

Algoritmo 4: Cálculo do *pixel* correspondente a duas coordenadas do campo de visualização

4.2.2 Classificação de Raios que Intersectam o *Holodeck*

Conforme a Figura 16, a operação de classificação de raios consiste em integrar a geometria do *Holodeck* na cena, através da intersecção dos raios provenientes do observador com o volume do *Holodeck*. O objectivo dessa intersecção é o calculo de dois escalares a aplicar à direcção do observador. Um dos escalares é relativo ao ponto de intersecção que se encontra mais próximo do observador e o outro escalar é relativo ao ponto de intersecção mais longínquo.

```
Função IntersectaRaioComHolodeck (pontoEntrada, pontoSaída)
    escalarEntrada ← -∞
    escalarSaída ← +∞
    Para cada eixo x,y,z
        Se (a direcção observaçãofor nula) Então
            Se (observador não estiver entre coordenadas do Holodeck) Então
                Devolver (não há intersecção)
            Fim Se
        Senão
            escalarMenor ← |menorPontoHolodeck – observador| / direcção observação
            escalarMaior ← |maiorPontoHolodeck – observador| / direcção observação
            Se (escalarMenor > escalarMaior) Então
                Trocar escalarMenor com escalarMaior
            Fim Se
            Se (escalarMenor > escalarEntrada) Então
                escalarEntrada ← escalarMenor
            Fim Se
            Se (escalarMaior < escalarSaída) Então
                escalarSaída ← escalarMaior
            Fim Se
            Se (escalarEntrada > escalarSaída) Então
                Devolver (não há intersecção)
            Fim Se
            Se (escalarSaída < 0) Então
                // Holodeck encontra-se atrás do observador
                Devolver (não há intersecção)
            Fim Se
        Fim Se
    Fim Para
    Para cada eixo x,y,z
      pontoEntrada ← observador + direcção observação * escalarEntrada
      pontoSaída ← observador + direcção observação * escalarSaída
    Fim Para
    Devolver (há intersecção)
Fim Função
```

Algoritmo 5: Cálculo de intersecção de um raio com o volume do *Holodeck*

A intersecção do raio com o *Holodeck* só ocorre quando esse *Holodeck* está no semi-espaço definido pelo ponto de observação (condição necessária mas não suficiente). A determinação da intersecção é feita explicitamente através do algoritmo seguinte.

Conhecidos os pontos de intersecção, é necessário classificar o raio definido por estes pontos, em função da geometria do *Holodeck*. Para o efeito, determinam-se as faces intersectadas pelo raio no *Holodeck*. Existe sempre uma coordenada x, y e z com igual valor à coordenada equivalente, ou do menor ponto e do maior ponto do *Holodeck*. A identificação dessa coordenada, permite detectar qual a face intersectada de acordo com o Algoritmo 6.

```
Função FaceIntersectada (ponto)
    nIntersecçõesPonto ← 0
    Se ponto.x = menorPontoHolodeck.x Então
        face ← 6; nIntersecções ← nIntersecções + 1
    Fim Se
    Se ponto.x = maiorPontoHolodeck.x Então
        face ← 1; nIntersecções ← nIntersecções + 1
    Fim Se
    Se ponto.y = menorPontoHolodeck.y Então
        face ← 4; nIntersecções ← nIntersecções + 1
    Fim Se
    Se ponto.y = maiorPontoHolodeck.y Então
        face ← 2; nIntersecções ← nIntersecções + 1
    Fim Se
    Se ponto.z = menorPontoHolodeck.z Então
        face ← 5; nIntersecções ← nIntersecções + 1
    Fim Se
    Se ponto.z = maiorPontoHolodeck.z Então
        face ← 3; nIntersecções ← nIntersecções + 1
    Fim Se
    Se nIntersecções > 1 Então
        Devolver (erro: intersecção com aresta ou vértice)
    Fim Se
    Devolver (face)
Fim Função
```

Algoritmo 6: Determinação da face do *Holodeck* que contém um ponto de intersecção

De cada vez que se determina uma face intersectada, incrementa-se o número de intersecções para se ter conhecimento da existência de uma intersecção com aresta ou vértice. Nesta situação, referida na secção 3.2.2, não se consegue relacionar um ponto de intersecção com uma única face, visto que a classificação de um raio é efectuada na correspondência unívoca do ponto de intersecção com a respectiva face. Para esta indeterminação, realiza-se o cálculo do raio recorrendo ao motor de radiância, sem classificar esse mesmo raio nem inserir o seu valor de radiância na estrutura de dados do *Holodeck*. A determinação de cada uma das faces intersectadas apresenta-se no Algoritmo 6.

Recorrendo à função que determina o número de divisões a aplicar a cada coordenada de cada uma das faces (Algoritmo 1) e à função que determina o número da divisão intersectada (Algoritmo 2), estão classificados os raios que passam pela posição do observador, atravessam o campo de visualização e intersectam o *Holodeck*, a fim de serem pesquisados e, eventualmente, armazenados na sua estrutura de dados.

4.2.3 Pesquisa dos Raios Classificados

Para realizar a pesquisa na estrutura de dados do *Holodeck*, começa-se por aceder ao índice primário que representa o par de faces intersectadas. Este índice é calculado a partir da face com menor identificação e da face com maior identificação do *Holodeck*, conforme a Figura 11. O Algoritmo 7 recebe as duas faces referidas e devolve o índice pretendido.

```
Função CalcularIndice (faceMenor, faceMaior)
 temporária ← 0;
 Para contadora ← 0 até faceMenor
  temporária ← temporária + contadora
    Fim Para
 Devolver ((faceMenor - 1) * 5 + (faceMaior - 2) - temporária)
Fim Função
```

Algoritmo 7: Cálculo do índice primário da estrutura em função da face de entrada e da face de saída

A função de pesquisa de um raio na estrutura do *Holodeck* necessita de conhecer as coordenadas dos dois pontos de intersecção que definem o raio e a respectiva classificação.

Em caso de pesquisa com sucesso, esta função devolve o valor de radiância armazenado na estrutura de dados do *Holodeck*.

Após ter sido determinado o índice primário da estrutura, através do Algoritmo 7, procede-se ao cálculo do índice secundário da estrutura que, de acordo com a secção 4.1.2.1, representa a numeração do par de quadrículas. A sequência desta numeração é determinada em função do número de divisões das faces em cada direcção e encontra-se ilustrada na Figura 18. No exemplo desta figura, pode verificar-se que este par de quadrículas se encontra numerado com o valor 153. O cálculo do número da quadrícula descreve-se no Algoritmo 9.

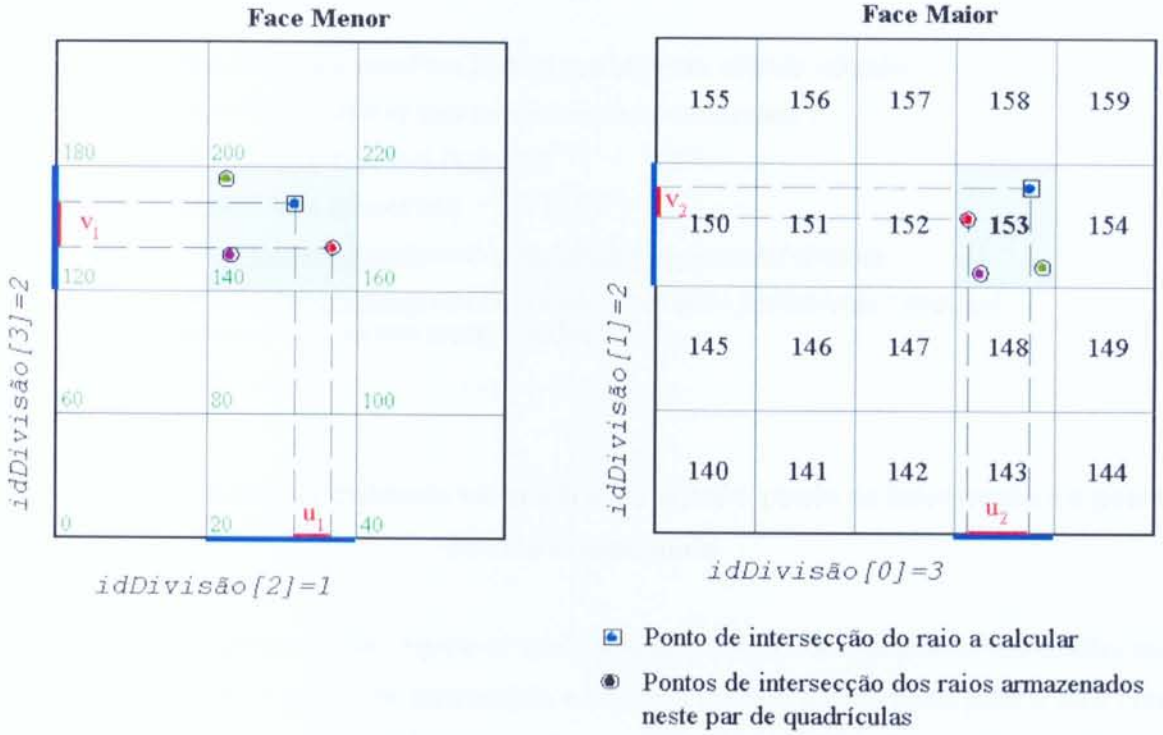


Figura 18: Visualização da numeração do par de quadrículas e distâncias entre os pontos de intersecção de um raio armazenado e do raio a calcular

Uma vez conhecidos os dois índices que dão acesso directo à estrutura de dados do *Holodeck*, realiza-se uma pesquisa sequencial sobre a lista de raios armazenados que possuem o mesmo sentido do raio em estudo. O objectivo desta pesquisa é encontrar um raio geometricamente próximo do raio a calcular.

Para o efeito, calcula-se a distância, em cada quadrícula, entre o ponto de intersecção e o ponto de um raio armazenado. Essa distância é convertida para um valor compreendido entre 0 e 1. Este valor é obtido pela razão entre a distância do ponto de intersecção ao ponto do raio armazenado.

