Faculdade de Engenharia da Universidade do Porto

Departamento de Engenharia Electrotécnica e de Computadores

Segmentação de Ambientes Virtuais Urbanos para Visualização em Arquitecturas com Recursos Limitados

Dissertação apresentada à Faculdade de Engenharia da Universidade do Porto para a obtenção do grau de Mestre

em Tecnologia Multimédia

Dissertação realizada sob a orientação científica de:

António Augusto de Sousa

(Professor Auxiliar da FEUP)

e co-orientação de

António Fernando Coelho

(Professor Auxiliar Convidado da FEUP)

Joana Gomes Rocha

2007

Agradecimentos

Este espaço é dedicado àqueles que deram a sua contribuição para que esta dissertação fosse realizada. A todos eles deixo aqui o meu agradecimento sincero.

Ao Professor Augusto Sousa, meu orientador, por ter tornado possível a realização desta tese, pelo apoio prestado e pela confiança depositada.

Ao Co-Orientador António Coelho, pela disponibilidade e cooperação durante a elaboração da tese.

Agradeço ao Instituto de Engenharia de Sistemas e Computadores do Porto pelo material que me facultaram para o desenvolvimento dos meus trabalhos de mestrado.

Agradeço também aos alunos do 3º ano de Engenharia Informática e Computação, que colaboraram na realização de testes ao protótipo desenvolvido.

A todas as pessoas mais próximas a mim pela coragem e força de vontade que sempre me transmitiram nos momentos de maior indecisão e que sempre me apoiaram e encorajaram ao longo de todos os meses de investigação e elaboração da tese.

Resumo

Os ambientes virtuais têm aplicações numa vasta gama de áreas do conhecimento e de uma forma bem diversificada. As necessidades do utilizador e a capacidade criativa fazem surgir novas e mais avançadas tecnologias.

Sistemas que recorrem a ambientes virtuais estão a invadir rapidamente a Internet com o objectivo de conjugar as vantagens das duas tecnologias e deste modo tirar o maior partido e rentabilidade. Neste contexto, os sistemas baseados em ambientes virtuais urbanos começam a ser vulgares, funcionando como *interface* em vários domínios de aplicação.

Os cenários urbanos tridimensionais colocam desafios de representação do mundo real devido às limitações da capacidade dos computadores. Os algoritmos de visibilidade e técnicas de oclusão orientadas para ambientes urbanos têm crescido significativamente, impulsionados pelo aumento na complexidade dos modelos e na necessidade de melhores formas de navegação interactiva.

Nesta dissertação foi desenvolvida uma metodologia que possibilita ao utilizador navegar num modelo virtual 3D de uma cidade, segmentando a cena de forma adaptativa, de modo a optimizar acessos, mas mantendo a preocupação de não excluir objectos cuja visualização é possível por sobre outros objectos, mais próximos. O protótipo de teste diz respeito a uma parte da zona baixa da cidade do Porto e o suporte tecnológico, julgado adequado ao desenvolvimento de ambientes virtuais para a Internet, é o VRML/X3D. Esta linguagem abre novas possibilidades de integração de conteúdos 3D com a *Web* e permite a visualização de informação com diferentes níveis de complexidade, em função do utilizador.

O sistema apresenta o modelo, de forma automática, em função da localização do utilizador, tendo em conta o campo de visão, os elementos considerados oclusores, as zonas a descoberto e ainda os edifícios com uma altura relevante. À medida que o utilizador navega pelo mundo virtual, os modelos são carregados parcialmente permitindo, deste modo, um passeio linear.

Abstract

The virtual environments have applications in several areas of knowledge and in quite diversified ways. User's needs and creative capacities make way for new and more advanced technologies.

Virtual environments systems are quickly invading the Internet with the intent to conjugate the advantages of the two technologies and in this way to get the most of both of them. In this context, urban virtual environment based systems start to be used frequently, functioning as interface between several application domains.

Urban three-dimensional scenes place challenges in presenting the real world due to the limitations of computers capacities. Visibility algorithms and guided techniques of occlusion for urban environments have grown significantly, stimulated by the increasing complexity of the models and the necessity of better forms of interactive navigation.

In this work a methodology was developed, that makes possible for the user to navigate in a 3D virtual model of a city, segmenting the scene in an interactive way, in order to optimise accesses, but keeping the concern not to exclude objects whose visualization is possible over near objects.

This test prototype is about a part of Porto's downtown, and the adequate technology for Internet virtual environment development was used, witch is the VRML/X3D. This language opens new possibilities for 3D contents integration with the Web and allows the visualization of information with different levels of complexity, depending on the user.

The system presents the model, in an automatic form, according to the position of the user, considering the field of vision, the elements considered to cause occlusions, the open areas and buildings with significant height.

While the user navigates in the virtual world, the models are partially loaded allowing a linear stroll.

Índice

Índice de Figuras

Índice de Gráficos

Índice de Listagens

Índice de Tabelas

CAPÍTULO 1

INTRODUÇÃO

Índice do Capítulo

1 - Introdução

O acesso a informação contextualizada no espaço tem vindo a tomar uma importância crescente nos últimos anos. Tal como diz Vasco Ferreira, em [Ferreira06], "Na sociedade de informação em que vivemos é necessário, cada vez mais, disponibilizar a informação de forma acessível e igualitária a todos os cidadãos. A possibilidade de consultar e analisar a informação com base em critérios geográficos veio enriquecer o significado da mesma, permitindo deduzir conclusões de forma mais intuitiva e visível. A informação georeferenciada reveste-se assim de bastante importância no apoio à decisão, em áreas como o urbanismo e gestão do território, ambiente, economia, *marking*, etc.".

Neste contexto, os ambientes virtuais têm vindo a surgir, nos últimos anos, como soluções de imersão, interacção e representação em diversas áreas como arquitectura, planeamento urbano, arqueologia, medicina, educação, etc.

Num ambiente virtual é de vital importância a qualidade da imagem e a interacção. A representação dos objectos que compõem o mundo tem de ser o mais realista possível, de modo a que o utilizador compreenda e analise correctamente a informação. Por outro lado, o utilizador pretende uma interacção com o mundo 3D em tempo real, isto é, a resposta a acções do utilizador e consequente visualização do ambiente têm de ser efectuadas a uma cadência suficientemente elevada para que a atenção do utilizador não seja desviada ou perturbada.

Assim, a criação de ambientes virtuais torna-se uma tarefa complexa, pela grande quantidade de objectos a implementar em 3D como ruas, quarteirões, casas, edifícios, e pelas limitações na capacidade gráfica do computador. Deste modo, é necessário definir critérios de relevância em função do objectivo a atingir, tendo ainda em conta, face aos objectivos de utilização em ambiente Internet, as limitações na largura de banda das redes. Dadas as limitações descritas ao nível da criação e visualização de ambientes virtuais, surge a necessidade de criar uma metodologia com o intuito de segmentar o ambiente virtual de modo a que qualquer utilizador, dotado de um dispositivo de recursos

limitados ou de um serviço de acesso limitado à rede, possa ter acesso à informação. Para que o utilizador se sinta completamente integrado no ambiente virtual, é inevitável que a metodologia permita a visualização dos elementos existentes no ambiente virtual em função do utilizador.

Deste modo, o objectivo desta dissertação é:

- Definir uma metodologia que, de forma dinâmica, realize a segmentação de um espaço urbano.
- A metodologia a desenvolver deve ter em conta que o utilizador navega na forma de "passeio" e, consequentemente, a segmentação deve ser em função das ruas. Os elementos visíveis no campo de visão do utilizador são representados dinamicamente, tendo em conta aspectos como a oclusão, espaços abertos e edifícios com uma altura superior à média dos restantes.
- Para a sua concretização, nomeadamente para efeitos de teste e avaliação, será então necessário criar um ambiente virtual de um espaço urbano específico. O ambiente virtual escolhido corresponde a uma área limitada da cidade do Porto, permitindo uma navegação 3D interactiva na forma de "passeio".

1.1 - Organização da Dissertação

Esta dissertação está dividida em 7 capítulos, adicionados de referências e anexos técnicos contendo, no final de cada um, com excepção do presente, uma síntese com o resumo dos principais tópicos abordados.

O presente capítulo, introdução, apresenta a relevância e os objectivos da dissertação.

O cálculo de visibilidade, a integrar no processo de síntese de imagem, desempenha um papel crucial no âmbito da computação gráfica. Os primeiros algoritmos tinham como objectivo determinar as superfícies visíveis dos objectos que compõem uma cena 3D [Sutherland74]. Existe actualmente uma grande diversidade destes algoritmos e de metodologias de aceleração, de acordo com a variedade de problemas nesta área. O capítulo 2 descreve de uma forma geral algumas das técnicas de cálculo de visibilidade e técnicas de redução da complexidade das cenas 3D, enquadrando-as com os projectos desenvolvidos, com particular destaque para as técnicas de oclusão.

VRML (*Virtual Reality Modeling Language*) é uma tecnologia utilizada na Internet para descrever cenas 3D. A última especificação apresentada é o VRML97, ainda com algumas ambiguidades e uma certa instabilidade no padrão. O X3D, seu descendente, surge também como forma de cobrir várias lacunas da especificação do VRML, promovendo uma maior flexibilidade. O capítulo 3 especifica, para estas duas linguagens,

a sua estrutura, os elementos que compõem uma cena, e as tecnologias associadas para a interacção entre o mundo virtual e o utilizador.

A visualização interactiva de cenas complexas torna-se cada vez mais importante em diversas áreas. Embora o desempenho do *hardware* gráfico tenha evoluído bastante nos últimos anos, continua a procura pela visualização interactiva de ambientes virtuais cada vez mais complexos, apesar de ainda existirem limitações com as tecnologias actuais, principalmente em relação à largura de banda das redes de telecomunicações quando se trata de visualização de cenas adquiridas, à distância, em servidores. No capítulo 4 são descritos os processos envolvidos na criação de um modelo de uma cidade virtual e especifica-se a metodologia desenvolvida para a realização da segmentação do ambiente virtual, assim como o cálculo de visibilidade aplicado, tendo em conta algumas das técnicas de oclusão descritas no capítulo 2.

A implementação do protótipo desenvolvido no contexto desta dissertação é descrita no capítulo 5, especificando os seus requisitos, a arquitectura e as tecnologias utilizadas. A implementação pressupõe uma fase de segmentação do modelo através de um Sistema GIS, a criação do ambiente em VRML e a posterior conversão em X3D, assim como a *interface* da aplicação.

No capítulo 6 apresenta-se uma análise comparativa ao protótipo desenvolvido e a uma aplicação mais *standard* com características pouco significativas em termos de segmentação. Por fim, é realizado um teste ao protótipo através do preenchimento de um inquérito por um conjunto de utilizadores, avaliando-se os respectivos resultados.

Finalmente, no capítulo 7, apresentam-se as principais conclusões, e apontam-se algumas pistas para desenvolvimentos futuros.

CAPÍTULO 2

TÉCNICAS DE ACELERAÇÃO DA VISUALIZAÇÃO DE CENAS COMPLEXAS

Índice do Capítulo

2 - Visualização de Cenas 3D Complexas

A quantidade de aplicações que surgem, actualmente, na área dos sistemas interactivos de visualização está a crescer exponencialmente, em áreas como o planeamento urbano, navegação para automóveis, a visualização de voo para uso militar, os estudos climatéricos e ambientais, o turismo virtual, a educação, etc.

Com o crescimento da utilização de ambientes 3D, também o *hardware* gráfico tem evoluído. No entanto, dispositivos móveis como o PDA (*Personal Digital Assistant*), o *Tablet PC* da Microsoft ou o *Smartphone*, ainda possuem limitações ao nível do desempenho, particularmente no caso da visualização de modelos 3D. Outra limitação bastante relevante na visualização de ambientes 3D prende-se com as tecnologias relativas à largura de banda das redes de telecomunicações. Devido a estes factos, têm sido desenvolvidas diversas técnicas com o intuito de adequar os ambientes 3D mais complexos à visualização, nomeadamente neste tipo de dispositivos.

Este capítulo encontra-se dividido em quatro secções principais. A primeira secção consiste numa descrição geral do processo de *rendering*. A segunda secção foca os algoritmos que têm sido desenvolvidos no âmbito do cálculo de visibilidade, sendo dado um maior ênfase ao algoritmo *Z-buffer*. A terceira secção diz respeito às técnicas de redução da complexidade das cenas 3D, na qual são focados alguns aspectos essenciais na visualização de objectos; descrevem-se as duas categorias de técnicas utilizadas para acelerar o processo de visualização de ambientes complexos, concretamente, as técnicas de remoção rápida de objectos e as técnicas de redução da complexidade geométrica. A última secção deste capítulo descreve com maior detalhe algumas das técnicas utilizadas em ambientes urbanos, visto ser a área na qual se centra esta dissertação.

2.1 - Processo de *Rendering*

O processo de criação de imagens realísticas, *rendering*, envolve diversas etapas, as quais são executadas em função do algoritmo. A primeira etapa consiste na definição dos objectos que compõem a cena isto é, a descrição da geometria dos objectos da cena incluindo as respectivas transformações geométricas, as informações sobre os materiais que os constituem, como cor, a textura, etc. A fase seguinte diz respeito à especificação do ponto de observação e às condições de iluminação. A etapa seguinte compreende a aplicação de algoritmos com o intuito de determinar as superfícies visíveis ao observador. Sucede-se o cálculo de iluminação das superfícies baseado na posição, na orientação e nas características das superfícies e na iluminação que lhes é projectada.

O processo de *rendering* pode ser executado previamente ou em tempo real. O pre-*rendering* é um processo computacional intensivo tipicamente usado para a criação de filmes, enquanto o *rendering* em tempo real é frequentemente utilizado em jogos de vídeo 3D ou em ambientes com uma intensa interacção com o utilizador.

Desta forma, o *pipeline* gráfico é o conjunto de etapas realizadas pelo *hardware* para a obtenção da imagem final, podendo ser esquematizado como é apresentado na Figura 1 [Foley97].

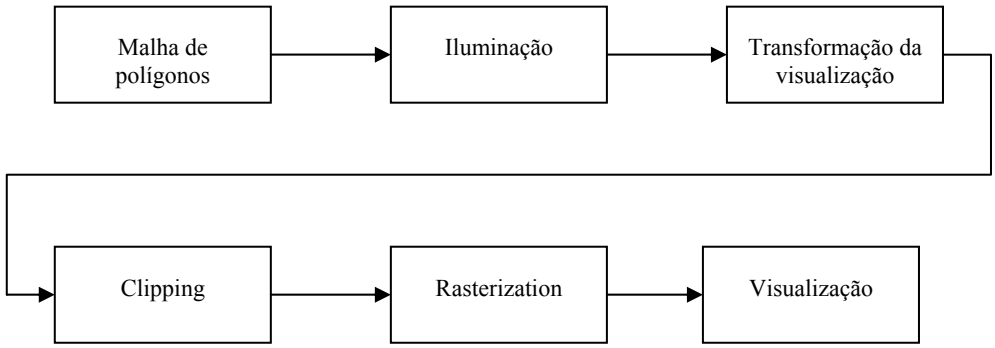


Figura 1 – *Pipeline* gráfico típico

A malha de polígonos diz respeito às estruturas de dados que compõem o espaço 3D e a forma como estão organizadas. A etapa seguinte consiste no cálculo da iluminação, aplicado, normalmente, apenas aos vértices de um polígono tendo em conta, a posição definida para a fonte de luz e as propriedades da superfície. Os resultados obtidos são mais tarde interpolados na fase de rasterização. A etapa de transformação de dados consiste na conversão das coordenadas no sistema local 3D (*object coordinate system*) para o sistema de coordenadas 3D em função da orientação do observador virtual ou sistema de coordenadas do olho (*eye coordinate system*). O primeiro é o sistema no qual o

objecto se encontra antes de ser posicionado na cena isto é, um sistema ortonormal de coordenadas locais. Isto permite que um determinado objecto possa ser muitas vezes utilizado num ambiente virtual com posições e orientações diferentes. O objecto é descrito em coordenadas locais e cada cópia do objecto sofre a transformação necessária em função do local onde é posicionado. O sistema de coordenadas do olho posiciona o observador na origem e, desta forma, todos os objectos da cena são posicionados em relação a estes eixos. Normalmente, o eixo do X é o eixo da direita, o eixo do Y corresponde ao eixo vertical, e o eixo do Z corresponde ao eixo na direcção da perpendicular ao monitor (Figura 2).

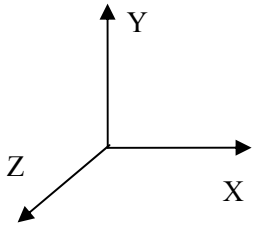


Figura 2 – Sistema de coordenadas do olho

A fase de *clipping* consiste em determinar as primitivas que se encontram visíveis ao observador reduzindo, desta forma, a quantidade de primitivas a representar (Figura 3) [Raposo04]. Nesta fase, as primitivas são testadas contra os limites do rectângulo do *clipping*, resultando dois conjuntos: um conjunto com as primitivas fora do rectângulo as quais, são extraidas do conjunto de elementos a representar, um segundo conjunto com as primitivas que se encontram dentro do rectângulo e consequentemente permanecem no conjunto de elementos a representar.