Numa implementação inicial, armazenava-se a distância do ponto de intersecção ao ponto do raio armazenado na estrutura de dados do *Holodeck*. Em vez dos 4 Bytes de memória ocupados pela referida distância, o valor armazenado (que passou a ser a razão referida anteriormente), passou a ocupar apenas 2 Bytes, diminuindo consideravelmente o tamanho do ficheiro *Holodeck*. Apresenta-se o cálculo deste valor no Algoritmo 8.

```
Função DistAoRaioArmaz (face, coordFace, pontoInt, posRaioArmaz, idDivisão, nDivisões)
    /* determinação do sistema local de duas coordenadas da face intersectada */
    3Dpara2D  ← { {y,z}, {x,z}, {x,y}, {x,z}, {x,y}, {y,z} }
    coord     ← 3Dpara2D[face-1][coordFace]
    medQuad   ← (pontoMaiorHolodeck.coord - pontoMenorHolodeck.coord) / nDivisões
    factor    ← (pontoMenorHolodeck.coord + idDivisão * medQuad + posRaioArmaz * medQuad /
      ValorMáximoArmazenar – pontoInt.coord) / medQuad
    Devolver (factor)
Fim Função
```

Algoritmo 8: Cálculo da distância na quadrícula entre o ponto de intersecção e o ponto do raio armazenado

Como medida de proximidade, optou-se por calcular a soma dos quadrados das distâncias, na quadrícula, entre o ponto de intersecção e o ponto do raio armazenado para a face com menor identificação e, em seguida, para a face com maior identificação. O raio escolhido é aquele que apresentar o menor valor do máximo referido, para ambas as faces.

Também se optou por realizar o cálculo do máximo de valores em vez da multiplicação, porque nesta operação, quando um dos factores tende para zero, o resultado final também tende para zero, independentemente do valor do outro factor.

```
Função PesquisaHolodeck (pontoEntrada, pontoSaída, faceMenor, faceMaior, idDivisão[4], sentido)
    indice ← CalcularIndice(faceMenor, faceMaior)
    indiceSec ← ((idDivisão[3] * NDivisões(faceMenor, 2) + idDivisão[2] ) * NDivisões(faceMaior, 1) + idDivisão[1] ) *
      NDivisões(faceMaior, 2) + idDivisão[0]
    Se sentido directo Então
        apontador ← Holodeck[indice].ParQuadr[indiceSec].raios_sdirecto
    Senão
        apontador ← Holodeck[indice].ParQuadr[indiceSec].raios_sinverso
    Fim Se
    Se apontador for nulo Então
        Devolver (pesquisa sem sucesso)
    Fim Se
    menorDistância ← limiteMinProximidadeFase1
    Enquanto apontador não for nulo
        u1 ← DistAoRaioArmaz(faceMenor, 0, pontoEntrada, apontador.u1, idDivisão[3], NDivisões(faceMenor,1))
        v1 ← DistAoRaioArmaz(faceMenor, 1, pontoEntrada, apontador.u2, idDivisão[2], NDivisões(faceMenor,2))
        factorProximidade ← u1*u1 + v1*v1
        Se (factorProximidade < menorDistância) Então
            u2 ← DistAoRaioArmaz(faceMaior, 0, pontoSaída, apontador.v1, idDivisão[1], NDivisões(faceMaior,1))
            v2 ← DistAoRaioArmaz(faceMaior, 1, pontoSaída, apontador.v2, idDivisão[0], NDivisões(faceMaior,2))
        Fim Se
        factorProximidade ← Máximo(factorProximidade, u2*u2 + v2*v2)
        Se (factorProximidade < menorDistância) Então
            menorDistância ← factorProximidade
            radiânciaRGB ← apontador.radiância
        Fim Se
        apontador ← apontador.próximo
    Fim Enquanto
    Devolver (menorDistância)
    Devolver (radiânciaRGB)
    Se (menorDistância >= limiteMinProximidadeFase1) Então
        Devolver (pesquisa excedeu limite)
    Fim Se
    Devolver (pesquisa com sucesso)
Fim Função
```

Algoritmo 9: Função de pesquisa na estrutura de dados do *Holodeck*

No Algoritmo 9 descreve-se a função de pesquisa na estrutura de dados do *Holodeck*, utilizando as operações de cálculo de índice primário, numeração das quadrículas, visita da estrutura de dados, determinação do factor de proximidade e consequente escolha do raio armazenado. O objectivo desta função é a determinação do valor de radiância de um raio ou a conclusão de que não existe, no *Holodeck*, qualquer raio suficientemente próximo do raio em processamento.

Com base no limite mínimo do factor de proximidade para a segunda fase de refinamento da imagem (secção 4.4.2), a função apresentada no Algoritmo 9 retorna ainda uma indicação da necessidade de executar ou não esse refinamento.

4.2.4 Activação do Motor de Radiância

Quando a pesquisa anterior não tem êxito, insere-se, por um lado, a informação de classificação do raio de intersecção com o *Holodeck* numa lista auxiliar e, por outro lado, a posição do observador e direcção de observação a serem processados pelo motor de radiância noutra lista auxiliar. Estas duas listas são necessárias para posterior armazenamento de dados na estrutura do *Holodeck*.

```
Procedimento RaioParaMotorRadiância(pontoEntrada, pontoSaída, faceMenor, faceMaior, idDivisão, sentido) )
    Inserir pontoEntrada, pontoSaída, faceMenor, faceMaior, idDivisão, sentido em
      classificaçãoRaio[nRaiosEmEspera]
    Inserir observador, direcção observação em listaRadiância [nRaiosEmEspera]
    nRaiosEmEspera ← nRaiosEmEspera + 1
    Se nRaiosEmEspera = nMáximoPedidosRaios Então
        MotorRadiância(cena, nRaiosEmEspera, listaRadiância)
        nRaiosEmEspera ← 0
    Fim Se
Fim Procedimento
```

Algoritmo 10: Activação do motor de radiância por pacotes de raios

Os raios que ainda não foram calculados são agrupados em pacotes com um número de pedidos definido pelo utilizador, como referido na secção 4.1.1. Uma vez formado um

pacote de pedidos de cálculo de raios, é activado o motor de radiância, o qual calcula e devolve as radiâncias desses raios.

O agrupamento de raios em pacotes deve-se ao facto de a inovação do motor de radiância ter um custo computacional significativo, pelo que se deve evitar a sua invocação para cálculo de pequenas quantidades de raios. Nessa medida, agrupam-se os raios a calcular em pacotes maiores por forma a diluir esse custo computacional.

4.2.5 Inserção dos Raios Calculados no *Holodeck*

Uma vez terminado o cálculo dos raios, a sua classificação é efectuada de acordo com a secção 4.2.2, e armazenada na estrutura de dados do *Holodeck*, juntamente com o valor de radiância obtido pelo motor de radiância.

4.2.6 Operação de Visualização

A visualização da imagem é constituída pelo conjunto de blocos descritos ao longo deste capítulo. Depois de determinada a posição de cada *pixel*, experimenta-se a intersecção do raio passante pelo observador e por este *pixel* com o volume do *Holodeck* para obter os pontos de intersecção. Com esses pontos de intersecção, determinam-se as faces intersectadas. Em seguida, calculam-se os números das divisões das quadrículas em que ocorrem as intersecções. Procede-se à pesquisa na estrutura de dados do *Holodeck*, a fim de atribuir ao *pixel* o valor de radiância e o factor de proximidade do raio armazenado mais próximo do raio a calcular. Se não existir informação na estrutura de dados do *Holodeck*, encaminha-se a informação do raio a calcular para um mecanismo que agrupa os pedidos dos raios em pacotes, a enviar para o motor de radiância. Caso o tamanho do último pacote não tenha atingido o limite definido pelo utilizador, envia-se os restantes raios para o motor de radiância. Todos os raios calculados são finalmente inseridos na estrutura de dados do *Holodeck*, com a respectiva contabilização do número de raios armazenados na estrutura de dados do *Holodeck*.

```
Procedimento OperacaoVisualizacao ()
    DistânciaEntrePixels()
    nRaiosEmEspera ← 0
    Para cada linha do campo de visualização
        Para cada coluna do campo de visualização
            pixel ← CálculoPixel(linha, coluna)
            Se pixel não estiver calculado e accionarHolodeck Então
                Se IntersectaRaioComHolodeck(pontoEntrada, pontoSaída) Então
                    faceMenor,intersectaMenor ← FaceIntersectada(pontoEntrada)
                    faceMaior,intersectaMenor ← FaceIntersectada(pontoSaída)
                    Se intersectaMenor E  intersectaMaior Então
                        sentido ← directo
                        Se faceMenor > faceMaior Então
                            sentido ← inverso
                            Trocar faceMenor com faceMaior
                            Trocar pontoEntrada com pontoSaída
                        Fim Se
                        idDivisão[3], idDivisão[2] ← IDDivisão(faceMenor, pontoEntrada)
                        idDivisão[1], idDivisão[0] ← IDDivisão(faceMaior, pontoSaída)
                        pixel.calculado, pixel.factorProximidade, pixel.radiância ← PesquisaHolodeck(pontoEntrada,
                          pontoSaída, faceMenor, faceMaior, idDivisão, sentido)
                        Se não pixel.calculado Então
                          RaioParaMotorRadiância(pontoEntrada, pontoSaída, faceMenor, faceMaior, idDivisão,
                          sentido)
                        Fim Se
                    Fim Se
                Fim Se
            Fim Se
        Fim Para
    Fim Para
    Se nRaiosEmEspera > 0 Então
        MotorRadiância(cena, nRaiosEmEspera, listaRadiância)
    Fim Se
    nRaiosEmEspera ← 0
    Para cada linha do campo de visualização
        Para cada coluna do campo de visualização
            Se pixel sem informação de radiância Então
```

```
            pixel ←InserçãoHolodeck(classificaçãoRaio[nRaiosEmEspera],
              listaRadiância[nRaiosEmEspera].radiânciaRGB)
            nRaiosEmEspera ← nRaiosEmEspera + 1
            nRaiosHolodeck ← nRaiosHolodeck + 1
        Fim Se
    Fim Para
Fim Para
Fim Procedimento
```

Algoritmo 11: Operação de Visualização

4.3 Visualização de Imagem

Nesta fase do protótipo, dispõe-se de uma grelha de radiâncias que pode não estar completa, dado que podem existir *pixels* para os quais não existem amostras relevantes no *Holodeck*. Para colmatar esta falta de informação realiza-se uma interpolação.

Além disso, a grelha de valores de radiâncias ainda não pode ser visualizada, uma vez que estas radiâncias têm de ser convertidas para valores de brilho. São estes valores de brilho que são perceptíveis pelo sistema visual humano. Na secção 4.3.2 faz-se uma descrição do operador de mapeamento de tons, o qual realiza a referida conversão.

4.3.1 Interpolação

A interpolação é efectuada de forma a obter os valores de radiância de pontos da grelha da imagem que não contenham informação. Para tal, faz-se um varrimento na grelha e, sempre que se detectem células sem informação, calcula-se a média pesada dos valores vizinhos com informação. O peso para o cálculo desta média é o inverso da distância Euclidiana entre os valores na grelha. O número de valores na grelha a considerar para cada iteração da interpolação é definido pelo utilizador no ficheiro (secção 4.1.1).

4.3.2 Operador de Mapeamento de Tons

Os valores de radiância obtidos representam uma quantificação da energia que se desloca ao longo de um raio enquanto este não intersecta a geometria da cena. Estes valores de radiância têm que ser convertidos para luminâncias (quantificação de luz) e posteriormente para brilho, sendo este perceptível pelo sistema visual humano.

O operador de mapeamento de tons utilizado é o apresentado em [Larson97]. Alguns operadores de mapeamento de tons preservam a visibilidade dos objectos alterando a impressão do contraste. Outros operadores preservam a impressão global de brilho em detrimento da visibilidade. O operador apresentado em [Larson97] tem em conta estes dois critérios, através da alteração do histograma de luminâncias e do seu mapeamento para valores de visualização, por forma a preservar visibilidade e o contraste.

Para aplicar este operador de mapeamento de tons, calcula-se a frequência de luminância de uma imagem, com base na conversão da grelha de radiâncias calculadas, para outra em que cada *pixel* representa um grau no campo de visualização da imagem original. A partir dos valores obtidos, calcula-se o histograma e a função cumulativa de distribuição de frequências correspondente. Em seguida, realiza-se um ajuste linear do histograma para garantir que os contrastes não sejam exagerados. Este ajuste é conseguido através da truncatura das frequências que excedam um dado limite. Aplica-se posteriormente um ajuste baseado na sensibilidade humana ao contraste. Assim, para a frequência de amostras em cada intervalo do histograma, é determinado um novo limite que recorre a uma função que calcula a diferença mínima de contraste detectável, representada na Figura 20. O Algoritmo 12 apresenta a implementação destes dois ajustes.

```
Função AjustarHistograma ()
    tolerância ← 0.025 * nTotalAmostras
    Fazer
        ajustes ← 0
        Se nTotalAmostras < tolerância Então
            /* intervalo dinâmico já representa luminâncias no histograma original */
            Devolver (insucesso)
        Fim Se
        Se ajuste linear Então
```

```
        limite ← (nTotalAmostras * intervaloFrequências) / (brilhoMáximoMonitor - brilhoMínimoMonitor)
    Fim Se
    Para cada intervalo do histograma
        Se ajuste baseado na sensibilidade humana ao contraste Então
            limite ← [ΔLt(luminânciaMonitor) / ΔLt(luminânciaMundo)] * [(nTotalAmostras * intervaloFrequências*
                luminânciaMundo) / [(brilhoMáximoMonitor - brilhoMínimoMonitor) * luminânciaMonitor]]
        Fim Se
        Se frequênciaIntervalo > limite Então
            ajustes ← ajustes + frequênciaIntervalo – limite
            frequênciaIntervalo ← limite
        Fim Se
    Fim Para
    nTotalAmostras ← nTotalAmostras – limite
  Até ajustes <= tolerância
  Devolver (sucesso)
Fim Função
```

Algoritmo 12: Função que garante que nenhuma frequência do histograma excede um dado limite

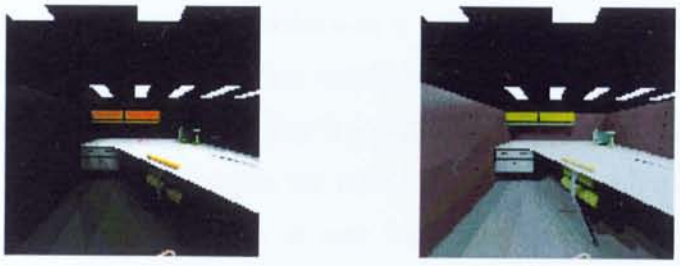
O olho humano tem a capacidade de se adaptar a um grande intervalo dinâmico de luminâncias. À medida que a luz diminui, o sistema visual humano tem mais dificuldade em detectar os contrastes.

$\Delta L_t(L_a)$ – diferença mínima detectável	Intervalo de luminâncias
-2.86	$\log10(L_a) < -3.94$
$(0.405 \log10(L_a)+16)^{2.18} - 2.86$	$-3.94 <= \log10(L_a) < -1.44$
$\log10(L_a) - 0.395$	$-1.44 <= \log10(L_a) < -0.0184$
$(0.249 \log(L_a) + 0.65)^{2.7} - 0.72$	$-0.0184 <= \log10(L_a) < 1.9$
$\log10(L_a) - 1.255$	$\log10(L_a) >= 1.9$

Figura 19: Função da diferença mínima detectável para cada um dos intervalos de luminância L_a

Na função de detecção de contrastes criada por [Ferwerda96] e aplicada em [Larson97], a sensibilidade é abrangida desde o limite mínimo da visão humana até aos níveis da luz do dia. A Figura 19 estabelece a relação entre a diferença mínima de contrastes detectável e o intervalo de luminâncias considerado.

Na Figura 20 apresentam-se, na alínea a), os valores de radiância obtidos para os pixels para uma imagem de um escritório. Estes valores são truncados quando ultrapassam o valor de radiância 1. Na alínea b) da mesma figura apresentam-se os valores de brilho obtidos por este operador de mapeamento de tons.



a) b)

Figura 20: a) Imagem sem operador de mapeamento de tons
b) A mesma imagem com operador de mapeamento de tons

Apesar deste operador atender ao facto do sistema visual humano possuir limitações, a implementação destas não foi realizada devido ao reduzido efeito produzido no fotograma final. Para além disso, a sua implementação provocaria uma diminuição no desempenho do sistema de navegação.

4.4 Exploração Interactiva

Numa situação de navegação interactiva, o utilizador move-se na cena através de pequenas variações em distância e em direcção. Portanto, é grande a probabilidade de um fotograma poder aproveitar um número significativo de raios calculados nos fotogramas anteriores. Tendo estes raios sido armazenados, existe assim a possibilidade de os reutilizar sem recorrer ao motor de radiância. Esta reutilização de raios é uma das grandes vantagens da técnica *Holodeck* e atinge o seu máximo quando a estrutura de dados do *Holodeck* se encontra tão preenchida que se torna exclusiva a consulta à estrutura de dados e é desnecessária a activação do motor de radiância.

Cada deslocação na cena implica forçosamente uma nova definição do campo de visualização. As imagens obtidas são, por vezes, grosseiras, uma vez que os raios utilizados

são geometricamente aproximados aos raios necessários. No entanto, na maioria das situações, a qualidade da imagem é suficiente para o utilizador se aperceber da sua posição na cena virtual.

Dado que, como resultado da navegação, a nova informação calculada vai sendo armazenada no *Holodeck* para posterior utilização, coloca-se o problema da coerência espaço-temporal. Efectivamente, os novos raios poderão ser reutilizados nas imagens seguintes, se a posição do observador não se alterar demasiado face à posição actual. Pelo contrário, se o observador se movimenta em passos muito largos, os novos raios têm uma posição espacial muito diferente dos que estão armazenados, o que resulta na necessidade de recorrer mais regularmente ao processamento de raios pelo motor de radiância.

O cálculo de raios necessário para representar cada fotograma processa-se em três fases:

- Fase "reutilização+falta de raios" – esta fase inicial pesquisa no *Holodeck* os raios a calcular; caso existam raios armazenados, geometricamente semelhantes (ie, dentro de uma tolerância especificada pelo utilizador), efectua-se a sua reutilização; se não existirem raios calculados no par de quadrículas correspondente ao raio a calcular (falta de raios), é realizada a interpolação desse raio. Na impossibilidade da interpolação, é efectuada a traçagem do raio e o resultado, obtido pelo motor de radiância, é inserido no *Holodeck*.

- Fase "raios inválidos" – nesta fase processam-se os raios que, ou foram interpolados, ou não satisfazem o factor de proximidade máximo definido, apesar de existirem raios armazenados no par de quadrículas correspondente; o processamento é realizado através de uma traçagem de raios, sendo os resultados inseridos no *Holodeck*.

- Fase "refinamento iterativo de raios" – esta fase destina-se a melhorar a qualidade da imagem obtida nas fases anteriores e só é accionada se o utilizador o pretender (por exemplo, após parar a sua navegação pela cena). A fim de se obter um refinamento progressivo da imagem, esta fase é iterativa; em cada iteração processam-se os *pixels* da imagem cujos raios foram substituídos por raios semelhantes, mas que

ultrapassam um certo valor percentual o factor de proximidade máximo; este valor percentual é decrementado a cada nova iteração.

Assim, na aplicação desenvolvida, assume-se que a navegação na cena pode fazer-se de três formas:

- *Rápida* – neste tipo de navegação, o utilizador pretende explorar rapidamente a cena e não necessita de ver imagens com elevada qualidade; assim sendo, a imagem resultante da fase "reutilização+falta de raios" é suficiente para a compreensão espacial da cena. Caso se pretenda uma grande rapidez, pode visualizar-se uma imagem simplificada, interpolada a partir dos *pixels* correspondentes a raios reutilizados, usando-se para tal uma técnica do tipo *flat-shaded cones* [Larson98a]).

- *Intermédia* – se o utilizador pretende navegar na cena e, simultaneamente, visualizar a imagem da cena com algum detalhe, então deverão ser usados os resultados das fases "reutilização+falta de raios" e "raios inválidos"; esta visualização será mais lenta mas, ainda assim, com tempos de processamento bastante inferiores aos que se obtêm por traçagem total de raios.

- *Detalhada* – caso o utilizador pretenda visualizar uma imagem com elevado nível de qualidade, então poderá accionar a fase "refinamento iterativo de raios" para esse efeito; esta fase é progressiva, apresentando-se sucessivamente imagens de maior qualidade e deve ser utilizada, quando o utilizador se imobiliza em determinado ponto da cena.

Quando termina o processo de produção de uma imagem, a Unidade Central de Processamento (CPU) entra em compasso de espera, o qual pode ser aproveitado para melhorar a qualidade da imagem produzida. Desta forma, utiliza-se esta disponibilidade para realizar uma sequência de refinamentos.

A justificação destes refinamentos deve-se ao facto da imagem resultante do processo de visualização ser constituída por *pixels* que têm um valor de brilho aproximado. Se esta aproximação não for suficientemente boa, deve proceder-se ao refinamento de raios. Este refinamento implica a activação do motor de radiância e é realizado em três fases, descritas

nas secções seguintes. Essas três fases podem ser activadas automaticamente se o utilizador assim o pretender.

4.4.1 Primeira Fase de Refinamento

Nesta fase de refinamento activa-se o motor de radiância para realizar o cálculo dos raios que foram obtidos por interpolação, dado que vários *pixels* da imagem foram determinados por esta operação. Na alínea a) da Figura 26 apresenta-se uma imagem com *pixels* obtidos por esta técnica, enquanto que na alínea b) esses *pixels* apresentam-se calculados pelo motor de radiância.

4.4.2 Segunda Fase de Refinamento

A segunda fase de refinamento permite ao utilizador obter uma imagem com razoável qualidade visual, o que depende do distanciamento entre o raio armazenado e o raio a calcular. Deve ser encontrado o equilíbrio entre o refinamento e a rapidez na obtenção de imagens, uma vez que a activação do motor de radiância é um processo moroso.