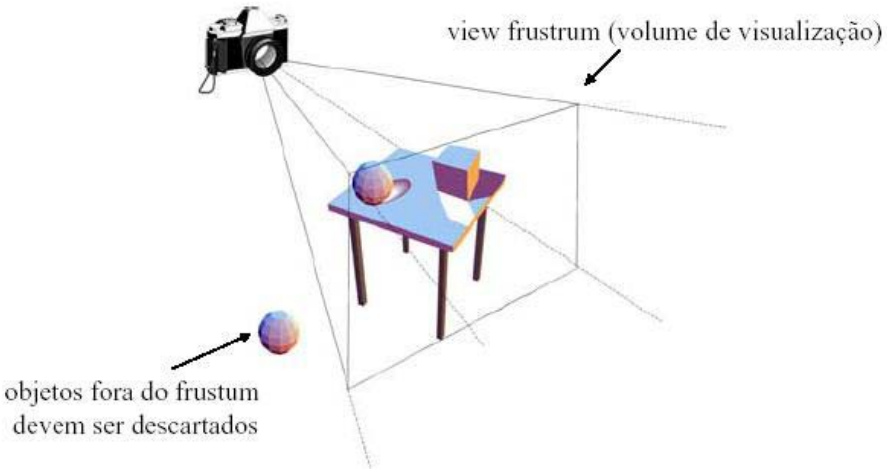


Figura 3 – *Clipping*

A *rasterization* é o processo de conversão de uma imagem constituída por primitivas geométricas do tipo linha e polígono (*vector graphics format*), em uma imagem com uma estrutura representada por uma grelha rectangular ou *pixels*. O objectivo desta operação consiste em determinar as quadrículas que representam cada uma das primitivas gráficas (Figura 4) [Sullivan06]

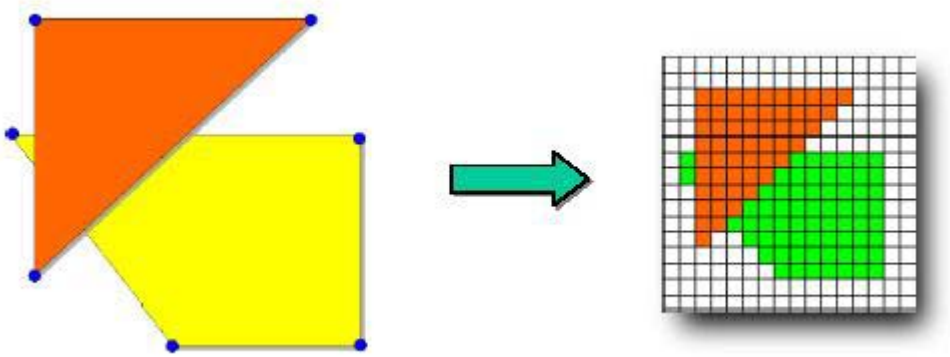


Figura 4 – *Rasterization*

Por fim a imagem final é visualizada no monitor do computador ou em outro dispositivo similar.

2.2 - Cálculo de Visibilidade

Os algoritmos de cálculo de visibilidade têm evoluído bastante devido à necessidade de representar modelos cada vez mais complexos e de permitir uma interacção mais realísta do utilizador com o ambiente. O utilizador procura, cada vez mais, interactuar em tempo real com os modelos 3D e não admite esperar, por vezes largos minutos, que uma nova imagem surja. Por outro lado e, de certa forma, em sentido inverso, quanto mais realista e complexo é o modelo, mais o utilizador se sente preso a ele.

O cálculo de visibilidade engloba as técnicas utilizadas pelo *hardware* gráfico de modo a simplificar a visualização da cena final. O cálculo de visibilidade é de grande importância na determinação das superfícies não visíveis de modo a remover as superfícies que se encontram ocultas ou parcialmente ocultas por outras. Este processo denomina-se de Técnica de Remoção de Faces Ocultas (*Hidden Surface Algorithm*), o qual normalmente, é utilizado em combinação com a fase de *rasterization* do *rendering pipeline*. Esta técnica engloba diversos tipos de algoritmos para executar a remoção das superfícies ocultas: algoritmo de *Ray tracing*, algoritmo do Pintor, algoritmo de Z-buffer, o *Scanline rendering*.

Os diversos algoritmos diferem entre si mas, contudo, baseiam-se na ordenação e coerência para melhorar os seus desempenhos [Sousa91]. Estes algoritmos podem ser

divididos em métodos que trabalham no espaço do objecto (*object-space*) e métodos que trabalham no espaço da imagem (*image-space*). Os primeiros métodos executam os cálculos à precisão arbitrária, geralmente a precisão disponível no computador. O objectivo destes métodos é determinar exactamente o que a imagem deve ser, pelo que esta será a correcta mesmo que ampliada várias vezes [Sutherland74]. Os segundos métodos procuram resolver o problema *pixel* a *pixel* e com uma menor definição, geralmente, em função do monitor. Por outras palavras, os algoritmos no espaço do objecto perguntam se cada objecto potencialmente visível na cena é visível, enquanto os algoritmos do espaço da imagem perguntam o que é visível dentro de um *pixel* do monitor [Sutherland74].

O algoritmo de *Ray tracing* consiste em gerar raios de luz do olho para trás através do plano de imagem na cena, sendo depois verificado quais os que intersecção as faces dos objectos na cena (Figura 5) [Campbell03]. Quando um raio intersecta um objecto passam a considerar-se também os raios reflectidos e refractados assim como os objectos que são visíveis através deles.

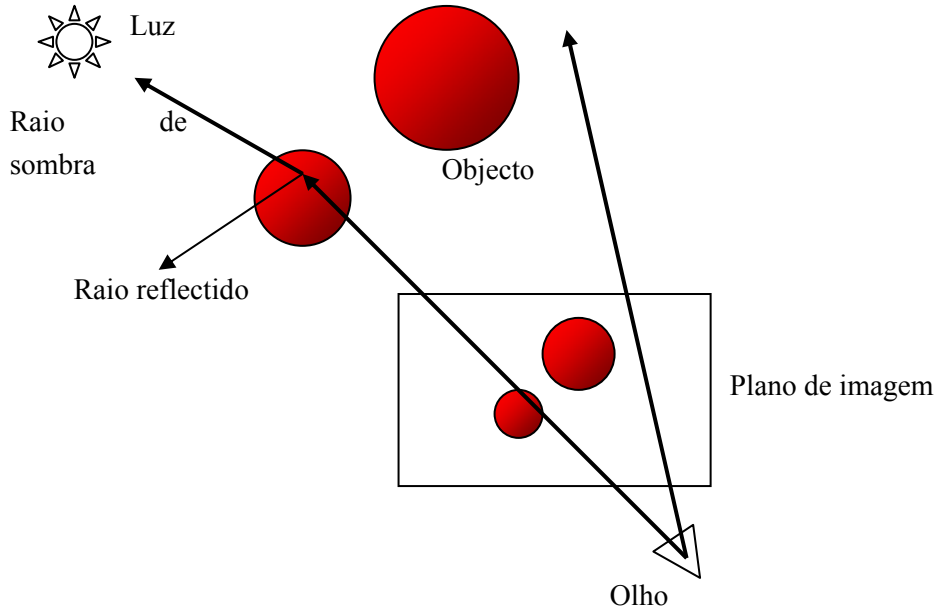


Figura 5 – Algoritmo *Ray tracing*

O algoritmo do Pintor baseia-se na ideia de pintar os objectos mais distantes na cena e depois pintar sucessivamente os que estão mais perto (Figura 6) [Wikipedia07]. Este algoritmo requer que os objectos estejam ordenadas de acordo com a distância entre cada objecto e o observador.

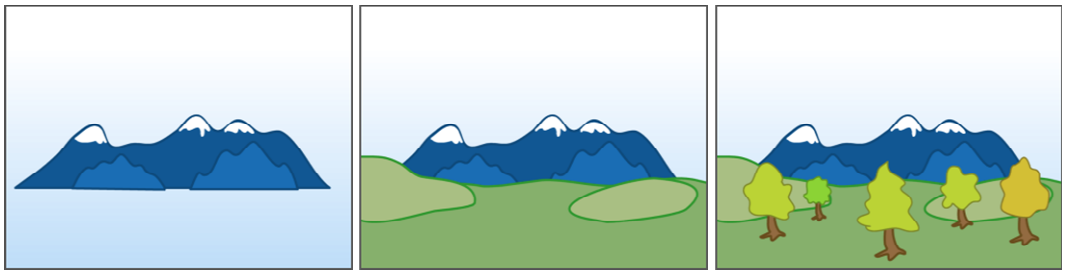


Figura 6 – Algoritmo do Pintor

O algoritmo Z-buffer foi criado por Catmull e requer, para cada *pixel* do monitor, a existência de dois *buffers*, um *color buffer* que armazena a cor actual de cada *pixel*, e um segundo *buffer*, *depth buffer*, que armazena, para cada *pixel*, a profundidade do objecto mais próximo até então, a ser desenhado naquele *pixel*. Este algoritmo é descrito com mais pormenores na secção 2.2.1.

O algoritmo *Scanline rendering* inicia-se com a ordenação da lista de todos os polígonos visiveis em função da coordenada em Y (Figura 7) [Esperança06]. Para cada plano de varrimento do monitor é calculada a intersecção deste com cada um dos polígonos da lista ordenada. Os polígonos que se revelam não visíveis são removidos da lista.

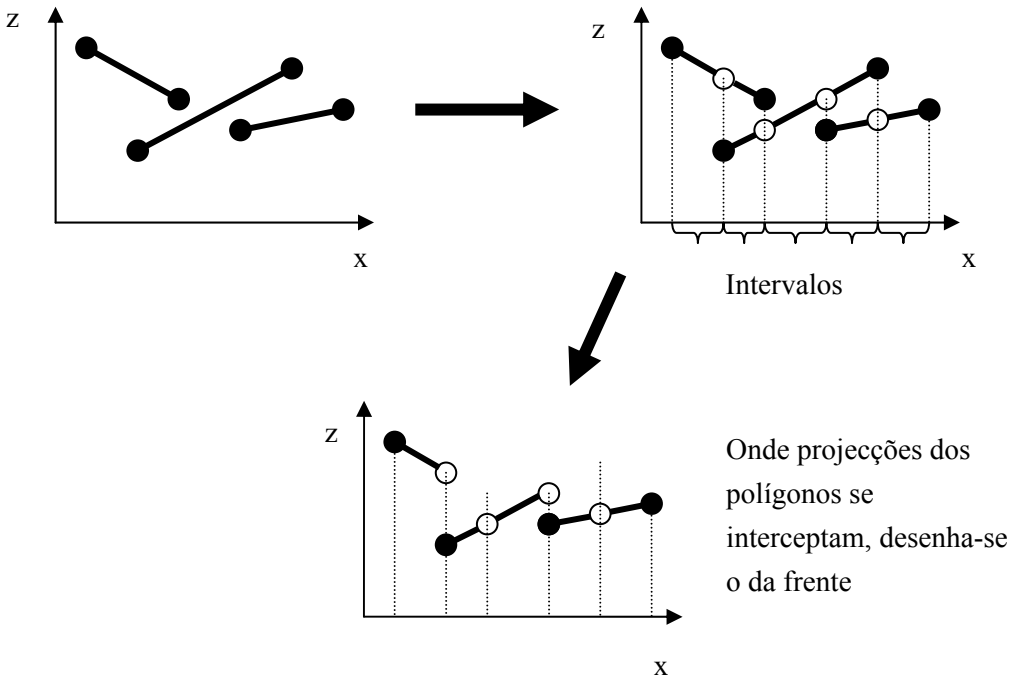


Figura 7 – Algoritmo *Scanline rendering*

2.2.1 - Z-buffer

O algoritmo *Z-buffer*, sendo simples e eficiente, consiste na utilização de dois *buffers* do tamanho da janela de exibição, um para guardar os valores de profundidade (*Z-buffer*) e outro para guardar valores de cor (*color buffer*). O *Z-buffer* é inicializado com o maior valor de profundidade possível e o *color buffer* é inicializado com a cor de fundo da imagem. Depois, cada face projectada é percorrida, *pixel* por *pixel*, e o valor de profundidade de cada *pixel* é comparado com o valor já armazenado no *Z-buffer*: se este *pixel* estiver mais próximo do observador, coloca-se o seu valor de profundidade no *Z-buffer* e o seu valor de cor no *color buffer*. Assim, no final do processamento o *color buffer* irá conter a imagem final.

Para representar cenas de grande dimensão quer de interiores quer de exteriores, são necessárias, frequentemente, grandes quantidades de polígonos. Ao visualizar estas cenas em tempo real, o *hardware* gráfico, baseado em cálculo de visibilidade por *Z-buffer* [Bittner03], não consegue processar um número tão elevado de primitivas. A secção seguinte apresenta algumas das técnicas desenvolvidas com o intuito de reduzir a quantidade de objectos a serem processados pelo *hardware* gráfico.

2.3 - Técnicas de Redução da Complexidade das Cenas 3D

A visualização de ambientes urbanos virtuais constitui um enorme desafio para os sistemas interactivos de visualização, devido à grande complexidade geométrica da informação, bem como à diversidade de condicionantes na síntese de imagem, como texturas, luzes, materiais, transparências, entre outras.

Um modelo 3D compreende, nomeadamente, a sua geometria, a topologia, os materiais, as normais e as texturas. A geometria de um modelo do tipo poliedro, o mais vulgar, diz respeito às coordenadas dos vértices, enquanto que a topologia especifica como os vértices, as arestas e as faces estão unidos. A topologia em poliedros é descrita usando triângulos ou quadriláteros, por serem as primitivas mais simples e as mais indicadas para o *hardware* gráfico. No entanto, outras primitivas como as NURBS, as curvas de Bézier, o cubo, a esfera, o cilindro ou o cone poderão ser utilizadas em modelos mais complexos. Os materiais têm várias propriedades para especificar como a luz é reflectida pela geometria dos objectos e como produzem a sua cor. Exemplos dessas propriedades são os coeficientes de reflexão difusa e especular, assim como a transparência. A cor para cada vértice de um polígono é calculada através da localização da fonte de luz e respectivas propriedades, bem como das propriedades do material associado ao vértice e do vector normal ao polígono naquele vértice. O vector normal descreve a orientação da superfície, e é necessário para calcular o ângulo de incidência dos raios de luz sobre o polígono. O *hardware* gráfico calcula a iluminação nos vértices do polígono, segundo um modelo de iluminação local, simples por natureza [Foley97].

Os métodos de redução da complexidade diminuem a quantidade de informação a ser tratada pelo *hardware* gráfico, nomeadamente pela etapa de cálculo de visibilidade. Trata-se no fundo de utilizar, do conjunto de polígonos a tratar pelo algoritmo *Z-buffer*, todos os polígonos que, com alguma certeza não são visíveis.

Existem várias técnicas para visualizar ambientes complexos o mais eficientemente possível. Existem métodos conservativos, os quais produzem imagens fidedignas, mas também existem métodos que sacrificam a fidelidade da imagem para conseguirem uma melhor interactividade. Um método é classificado de conservativo quando inclui pelo menos todos os objectos visíveis além de um certo número de objectos não visíveis [Cohen03]. Um objecto ocluso pode ser classificado de visível mas nunca um objecto visível pode ser definido como ocludido [Cohen03].

As diversas técnicas que foram surgindo para resolver o problema da visibilidade de cenas complexas em tempo real podem ser divididas em duas áreas. A primeira, diz respeito às técnicas de remoção rápida de objectos e divide-se em técnicas de remoção por campo de visão (*view-frustum culling*), técnicas de oclusão (*occlussion culling*) e técnicas de remoção de faces ocultas (*back-face culling*). A segunda área diz respeito às técnicas de redução da complexidade geométrica e divide-se em técnicas de redução de detalhes geométricos (*geometric detail reduction*) e técnicas de aceleração baseadas na imagem (*image-based acceleration*).

A implementação de uma ou de várias técnicas depende, principalmente, da localização do utilizador no ambiente virtual e do seu campo de visão.

As secções seguintes descrevem, de uma forma geral, cada uma das técnicas apresentadas.

2.3.1 - Técnicas de Remoção Rápida de Objectos

Na visualização de um ambiente que contém um número muito elevado de polígonos acontece que, normalmente, só uma parte dos polígonos é visível. Assim, de forma a acelerar a fase de *rendering*, utilizam-se vários métodos de remoção de polígonos. Estes métodos de remoção, de uma forma geral, determinam um conjunto de objectos potencialmente visíveis (PVS – *Potentially Visible Set*) para uma dada região ou para um determinado ponto de vista.

As principais técnicas de remoção rápida (*cull*) de objectos são o *occlusion culling*, *view-frustum culling* e *back-face culling* [Martin00]. As técnicas serão apresentadas nas secções seguintes pela ordem pela qual são tipicamente executadas num sistema de *rendering*.

2.3.1.1. Occlusion Culling

Como se viu anteriormente, o cálculo de visibilidade é feito, geralmente, pelo *hardware* gráfico presente na placa gráfica do computador, através do algoritmo *Z-buffer* (secção 2.2.1). Em ambientes virtuais complexos, em que estão presentes vários milhares de objectos, a sua utilização torna-se ineficiente devido ao grande volume de dados a serem carregados na memória principal e ao enorme desperdício de recursos computacionais. Mesmo após a eliminação de objectos por via das técnicas de remoção por campo de visão, a quantidade de objectos restantes continua muito elevada, sendo que, destes, somente um pequeno número será finalmente visível.

Uma família de técnicas vulgarizadas nos últimos tempos com o objectivo de reduzir ainda mais o número de objectos é a remoção por oclusão.

Estas técnicas, como foi referido na secção 2.3, podem ser conservativas ou não-conservativas: o conjunto da visibilidade conservativa inclui pelo menos todos os objectos visíveis e ainda um certo número pequeno de objectos não visíveis; o conjunto da visibilidade não conservativa não garante que todos os polígonos visíveis sejam identificados.

O trabalho apresentado por Agarwal [Agarwal01] classifica-se como sendo uma técnica de oclusão não-conservativa, executada em tempo real. Esta abordagem consiste em calcular a oclusão com base em coerência temporal, extraída de *frames* consecutivas de um ambiente constituído pelos mesmos objectos e com pequenas variações na sua geometria. O algoritmo também calcula uma estimativa conservativa do tempo, denominada *time stamp*, para cada cena, até um tempo limite pré-definido. Terminado o tempo limite, a cena deve ter a sua oclusão processada. As *frames* que exigem re-processamento, denominadas *critical frames*, ocorrem devido a mudança brusca na direcção do observador ou remoção de oclusores. Nestes casos é necessário executar previamente uma operação de oclusão, de modo a tirar vantagens da coerência temporal através da selecção de um conjunto de oclusores e posterior geração de máscaras hierárquicas de oclusão de forma a que objectos marcados permaneçam oclusos para diversas *frames* subsequentes.