Na implementação desta fase, analisa-se o factor de proximidade calculado para cada *pixel* e, quando é ultrapassado o limite definido pelo utilizador (*limiteMinProximidadeFase1*), recorre-se à função que adiciona o raio a refinar ao pacote de pedidos de raios para o motor de radiância. Depois de calculado o referido pacote, actualizam-se os valores de radiância dos *pixels* e armazenam-se os raios na estrutura de dados do *Holodeck*. O factor de proximidade a atribuir ao *pixel*, resultante do refinamento, passa a ter o valor zero, uma vez que são nulas as distâncias entre os pontos de intersecção nas quadrículas do raio calculado e do raio armazenado (raios coincidentes).

Em seguida, procede-se ao refrescamento da imagem a fim de serem visualizados os novos *pixels* calculados.

A visualização das alterações consequentes desta fase de refinamento são apresentadas nas alíneas b) e c) da Figura 31.

4.4.3 Terceira Fase de Refinamento

Se o utilizador considerar que a fase anterior não é suficiente para a qualidade pretendida, tem a possibilidade de recorrer a uma terceira fase de refinamento. Esta fase inclui várias iterações.

Reduções sucessivas são aplicadas ao limite do factor de proximidade, utilizado na segunda fase de refinamento, de modo que sejam calculados, pelo motor de radiância, os *pixels* que não se encontram dentro dos novos limites. O limite usado em cada iteração é calculado através da aplicação de uma percentagem ao limite de proximidade da iteração anterior. Esta percentagem é definida pelo utilizador, bem como o limite mínimo do factor de proximidade, para concluir esta fase de refinamento. De salientar que cada fase termina com o respectivo refrescamento da imagem.

4.5 Considerações Sobre o Pré-preenchimento do Holodeck

Vários são os métodos implementados para concretizar o pré-preenchimento do *Holodeck* (como referido na secção 3.2.3).

O método sequencial garante que todos os pares de quadrículas são visitados. Este método tem a desvantagem de criar um ficheiro físico do *Holodeck* muito grande, ultrapassando, mesmo a capacidade de armazenamento de um CD-ROM.

O método completamente aleatório permite o pré-preenchimento até uma percentagem do número total de raios definido de acordo com o total de pares de quadrículas. A desvantagem deste método caracteriza-se pela baixa probabilidade de acertar em pares de quadrículas que não estejam ainda preenchidas, uma vez ultrapassada uma certa percentagem de preenchimento.

Para ultrapassar esta desvantagem, pode utilizar-se outro método em que se determina aleatoriamente um valor percentual para cada raio e se esse valor estiver contido na percentagem definida, calcula-se o raio e, faz-se o respectivo armazenamento.

Um outro método realiza visitas apenas aos pares de quadrículas que se encontram em pares de faces visíveis/invisíveis ao observador.

A combinação dos dois últimos métodos de pré-preenchimento referidos deu origem ao método descrito em seguida, o qual é o utilizado para a obtenção dos resultados descritos na secção 5.

Esta escolha teve em conta o facto da menor probabilidade dos raios intersectarem faces visíveis que anteriormente eram invisíveis (ou vice-versa), uma vez que, geralmente, a deslocação na cena ser geralmente efectuada em passos curtos. Como os raios estão armazenados num único sentido, origina uma menor ocupação de memória.

A classificação das faces visíveis e invisíveis do *Holodeck* inicia-se pelo cálculo do campo de visualização. Determinando a intersecção dos raios do campo de visualização com o *Holodeck*, obtêm-se os pontos de intersecção para calcular as faces intersectadas. Pode observar-se que as instruções do algoritmo de visualização da imagem (Algoritmo 11) e do algoritmo de determinação das faces visíveis e invisíveis (Algoritmo 13) são inicialmente as mesmas.

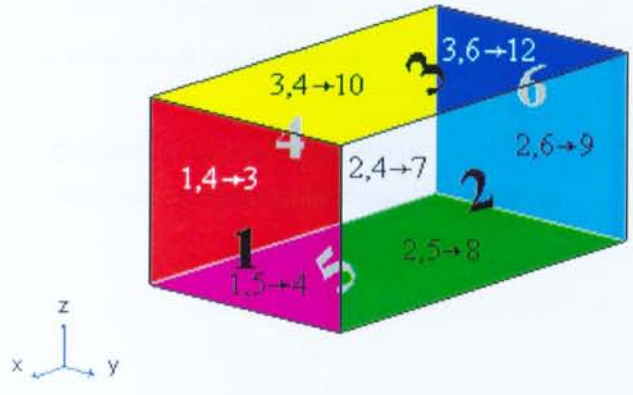


Figura 21: Visualização da classificação das faces visíveis e invisíveis através do cálculo dos índices primários

Classificam-se as faces visíveis e invisíveis através do índice primário da estrutura de dados do *Holodeck* e do sentido do raio. De cada vez que se detecta um índice novo, insere-se

numa lista que, no final deste processamento, contém todos os índices que identificam as faces visíveis e respectivas faces invisíveis para o utilizador.

No exemplo da Figura 21, observa-se que os índices 3, 4, 7, 8, 9, 10 e 12 representam áreas do campo de visualização que correspondem a um par de faces visível/invisível.

Apresenta-se no Algoritmo 14 o mecanismo anteriormente exposto.

```
Função FacesVisiveisInvisiveis ()
    DistânciaEntrePixels()
    Para cada linha do campo de visualização
        Para cada coluna do campo de visualização
            pixel ← CálculoPixel(linha, coluna)
            Se IntersectaRaioComHolodeck(pontoEntrada, pontoSaída) Então
                intersectaMenor,faceMenor ← FaceIntersectada(pontoEntrada)
                intersectaMaior,faceMaior ← FaceIntersectada(pontoSaída)
                Se intersectaMenor E intersectaMaior Então
                    sentido ← directo
                    Se faceMenor > faceMaior Então
                        sentido ← inverso
                        Trocar faceMenor com faceMaior
                        Trocar pontoEntrada com pontoSaída
                    Fim Se
                    indice ← CalcularIndice(faceMenor, faceMaior)
                    Adicionar indice sem repetições na listaFacesVisiveisInvisiveis
                Fim Se
            Fim Se
        Fim Para
    Fim Para
    Devolver (listaFacesVisiveisInvisiveis)
Fim Função
```

Algoritmo 13: Determinação das faces visíveis e invisíveis do *Holodeck*

Apresenta-se, em seguida, o cálculo das duas faces intersectadas a partir do índice primário. Estas faces são determinadas através da correspondência estabelecida entre o índice primário

e as faces de entrada e de saída, de acordo com Figura 11. O par de faces resultante permite obter o número total de raios a inserir na estrutura de dados do *Holodeck*.

```
Função IndiceParaFaceMenorEMaior (indicePrimario)
    /* correspondência entre o indice primario e as faces de entrada e de saida */
    IndPrim_Faces ← {{1,2}, {1,3}, {1,4}, {1,5}, {1,6}, {2,3}, {2,4}, {2,5}, {2,6}, {3,4}, {3,5}, {3,6}, {4,5}, {4,6}, {5,6}}
    Devolver (IndPrim_Faces [indicePrimario, 1])
    Devolver (IndPrim_Faces [indicePrimario, 2])
Fim Função
```

Algoritmo 14: Atribuição da face de entrada e da face de saída ao índice da estrutura de dados do *Holodeck*

Este número total de raios é determinado pelo produto do número de divisões a aplicar às duas dimensões de uma face pelo número de divisões a aplicar às duas dimensões da outra face intersectada. Após o somatório de todas as multiplicações, calcula-se a percentagem definida pelo utilizador, de acordo com o algoritmo seguinte.

```
Função NTotalRaiosFacesEntrada ()
    NParcialRaios ← 0
    Para cada indice da estrutura de dados
        Se indice pertence à listaFacesVisiveisInvisiveis Então
            faceMenor, faceMaior ← IndiceParaFaceMenorEMaior (indice)
            nParcialRaios ← nParcialRaios + ( NDivisões(faceMenor, 1) * NDivisões(faceMenor, 2) *
              NDivisões(faceMaior, 1) * NDivisões(faceMaior, 2) )
        Fim Se
    Fim Para
    Devolver ( (nRaiosParQuadrícula * nParcialRaios / nParesFacesIntersectadas) * (percNRaios / 100) )
Fim Função
```

Algoritmo 15: Contagem do número total de raios que vão atravessar o *Holodeck*

Nesta etapa, determina-se o ponto de intersecção na quadrícula de cada face. Para o efeito, estabelece-se uma posição aleatória para cada coordenada bidimensional nessa quadrícula. O valor da coordenada x do ponto de intersecção para a face 1 equivale ao valor da mesma coordenada do maior ponto do *Holodeck*. Para a face 6, o valor desta coordenada equivale ao

menor ponto do *Holodeck*. Para as restantes faces, ao menor ponto da quadrícula intersectada adiciona-se a posição aleatória relativa na quadrícula.

```
Função PontoNaQuadricula (face, idDivisãoX, idDivisãoY)
    posiçãoQuadX ← valor aleatório até 1
    posiçãoQuadY ← valor aleatório até 1
// ---------- Cálculo da coordenada x ----------
    Caso face = 2,3,4,5
      ponto.x ← menorPontoHolodeck.x + ((maiorPontoHolodeck.x - menorPontoHolodeck.x) / nº de divisões em x) *
          (idDivisãoX + posiçãoQuadX)
    Caso face = 6
      ponto.x ← menorPontoHolodeck.x
    Caso face = 1
      ponto.x ← maiorPontoHolodeck.x
    Fim Caso
// ---------- Cálculo da coordenada y ----------
    Caso face = 1,6
      ponto.y ← menorPontoHolodeck.y + ((maiorPontoHolodeck.y - menorPontoHolodeck.y) / nº de divisões em y) *
          (idDivisãoX + posiçãoQuadX)
    Caso face = 3,5
      ponto.y ← menorPontoHolodeck.y + ((maiorPontoHolodeck.y - menorPontoHolodeck.y) / nº de divisões em y) *
          (idDivisãoY + posiçãoQuadY)
    Caso face = 4
      ponto.y ← menorPontoHolodeck.y
    Caso face = 2
      ponto.y ← maiorPontoHolodeck.y
    Fim Caso
// ---------- Cálculo da coordenada z ----------
    Caso face = 1,2,4,6
      ponto.z ← menorPontoHolodeck.z + ((maiorPontoHolodeck.z - menorPontoHolodeck.z) / nº de divisões em z) *
          (idDivisãoY + posiçãoQuadY)
    Caso face = 5
      ponto.z ← menorPontoHolodeck.z
    Caso face = 3
      ponto.z ← maiorPontoHolodeck.z
    Fim Caso
    Devolver (ponto)
Fim Função
```

Algoritmo 16: Cálculo de um ponto aleatório numa dada quadrícula

Este processo repete-se para as coordenadas y e z do ponto de intersecção na quadrícula.