Uma abordagem conservativa é apresentada por Cohen [Cohen98], a qual se baseia na divisão do ambiente em células bidimensionais. A estas células é feito um pré-processamento de forma a calcular grandes oclusores e a obter o conjunto de objectos potencialmente visíveis (PVS).

Em Wonka [Wonka00], o algoritmo desenvolvido permite acelerar o cálculo de visibilidade, possibilitando a sua utilização em tempo real para ambientes urbanos extensos, pela utilização de um novo método de pré-processamento com fusão dos oclusores. Este algoritmo determina os limites da sombra do objecto (*umbra*) como consequência de oclusores múltiplos, sendo calculadas as fusões das sombras de vários

oclusores e das penumbras. A sombra diz respeito à área que não pode ser visualizada a partir de um ponto de vista, enquanto que a penumbra é a região na qual pode ser vista parte ou o todo, a partir de um determinado ponto de vista. O método é conservativo, pois nunca declara objectos visíveis como invisíveis.

Uma outra abordagem conservativa de pré-processamento da oclusão é a desenvolvida por Schaufler [Schaufler00], através da fusão de pequenos oclusores. Os *voxels*[1] que constituem o ambiente são divididos em "opacos" (internos a objectos), em "vazios" (não pertencentes a nenhum objecto), e em "limites", os quais limitam os objectos. Terminada a classificação, as regiões de *voxels* vazios são divididas em células visíveis, através das quais são definidas linhas que se ligam pelos limites aos limites dos objectos, originando volumes de oclusão. A fusão de pequenos oclusores é testada de forma a tornar mais eficiente a oclusão.

Para além da divisão da técnica de remoção de objectos por oclusão em métodos conservativos e não conservativos, existem vários outros aspectos a considerar, como a região de interesse, o critério da divisão da região em células e portais e ainda a possibilidade de restrição a ambientes 2D ou 2.5D.

A região de interesse no cálculo de visibilidade relaciona-se com a possibilidade de um dado algoritmo executar os cálculos apenas relativamente à posição do observador (baseadas em pontos), ou executar os cálculos de forma a estes serem válidos para qualquer lugar em uma dada região do espaço (baseadas em regiões). As primeiras são mais simples de implementar mas atingem um menor desempenho em ambientes de navegação em tempo real. As segundas envolvem algoritmos mais complexos mas possibilitam a previsão das regiões a serem visitadas num momento próximo e, portanto, a realização de pré-processamento da visibilidade.

Os algoritmos baseados em regiões exploram tipicamente a característica da arquitectura de interiores ou ambientes similares e organizam-nas em células (salas), ligadas por portais (portas ou janelas) (Figura 8) [Pires01]. Estes métodos têm em consideração que outras células serão visíveis apenas através dos portais.

[1] Um *voxel* representa um valor numa grelha regular de um espaço tridimensional. O *voxel* é análogo ao *pixel* mas em 3D.

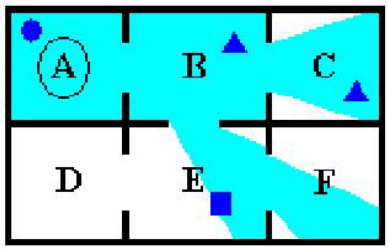


Figura 8 – Exemplo da técnica Portals and Cells

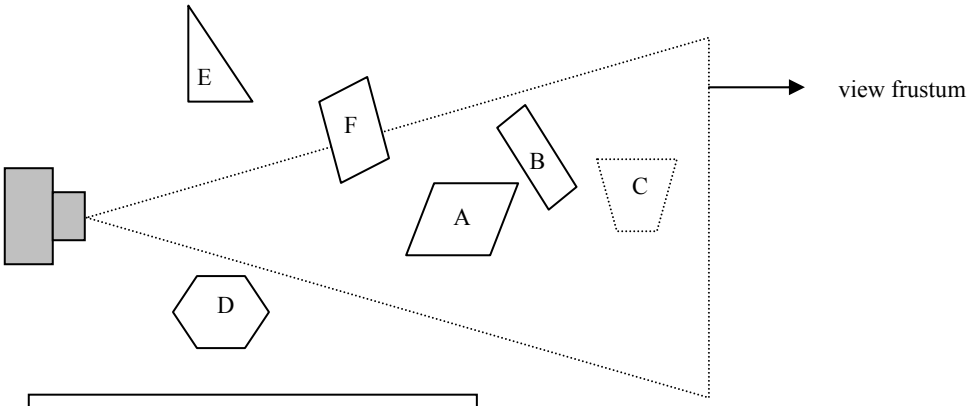
Algumas cenas 3D são "mapeáveis" apenas em duas dimensões ou duas dimensões e meia. Teller [Teller92] construiu uma solução analítica para o problema de visibilidade com uma fase de pré-processamento onde é feita a divisão da cena em células e a identificação dos portais. Este processamento inclui: *Cell-to-cell visibility* onde, para cada célula, se determina quais são as células visíveis através de rectas (*sightlines*); *Cell-to-object visibility* onde, se determinam rectas de cada uma das células para objectos das células visíveis correspondentes, isto é, objectos potencialmente visíveis (PVS); *Eye-to-cell visibility* onde, para a célula onde se encontra o observador, se elimina do conjunto das células visíveis, aquelas que estão fora do campo de visão. Esta técnica de Teller exige uma demorada etapa de pré-processamento e restringe-se a ambientes 2D ou 2.5D.

As técnicas anteriores de remoção de objectos por oclusão permitem em grande parte dos casos, reduzir substancialmente o número de polígonos a processar na etapa de cálculo de visibilidade executado pelo *hardware* gráfico. As técnicas de remoção por campo de visão, são complementares e verificam quais os objectos que se encontram fora do campo de visão do observador, permitindo assim a sua remoção. Uma descrição é efectuada na secção seguinte.

2.3.1.2. View-Frustum Culling

A técnica *view-frustum culling* ou remoção por campo de visão tem como objectivo eliminar rapidamente os objectos que se encontram fora do campo de visão do utilizador.

Sendo S o conjunto de todos os objectos existentes numa cena, este conjunto é dividido em três sub-conjuntos [Pires01]: um sub-conjunto I que contém todos os objectos que se encontram totalmente dentro do campo de visão (Figura 9), um sub-conjunto O que engloba todos os objectos fora do campo de visão e um último sub-conjunto B com todos os objectos que estão posicionados nos limites do campo de visão. O conjunto R de objectos resultantes é o que resulta da reunião dos conjuntos B e I, sendo os do conjunto O descartados, não seguindo para a etapa seguinte, de cálculo de visibilidade:



Legenda:

A – tem 4 faces onde se pode utilizar o *back face culling*.

B – tem 3 faces onde se pode utilizar o *back face culling*.

C – occlusion culling.

D, E – view frustum culling.

$$I = \{A, B, C\}$$
$$O = \{E, D\}$$
$$B = \{F\}$$
$$R = I \cup B = \{A, B, C, F\}$$

Figura 9 - Técnicas de remoção rápida de objectos: view-frustum culling, occlusion culling e back-face culling

Sendo o campo de visão limitado por uma área de seis planos (plano próximo, plano afastado e quatro lados [Hudson97]), o algoritmo verifica a intersecção de cada objecto com o campo de visão. Apenas os nós parcialmente ou completamente contidos no campo de visão são passados à fase seguinte. A vantagem de se utilizar esta técnica é que uma grande quantidade de polígonos pode ser excluída do processo de *rendering*, antes de serem enviados ao *hardware* gráfico [Pires01]. Se um objecto está totalmente incluído no *frustum*, ele é enviado para o *hardware*, mas se um objecto está parcialmente dentro do *frustum*, mais trabalho de decisão é necessário, podendo diversas estratégias serem aplicadas. Por exemplo, pode-se optar por enviar o objecto completo para o *hardware* ou dividi-lo e enviar apenas a parte que se encontra no interior do *frustum*. Os objectos que se encontram fora do *frustum* são eliminados imediatamente.

Visto que um ambiente virtual possui um grande número de objectos, o método descrito pode ser melhorado através do agrupamento em hierarquias de objectos (*bounding volume*) ou da divisão do espaço (*spacial partitioning*) (secção 2.3.1.4).

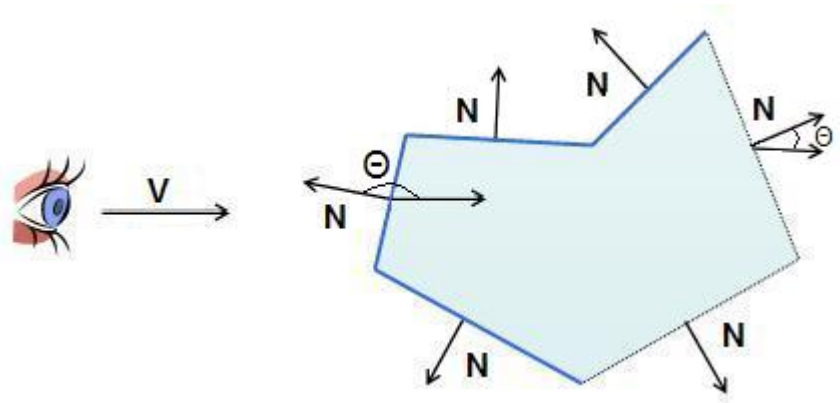
2.3.1.3. Back-Face Culling

O *back-face culling* ou remoção das "faces de trás", é processada ao nível do polígono. Baseia-se no facto de um polígono possuir duas faces, normalmente identificáveis pelo sentido da normal ao polígono. Atendendo à posição do observador, somente uma face

será visível por efeito de oclusão do objecto a que pertence, o que permite reduzir para cerca de metade a complexidade do cálculo de visibilidade.

Por norma, o *hardware* gráfico executa o *rendering* para ambas as faces, o que justifica a utilização deste método, dado que elimina os polígonos que não contribuem para a imagem final, por se encontrarem ocultos por outros do próprio objecto (Figura 10).

Um método vulgar de determinação, num polígono, da sua "face de trás" consiste na determinação do ângulo Θ entre a direcção de observação (V) e o vector normal de cada face (N) (Figura 10): considera-se "face de trás" aquela que apresenta um ângulo Θ inferior a Π/2, o que é avaliável pelo sinal do produto escalar dos dois vectores [Esperança06].



Se V · N = 0 → vectores perpendiculares, face não visível;
Se V · N > 0 → vectores com ângulo menor que 90º, face virada para trás;
Se V · N < 0 → vectores com ângulo maior que 90º, face virada para a frente.

Figura 10 – Cálculo da face de trás do polígono

A abordagem introduzida por Zhang [Zhang97] é uma aproximação eficiente e pouco pesada na remoção das faces ocultas, a qual requer pouco esforço de integração nos sistemas gráficos já existentes. O algoritmo, de uma forma geral, usa uma máscara normal (*normal mask*) para representar as normais do polígono e as normais do espaço, reduzindo a operação de remoção a uma operação lógica AND entre a máscara normal do polígono e a memória de faces ocultas.

2.3.1.4. Estruturação de Dados

De forma a permitir a maior eficácia dos métodos anteriores de remoção de objectos, torna-se necessária a utilização de uma estrutura de dados adequada, tipicamente baseada em volumes envolventes.

Um *bounding volume* (volume envolvente) é um elemento geométrico simples, que envolve um ou mais objectos geométricos mais complexos. Este tipo de hierarquia é utilizado como uma aproximação dos objectos complexos no intuito de reduzir o número

de operações a serem realizadas. Na Figura 11, extraído de [Fernandes06], apresentam-se alguns tipos de volumes envolventes que se descrevem seguidamente.

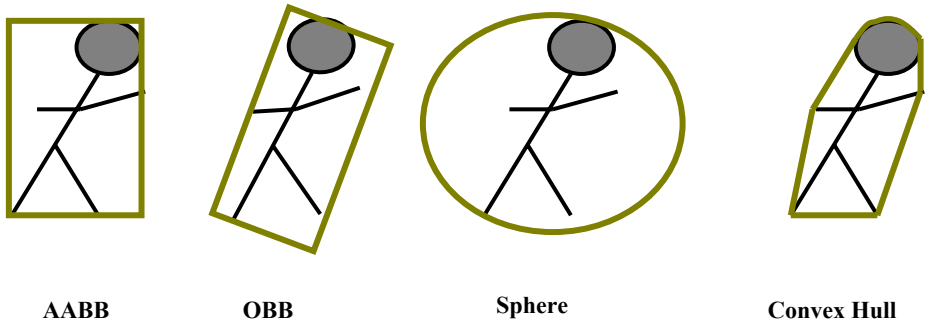


Figura 11 – Diferentes tipos de volumes envolventes

A caixa envolvente (*Bounding Box*) é um paralelepípedo cujas dimensões se ajustam às do objecto que contém e pode ser definida de duas formas: caixa envolvente orientada (OBB – *Oriented Bounding Box*), ou caixa alinhada com os eixos (AABB – *Axis Aligned Bounding Boxes*). Os testes de intersecção são mais simples de realizar com caixas AABB mas com a desvantagem de, para objectos que não se encontrem alinhados com os eixos, uma caixa possuir um volume superior ao do objecto respectivo.

A esfera envolvente (*Bounding Sphere*) é uma esfera hipotética que abrange completamente um objecto. É definida por uma coordenada 3D representando o centro da esfera e um valor escalar que define o raio máximo que envolve todos os pontos do objecto. A reduzida informação que a caracteriza relaciona-se com a maior simplicidade da sua utilização como volume envolvente do que as caixas anteriores, embora com a desvantagem de não ser o tipo de volume que limita o objecto de forma mais justa.

O *convex hull* é o mais pequeno poliedro que cerca um objecto, possuindo a vantagem de uma aproximação muito rigorosa aos limites do objecto. Em contrapartida, a sua implementação engloba cálculos muito complexos.

A Tabela 1 [Rodrigues05] resume as principais vantagens e desvantagens de cada um dos tipos de volumes envolventes referidos.

Tabela 1 – Vantagens e desvantagens dos vários volumes envolventes

	Vantagens	Desvantagens
AABB	Simples de manipular	Mais complexa que a BS; Menos rigorosa que OBB
OBB	Rigorosa	Complexa de manipular
BS	Muito simples de criar e manipular	Aproximação geralmente fraca
Convex Hull	Muito rigorosa	Muito complexa

O trabalho apresentado por Assorsson [Assarsson00] permite acelerar significativamente o cálculo de remoção por campo de visão através do uso de uma árvore hierárquica de volumes envolventes do tipo AABB e OBB. O algoritmo testa recursivamente o volume envolvente contra os seis planos que definem o *view-frustum*. O algoritmo destina-se a acelerar a técnica de remoção por campo de visão em animação, optimizando a relativa coerência da posição do objecto em imagens (*frames*) consecutivas.

[Slater97] introduz uma nova abordagem à técnica de remoção do campo de visão, com o objectivo de reduzir a quantidade de cálculos necessários à selecção dos objectos visíveis. Tendo em conta o princípio da coerência, objectos dentro e fora do campo de visão, ou que intersectam os limites do campo de visão, alteram-se lentamente durante o tempo. O algoritmo identifica os três conjuntos de objectos e divide os que se encontram fora dos limites do campo de visão em subconjuntos, provados probabilisticamente de acordo com a distância ao campo de visão. Com esta informação, acelera os cálculos de remoção do campo de visão nas imagens seguintes.

A hierarquia *spacial partitioning* tem como objectivo dividir o ambiente em diversas regiões definindo, cada uma, de acordo com o seu conteúdo. As estruturas mais usadas na divisão de espaço no campo de visualização interactiva são a *quadtrees/octrees*, *kd-trees* e várias *BSP trees*.

Uma *quadtree* é uma estrutura de dados, em duas dimensões, a qual divide o espaço em partes de dimensão igual. Sendo que cada quadrante pode ser dividido novamente em quatro subquadrantes e assim sucessivamente. Numa estrutura em *quadtree* existe um nó pai, em que os quatro quadrantes são representados por quatro nós filhos, numa ordem pré-definida. Nas *quadtrees*, todos os nós ou são nós folha ou têm quatro nós filhos (Figura 12) [Fernandes06].

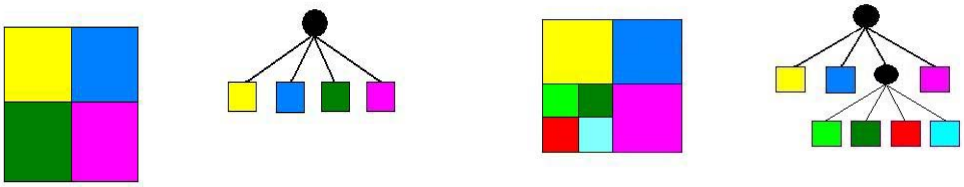
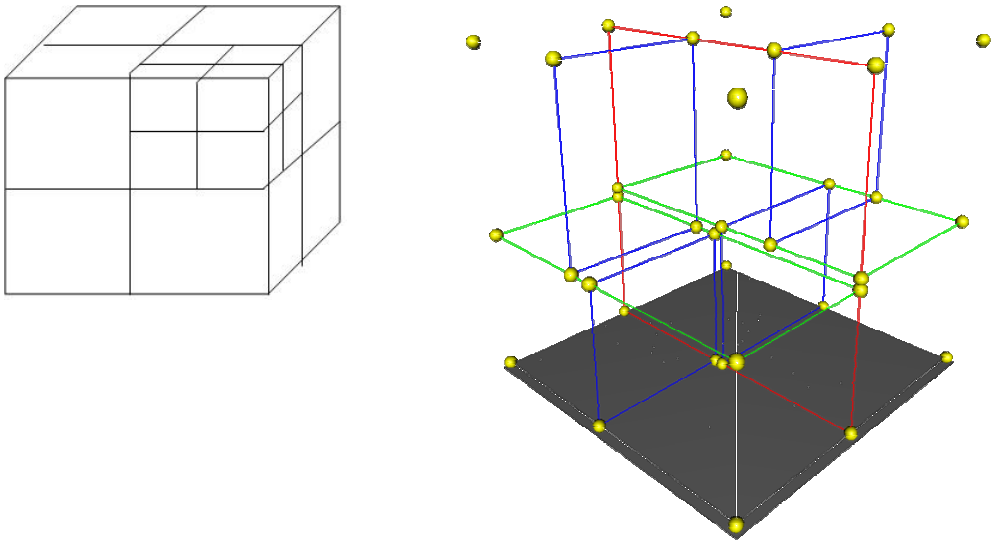


Figura 12 – Exemplo de uma estrutura *quadtree*

Octrees e *kd-trees* são estruturas de dados hierárquicas em forma de árvore para dividir o espaço 3D em células paralelepipédicas. O nó raiz corresponde ao espaço do modelo e é representado por uma caixa que envolve completamente o modelo. Cada nó interno na árvore representa uma subdivisão da sua região em regiões menores, às quais correspondem os nós filhos. As regiões dos nós filhos são células nas quais os objectos ou primitivas são mantidas.

A diferença entre uma *octree* e uma *kd-tree* é que na primeira cada nó interno divide a sua região ao longo dos três eixos, obtendo oito sub-regiões com a mesma dimensão (Figura 13a)) [Shen06]. A segunda estrutura (*kd-tree*) divide recursivamente o espaço em duas partes, não necessariamente iguais, por um plano ortogonal a um dos três eixos coordenados (Figura 13b)) [Wikipedia07].



a) **b)**

Figura 13 – Exemplo de uma estrutura *octree* (a) e *kd-tree* (b)

As *BSP-trees* (*Binary Space Partion Trees*) são estruturas hierárquicas para dividir o espaço recursivamente em células convexas. Na árvore, cada nó interno divide a região convexa em duas regiões através de um plano de orientação livre (Figura 14) [Muniz05].

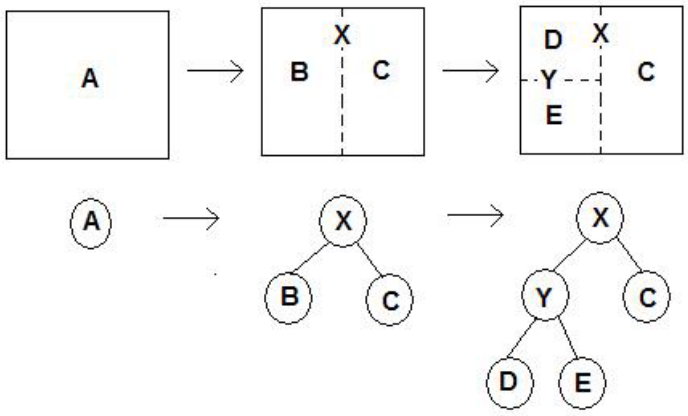


Figura 14 – Exemplo de uma *BSP-tree*

2.3.2 - Técnicas de Redução da Complexidade da Cena

A secção 2.3.1 descreve técnicas que resolvem o problema da redução do número de objectos a serem processados pelo *hardware* gráfico para efeitos de cálculo de visibilidade. Mas para cenas complexas em que o observador pode visualizar muitos objectos, estas técnicas podem também ser complementadas por técnicas que reduzem a complexidade com que os objectos visíveis são representados.

As técnicas de redução da complexidade geométrica dividem-se em duas categorias: técnicas de redução geométrica dos modelos, com o objectivo de reduzir a complexidade através do uso de modelos geométricos simplificados, e as técnicas de aceleração baseadas na imagem, as quais sintetizam imagens a partir de imagens, permitindo tempos de *rendering* independentes do número de objectos a descrever. Estas duas técnicas são descritas com mais pormenor nas secções seguintes.

2.3.2.1. Técnicas de Redução Geométrica do Modelo

Como referido anteriormente, a navegação em ambientes virtuais complexos implica, normalmente, ficheiros de grandes dimensões disponíveis num servidor, a partir do qual os diversos utilizadores têm acesso. Devido à limitada capacidade das redes, a transmissão da informação torna-se lenta, agravando-se o problema quando a informação necessita de ser transmitida continuamente devido, por exemplo, a ser gerada dinamicamente pelo servidor. Uma das formas de resolver este problema consiste na aplicação de técnicas de simplificação com o intuito de reduzir a geometria do modelo através da geração de uma ou várias versões simplificadas da geometria do modelo. Esta secção aborda a técnica de redução da geometria do modelo através da simplificação da malha poligonal e através de modelos de multi-resolução.

Uma malha poligonal é uma superfície tridimensional representada através de um conjunto de polígonos constituído por vértices, arestas e faces. O tipo de malha poligonal mais utilizada constitui-se de triângulos (Figura 15) [Fluschmann99], por ser suportado por diversos *softwares* de visualização e pelo *hardware gráfico*.

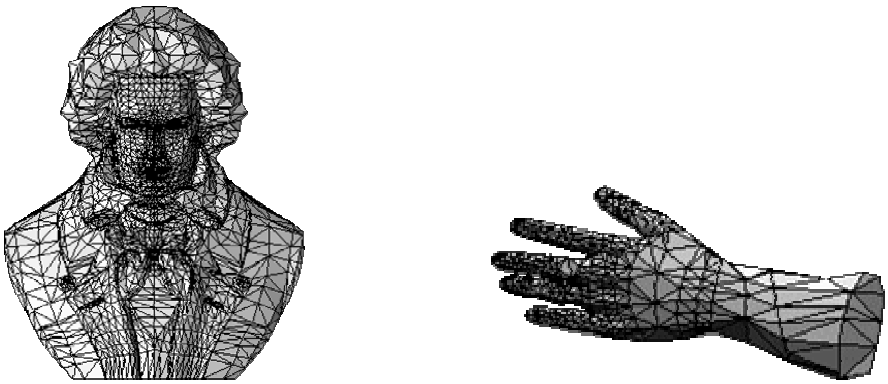


Figura 15 – Exemplos de malha poligonal

O objectivo do método de simplificação da malha poligonal é reduzir o número de vértices, arestas e triângulos da malha, de forma a diminuir a quantidade de informação necessária para representar a triangulação, tendo em conta que a malha simplificada deve aproximar-se da malha original.

De acordo com [Garland99], estes métodos são divididos nas seguintes categorias: *vertex clustering*, *vertex decimation* e *edge contraction*.

A categoria *vertex clustering* agrupa vértices em conjuntos e determina um único vértice representativo. Este método agrupa iterativamente conjuntos em conjuntos maiores, até que apenas um único vértice-conjunto permaneça.

O trabalho desenvolvido por Rossignac [Rossignac93] propõe uma técnica simples e robusta para reduzir a complexidade do modelo através do *vertex clustering*. O modelo é colocado dentro de uma grelha uniforme de células e os vértices de cada uma das células são agrupados para formar um único vértice. Apenas os triângulos cujos três vértices estão dentro de diferentes células continuam iguais no modelo simplificado.

A categoria *vertex decimation* é a mais utilizada das três enunciadas. Em cada etapa de simplificação, um vértice é seleccionado para ser eliminado com as respectivas faces. Este tipo de simplificação é utilizado no algoritmo proposto por Schroeder [Schroeder92], no qual são feitos vários testes sobre todos os vértices da malha poligonal. Durante cada teste, um vértice é um candidato a ser removido e, se se reunirem todos os critérios específicos da decimação, então o vértice e todos os triângulos associados são eliminados. O furo resultante na malha poligonal é remendado dando origem a uma triangulação. O processo da remoção de vértices repete-se, com um possível ajuste nos critérios de decimação até ser necessário terminar, geralmente devido a uma redução percentual da malha poligonal original ou devido a um valor máximo pré-definido atingido da decimação.

Para um controlo mais preciso durante o processo de simplificação, algumas abordagens optaram pelo uso de *edge contractions* [Andersson06]. Esta técnica une dois vértices que se encontram ligados removendo um limite do grafo. De uma forma geral, esta técnica remove duas faces triangulares por cada *edge contraction*.

Como foi referido anteriormente, outra das técnicas existentes que permitem a redução da geometria do modelo são as que utilizam modelos de multi-resolução. Os modelos de multi-resolução (*multi-resolution models*[2]) são uma forma de resolver os problemas de *rendering* de objectos dotados de uma geometria muito complexa. De uma forma geral, este tipo de técnicas centra-se na ideia de que, uma vez que os objectos distantes do

[2] Modelos que são representados e visualizados em diversas resoluções geométricas.

observador são visualizados numa pequena área do ecrã, estes objectos podem sofrer do processo de *rendering* a partir de malhas de pequena resolução sem que a sua aparência seja praticamente afectada. Assim, na técnica *Level-of-detail* (*LOD*) são usadas representações progressivamente mais simples da geometria de um modelo baseadas em parâmetros pré-definidos e dependentes, por exemplo, da distância em relação ao observador (Figura 16 e Figura 17) [Valente04] [Luebke05].

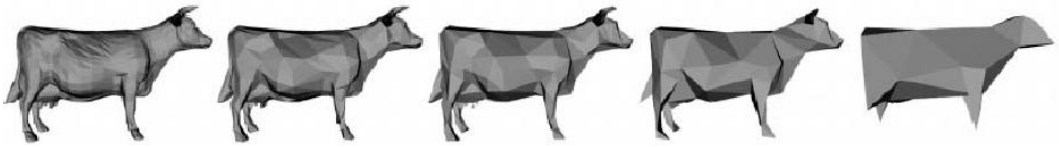


Figura 16 – Exemplo de LOD para a representação de uma vaca

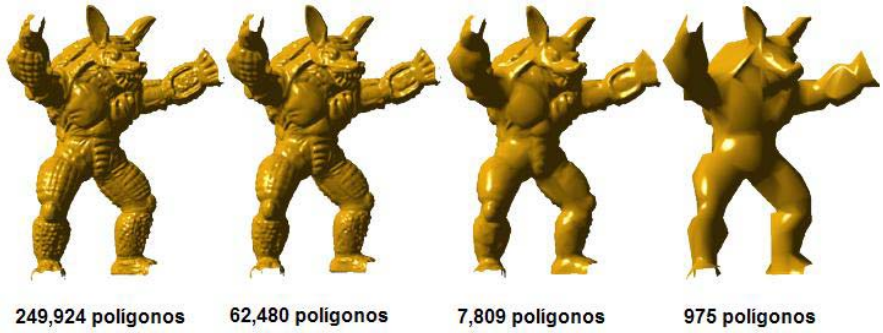


Figura 17 – Exemplo de LOD para representação de um dragão

Estas técnicas já são usadas desde o início dos simuladores de voo e foram introduzidas em ambientes do tipo *walkthrough* por Funkhouser [Funkhouser93]. O algoritmo proposto por Funkhouser determina uma representação hierárquica do modelo, na qual os objectos são descritos em múltiplos níveis de detalhe e podem ser representados por vários algoritmos de *rendering*. Desta forma, o algoritmo selecciona um nível de detalhe e o algoritmo de *rendering* para cada objecto potencialmente visível, com o objectivo de gerar a melhor imagem possível, ajustando a sua qualidade para manter uma *frame rate* uniforme. Esta técnica de aceleração requer o armazenamento das diversas resoluções do modelo, o qual se divide em duas categorias: *LOD* discreto e *LOD* contínuo. No *LOD* discreto é guardada a malha triangular completa para cada resolução possível do objecto representado. Desta forma, é necessário determinar quais as resoluções que podem ocorrer durante o processo de visualização e armazenar uma representação simplificada para cada uma delas. Este tipo de *LOD* obriga a ter disponível um grande espaço para armazenamento e não ser possível ter nenhum tipo de adaptação do modelo em função da posição do objecto relativamente ao observador. No *LOD* contínuo, a estrutura de armazenamento da malha triangular é mais elaborada, possibilitando a selecção de uma malha com nível de detalhe em função da resolução do objecto na cena. A resolução do

objecto a apresentar durante a visualização é processada no momento, obrigando a um maior esforço nesta etapa. Todavia, torna-se viável se forem utilizadas versões eficientes da malha com o detalhe requerido.

O trabalho desenvolvido por Maciel [Maciel95] descreve um sistema de navegação, o qual utiliza primitivas de texturas (*texture mapped primitives*) para representar conjuntos de objectos, de forma a manter a *frame rate* elevada e constante. Nos casos em que, no campo de visão, existem mais primitivas visíveis do que as que são possíveis de remover em tempo real, então o sistema assegura-se de que cada objecto visível, ou o conjunto em que estão incluídos, seja removido em cada *frame*. O algoritmo suporta as representações do tipo *levels-of-detail* para objectos individuais e a geração automática de alguns tipos de *levels-of-detail* para objectos e conjuntos de objectos. Suporta ainda o conceito de escolher uma representação do objecto de entre as associadas a este, em função da direcção em que está a ser visto. De uma forma geral, o sistema pode ser visto como uma generalização do conceito de *levels-of-detail*, onde a cena é armazenada numa hierarquia *levels-of-detail* em *top-down*. Cada *LOD* tem uma contribuição na qualidade da simulação, a qual pode ser estimada através de um benefício heurístico. A representação com maior detalhe tem um benefício mais elevado, independentemente do ângulo, sendo a que tem um custo superior no *rendering*. As representações com menor detalhe têm um benefício em função do ângulo de visão e da sua distância. O objectivo de incorporar informação da dependência da vista no benefício heurístico, tem como fim ser possível determinar a exactidão de cada uma das representações do objecto de acordo com as direcções de visão possíveis. O modelo hierárquico é uma estrutura em árvore, na qual os nós são meta-objectos que fornecem múltiplas representações do modelo, cada uma representando um determinado tempo de *rendering*. A raiz da árvore contém a representação mais simples do modelo e a de menor custo de *rendering* para o modelo completo, enquanto que as folhas da árvore contêm as representações mais perceptíveis do modelo e as de maior custo de *rendering*. A abordagem divide, ainda, as representações em *view-dependent* (dependentes do observador) e *view-independent* (independentes do observador), sendo a primeira considerada apenas para um subconjunto das direcções de visão possíveis, enquanto que as segundas *LOD* são consideradas para todos os ângulos de visão.

O projecto desenvolvido por Arnold [Arnold99] na cidade de Norwich consistiu na modelação em VRML (*Virtual Reality Modelling Language*) da área que envolvia o castelo de Norman, a catedral e outros edifícios considerados relevantes. Estes foram modelados utilizando *LOD*. Para os edifícios considerados com pouco interesse histórico, foi empregue uma prototipagem rápida, que consistiu num conjunto genérico e parametrizado de edifícios que puderam ser rapidamente definidos e construídos. O

projecto incorporou ainda *avatars*[3] com funções de se movimentarem pela cidade e a capacidade de interactuar uns com os outros.

A abordagem proposta por Chamberlain [Chamberlain96] descreve um método hierárquico que acelera o processo de *rendering* sem sacrificar muito a qualidade da imagem. Neste algoritmo a cena é dividida utilizando uma estrutura em *octree*, na qual, cada nó representa uma região da cena. Cada região da cena tem associada uma aproximação da sua aparência distante, podendo esta aproximação sofrer o processo de *rendering* em menos tempo do que a própria geometria da região. Quando o tamanho da região a ser visualizada é suficientemente pequena, isto é, a região encontra-se a uma grande distância do observador, então a aproximação é utilizada em substituição da geometria da respectiva região. Este método tem a vantagem de ser totalmente automático e de permitir desempenhos satisfatórios em cenas de exterior, nas quais uma grande quantidade de primitivas é parcialmente visível.

Constantinescu [Constantinescu01] apresenta no seu trabalho algumas técnicas de selecção de *LOD* como a distância, o tamanho e velocidade. A selecção pela distância é a mais simples, pois apenas necessita de definir um ponto ou vários pontos a partir dos quais os *LODs* correspondentes serão lidos. A selecção a partir do tamanho considera a incapacidade do olho humano para detectar detalhes de objectos pequenos. Desta forma, os objectos pequenos são representados com pouco detalhe, enquanto que os objectos de grandes dimensões apresentam uma maior riqueza. A velocidade baseia-se na capacidade limitada do olho humano em fixar os objectos que se movem. Assim, para estes últimos, o detalhe diminui com o aumento da velocidade.

2.3.2.2. Técnicas de Aceleração Baseadas na Imagem

Durante as últimas décadas, investigadores em computação gráfica desenvolveram diversas técnicas para tratamento e visualização da imagem, como a criação de imagens fotorealístas interactivas. Infelizmente, o objectivo de atingir simultâneamente interactividade e fotorealismo é complexo.

O processo de criar imagens fotorealistas, inicialmente, envolvia a simulação da propagação da luz pelo ambiente. Isto requeria a modelação da geometria do objecto no ambiente, as propriedades do material que constituía a geometria, a fonte de luz que emitia a energia para o ambiente, e as câmaras que captavam a luz reflectida. De seguida

[3] Representação ou aparência visual utilizada para interpretar o utilizador [TechTarget06].

era utilizado um algoritmo de *ray-tracing*[4] ou radiosidade[5] para calcular a distribuição da luz pela cena.

A síntese de imagens de uma cena que se assemelhe a uma fotografia real através deste processo é extremamente difícil, devido à complexidade da geometria do mundo real e à simulação da propagação da luz que leva imenso tempo a determinar. No entanto, uma vez que o objectivo é produzir imagens fotorealistas, o que interessa é a luz que chega ao olho do observador. Assim, no processo de *rendering* baseado na imagem, imagens "substituem" a geometria, sendo usadas como primitivas fundamentais, executando-se então o *rendering* através da reprojecção e interpolação.

Maciel [Maciel95], já referido a propósito de LODs na secção 2.3.2.1, foi dos primeiros a utilizar a imagem no contexto de aceleração do *rendering*, introduzindo o conceito de Impostores (*impostors*).

Impostores são superfícies texturadas, mais simples do que o objecto real e que contêm as características visuais mais importantes do mesmo. A utilização de impostores permite simular os pormenores do objecto real sem a necessidade de representar detalhes geométricos do mesmo.