Finalmente, para realizar o preenchimento da estrutura de dados do *Holodeck*, determinam-se as faces visíveis e invisíveis e o respectivo índice primário. Conhecido o número total de raios a inserir na estrutura de dados, faz-se atravessar, para cada par de quadrículas, o número de raios adequado.

Para cada raio a calcular, determina-se um valor percentual aleatório. Se esse valor estiver contido na percentagem de raios a inserir por par de quadrículas (definido pelo utilizador), determinam-se os pontos de intersecção do raio nas quadrículas, procede-se ao cálculo deste raio e insere-se o mesmo na estrutura de dados do *Holodeck*.

```
Procedimento PreenchimentoFacesEntrada ()
    listaFacesVisiveisInvisiveis ← FacesVisiveisInvisiveis()
    Para cada indice da estrutura de dados
        Se indice pertence à listaFacesVisiveisInvisiveis Então
            faceMenor, faceMaior ← IndiceParaFaceMenorEMaior (indice)
            nTotalRaiosFacesEntrada ← NTotalRaiosFacesEntrada()
            Enquanto nRaiosHolodeck <= nTotalRaiosFacesEntrada
                Para idDivisão[3] ← 0 até NDivisões(faceMenor,1)
                    Para idDivisão[2] ← 0 até NDivisões(faceMenor,2)
                        Para idDivisão[1] ← 0 até NDivisões(faceMaior,1)
                            Para idDivisão[0] ← 0 até NDivisões(faceMaior,2)
                                Para nraio ← 0 até nRaiosParQuadrícula
                                    decisão ← valor percentual aleatório
                                    Se (decisão < percentagemNRaios) Então
                                        pontoEntrada ← PontoNaQuadricula(faceMenor, idDivisão[3], idDivisão[2])
                                        pontoSaída ← PontoNaQuadricula(faceMaior, idDivisão[1], idDivisão[0])
                                        InserçãoHolodeck(idDivisão, indice, sentido, pontoEntrada, pontoSaída,
                                          radiânciaRGB)
                                        nRaiosHolodeck ← nRaiosHolodeck + 1
                                    Fim Se
                                Fim Para
                            Fim Para
                        Fim Para
                    Fim Para
                Fim Para
```

Fim Enquanto

Fim Se

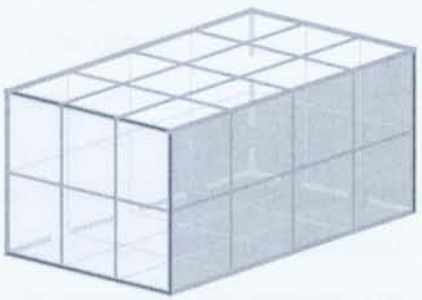
Fim Para

Fim Procedimento

Algoritmo 17: Preenchimento da estrutura de dados do *Holodeck*

4.6 Síntese do Capítulo

Neste capítulo descreve-se uma implementação do protótipo da técnica *Holodeck*. Esta implementação é constituída por um conjunto de processos tais como: inicialização do sistema, produção de imagem, visualização de imagem e exploração interactiva. Descrevem-se também as várias fases de refinamento que se aplicam a cada imagem em ambiente de navegação. Finalmente, fazem-se algumas considerações sobre o nível de pré-preenchimento do *Holodeck*.



5 Avaliação de Resultados

Nesta secção avalia-se o desempenho do protótipo segundo diversas vertentes. Um desses objectivos de avaliação passa pela comparação do seu desempenho em comparação com a traçagem total de raios através do motor de radiância. Também se pretende estudar a influência do grau de divisão de faces em quadrículas e do número de raios a armazenar por par de quadrículas. Em termos de pré-preenchimento do *Holodeck*, visa-se determinar qual a gama óptima de enchimento, a realizar numa fase de pré-processamento, por forma a obter os melhores desempenhos interactivos do protótipo. Para além destes aspectos, outros são ainda abordados nesta secção.

A cena de teste (fonte [NRC]) é formada por uma célula de escritório, a qual inclui uma mesa em forma de L, uma cadeira, armários, computador e outros objectos mais pequenos, dentro de uma sala com 6 luminárias no tecto. As dimensões da cena são 12m x 10m x 2.5m. Um *Holodeck* foi colocado na cena de forma a envolver toda a célula de trabalho.

Como plataforma de teste foi utilizado um PC (CPU *Pentium* III a 600Mhz, 128 Mb RAM) com sistema operativo *Windows* 98.

As condições de funcionamento do motor de radiância foram definidas com nível de qualidade elevado, destacando-se as seguintes:

- um nível de interreflexão difusa (isto é, a iluminação global num ponto depende da iluminação directa e da iluminação indirecta devida aos objectos vizinhos);
- cerca de 400 amostras de raios para calcular a interreflexão difusa;
- simulação de luminárias artificiais nos objectos mais iluminados.

Esta parametrização do motor de radiância conduz a tempos de cálculo, por raio luminoso, consideravelmente elevados, o que invalida imediatamente a traçagem de raios pelo motor de radiância como forma única de produção interactiva de imagens. No entanto, uma vantagem decorrente dessa parametrização é a informação radiométrica calculada ser bastante próxima da realidade ou, dito de outra forma, fisicamente válida.

Um tipo de resultados que se pretende avaliar é a dependência da resposta do protótipo, em termos de qualidade de imagem e do tempo de cálculo, com os parâmetros que caracterizam a estrutura do *Holodeck*: o número de divisões das três dimensões do *Holodeck*, o número máximo de raios que atravessam cada par de quadrículas, a resolução da imagem e o valor máximo do factor de proximidade admitido para a semelhança entre raios a calcular e raios existentes.

Sendo N o número de divisões de cada dimensão do *Holodeck* e F o número máximo de raios passantes por cada par de quadrículas, conclui-se que o número máximo de raios inseridos no *Holodeck* é:

$$\mathbf{K = 30 * N^4 * F} \qquad \textbf{(3)}$$

Para uma dada imagem e considerando um caso ideal em que todos os raios a calcular existam no *Holodeck*, a visualização dos *pixels* respectivos depende apenas do tempo de acesso directo aos raios previamente calculados e armazenados. Ou seja, nesta situação, não é necessária qualquer traçagem de raios, sendo a imagem criada por um processo designado por operação de visualização, conforme Figura 15.

5.1 Comparação de Tempos

Na Figura 22 apresentam-se os tempos do processo de operação de visualização de uma imagem (*Holodeck* ideal) em comparação com os respectivos tempos de cálculo completo pelo motor de radiância, em quatro situações que apresentam a mesma quantidade total de raios no *Holodeck*[5]:

- Caso 1: N=5, F=16384
- Caso 2: N=10, F=1024
- Caso 3: N=15, F=203
- Caso 4: N=20, F=64

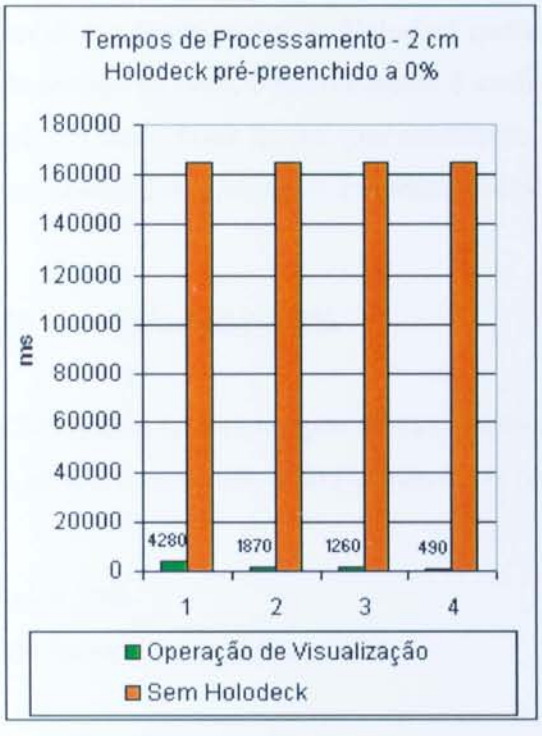


Figura 22: Tempos de processamento com *Holodeck* ideal e sem *Holodeck*

[5] A manutenção do número raios no *Holodeck*, para os quatro casos analisados, permite simular uma situação de limitação de memória física.

A análise da Figura 22 mostra que o refrescamento com base no *Holodeck* é altamente vantajoso em comparação com a traçagem total de raios. Além disso, também parece mostrar que é preferível aumentar o número de quadrículas em detrimento do número de raios por par de quadrículas.

Em seguida, são analisadas situações de navegação na cena de teste que, apesar de serem simples, apresentam tempos de processamento que são semelhantes a outras navegações da mesma cena, podendo, por isso mesmo, ser consideradas como casos representativos.

5.2 Variação do Pré-preenchimento do Holodeck

Nesta secção pretende-se avaliar o desempenho do *Holodeck* quando este se encontra vazio, fazendo variar os passos de avanço na cena. Posteriormente é avaliado o número de divisões de cada dimensão (N) para um dado nível de pré-preenchimento. Por último, entre vários níveis de pré-preenchimento analisados, procura-se encontrar um ponto óptimo.

5.2.1 Pré-enchimento do *Holodeck* a 0%

Na Figura 23 e na Figura 24, apresentam-se tempos de navegação na cena, em seis imagens da sequência de animação, sem pré-enchimento do *Holodeck*. Os parâmetros gerais definidos pelo utilizador foram:

- Factor de proximidade 50%
- Redução iterativa do factor de proximidade 60%

O estudo apresentado foi realizado sobre um *Holodeck* com 20 divisões em cada dimensão (N), por ser a situação que apresentava os menores tempos de processamento.

Os resultados apresentados foram obtidos com duas velocidades diferentes de movimentação do observador (correspondentes a passos de avanço na cena de 2 cm e de 5 cm, respectivamente, alternados com rotações de 10º) entre imagens consecutivas. Em termos práticos, os valores de abcissa dos gráficos, compreendidos entre 2 e 6, representam um avanço, rotação à direita, avanço, rotação à esquerda e avanço, respectivamente.

As imagens podem ser sujeitas a processos de refinamento até atingirem uma qualidade visual idêntica entre si, embora possam também surgir imagens cuja qualidade inicial não justifique refinamento.