Existem duas classes de impostores: dependentes e independentes do ponto de vista [Martin00]. A primeira classe corresponde a texturas inseridas nas faces dos objectos. Encontram-se neste caso os chamados *billboards*[6], que substituem a geometria complexa por geometria simples com mapeamento de textura, de modo a obter-se realismo e rapidez. A geometria simplificada é orientada em função da câmara. Impostores independentes do ponto de vista dizem respeito a versões simplificadas da malha poligonal original.

O algoritmo apresentado por Sillion [Sillion97] faz uma eficiente segmentação do modelo urbano baseado na informação da cidade resultando, para cada *frame*, dois subconjuntos distintos: a vizinhança local e o cenário distante. Para disponibilizar informação geométrica detalhada ao observador, a representação da vizinhança local é efectuada em três dimensões. O cenário distante, pelo contrário, é substituído por impostores, com a

---

[4] *Ray-tracing* é a técnica de *rendering* de imagens baseada no traçado de raios de luz. Consiste na análise do caminho percorrido por raios que vão desde a câmara até ao plano de fundo da imagem, para determinar se o raio intersecta algum objecto da cena. A cor do *pixel* a representar no ecrã será a cor do objecto atingido pelo raio que se encontra mais próximo [Silva].

[5] A radiosidade é um método de simular as diversas reflexões da luz na cena, resultando em sombras mais suaves e naturais [Velho00].

[6] Tipo de nó que modifica a sua rotação em volta de um eixo pré-definido de acordo com a posição do utilizador.

aplicação de informação tridimensional apropriada. Estes impostores são criados *off-line* como pré-processamento ou a pedido quando o observador entra numa nova área do modelo. Existe no modelo o dobro de impostores que entradas, visto que os impostores se encontram associados a limites de objectos. Dos testes elaborados, alguns erros foram encontrados, como é o caso do problema de ruptura[7].

Quando um algoritmo de visibilidade utiliza impostores existem várias limitações que nem sempre são colmatadas com sucesso. Os principais problemas que surgem são: deformação da imagem devido à representação do impostor, resolução incorrecta, representação incompleta da imagem e imagem partida[8].

O trabalho desenvolvido por Wimmer [Wimmer01] apresenta uma estrutura nova para codificar a aparência de um modelo geométrico visualizado numa dada região. Esta representação pode ser usada em aplicações interactivas ou de visualização em tempo real para substituir um modelo complexo por um impostor, mantendo uma qualidade elevada e com baixos custos de *rendering*. A abordagem cria uma representação espaço-objecto idêntica a um ponto nuvem (*point cloud*) ou a uma imagem com profundidade, com uma taxa de amostragem controlada, de forma a fornecer a densidade suficiente para todas as possíveis condições de visibilidade de um ponto de vista específico. A aparência de cada ponto é calculada para as diferentes localizações do campo de visão, utilizando a integração Monte Carlo, isto é, são disparados raios de uma região rectangular que está contida no triângulo formado pelas três câmaras pré-definidas como campo de visão. Cada ponto do raio é associado a um mapa de textura que codifica as contribuições da aparência para as diferentes localizações do campo de visão. Os impostores baseados no ponto fazem uso do *hardware* de *rendering* para transformar cada ponto com o respectivo mapa de textura associado em função da posição do observador.

O grande problema na utilização de impostores é a quantidade de memória necessária. Visto isto, Jeschke [Jeschke05] apresenta um algoritmo que coloca automaticamente os impostores na cena com uma qualidade desejável para a *frame rate* e para a imagem, sem a necessidade de requerer grande quantidade de memória nos impostores. De uma forma geral, o algoritmo, dada uma cena estática e o campo de visão, define para cada posição visível quais os objectos que serão visualizados como impostores. Inicialmente, são identificadas as áreas visíveis que não podem ser rapidamente processadas, seguindo-se a criação de um conjunto de impostores candidatos. Um impostor candidato é parte da cena, tendo em conta o campo de visão, de forma a este ter a qualidade desejada. De

[7] Problema relacionado com a utilização de impostores que acontece quando o utilizador se encontra perto do limite entre o modelo segmentado e o modelo representado por impostores.

[8] Impostores em pontos causam normalmente problemas de 'rachar', que acontecem quando nenhuma informação é apresentada para determinada direcção do ponto de vista do utilizador [Decoret99].

seguida, o algoritmo de optimização selecciona dos candidatos, os que serão impostores, em função da sua capacidade de *rendering* e memória necessária.

É de salientar que a simplificação de modelos nem sempre se deve a razões de desempenho. Devido à complexidade e precisão geométrica dos modelos, normalmente é necessário incorporar, a um modelo virtual, outros modelos de pequenos detalhes, como é o caso de texturas de melhor qualidade. Assim, problemas de *aliasing* e, consequentemente, modelos que têm de ser *resampled* (pré-filtrados) para minimizar o cintilar e o desaparecimento de objectos, surgem devido às limitações da resolução na tecnologia. Como forma de contornar este problema, Decoret [Decoret02] introduziu uma nova abordagem de simplificação baseada em *billboard clouds*, definida como um conjunto de texturas, polígonos parcialmente transparentes, com um tamanho, orientação e resolução da textura independentes. A *billboard clouds* é construída através da selecção de um conjunto de planos que capturam a geometria do modelo e projectam triângulos sobre estes planos de forma a calcular os mapas de textura e transparência associadas a cada plano do *billboard*. As texturas podem incluir mais do que uma informação simples da cor. Nos *billboard clouds* é executado o *rendering* para cada polígono individualmente, não sendo necessária a introdução de informação topológica, sendo esta a sua principal vantagem. A sua complexidade visual resulta principalmente das texturas.

2.4 - Técnicas Direccionadas para Ambientes Urbanos

Cada uma das técnicas descritas nas secções anteriores é, normalmente, adequada para um tipo de cena específico ou abrange diversos tipos de cenas. Algumas classificações possíveis são: cenas estáticas/dinâmicas e cenas interiores/exteriores.

No trabalho desenvolvido por Pires [Pires01] é apresentada uma taxonomia para as técnicas de aceleração da remoção de objectos e para as técnicas de aceleração da redução da complexidade com as respectivas áreas de aplicação.

Visto que esta dissertação se foca em ambientes urbanos, esta secção descreve mais detalhadamente algumas das técnicas desenvolvidas, tendo em conta este requisito e ainda outras técnicas não classificadas para este tipo de ambientes mas susceptíveis de se adaptarem.

Os meios urbanos apresentam-se como um verdadeiro desafio relativamente aos sistemas de visualização interactiva, devido à enorme complexidade de informação geométrica a representar, às limitações dos dispositivos móveis e, no caso de serviços em rede, à largura de banda das redes. Ao longo dos anos, algumas técnicas para a aceleração do *rendering* de cenas complexas em modo *walkthrough* têm sido desenvolvidas. Uma das técnicas é a utilização de modelos *levels-of-detail* na cena (secção 2.3.2.1), permitindo um número reduzido de cálculos em tempo real. Contudo, podem surgir problemas, como é o caso de métodos heurísticos simples que utilizam esta técnica e que, por serem

estáticos, não suportam uma grande quantidade de objectos variável no *pipeline* gráfico, originando um desempenho deficiente por parte do *rendering*. Embora este problema já tenha sofrido várias tentativas de resolução, como por exemplo, uma previsão selectiva, através da relação custo/benefício, de nível de detalhe para cada objecto, e o nível de percepção nos diferentes *levels-of-detail* do objecto ser excelente, a troca imediata de um *level-of-detail* para outro causa efeitos perceptíveis devido, principalmente, à mudança na definição geométrica do objecto [Funkhouser93].

Outra abordagem é a utilização de impostores, os quais permitem simular os pormenores do objecto real sem representar detalhes geométricos do mesmo (secção 2.3.2.2). Estes impostores infelizmente, são válidos apenas durante algumas *frames* e consequentemente têm de ser constantemente actualizados.

As técnicas mais vulgarmente utilizadas em sistemas em modo *walkthrough* são as técnicas de remoção rápida de objectos (secção 2.3.1).

2.4.1 - Técnicas de Remoção Orientadas para Ambientes Urbanos

As técnicas de aceleração baseadas na remoção, em que a divisão do espaço é feito em função de regiões, têm a vantagem de permitirem o cálculo da oclusão *offline*, comparativamente às técnicas em que a região de interesse se baseia em pontos. As técnicas de remoção com a região de interesse baseada em pontos são importantes para o caso do ambiente ser, por exemplo, uma floresta, na qual cada folha esconde muito pouco, mas todas as árvores juntas ocultam tudo que se encontra atrás.

Contudo, estas técnicas originam cálculos computacionais significativos durante a visualização e não podem ser facilmente adaptados para o uso de *pré-fetching* quando o modelo ocupa uma capacidade superior à da memória existente no computador. Desta forma, conclui-se que os algoritmos baseados em pontos não são adequados para ambientes em modo de *walkthrough*, visto que estes necessitam de transmitir alterações na visibilidade em cada *frame*, causando atrasos inaceitáveis na comunicação, independentemente da rapidez com que é feito o cálculo da oclusão.

Tendo por base este conceito, Koltun [Koltun01] introduz uma nova abordagem para o cálculo de oclusão baseado em regiões, eliminando a necessidade de pré-processamento e o armazenamento de grandes quantidades de informação, sem haver atraso na imagem. O algoritmo utiliza uma transformação geométrica, representando a visibilidade num espaço a duas dimensões, *dual ray space*. Enquanto que o observador percorre a região, o servidor calcula a informação visível para as regiões adjacentes. A rapidez do algoritmo permite calcular e transmitir a informação visível antes do observador chegar à região seguinte e a sua precisão garante que o PVS (*Potencially Visible Set*) é suficientemente pequeno para ser visualizado pelo cliente em tempo real. O algoritmo utiliza uma *kd-tree*,

padrão para a divisão da cena em regiões, na qual, cada nó está associado a uma AABB e com os objectos que estão contidos nessa caixa. Para uma determinada região, o algoritmo percorre hierarquicamente a estrutura em modo *top-down* verificando, para cada nó, se a respectiva caixa envolvente é visível. Quando um nó ocludido é encontrado termina a recursividade.

Nos métodos de oclusão, qualquer objecto pode ser usado como oclusor ou, em alternativa, pode ser seleccionado um conjunto de objectos que serão os oclusores. No primeiro caso há a vantagem de se maximizar a oclusão, embora grande parte dos métodos seleccionem heuristicamente quais os oclusores a utilizar. É o caso do algoritmo apresentado por Bittner [Bittner99]. Este permite a eficiente remoção das partes invisíveis de um modelo, utilizando para a representação da topologia do modelo uma hierarquia espacial em árvore. Esta árvore é construída por todos os polígonos oclusores para um determinado ponto de vista. A particularidade deste algoritmo é que requer que todos os oclusores sejam polígonos convexos, o que na prática nem sempre acontece.

Hudson [Hudson97] propõe um algoritmo que escolhe dinamicamente um conjunto de oclusores e calcula a respectiva sombra (*shadow frustum*) e todos os objectos que lhe estão contidos, de forma a removê-los. A abordagem requer que o modelo seja representado através de uma divisão espacial e numa hierarquia espacial. A primeira é construída em pré-processamento e utilizada para a selecção dos oclusores; a segunda hierarquia é utilizada para a remoção rápida da geometria dos oclusores. A fase de selecção dos oclusores decorre em pré-processamento, seguindo os seguintes princípios: ângulo contínuo, complexidade da profundidade e coerência. O ângulo contínuo visto de um objecto convexo é facilmente calculado e mede a fracção do campo visual que este ocupa. No principio da complexidade da profundidade, o algoritmo estima o valor de um oclusor através da amostragem, isto é, selecciona aleatoriamente alguns observadores para cada região e constrói a sombra a partir do observador e do potencial oclusor, determinando o número de objectos contidos na respectiva sombra. A média das várias amostras é uma estimativa directa do valor do oclusor. Para cada oclusor encontrado, é construída a respectiva sombra: seleccionando-se o observador como o cume da área visível, o respectivo "plano perto" é definido como o plano que passa pelo ponto mais afastado da silhueta do oclusor e nos pontos que se encontram na direcção deste último ponto com o observador. Cada lado da área visível é um plano que contem o observador e a borda da silhueta do objecto. Esta área visível contém a área que o oclusor oculta, na qual todos os objectos contidos estão oclusos, não sendo, portanto, efectuado o *rendering*. Em comparação com outros algoritmos de detecção de colisões entre polígonos convexos, a abordagem proposta é pelo menos duas vezes mais rápida e mais robusta.

Wonka [Wonka99] desenvolveu um trabalho interessante em que são calculadas as sombras projectadas por um objecto em função do campo de visão do utilizador. As sombras determinadas, designadas de sombras do oclusor (*occluder shadows*), definem

uma área (*shadow frustum*), na qual os objectos contidos são definidos como invisíveis. Inicialmente, o algoritmo constrói duas estruturas: uma grelha da cena e uma grelha de oclusão. Para construir a grelha da cena, todos os objectos são organizados em função de uma grelha regular que abrange todo o ambiente. De seguida, os objectos são anexados à célula da grelha que intersectam. A grelha de oclusão diz respeito aos objectos da cena que são potenciais oclusores, isto é, polígonos totalmente opacos e de grandes dimensões. Durante a execução, o algoritmo selecciona para cada *frame* um número de oclusores, para os quais se determina as respectivas sombras, sendo efectuado o *rendering* para um mapa de remoção (*cull map*). De seguida, o mapa de remoção é percorrido para serem seleccionados os objectos potencialmente visíveis, para posterior teste pelo algoritmo *Z-buffer*.

O trabalho desenvolvido por Kultun [Kultun00] ocupa-se do problema da identificação de oclusores eficientes, isto é, oclusores que efectivamente possuem um volume de sombra que abrange uma grande área. O algoritmo proposto mede o potencial contributo de oclusão de um objecto para ambientes baseados em regiões e em pontos. A sombra de um objecto é determinada em função dos limites dos seus lados relativamente ao observador (Figura 18). Para o cálculo da medida do potencial contributo do oclusor, o algoritmo coloca dois planos de projecção atrás do oclusor; os dois planos são paralelos entre si e distam de uma distância pequena, pré-definida. De seguida, é projectado o oclusor a partir do observador (intersecção da sombra do objecto com o plano de projecção) para ambos os planos de projecção, sendo posteriormente calculada a área projectada nos dois planos. A medida do potencial contributo do oclusor é dada pela divisão das duas áreas da projecção. Esta relação permite obter uma estimativa da taxa de crescimento da sombra, a qual reflecte o tamanho da sombra. O valor obtido é maior do que 1.0 para os oclusores cuja sombra cresce com a distância. Se o oclusor for ampliado, esta relação torna-se maior. No caso do oclusor possuir uma área maior do que a área onde se encontra o observador, quanto maior for a distância entre ambos, menor é a relação. Pelo contrário, quando o oclusor possui uma área menor que a área do observador, quanto maior for a distância entre ambos, maior é a relação, sucedendo o mesmo com a sombra do oclusor. Desta forma, a abordagem mostra que os métodos que utilizam esta técnica podem produzir resultados incorrectos quando a visibilidade é calculada em função de uma região.

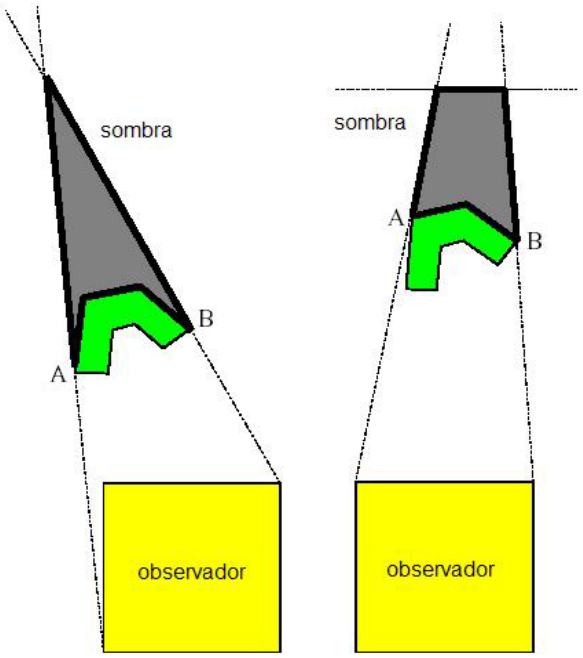


Figura 18 – Cálculo da sombra de um objecto relativamente ao observador

Durand [Durand00] introduz um algoritmo de pré-processamento da visibilidade que subdivide a cena em células visíveis (*viewing cells*), as quais representam as regiões por onde se movimenta o observador. Para cada célula, é calculado um conjunto de objectos que são potencialmente visíveis para todos os pontos dentro dessa célula (PVS). De modo a calcular eficientemente o PVS, é introduzido o conceito de operador de projecção estendida (*extended projection*), definida a partir de uma sub-estimativa das projecções do oclusor e uma sobre-estimativa para os ocludidos, relativamente a cada observador de uma célula. A projecção estendida de um oclusor é a intersecção de todas as vistas no plano projector, enquanto que a projecção estendida do triângulo ocludido é a união das vistas (Figura 19). O algoritmo, do tipo *occlussion culling*, está dividido em duas partes: uma etapa de pré-processamento e outra, de visualização interactiva. A etapa de pré-processamento cria as células visíveis e o PVS relativo à cena original. As células visíveis são organizadas numa estrutura espacial hierárquica, enquanto que a geometria da cena se encontra organizada em uma outra estrutura espacial. Esta fase inclui ainda o cálculo de oclusão para cada célula visível, na qual, são, inicialmente, seleccionados os oclusores, os planos projectores e a projecção estendida do oclusor, de forma a construir o mapa hierárquico em profundidade (*Hierarchical Depth Map*). Por fim, os oclusores são testados recursivamente e identificados como ocultos ou visíveis, sendo estes últimos inseridos no PVS da respectiva célula. Na etapa de visualização interactiva, sempre que o observador entra numa nova célula, o estado de visibilidade de um nó é actualizado. A abordagem processa objectos dinâmicos com oclusores estáticos mas, contudo, oclusores dinâmicos não são permitidos. As vantagens da fase de pré-processamento são a

possibilidade do algoritmo ser utilizado em cenas que ocupam maior capacidade do que a memória disponível e facilidade de transmissão a partir da rede.