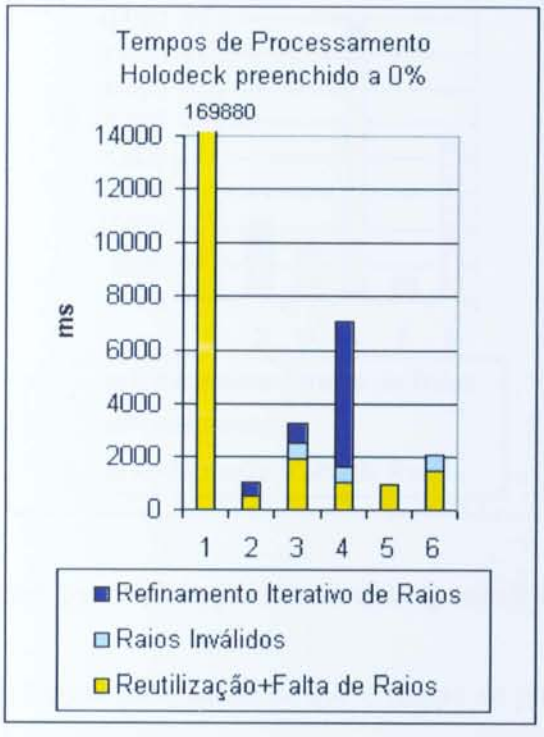


Figura 23: Tempos parciais de processamento (passo 2 cm; rotação 10º)

Tendo em conta que uma traçagem total de raios demora cerca de 160.000ms (Figura 22), a Figura 23 mostra claramente os benefícios da navegação usando técnicas de reutilização de informação e de melhoramento progressivo da imagem.

A Figura 24 apresenta os tempos equivalentes, mas os passos de avanço na cena são de 5 cm, mantendo-se o ângulo de rotação. Neste caso, os tempos de cálculo correspondentes a raios inválidos e o refinamento são diminutos. Tal facto deve-se à perda de coerência espacial quando o observador se desloca com um passo de avanço significativamente maior do que 2 cm, o que obriga a que muitos mais raios sejam calculados pelo motor de radiância.

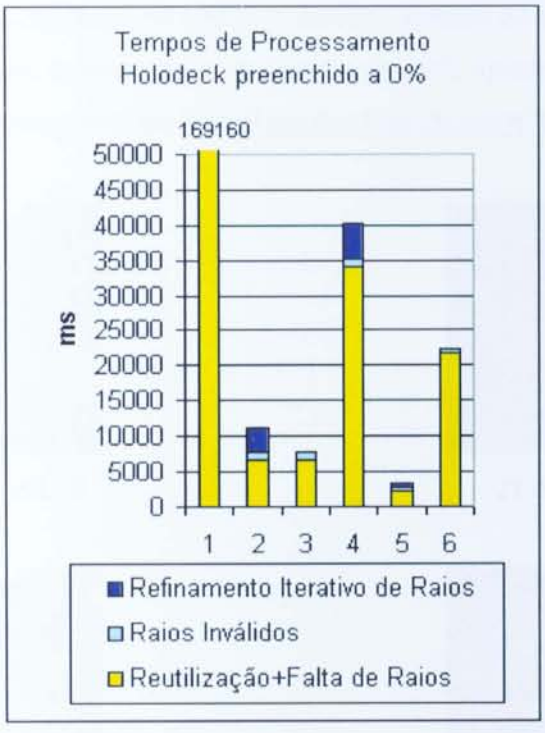


Figura 24: Tempos parciais de processamento (passo 5 cm; rotação 10º)

Da análise da Figura 23 e Figura 24 constata-se que o tempo de processamento para obter a imagem inicial é manifestamente superior ao tempo necessário para obter as imagens posteriores resultantes da navegação na cena. Este facto justifica-se pela inexistência de raios armazenados quando se procede ao cálculo da imagem inicial (o *Holodeck* encontra-se vazio). Os raios calculados para a imagem inicial são armazenados e reutilizados, em grande parte, nas imagens posteriores.

Comparando a Figura 23 com a Figura 24, verifica-se por vezes que, em termos percentuais, o tempo da fase reutilização+falta de raios é inversamente proporcional à soma dos tempos das fases raios inválidos e refinamento iterativo de raios. Com efeito, devido ao facto de um grande número de raios ser calculado na fase reutilização+falta de raios, tal acarreta que se use um número inferior de raios aproximados.

Também se salienta que, quando é refinado um grande número de raios numa imagem, a seguinte reutiliza muitos desses raios, não necessitando, portanto, de um grande nível de refinamento, como se exemplifica na passagem da abcissa 4 para a abcissa 5 na Figura 23.

A Figura 25 apresenta a sequência de imagens correspondente ao caso da Figura 23 (passo de avanço na cena de 2 cm, intercalado com rotação de 10º), apenas com a primeira fase do processo de produção de imagem (fase "reutilização+falta de raios").

1) Imagem inicial

2) Avanço 1

3) Rotação 1

4) Avanço 2

5) Rotação 2

6) Avanço 3

Figura 25: Evolução das imagens da navegação da cena (*Holodeck* 0%, passo 5cm, rotação 10º)

Da observação da Figura 26 constata-se que a imagem inicial (fase "reutilização+falta de raios") apresenta uma qualidade aceitável. Os *pixels* que foram obtidos por interpolação ou que não respeitam o factor de proximidade são preenchidos pela fase seguinte ("raios inválidos"), com a qual se obtém uma imagem de qualidade quase final. As luminárias da alínea a) são um exemplo desses *pixels* determinados por interpolação. Na alínea b) da

mesma figura, os mesmos *pixels* são resultado do cálculo pelo motor de radiância. Os resultados das iterações da fase "refinamento iterativo de raios" notam-se essencialmente nos *pixels* correspondentes a arestas de objectos nas alíneas c) e d).

a) Rotação 2 da Figura 25

b) Cálculo de *pixels* interpolados (tempo 1''3s; nº de pixels 701)

c) 1ª iteração do refinamento progressivo (tempo 0''43s; nº de *pixels*: 88)

d) 2ª iteração do refinamento progressivo (tempo 2''30s; nº de pixels 1968)

Figura 26: Fases de refinamento da rotação 2 da Figura 25

5.2.2 Pré-enchimento do Holodeck a 33%

O método de pré-preenchimento utilizado para obter os resultados que se apresentam em seguida foi o Método Direccional Parcial Controlado referido na secção 3.2.3.4.

O pré-preenchimento da estrutura de dados a 33% equivale afirmar que os pares de quadrículas contêm 33% do número máximo de raios que a correspondente lista pode albergar. Desta forma, garante-se que a estrutura de dados fornece pelo menos uma aproximação do raio a calcular. Por esse motivo, a fase raios inválidos é inexistente para os resultados apresentados nesta secção.

Nas Figura 27, Figura 28 e Figura 29 apresentam-se tempos de navegação na cena que correspondem à evolução dos seguintes parâmetros:

- Factores de proximidade: 25% (N=5); 50% (N=10); 100% (N=20)
- Redução iterativa do factor de proximidade 60%
- Passo de avanço na cena: 5 cm

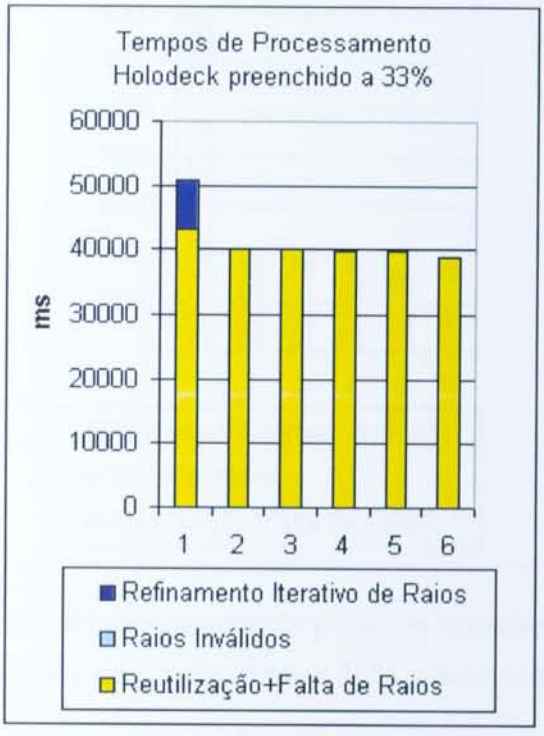


Figura 27: Tempos parciais de processamento para um *Holodeck* com N=5 preenchido a 33% e factor de proximidade a 25% – 5cm; rotação 10°

O motivo da utilização de factores de proximidade dependentes de N visa definir semelhantes distâncias físicas no processo de pesquisa dos melhores raios para reutilização (note-se que a dimensão da quadrícula depende de N, o número de divisões de cada face).

Da análise da Figura 27, da Figura 28 e da Figura 29, conclui-se que os tempos de produção de imagem são bastante mais favoráveis para valores de N elevados. A explicação deste facto deve-se ao tamanho das listas que armazenam os raios previamente calculados e que, no caso de N=5 (listas com 16.384 elementos), conduzem a tempos de pesquisa significativamente elevados. Ao invés, no caso N=20, as listas são bastante mais curtas (64 elementos) e a pesquisa é muito mais rápida. Daqui decorre que, quando o número de elementos armazenados por par de quadrículas é elevado (N baixo), uma lista ligada não é a estrutura de dados adequada, necessitando-se de uma estrutura de dados mais elaborada.

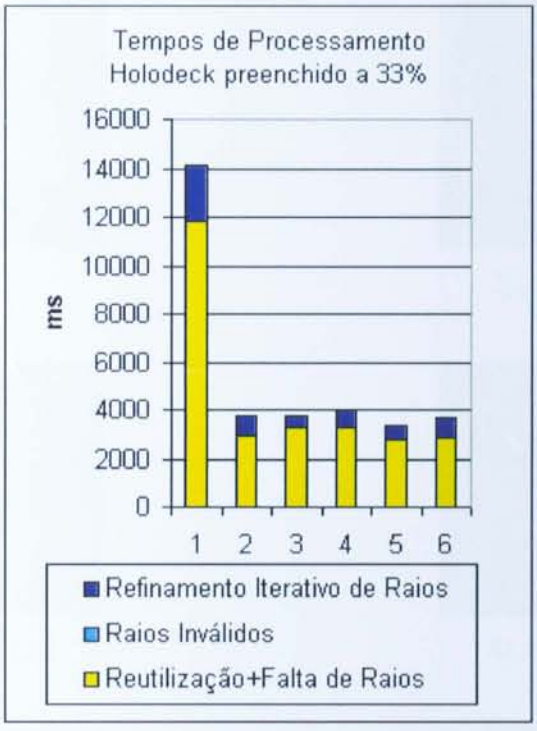


Figura 28: Tempos parciais de processamento para um *Holodeck* com N=10 preenchido a 33% e factor de proximidade a 50% – 5cm; rotação 10º

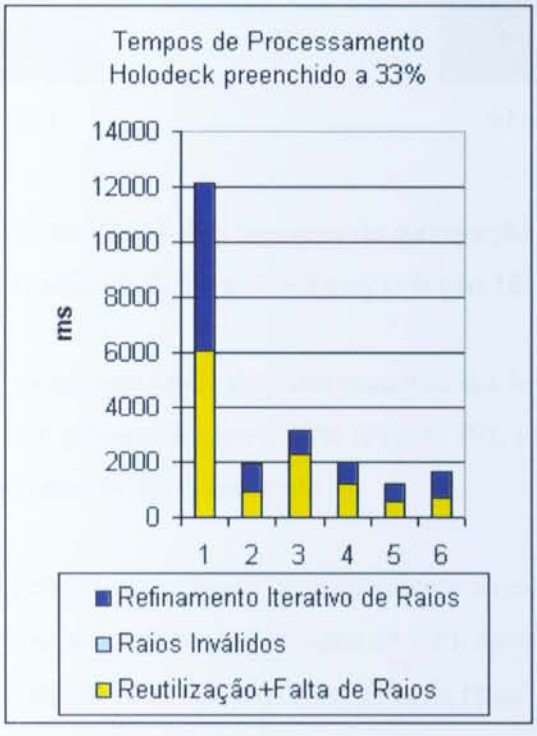
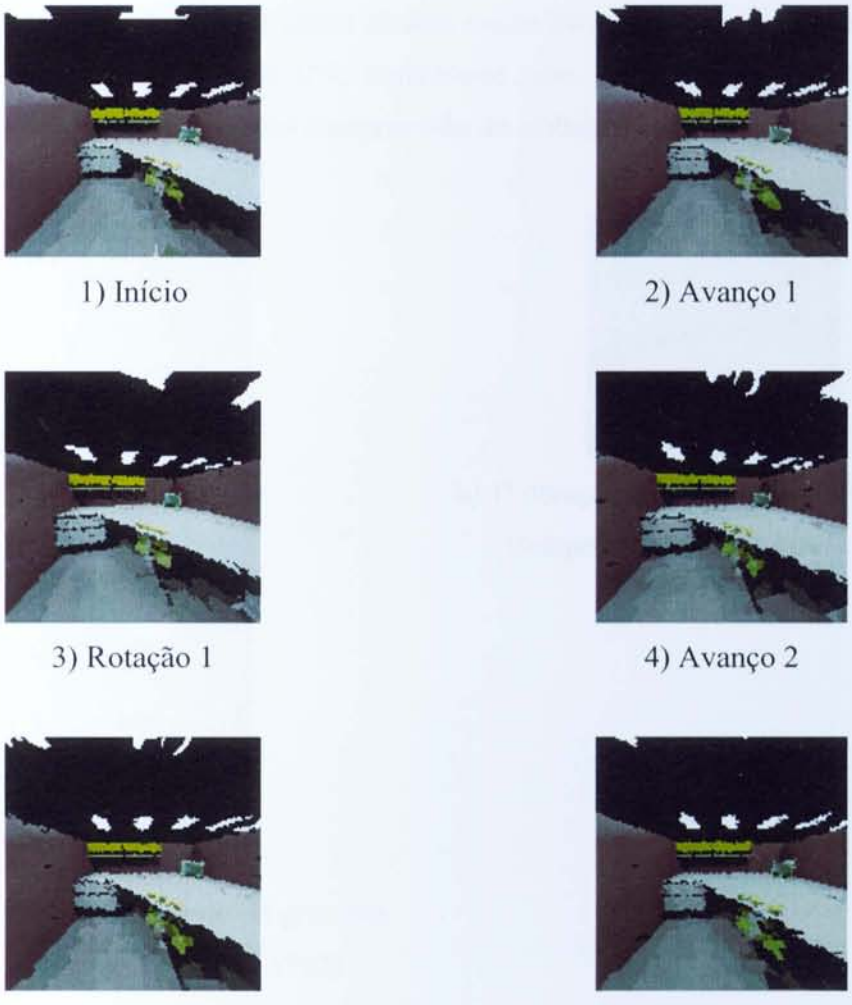


Figura 29: Tempos parciais de processamento para um *Holodeck* com N=20 preenchido a 33% e factor de proximidade a 100% – 5cm; rotação 10º



1) Início 2) Avanço 1

3) Rotação 1 4) Avanço 2

5) Rotação 2 6) Avanço 3

Figura 30: Evolução das imagens da navegação da cena (*Holodeck* 33%, passo 5 cm, rotação 10°)

Comparando, para N=20, os tempos totais de processamento das imagens do *Holodeck* vazio (Figura 24) com o *Holodeck* pré-preenchido a 33% (Figura 29), verifica-se uma redução na ordem de metade desses tempos de processamento.

A Figura 30 apresenta a evolução da imagem correspondente ao caso da Figura 29 (passo de avanço na cena igual a 5 cm, intercalado com rotação de 10°), apenas com a primeira fase do processo de produção de imagem (fase "reutilização+falta de raios").

Apesar das imagens da Figura 30 serem obtidas muito mais rapidamente do que na situação equivalente com o *Holodeck* a 0%, constata-se que possuem uma qualidade visual satisfatória, suficiente para uma boa compreensão da ambiência da cena.

a) Início da Figura 30

b) 1ª iteração do refinamento progressivo (tempo 3''68s; nº de *pixels*: 300)

c) 2ª iteração do refinamento progressivo (tempo 5''53s; nº de *pixels*: 5982)

Figura 31: Fases de refinamento da imagem inicial da Figura 30

Nas imagens da Figura 31, a fase "refinamento iterativo de raios" actua apenas para eliminar artefactos visuais em regiões da imagem nas quais existem variações bruscas de visibilidade (muito perceptíveis, por exemplo, nas arestas de objectos) ou de iluminação (pouco perceptíveis).

5.2.3 Ponto Óptimo de Pré-preenchimento do *Holodeck*

A Figura 32 apresenta tempos de processamento para navegação com N=20, F=64 e passo de avanço de 5 cm. A percentagem de preenchimento do *Holodeck* varia, tomando os valores

de 0%, 33% e 66% para cada um dos gráficos[6]. Da análise destes gráficos constata-se que é vantajosa a utilização do *Holodeck* preenchido a 33%, em comparação com o *Holodeck* vazio, dado que, neste caso, o tempo de cálculo de todas as imagens é superior. Também se pode constatar que o *Holodeck* preenchido a 66% apresenta tempos de processamento mais elevados do que a 33%. Isto deve-se ao facto da lista de raios por par de quadrícula aumentar, originando um acréscimo de tempo na pesquisa dessa lista.

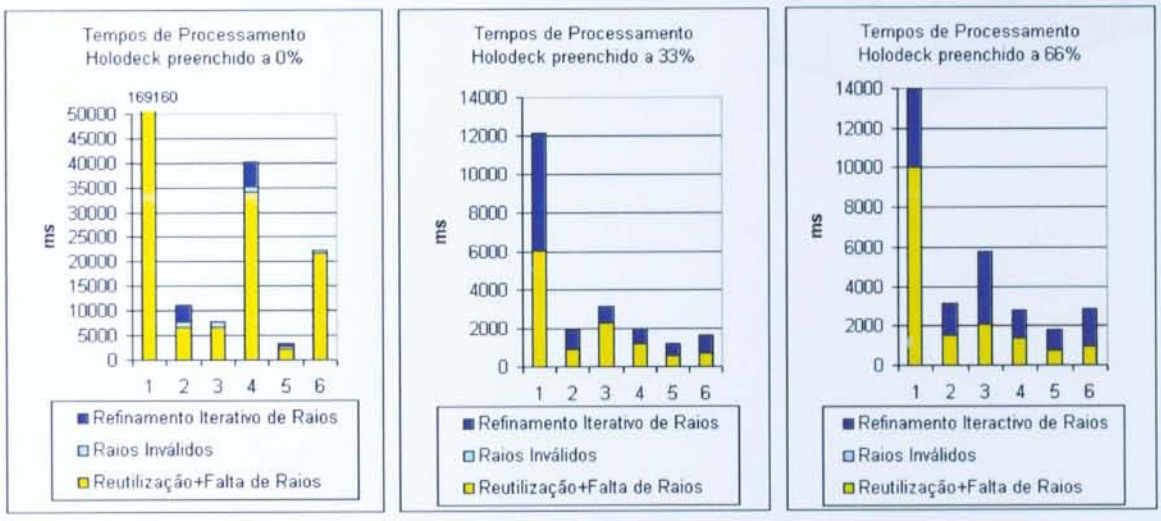
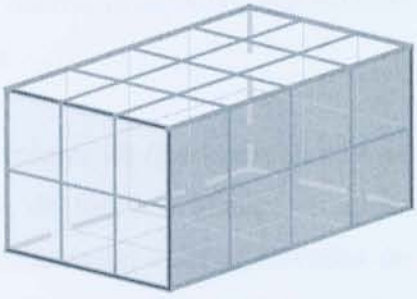


Figura 32: Comparação de tempos de processamento para diversos valores de pré-preenchimento do *Holodeck*

É preferível a existência de um ponto óptimo de enchimento inicial do *Holodeck*, a partir do qual o desempenho do protótipo não melhora substancialmente. Pelos resultados obtidos, esse ponto deverá estar próximo dos 33%. No entanto, o seu valor poderá depender da resolução da imagem a produzir e da parametrização do *Holodeck* (divisão de faces em quadrículas, raios a armazenar por par de quadrículas, tolerâncias definidas pelo utilizador). Dado que o *Holodeck* é uma caixa negra à volta de uma parte bem definida da geometria da cena, será de prever que esse ponto óptimo não dependa dos objectos nele contidos.

---

[6] O espaço físico necessário para armazenar *Holodecks* pré-preenchidos a 33%, 66% e 100% é da ordem de 200MB, 400MB e 600MB, respectivamente.



6 Conclusões

Esta dissertação descreve uma implementação de avaliação do método *Holodeck*, proposto por [Larson98a]. Relativamente a outras implementações, incluindo a de [Larson98a], esta tem a vantagem de permitir a utilização, em ambiente de *walkthrough*, de um motor de radiância fisicamente válido sobre uma plataforma computacional comum (PC/*Windows* 98). Para além disso, apresenta um faseamento mais explícito do refinamento da imagem e a possibilidade de controlo sobre esse refinamento por parte do utilizador.

Os resultados obtidos permitem concluir que o método *Holodeck* apresenta tempos de geração de imagem bastante inferiores aos tempos necessários quando se utiliza exaustivamente o motor de radiância.

Esta redução de tempo de processamento implica uma redução de qualidade de imagem. No entanto, presume-se que, durante o movimento do observador (em *walkthrough*), essa redução é aceitável. Numa posição de observação fixa, é possível incrementar a qualidade de imagem sob o controlo do utilizador, sem perda de tempo relativamente ao cálculo completo da imagem.