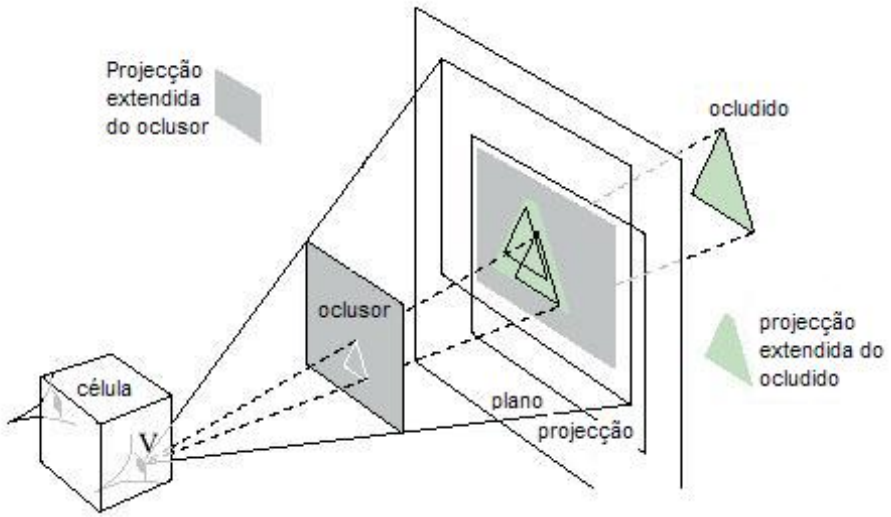


Figura 19 – Projecção estendida do oclusor e do ocludido

Uma abordagem que utiliza diversas técnicas, normalmente orientadas para a aceleração da interactividade de cenas complexas 3D em modo *walkthrough*, é a apresentada por Teler [Teler01]. A descrição da cena completa é inicialmente armazenada numa base de dados no servidor, o qual recebe periodicamente parâmetros do campo de visão do cliente, assim como a velocidade e a aceleração. O servidor determina previsões de movimento e decide quais as representações dos objectos a serem transmitidas ao cliente. Este, por sua vez, armazena na base de dados local, todas as representações dos objectos que recebe. O objectivo do servidor seleccionar as partes da cena a enviar é permitir que as *frames* obtidas após o processo de *rendering* pelo cliente sejam idênticas às *frames* que seriam obtidas após o processo de *rendering* se o modelo inteiro tivesse sido disponibilizado. Esta selecção é feita por um algoritmo *on-line* optimizado. A descrição da cena consiste numa colecção de objectos, sendo cada um destes, constituído por um conjunto de representações, as quais podem incluir o modelo geométrico completo do objecto, diversos *levels-of-detail* ou representações progressivas em malha, assim como impostores pré-calculados ou gerados dinamicamente. Cada representação do objecto tem associado um custo, representativo da sua previsão de duração na transmissão, e um benefício em função do observador, referente ao contributo do objecto para a qualidade da visibilidade da cena. Os parâmetros do custo e do benefício da representação de um objecto são analisados pelo servidor sempre que o servidor determina um possível caminho para o cliente. A previsão do caminho a efectuar pelo cliente pressupõe que, iniciado um tipo de movimento (parado num sítio, virar ou mover em linha recta), este continuará num futuro próximo. Como resultado final, conclui-se que o algoritmo ignora totalmente o *frame rate*, centrando-se em maximizar a qualidade ao longo do tempo,

dependendo da largura de rede disponível. Todavia, a navegação pela cena é sempre possível mesmo com uma largura de rede fraca.

Os ambientes interiores utilizam de uma forma eficiente as técnicas de células e portais (secção 2.3.1.2). Estas têm efectivamente sido pouco exploradas em ambientes exteriores urbanos, nomeadamente devido à decomposição do ambiente em células conectadas por portais, possibilitando assim, uma eficiente e rápida eliminação em massa de regiões da geometria do mundo do conjunto visível. A vantagem na utilização desta técnica é que qualquer região que não pode ser vista através do conjunto de portais que pertencem a uma região, então essa região não é considerada para o *rendering*. Da mesma forma, uma região contígua a outra não é considerada para o *rendering* caso entre elas não exista um portal. Consequentemente o campo de visão, neste tipo de técnicas, limita-se a uma pequena área comparativamente ao mundo completo pelo facto de a visibilidade só ser possível através dos portais.

Luebke e Georges [Luebke95] inovaram o algoritmo proposto por Teller [Teller92] tornando-o mais rápido para o cálculo de oclusão com portais. O algoritmo faz uso da subdivisão do ambiente em células e portais, definindo as primeiras como volumes poligonais do espaço e os portais como sendo as regiões 2D transparentes colocadas nos limites das células que se conectam a células adjacentes. Depois de dividido o ambiente em células e portais, é possível determinar quais as células visíveis para o observador ao atravessar apenas as células do conjunto potencialmente visível (PVS). O algoritmo utiliza caixas envolventes e inicia-se com a projecção dos vértices de cada portal para o espaço e determina a caixa envolvente 2D relativa aos eixos, designada de *cull box*, resultante dos pontos encontrados. Esta caixa representa os limites conservativos para o portal, isto é, os objectos cujas projecções no espaço ficam totalmente fora da *cull box*, são definidos como invisíveis e, desta forma, podem ser removidos. Se a caixa envolvente projectada intersecta a *cull box* associada, o objecto é definido como potencialmente visível através do portal, sendo posteriormente executado o *rendering*. Visto que um objecto pode ser visível através de diversos portais, este é rotulado quando é executado o *rendering*, permitindo assim, que os objectos sejam processados apenas uma vez por cada *frame*. À medida que os sucessivos portais são analisados, a respectiva caixa é intersectada pela *cull box* associada, com apenas algumas comparações. Com base no conceito dos espelhos serem uma extensão dos portais, o PVS é automaticamente restringido pelo espelho, envolvendo grandes dificuldades durante o *rendering*, como é o caso, por exemplo, da geometria que se encontra à frente do espelho não poder aparecer reflectida no mesmo. O algoritmo permite apenas o conceito de espelhos estáticos, estando previsto para trabalho futuro a utilização de espelhos dinâmicos.

Mais recentemente Jiménez [Jiménez00] propôs um algoritmo mais simples do que o desenvolvido por Teller [Teller92], para computar recursivamente a informação visível de toda a cena através de portais, determinando simultaneamente as aproximações

conservativas de visibilidade *cell-to-cell* e *cell-to-region* para cada uma das células. Este algoritmo utiliza uma adaptação do *Visibility Skeleton*, ferramenta poderosa que permite resolver diferentes problemas que requerem informação global de visibilidade, de forma a resolver conservativamente o problema de cálculo da *antipenumbra*. Isto é, para uma sequência de portais, a *antipenumbra* caracteriza-se pelo volume iluminado por uma fonte de luz até à célula atingida a partir do último portal da sequência. O cálculo deste volume para a sequência de um único portal é simples: a *antipenumbra* é o espaço 2D para além do portal, intersectado com as células atingidas, sendo estas últimas as células imediatamente vizinhas. O cálculo da *antipenumbra* de uma sequência de portais de tamanho dois ou superior, envolve interacções entre vértices e limites dos diferentes portais, requerendo ainda primitivas lineares e quadráticas, para descrever correctamente o volume iluminado.

Com o intuito de acelerar o cálculo de remoção de objectos e aproveitando a característica de algumas cenas relativamente à densa oclusão e à coerência da visibilidade, Haumont [Haumont03] propõe reformular o problema de CPG (*cell-and-portal graph* – grafo célula-portal) em segmentação de imagens 3D. Um CPG é um grafo que codifica a estrutura da visibilidade de uma cena, no qual os nós são células, correspondentes a salas de um edifício, ligados por portais, que correspondem a aberturas como portas, ou janelas. Uma célula só pode ver outras, através dos portais. O algoritmo encontra-se organizado em diversas etapas: primeiro, o campo distância da representação da cena é calculado, sendo este definido por um valor escalar em que cada ponto contém a distância da geometria da cena mais próxima. De seguida, os valores determinados são introduzidos numa grelha, podendo assim o mapa das distâncias ser utilizado no processo de *watershed*[9]. Quando necessário, os portais são construídos com base na geometria da cena. O CPG é calculado enquanto que decorre o processo *watershed*. Por fim, o algoritmo atribui a geometria da cena a diferentes células. Os resultados mostram que a abordagem determina uma decomposição muito semelhante à estrutura do modelo, não impondo nenhuma condição particular na modelação.

Outra abordagem orientada para a técnica das células e portais é a proposta por Lefebvre [Lefebvre03], que calcula automaticamente uma decomposição dos polígonos da cena num grafo célula-portal (CPG), de forma a reduzir o tempo de pré-processamento e como alternativa ao cálculo de um PVS (*potencially visible set*). O método define o ambiente 3D como una cena à prova de água (*watertight*), com um interior bem definido, isto é,

[9] A técnica *watersheds* é uma técnica de segmentação de imagens baseada no principio de "inundações de relevos topográficos", sendo que a imagem em níveis de cinza representa um relevo topográfico que vai inundando com água, formando-se, assim, bacias de captação e de linhas *watersheds* ou linhas divisoras de água [Peccini03]. Esta abordagem pode ser implementada em imagens 3D mas com custos elevados em cálculos e em memória.

ocupado por materiais reais. Esta cena é decomposta numa árvore BSP (*Binary Space Partion*), em que as *bsp-cells* são as folhas da árvore BSP e as *bsp-portals* são as faces transparentes de cada *bsp-celll*, e inicia-se com a procura de "bons" portais, seguindo-se a construção das células através dos componentes comuns do grafo das células menores. No passo seguinte, constroem-se os separadores válidos (polígonos grandes e transparentes feitos de pequenos *bsp-portals*) na cena, à parte do conjunto de *bsp-portals* da árvore BSP. Intuitivamente, os separadores adicionados recentemente devem fechar hermeticamente um caminho na cena, garantindo que nada é visível para além do separador, se este estiver escondido. Estes separadores definem o grafo célula-portal: as células correspondem aos volumes incluídos entre os separadores e os polígonos da cena. Os separadores são criados a partir dos limites da subdivisão inicial da cena e as células são construídas através do conjunto de ligações da *bsp-cells*, isto é, duas *bsp-cells* estão ligadas se se encontram conectadas na BSP e a *bsp-portal* que as separa não faz parte de um separador válido, obtendo-se assim, a CPG com separadores válidos. O último passo do algoritmo consiste em processar as células para restringir ao máximo os custos de *rendering*. Deste modo, se um polígono da cena não gera bons portais, este passo permite adicionar novos separadores que não têm obrigatoriamente de estar alinhados com a geometria. Por fim, é aplicado o algoritmo de simplificação à CPG de forma a manter apenas os portais que são relevantes. Os resultados apresentados mostram que os portais criados pelo método possuem boas propriedades geométricas, isto é, frequentemente coincidem com portas ou janelas, e a decomposição gerada pode ser usada *on-line* para *occlusion culling*.

2.5 - Síntese do Capítulo

O processo de criação de uma imagem a partir de uma cena tridimensional denomina-se de *rendering*, que envolve o cálculo de visibilidade. O algoritmo de cálculo de visibilidade mais simples e mais utilizado é o Z-buffer que, apesar de grande rapidez, decorrente da implementação em *hardware*, não consegue processar cenas muito complexas a um *frame rate* adequado.

A necessidade crescente de representar ambientes com uma qualidade tendencionalmente imperceptível do real, assim como de permitir ao utilizador interagir directamente com ele, tem fomentado o aparecimento de novas técnicas, de cálculo de visibilidade, ou suas complementares.

Existem várias técnicas de aceleração do cálculo de visibilidade, sendo as mais utilizadas as técnicas de remoção rápida de objectos (*Culling*) e as técnicas de redução da complexidade geométrica.

As técnicas de remoção rápida de objectos reduzem o número de polígonos a serem enviados para o *hardware* gráfico, baseando-se no princípio de que os objectos não

visíveis não precisam de ser representados. Nesta categoria de algoritmos, podem encontrar-se as técnicas de *occlusion culling*, a *view-frustum culling* e *back-face culling*.

O modo mais óbvio de remover os objectos é através da utilização da técnica *view-frustum culling*, a qual se fundamenta no facto de que apenas os objectos que se encontram na área visível do observador devem ser representados.

A forma primitiva de remoção baseia-se na observação de que, se os objectos são fechados, então os polígonos que não estão virados para o observador, não são visíveis. O *back-face culling* baseia-se nesta assumpção e permite a remoção de quase metade dos polígonos localizados dentro da área visível do observador.

Como a técnica *view-frustum culling* e *back-face culling* ainda não são suficientes na redução do número de polígonos, a técnica *occlusion culling* tem como objectivo identificar as partes visíveis da cena.

As técnicas de redução da complexidade geométrica dividem-se em técnicas de redução geométrica do modelo e em técnicas de aceleração baseadas na imagem. A primeira categoria tem como objectivo a redução da complexidade através do uso de modelos geométricos simplificados, como é o caso da simplificação da malha poligonal ou a aplicação de *levels-of-detail*. A segunda categoria utiliza imagens que permitem tempos de *rendering* independentes do número de objectos a descrever na cena, designadas por impostores.

Nenhuma técnica pode ser definida genericamente como a melhor, visto cada uma delas ter as suas próprias características e função específica. Os diversos projectos que têm vindo a ser desenvolvidos aplicam uma determinada técnica ou conjugam diferentes técnicas, permitindo deste modo aproveitar as melhores características de cada uma delas.

Uma área da computação gráfica com um forte desenvolvimento actualmente é a dos ambientes virtuais urbanos do tipo *walkthrough*. Para estes ambientes, têm sido propostas várias técnicas de aceleração, devido à enorme quantidade de objectos a representar. As propostas baseiam-se, normalmente, numa das anteriores categorias de técnicas apresentadas mas seguindo, praticamente, todas na mesma direcção, sem se questionarem sobre uma possível utilização de uma técnica orientada especificamente para ambientes do tipo *walkthrough*. Possivelmente seria interessante combinar a técnica *occlusion culling* com volume de sombra com uma técnica classificada para outro tipo de ambiente, como é o caso da técnica células e portais.

CAPÍTULO 3

MODELAÇÃO 3D PARA A WEB

Índice do Capítulo

3 - Modelação 3D para a Web

Com o intuito de desenvolver um ambiente que permitisse a criação de um conjunto diversificado de aplicações 3D distribuídas e interactivas e que possibilitasse a construção de uma nova *interface* 3D, surgiu o *Open Inventor*. Este serviu de base à especificação VRML 1.0 (*Virtual Reality Markup Language*), cujo principal intuito foi o de conseguir um formato de intercâmbio de ficheiros 3D na World Wide Web. Após a última actualização, na versão VRML97, e devido à persistência de algumas limitações, o Consórcio Web 3D desenvolveu uma nova especificação designada como X3D (*Extensible 3D*), que possui uma representação baseada em XML (*Extensible Markup Lamguage*).

Os avanços progressivos da Computação Gráfica têm vindo a criar expectativas para a simulação e modelação de ambientes virtuais. O aumento do uso das tecnologias 3D tem encorajado o desenvolvimento de ferramentas com potencialidades para visualizar rapidamente os recursos 3D. As actualizações às especificações 3D trouxeram as novas tecnologias X3D e SAI em substituição das anteriores VRML e EAI. De uma forma geral, VRML é uma linguagem *standard* para representar informação interactiva 3D para a Internet. Para que o utilizador possa interagir com o mundo VRML, a aplicação tem de implementar a tecnologia EAI (*External Authoring Interface* a qual, através da sua *interface*, possibilita a comunicação com a cena VRML a partir de uma *applet*[10] Java embebida numa página HTML (*HyperText Markup Language*). O X3D define-se como um novo formato *standard* para visualização de conteúdos 3D, que veio substituir o VRML e, consequentemente, a EAI foi substituída pela SAI (*Scene Access Interface*), a

[10] Pequeno aplicativo escrito em Java, em que o seu *bytecode* é interpretado através a Java Virtual Machine na máquina do cliente ou embutida no próprio *browser* do cliente.

qual permite fazer a comunicação entre o X3D e o Java, através das funções definidas por esta API (*Application Programming Interface*).

Neste capítulo são descritas, de uma forma geral, as linguagens de anotação para realidade virtual na Internet VRML e X3D, especificando, para cada uma, a sua estrutura e os elementos que compõe uma cena. Visto que cada uma das linguagens tem associados meios que possibilitam a interacção entre o respectivo ambiente e o utilizador, a secção 3.2 expõe sucintamente as respectivas *interfaces*, a EAI e a SAI, respectivamente para VRML e X3D.

3.1 - Linguagens de Anotação para VR na Internet

3.1.1 - VRML

A linguagem VRML (*Virtual Reality Modeling Language*) [Carey97] é um padrão ISO (*International Organization for Standardization*) para a representação de informação tridimensional na Internet, permitindo a construção de uma gama variada de aplicações em áreas desde a engenharia até à educação. A primeira versão (VRML 1.0) surgiu em 1995 mas possuía limitações de interactividade entre os utilizadores e as aplicações virtuais.

Entretanto, esta linguagem sofreu contínuas actualizações enquadradas em sucessivos *standards* e especificações, até Dezembro de 1997, em que foi publicado o VRML97 como padrão internacional ISO/IEC 14772. As principais características desta especificação é a sua capacidade de criar objectos 3D estáticos e animados, com ligações para elementos multimédia como sons, textos, imagens e filmes, a capacidade de encapsular novos recursos de forma a criar novos nós, interacção directa com o utilizador através de sensores e interpoladores e a criação de animações usando *scripts*.