Quando o *Holodeck* se encontra inicialmente vazio e a navegação na cena é realizada através de avanços/rotações pequenos, conclui-se que se explora efectivamente a coerência espacial e verifica-se ser mínimo o recurso ao motor de radiância. Por outro lado, à medida que aumenta a distância do avanço do observador, perde-se essa coerência espacial, aumentando consideravelmente o cálculo de raios pelo motor de radiância.

Relativamente ao estudo do número de divisões a aplicar ao volume do *Holodeck*, conclui-se que, face a uma mesma quantidade de raios no *Holodeck*, de forma a manter a mesma quantidade de memória ocupada, se obtêm melhores resultados com valores elevados do número de divisões (20, nos casos testados). A justificação encontra-se no facto de as acções de pesquisa de raios em cada par de quadrículas se efectuar numa estrutura do tipo lista ligada. Realmente, com menos divisões por face o comprimento destas listas é maior, o que implica maiores tempos de pesquisa

O estudo do pré-preenchimento do *Holodeck* permite concluir que, para deslocações consideráveis na cena, a utilização do *Holodeck* vazio não é muito eficiente, apesar dos tempos de processamento serem inferiores aos correspondentes com uso exclusivo ao motor de radiância. Para o *Holodeck* pré-preenchido a 33%, quando o número de divisões é da ordem das 20, verifica-se que os tempos de processamento da imagem são bastante favoráveis. Para percentagens superiores a 33%, conclui-se que as listas de raios da estrutura de dados se tornam mais extensas, dificultando a sua pesquisa, pelo que os tempos de processamento voltam a piorar.

6.1 Trabalho Futuro

O motor de radiância usado neste trabalho para a traçagem de raios funciona como uma "caixa negra" (recebe raios e devolve radiâncias), pelo que o emprego de técnicas de paralelização na traçagem desses raios poderá aumentar o desempenho interactivo desta aplicação.

O pré-preenchimento do *Holodeck* com base na geometria da cena, a fim de lançar maior número de raios provenientes de locais mais prováveis de passagem, tais como corredores, portas, escadas, etc., torna-se muito interessante para futura implementação.

Como foi referido anteriormente, um ponto óptimo de enchimento inicial do *Holodeck*, pode ser encontrado, a partir do qual o desempenho do protótipo não melhora substancialmente. Este ponto óptimo pode, inclusive, ser dependente da complexidade da cena ou, caso esta situação não se verifique, poderiam ser encontrados os factores determinantes para a obtenção do referido ponto óptimo de enchimento.

A utilização de vários *Holodecks* numa cena mostra-se, face aos testes efectuados, bastante promissora, na medida em que estes podem ser colocados em regiões de geometria ou iluminação complicada, evitando-se assim traçagens demoradas de raios. A localização destes *Holodecks* poderá resultar de um processo semi-automático de análise da complexidade da cena.

Uma possibilidade para trabalho futuro é prever o próximo movimento do observador a partir do percurso efectuado, realizando o cálculo da direcção a seguir através da média de todas as direcções tomadas até à posição actual do observador.

Quando o utilizador solicita passos extensos na cena, o protótipo poderá apresentar o movimento cadenciado, de forma a aproveitar a coerência espacial. A sequência total de imagens, até atingir a posição final, poderá ser mais demorada, mas, em contrapartida, é enriquecida com a ilusão de movimento. Desta forma, o utilizador viaja na cena (através da evolução do movimento), não ficando tanto tempo em compasso de espera.

Um melhoramento adicional a este protótipo é a manutenção de distâncias aos objectos intersectados quando o utilizador se encontra dentro do *Holodeck*. Na presente implementação, se existirem objectos posicionados atrás do observador, embora dentro do *Holodeck*, obtém-se uma imagem que contém objectos mal posicionados. Uma análise de distâncias e direcções pode evitar esta situação.

O método de pré-preenchimento do *Holodeck* de acordo com a direcção retirada da posição inicial do observador pode ser melhorado se se realizar o enchimento pelas quadrículas visíveis, ao invés de se realizar o enchimento pelas faces visíveis. O número de raios que intersectam os pares de faces visíveis/invisíveis é bastante superior ao número de raios que intersectam os pares de quadrículas visíveis/invisíveis, uma vez que uma face que oculta outra não se sobrepõe, forçosamente, a todas as quadrículas pertencentes à face invisível.

Desta forma, o lançamento aleatório de raios que passam apenas pelos pares de quadrículas, permite a posterior reutilização de uma grande percentagem de raios armazenados no *Holodeck*, enquanto o observador se mantém numa posição próxima da inicial. Assim sendo, o número de raios a calcular no processo de pré-preenchimento será muito inferior e, consequentemente, este processo torna-se mais rápido. Este método permitiria diminuir o tempo de pré processamento, o tamanho do *Holodeck* ou, para o mesmo tamanho, concentrar mais raios em zonas importantes do *Holodeck*.

Dos estudos realizados, pode concluir-se que a pesquisa à lista de raios nos sentidos directo e inverso se torna ineficiente quando esta possui um elevado número de raios, ou seja, em situações em que o número de quadrículas por face é baixo. Para reduzir este problema na presente implementação coloca-se, à cabeça da lista, o primeiro raio encontrado para o correspondente par de quadrícula que se encontra dentro do limite do factor de proximidade durante uma operação de pesquisa. Assim, é grande a probabilidade de, na pesquisa seguinte, o primeiro raio na lista se encontrar dentro do limite do factor de proximidade para esse raio. Esta solução tem a desvantagem de as imagens obtidas serem muito grosseiras, obrigando a muitas iterações de refinamento. Como trabalho futuro, existe a possibilidade de encontrar uma método de aceleração que permita catalogar os raios que se encontram nessas listas para facilitar a sua pesquisa. Como exemplo de métodos de aceleração pode-se recorrer a [Kirk87] e [Arvo88].

9 Referências

[Arvo88] James Arvo, David Kirk, "A survey of Ray Tracing Acceleration Techcniques," SIGGRAPH'88 – Course Notes

[Blinn76] J. F. Blinn, M. E. Newell, "Texture and Reflection in Computer Generated Images," Communications of the ACM, Vol. 19, Nº 10, Outubro 1976, pp. 543-547.

[Catmull75] E. A. Catmull, "Computer Display of Curved Surfaces," Conference Proceedings on Computer Graphics, Pattern Recognition and Data Structure, Maio 1975, pp. 11-17.

[Chen95] S. E. Chen, "QuickTime VR: An image-Based Approach to Virtual Environment Navigation," Computer Graphics (SIGGRAPH'95 Conference Proceedings), Agosto 1995, pp. 29-38.

[Ferwerda96] J. Ferwerda, S. Pattanaik, P. Shirley, D. P. Greenberg, "A Model of Visual Adaptation for Realistic Image Synthesis," Proceedings of ACM SIGGRAPH'96, p. 249-258.

[Glassner95] Andrew Glassner, "Principles of Digital Image Synthesis", 1995.

[Gortler96] Steven Gortler, Radek Grzeszczuk, Richard Szeliski, Michael Cohen, "The Lumigraph," Computer Graphics Proceedings, Annual Conference Series, 1996.

[Kirk87] David Kirk, James Arvo, "Fast Ray Tracing by Ray Classification," SIGGRAPH'87

[Larson97] Greg Ward Larson, Holly Rushmeier, Christine Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes", IEEE Transactions on Visualization and Computer Graphics, Dezembro 1997.

[Larson98a] Greg Ward Larson, "The Holodeck: A parallel Ray-caching Rendering", 2nd Eurographics Workshop on Parallel Graphics and Visualization, 1998.

[Larson98b] Greg Ward Larson, Rob Shakespeare, "Renderig with Radiance: The Art and Science of Lighting Visualization," Morgan Kaufmann, 1998.

[Levoy96] Marc Levoy, Pat Hanrahan, "Light Field Rendering", Computer Graphics Proceedings, Annual Conference Series, 1996.

[Lippman80] A. Lippman, "Movie-Maps: An Application of the Optical Videodisc to Computer Graphics," Computer Graphics (SIGGRAPH'80 Conference Proceedings), 1980.

[Maciel95] P. W. C. Maciel, P. Shirley, "Visual Navigation of Large Environments Using Textured Clusters," in Symposium on Interactive 3D Graphics, ACM SIGGRAPH'95, 1995, pp. 95-102.

[McMillan97] Leonard McMillan Jr., "An Image Based Approach to Three Dimensional Computer Graphics," Chapel Hill, 1997

[NRC] Institute for Research in Construction do National Research Council, Canadá, www.nrc.ca/irc/

[Ofek98] E. Ofek, A. Rappoport, "Interactive Reflections on Curved Objects," Computer Graphics (SIGGRAPH'98 Conference Proceedings), Julho 1998, pp. 333-342.

[Regan94] M. Regan, R. Pose, "Priority Rendering with a Virtual Reality Address Recalculation Pipeline," Computer Graphics (SIGGRAPH'94 Conference Proceedings), 1994.

[Schaufler98] G. Schaufler, "Per-Object Warping with Layered Imposters," Proceedings of the 1998 Eurographics Workshop on Rendering Techniques, 1998.

[Segal92] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, P. Haeberli, "Fast Shadows and Lighting Effects Using Texture Mapping," Computer Graphics (SIGGRAPH'92 Conference Proceedings), Vol. 26, Nº 2, Julho 1992, pp. 249-252.

[Shade96] J. Shade, D. Lischinski, D. Salesin, T. DeRose, "Hierarchical Image Caching for accelerated Walkthroughs of Complex Environments," Computer Graphics (SIGGRAPH'96 Conference Proceedings), pp. 75-82, Agosto 1996.

[Sillion97] F. Sillion, G. Drettakis, B. Bodelet, "Efficient Imposter Manipulation for Real-Time Visualization of Urban Scenery," Proceedings of the 1997 Eurographics Forum on Computer Graphics, Dublin, 1997, pp. 207-218.

[Walter97] B. Walter, G. Alppay, E. Lafortune, S. Fernandez D. P. Greenberg, "Fitting Virtual Lights for Non-Diffuse Walkthroughs," Computer Graphics and Interactive Techniques (SIGGRAPH'97 Conference Proceedings), Agosto 1997, pp. 45-48.

[Walter99] B. Walter, G. Drettakis, S. Parker, "Interactive Rendering Using the Render Cache," Proceedings of the 10th Eurographics Workshop on Rendering, Junho 1995, pp. 27-38.

[Ward94] Greg Ward, "The RADIANCE Lighting Simulation and Rendering System", Computer Graphics Proceedings, Annual Conference Series, 1994.

[Whitted80] T. Whitted, "An Improved Illumination Model for Shaded Display," Communications of the ACM, vol. 23, no. 6, Junho 1980, pp. 343-349.