Os ficheiros *wrl*, em linguagem VRML, são constituídos por texto puro, pelo que podem ser editados com editores simples. No entanto, dada a relativa complexidade dos objectos a modelar é de supor a utilização de editores gráficos de VRML como o *VRMLPad* [Parallel], o *ISB* (*ParallelGraphics Internet Scene Builder*) [Parallel], o *CosmoWorls* [SGI] ou o *VRCreator* [VRCreator]. A visualização e interacção com os ficheiros *wrl* é feita, usualmente, através de um *plugin* instalado num programa de navegação Web, capaz de traduzir a descrição da cena e dos objectos para uma forma gráfica 3D. Alguns dos *plugins* existentes são o *Cosmo Player* [SGI], o *Cortona* [Parallel] e o *Contact3D* [Bitmanagement].

A construção de um ficheiro VRML implica a utilização de um conjunto de componentes e regras de estruturação de um arquivo. Os principais componentes do VRML são a estrutura gráfica da cena, a arquitectura de eventos, sensores, *scripts* e interpoladores, o encapsulamento e a reutilização, bem como as facilidades de distribuição de cenas.

3.1.1.1. Componentes do VRML

Estrutura gráfica da cena

Um ambiente modelado em VRML é uma colecção de objectos 3D e mundos organizados numa estrutura hierárquica (*scene graph*), contendo entidades chamadas de nós (*nodes*). Estes armazenam a informação em campos podendo conter outros nós "filhos". Alguns nós podem ter mais do que um "pai", não podendo conter-se a si mesmo. Ao nó que não está contido em nenhum outro nó, dá-se o nome de nó raiz (*root node*). O VRML permite que o utilizador crie identificadores para qualquer nó, possibilitando a posterior referência a uma instância desse nó.

A Tabela 2 [Pollo97] representa os diversos tipos de nós existentes em VRML agrupados por tipo de funcionalidade em que se inserem.

Tabela 2 – Tipos de nós

Geometry & Appearances	Scene Environment Components	Grouping	Behaviors	Miscellaneous Nodes and Statements
Appearance Box Color Cone Coordinate Cylinder ElevationGrid Extrusion FontSyle IndexedFaceSet IndexedLineSet ImageTexture Material MovieTexture Normal PixelTexture PointSet Shape Sphere Text TextureCoordinate TextureTransform	AudioClip Background DirectionalLight Fog NavigationInfo PointLight Sound SpotLight Viewpoint	Anchor Billboard Collision Group Inline LOD Switch Transform	ColorInterpolator CoordinateInterpolator CylinderSensor NormalInterpolator OrientationInterpolato r PlaneSensor PositionInterpolator ProximitySensor ScalarInterpolator Script SphereSensor TimeSensor TouchSensor VisibilitySensor	DEF EXTERNPROTO PROTO ROUTE USE WorldInfo

Arquitectura de eventos

Os nós comunicam entre si através de eventos (*event*). Para o efeito, cada tipo de nó define o nome e o tipo de eventos que pode gerar (*EventOut*) ou receber (*EventIn*), como se pode ver na Figura 20.

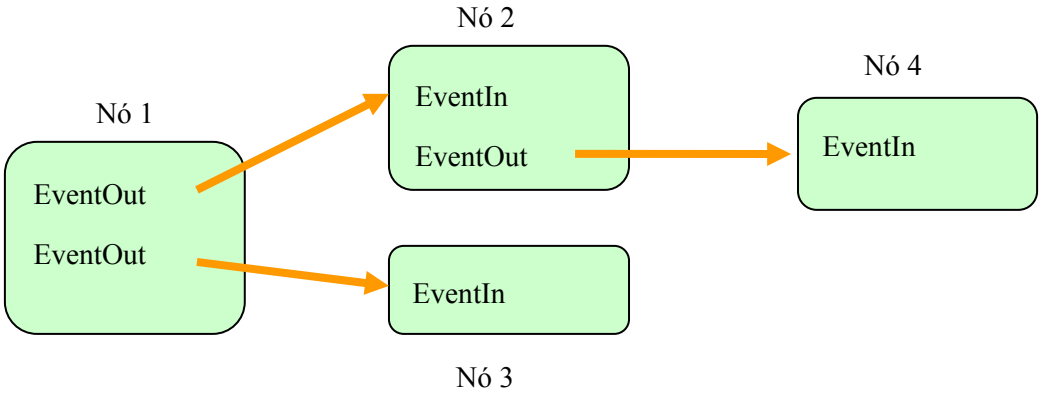


Figura 20 – Eventos entre vários nós

Sensores

Os sensores são as primitivas básicas de animação e interacção com o utilizador, através da geração de eventos. Isto é, quando um sensor é accionado por uma acção do utilizador ou devido à passagem do tempo, gera-se um evento, o qual pode ser redireccionado para um evento de entrada de outro nó através de uma rota (secção 3.1.1.2). No caso de a rota terminar num nó comum, o evento modifica o valor de um campo. Caso a rota termine num nó *Script* ou um nó prototipado, então é possível realizar qualquer tipo de processamento. O sensor *ProximitySensor* pode ser usado, por exemplo, quando o utilizador se aproxima de uma porta, de forma a gerar uma animação que a abra imediatamente. No caso de um sensor de visibilidade (*VisibilitySensor*), quando um determinado objecto se torna visível para o utilizador, ele executa uma determinada acção, podendo ser usado, por exemplo, para accionar uma fonte de água num parque natural apenas quando esta se torna visível ao utilizador, de forma a diminuir o esforço do *browser*.

A Tabela 3 [Miranda99] descreve os diversos tipos de sensores existentes em VRML e respectiva função.

Tabela 3 – Sensores e respectiva função

Sensor	Função
CylinderSensor	Desloca o objecto seleccionado pelo utilizador de forma a executar um movimento cilíndrico do eixo Y do sistema de coordenadas do sensor.
PlaneSensor	Desloca o objecto seleccionado pelo utilizador de forma a executar um movimento de translação no plano XY de coordenadas do sensor.
ProximitySensor	Gera eventos quando o observador entra ou sai da região do sensor.
SphereSensor	Desloca o objecto seleccionado pelo utilizador de forma a executar um movimento esférico relativamente à origem do sistema de coordenadas do sensor.
TimeSensor	Gera eventos de tempo (segundos) enquanto o tempo decorre.
TouchSensor	Gera um evento quando detecta que é activado um objecto do grupo do nó pai.
VisibilitySensor	Detecta mudanças na visibilidade de uma caixa rectangular virtual do ponto de vista do observador

Scripts e interpoladores

Os *Scripts* são pequenos trechos de programa escritos em Java ou JavaScript que podem ser inseridos entre geradores (sensores) e receptores de eventos, permitindo definir comportamentos arbitrários. Estes nós recebem um evento, o qual significa uma alteração ou uma acção do utilizador, originando a execução do *script* associado que, eventualmente, efectua alterações na cena enviando eventos de saída, comunicando com outros servidores da Internet, etc. Cada evento recebido é declarado no nó *Script* como um evento de entrada: `eventIn`. Da mesma forma, cada evento enviado tem de ser declarado como um evento de saída: `eventOut`.

A existência de *scripts* associados a vários nós permite controlar toda a interacção e o comportamento dos elementos do mundo virtual.

Os interpoladores são *scripts* embutidos que possibilitam a execução de animações[11], podendo ser combinados com vários nós da cena. São utilizados, normalmente, para executar animações, através de uma lista de pontos chave (*keyframe*) e de tempos, isto é,

[11] Animações mais complexas podem ser construídas utilizando nós do tipo *Script* e técnicas mais sofisticadas do que a interpolação linear.

uma função linear de *n* valores de tempo *t* (*key*), e de valores *keyvalue* os quais dizem respeito a *n* valores correspondentes a *f(t)*.

A Tabela 4 expõe os diferentes tipos de interpoladores e respectiva função.

Tabela 4 – Interpoladores e respectiva função

Interpolador	Função
ColorInterpolator	Interpola linearmente um conjunto de valores de cores RGB.
CoordinateInterpolator	Interpola linearmente um conjunto de coordenadas.
NormalInterpolator	Interpola linearmente um conjunto de vectores normais.
OrientationInterpolator	Interpola linearmente um conjunto de valores de rotação.
PositionInterpolator	Interpola linearmente entre um conjunto de valores de translação.
ScalarInterpolator	Interpola linearmente um conjunto de valores escalares.

Encapsulamento e reutilização

O VRML suporta a prototipagem para o encapsulamento e reutilização da geometria, propriedades, animações ou comportamentos. A prototipagem permite a definição de um novo tipo de nó a partir da combinação de tipos de nós já existentes, sendo muito semelhante às definições de classes em programação orientada a objectos.

Cenas distribuídas

O VRML inclui duas primitivas que permitem que a definição de uma cena possa ser distribuída: o nó *Inline*, que faculta a inclusão de outra cena armazenada em qualquer lugar da Web, e o nó *Externproto* que possibilita que definições de novos nós possam ser acedidos de qualquer lugar da Web. Ao contrário de um protótipo, o nó *Externproto* é apenas uma referência a um comando *Proto* em outro ficheiro VRML, sendo necessário declarar a *interface* do protótipo sem que os campos tenham os valores por defeito.

Um arquivo VRML pode tornar-se muito grande e complicado de gerir quando os objectos ou cenas são complexas. Desta forma, a utilização de nós *Inline* permite dividir o ficheiro em pequenas partes de uma cena, sendo apenas especificado o *url* de cada ficheiro constituinte.

3.1.1.2. Estrutura de um Arquivo

Um arquivo VRML contém uma estrutura com os seguintes elementos: cabeçalho, cena, protótipos e eventos.

Cabeçalho

O cabeçalho num arquivo VRML é obrigatório, identificando o ficheiro como VRML e informando do tipo de codificação usada:

```
#VRML V2.0 utf8
```

Cena

A cena compõe-se da definição dos objectos gráficos, os nós e respectivas propriedades. Um nó indica o tipo de objecto que descreve e as respectivas propriedades. O VRML define 54 tipos diferentes de nós, incluindo, por exemplo, a geometria primitiva, as propriedades da aparência, o som e vários tipos de nós de agrupamento. Estes nós podem ser divididos em sete categorias como é apresentado na Tabela 5 [Miranda99]. A definição completa de todos os nós, pode ser encontrada em [VRML97].

Tabela 5 – As categorias em que se dividem os diferentes nós

Tipos de nós	Objectivo
Geometria	Representam a geometria no mundo VRML.
Aparência	Usados para determinar o aspecto de um objecto.
Agrupamento	Servem para agrupar vários nós.
Animação e comportamentos	Fornecem suporte para animação, som e outras actividades baseadas no tempo, assim como para programação de comportamentos.
Sensores	Detectam os movimentos e as acções do utilizador
Ambientes	Servem para criar ambientes em determinadas áreas do mundo.
Navegação	Afectam a navegação do utilizador através do mundo.

Protótipos

O nó *Proto* permite a criação de novos nós através do agrupamento de um conjunto de outros nós, com o intuito de possibilitar a sua reutilização. Este nó é constituído por duas partes: a primeira parte representa o cabeçalho, o qual possui o nome do novo nó, vários campos e eventos; a segunda parte contém vários nós, os quais dizem respeito à geometria do protótipo.

Eventos

Quando um campo de um nó sofre alterações depois de ser carregado, gera um evento. A conexão percorrida quando um evento é enviado de um nó para outro faz-se utilizando rotas ou *Scripts*.

A rota (*Route*) define o caminho que será percorrido pelo evento gerado ou recebido. A sua sintaxe é:

```
ROUTE Node.eventOut_changed TO Node.set_eventIn
```

A Figura 21 esquematiza um exemplo de uma rota entre dois sensores.

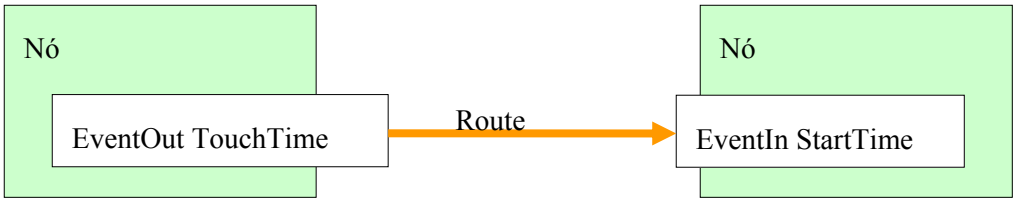


Figura 21 – Definição de uma rota

3.1.2 - X3D

O X3D (*Extensible 3D*) [X3D] é o *standard* sucessor do VRML97, oferecendo características novas relativamente a este, tais como uma maior flexibilidade, a inclusão de novas características na codificação (formatos VRML, XML e binário comprimido), introdução de sensores de teclado, conceitos abstractos como *NURBS* (componente que define uma colecção específica de nós com um conjunto de funcionalidades comuns), a possível implementação em *browsers* com diferentes níveis de funcionalidade, uma estrutura da cena (*scene graph)* com novos nós e campos de dados, maior precisão com a iluminação e modelo de eventos, bem como a revisão e unificação do modelo API[12]. A API unificada do X3D esclarece e resolve diversos problemas que existem no VRML, resultando numa implementação mais robusta e segura (Figura 22) [Chierici04].

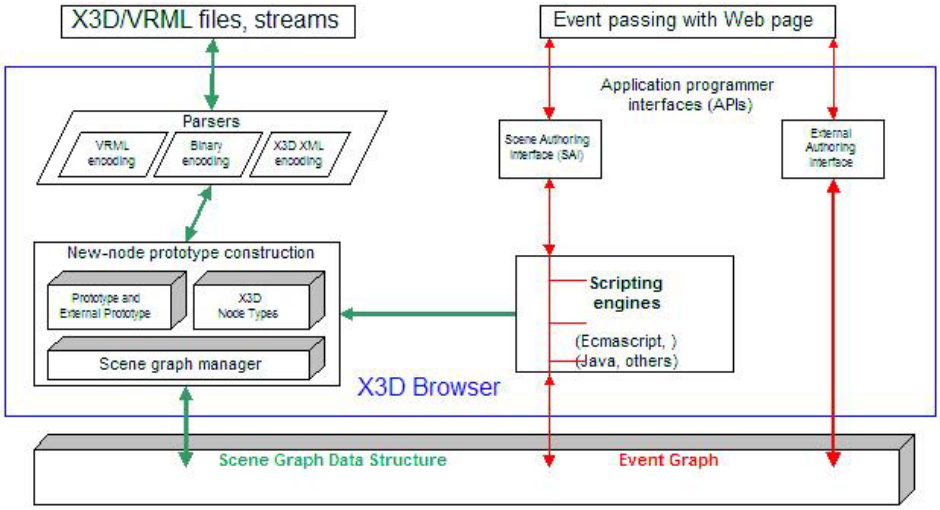


Figura 22 – Arquitectura e componentes básicos do X3D

[12] Uma *Application Programming Interface* (API) ou Interface de Programação de Aplicativos, é uma *interface* do código fonte, constituída por um conjunto de rotinas e padrões acessíveis apenas por programação, que permitem utilizar as características do programa de uma forma simples para o utilizador.

Esta especificação contém requisitos de como um documento X3D deve ser visualizado, possui uma linguagem bastante robusta e suporta conteúdos genéricos 3D para interactividade na *Web* e outros media. Novos conceitos introduzidos na arquitectura modular do X3D são o componente (*component*) e o perfil (*profile*). Um componente é um conjunto específico de nós que possui um conjunto de funcionalidades. Um perfil é um conjunto de componentes e de níveis para cada um destes componentes, assim como os critérios mínimos que suportam todos os objectos desse conjunto. É necessária a definição do perfil a usar em cada ficheiro X3D (Figura 23) [Chierici04].

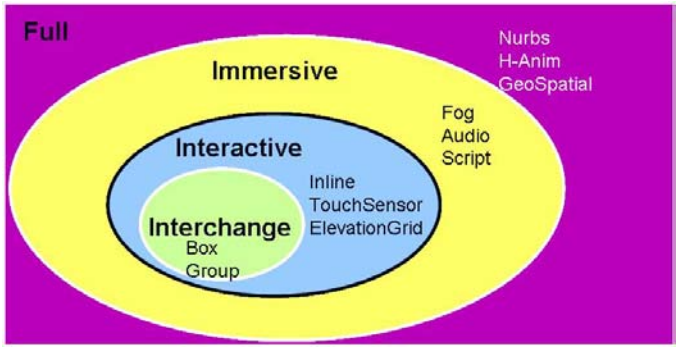


Figura 23 – Representação gráfica do conceito de perfil

Estes dois conceitos oferecem ao X3D uma arquitectura modular de maior expansibilidade e flexibilidade, permitindo vários graus de suporte do *standard* para suprimir uma variedade de necessidades detectadas no VRML.

3.1.2.1. Estrutura de um Documento

A estrutura de um documento XML é idêntica à de um documento VRML, iniciando-se com o cabeçalho, seguido das declarações de protótipos, do cenário gráfico e termina com a área de eventos. Da mesma forma, o X3D adoptou a estrutura hierárquica da cena do VRML, isto é, o *scene graph* que contém todos os objectos do sistema e as relações entre eles. Esta hierarquia possui elementos que podem conter atributos específicos com um ou mais elementos "filhos".

O X3D adoptou a sintaxe do XML (*Extensible Markup Language*) [XML] como forma de resolver problemas relativos à integração e interoperabilidade com as restantes ferramentas da *Web*. Os principais editores existentes no mercado são o *X3D-Edit* e o editor de XML *Xeena* da IBM.

Os nós e os campos em X3D são conhecidos como elementos e atributos respectivamente. O documento é constituído por *tags* de início e *tags* de fim (`<tag>…</tag>`), em que os atributos estão contidos dentro da *tag* inicial. Tudo que se encontra entre a *tag* inicial e a final é designado de *content*, o qual pode ser uma nova *tag*

ou um conjunto de *tags*. Se o elemento contém apenas atributos, a *tag* de início e fim pode ser combinada numa só *tag*.

3.1.2.2. Componentes Funcionais

Um ficheiro X3D é composto pelos mesmos componentes funcionais que um ficheiro em VRML. A Tabela 6 expõe um comparativo entre alguns componentes em VRML e X3D.

Tabela 6 – Tabela comparativa entre alguns elementos VRML e X3D

VRML	X3D
`Shape { Appearance {} Geometry {}}`	`<Shape> <Appearance> </Appearance> </Shape>`
`Appearance { Material {} Texture {} TextureTransform {}}`	`<Appearance> <Material/> <Texture/> <TextureTransform/> </Appearance>`
`Material { diffuseColor <valor> emissiveColor <valor> ambientIntensity <valor> shininess <valor> specularColor <valor> transparency <valor>}`	`<Material diffuseColor=”<valor>” emissiveColor=”<valor”> ambientIntensity=”<valor>” shininess=”<valor>” specularColor=”<valor>” transparency=”<valor>”/>`
`Box { size <valorX> <valorY> <valorZ>}`	`<Box size=”<valorX> <valorY> <valorZ>”/>`

O nó *Shape* é usado para a construção das formas geométricas. Em VRML este nó contém os campos *Appearance* e *Geometry*. O campo *Appearance* contém o nó *Appearance*, o qual especifica os atributos visuais do objecto. O campo *Geometry* especifica o nó *Geometry*, que define a geometria do objecto. Em X3D as *tag* são idênticas com a excepção do nó *Geometry*. O nó *Appearance* especifica as propriedades visuais de uma forma geométrica. Pode conter os nós *Material*, *Texture* e *TextureTransform*.

O nó *Material* especifica as propriedades da superfície para o nó de geometria associado. Possui, em VRML e em X3D, seis campos: *diffuseColor*, *emissiveColor*, *ambientIntensity*, *shininess*, *specularColor* e *transparency*. O primeiro campo define a cor do objecto; o segundo é usado para definir objectos como fosforescentes; o campo *ambientIntensity* especifica a percentagem de luz reflectida; o valor *shininess* controla a abertura do brilho; o campo *specularColor* define a cor do brilho e o *transparency* especifica a percentagem de luz que é transmitida através do objecto.

O nó *Box* é usado para definir paralelepípedos. O campo *size* especifica as dimensões da caixa segundo as direcção de X, Y e Z.

3.2 - EAI e SAI

Para controlar o ambiente virtual em VRML ou em X3D, é necessário utilizar uma *interface* denominada *External Authoring Interface* [Parallel] ou *Scene Access Interface* [X3D] respectivamente. Esta *interface* permite definir um conjunto de funções que

afectam o ambiente virtual, possibilitando o acesso às funcionalidades do *browser*, o envio e recepção de eventos para a cena e a notificação da ocorrência de eventos na cena. As secções seguintes descrevem resumidamente as duas *interfaces*.

3.2.1 - EAI

A EAI (*External Authoring Interface*), é constituída por um conjunto de classes Java que permitem a comunicação entre um mundo VRML e uma aplicação Java; a sua função é permitir uma maior liberdade na criação de *interfaces* sofisticados para interagir com estes mundos VRML, de forma a tornar possível a criação de sistemas multimédia complexos.

A *interface* EAI e o nó *Script* permitem ao programador o controlo dos nós da cena através da linguagem Java: a primeira oferece um método generalizado de aceder aos nós e eventos do *scene graph* através da conexão do *plugin* do *browser* VRML (como o *LiveConnect* ou o *Cortona*) ao *browser* da Internet, possibilitando a utilização e a afectação de eventos e campos de uma forma dinâmica; enquanto que o segundo disponibiliza um conjunto de campos e eventos predefinidos para controlar o fluxo de eventos, sem sair do *browser* VRML ou para adicionar comportamentos aos objectos (isto é, define comportamentos dentro do *browser*, sem comunicação com código externo).

A EAI contém um conjunto de classes Java, com o objectivo principal de estabelecer a comunicação entre o VRML e a aplicação Java (Figura 24) [Pecis]. Desta forma, em função da acção executada pelo utilizador é possível, a um programa externo, como uma *applet* Java, aceder aos diversos objectos que compõem o mundo virtual em VRML e modificar as suas características.

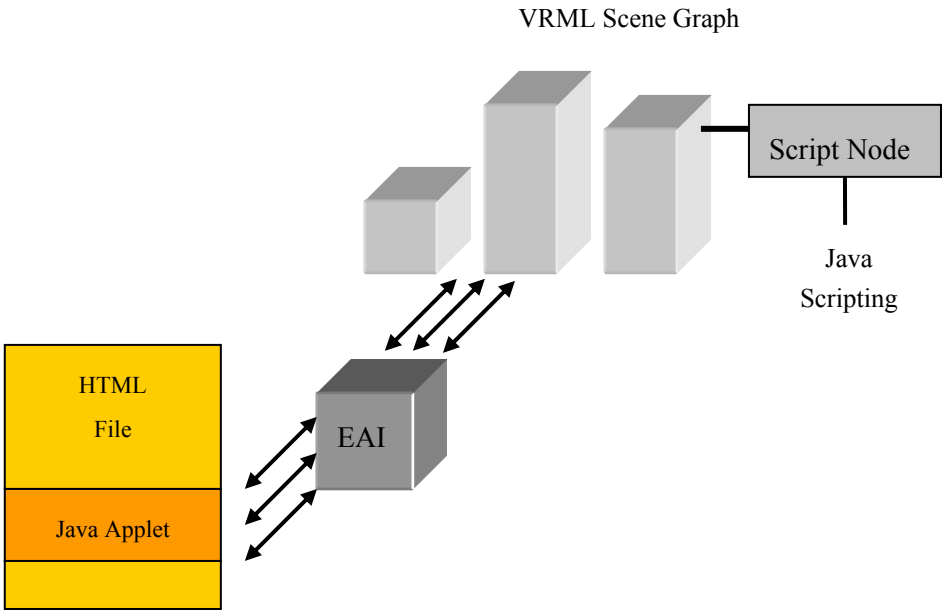


Figura 24 – Esquema representativo da relação entre o VRML, a EAI e a Java.

A EAI permite uma maior modularidade e simplicidade dos programas e uma maior liberdade para a criação de sofisticadas *interfaces* de interacção com os mundos VRML, mas com a desvantagem de não possibilitar uma fácil portabilidade entre as diversas plataformas existentes.

O Anexo A apresenta uma breve descrição da *interface* EAI, referindo os princípios fundamentais para se estabelecer uma comunicação entre o mundo VRML e o utilizador.

3.2.2 - SAI

Para se comunicar com a cena X3D é necessário utilizar a *interface* SAI (*Scene Access Interface*). Esta *interface* elimina a incompatibilidade entre as diversas linguagens de programação ligadas ao X3D, como a ECMAScript e a Java, assim como entre aplicações externas e *scripts* internos. Devido às suas especificações, as implementações que utilizam a SAI são automaticamente *ready-to-use* para todos os *browsers* em qualquer plataforma.

Conforme visto na secção anterior, a EAI constitui a forma de aceder externamente às cenas VRML, distinguindo-se por isso do método interno do *script*. No contexto do X3D, a SAI junta efectivamente estes dois métodos, implementando melhoramentos na portabilidade dos *browsers* e na interacção da cena. Outra vantagem da utilização da SAI na implementação é a sua característica de livre instalação, isto é, não é necessário fazer o *download* ou a instalação de qualquer tipo de *software* ou módulo, à excepção da *Java Runtime Environment* (JRE) a qual, normalmente, já se encontra instalada no

computador. Utilizando o nó *Script* e o mecanismo de prototipagem em X3D, os protótipos e os *scripts* são actualizados automaticamente pelo *browser* a partir da *Web*. Quando o *browser* não suporta a SAI, não existe o problema de retornar uma mensagem de erro pois os nós são simplesmente ignorados.

A integração entre X3D e Java pode ser feita utilizando um acesso interno ou um acesso externo. No primeiro tipo de acesso (Figura 25), o ficheiro X3D faz a ligação com a classe Java através do nó *Script*, sendo obrigatório implementar a *interface X3DScriptImplementation* [Martins]. Quando a cena X3D é carregada, a classe Java é chamada de forma a que as interacções ocorram, como por exemplo, a alteração da cor de um objecto ou as respectivas coordenadas quando lhe é feito um clique pelo utilizador. Sucintamente, os passos a executar na programação são: no ficheiro X3D define-se a *interface* e as rotas; no ficheiro Java acede-se aos campos de entrada e de saída a partir do método *setFields()*; seguidamente, adicionam-se os *listeners* para que a Java seja notificada da ocorrência dos eventos de entrada; por fim calculam-se os valores de saída sendo necessário definir no método *readableFieldChanged* os respectivos valores através do método *setValue()*. O Anexo B descreve de uma forma detalhada os procedimentos necessários para este tipo de acesso.

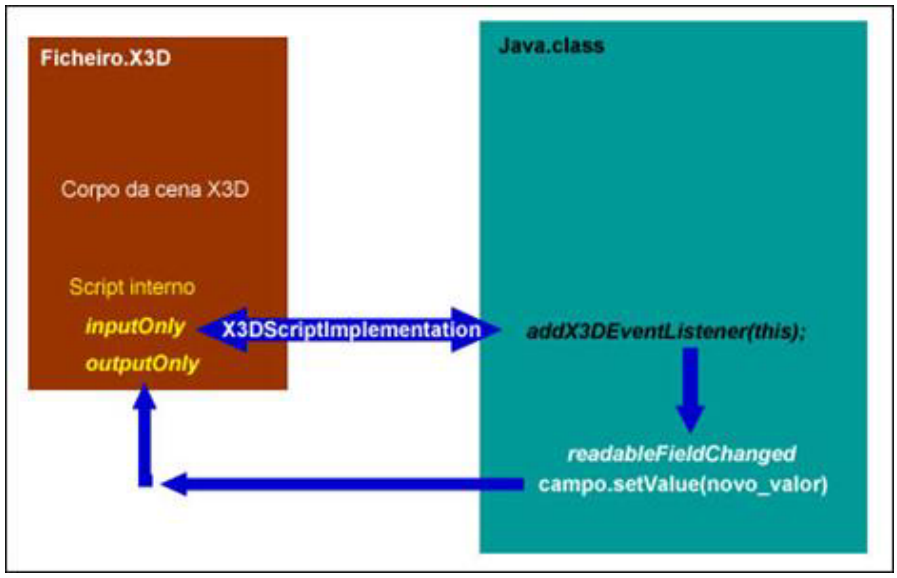


Figura 25 – Filosofia de programação do acesso interno

No acesso externo (Figura 26) existe uma janela referente à cena X3D e outra janela com a *interface* Java, cabendo à classe Java a responsabilidade de chamar o ficheiro X3D [Martins]. De uma forma geral, os passos a executar na programação são: criar um componente X3D; adicionar o componente X3D à *interface* do utilizador; acesso ao *browser* através do método *getbrowser()*; criar a cena 3D com um ficheiro X3D; finalmente, actualizar o *browser* com a cena criada. O Anexo B descreve de uma forma detalhada os procedimentos necessários para este tipo de acesso.

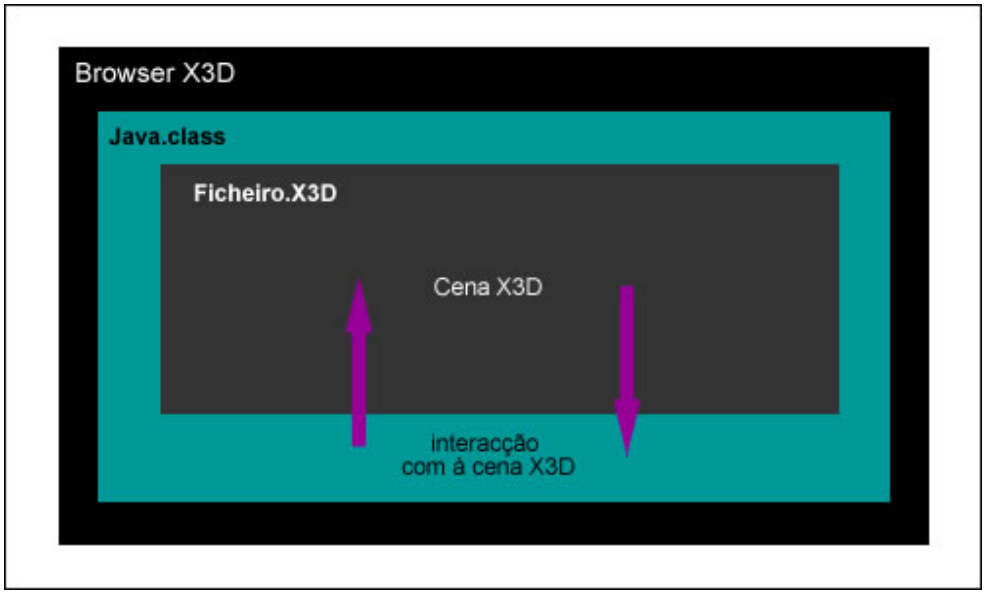


Figura 26 – Representação da filosofia do acesso externo

3.3 - Síntese do Capítulo

O VRML surgiu, com a rápida expansão da *Web*, como uma tentativa de juntar os conceitos de *view source* e de realidade virtual. O seu conceito foi revolucionário mas continha bastantes limitações, o que levou o *Web3D Consortium* a reformular as suas especificações, adoptando o X3D como um padrão ISO. A estrutura gráfica do VRML é comparável a uma árvore, dado que contém nós estruturados de um modo hierárquico. Esta árvore descreve os aspectos estáticos da cena 3D, enquanto que os aspectos dinâmicos são efectuados por eventos que percorrem rotas entre os diversos nós.

O X3D é uma evolução do VRML, incorporando melhoramentos relativamente a este último, tais como o suporte de novas primitivas gráficas, comportamentos e tipos de interactividade, maior flexibilidade, melhoramentos aos nível da *interface* de programação de aplicativos (API) na cena 3D e a definição de componentes e perfis. Prevê ainda, por outro lado, a possibilidade de a especificação ser facilmente expandida.

Para ser possível a interacção do utilizador com o ambiente virtual VRML, é necessário utilizar a EAI. Esta permite que um programa externo como uma *applet* Java aceda aos objectos na cena VRML e modifique as suas características. Com a passagem para o X3D, esta tecnologia evoluiu para a SAI, integrando aperfeiçoamentos na portabilidade dos *browsers* e na interacção da cena.

CAPÍTULO 4

UMA METODOLOGIA PARA A SEGMENTAÇÃO DE UM AMBIENTE VIRTUAL URBANO

Índice do Capítulo

4 - Uma Metodologia para a Segmentação de um Ambiente Virtual Urbano

A criação e consequente disponibilização de modelos tridimensionais de cidades reveste-se, cada vez mais, de uma grande importância em diversas áreas. Estes modelos constituídos em ambientes virtuais, fornecem uma resposta mais completa e eficaz às necessidades dos cidadãos e das organizações.

O número de utilizadores que estabelecem acesso a bases de dados com modelos 3D tem crescido drasticamente nos últimos anos. Enquanto que alguns acedem a estas bases de dados utilizando *hardware* gráfico de elevado desempenho sobre conexões de rede em banda larga, outros acedem à informação com dispositivos limitados em termos de processamento gráfico e com baixa velocidade de ligação à rede.

Desta forma, na criação de ambientes virtuais, é muito importante considerar a hierarquia da informação, a qual permite a interacção do utilizador com os objectos a diferentes níveis. O esquema de representação que se adopta para representar a informação geométrica que constituiu o modelo é de grande relevância para que o ficheiro seja de pequena dimensão e de rápida visualização. Este processo é designado de segmentação, possibilitando uma diminuição da complexidade dos ambientes virtuais, através da divisão do ambiente original em ambientes mais pequenos, de forma a permitir a transmissão destes últimos por redes com pequena largura de banda e a sua utilização em computadores com baixo processamento gráfico.

Neste capítulo, são descritos os processos envolvidos na criação de um modelo de uma cidade virtual, dando especial atenção à sua segmentação. São definidas algumas directrizes relativamente ao cálculo de oclusão aplicado no âmbito desta dissertação.

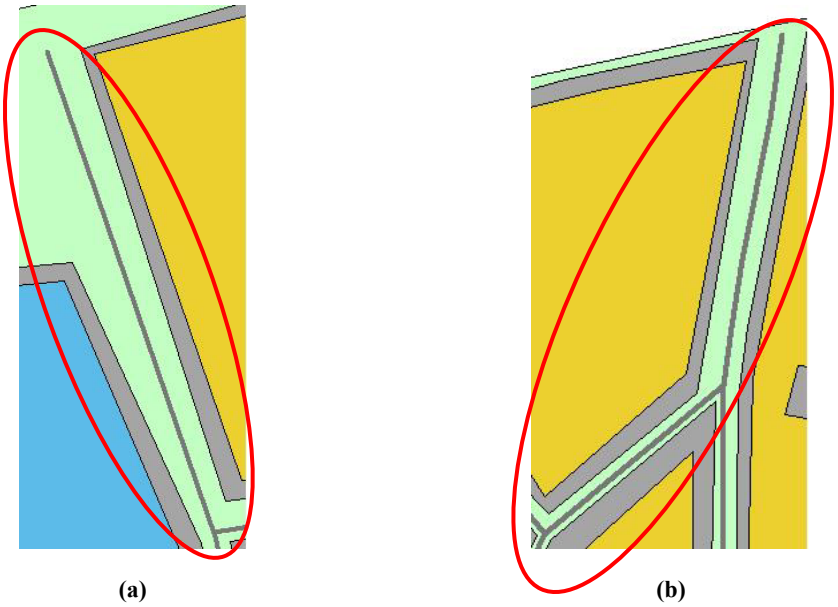
4.1 - Criação de um Modelo Urbano

Para construir um ambiente virtual urbano é necessário, em primeiro lugar, definir os elementos essenciais, as suas características e as relações entre os diversos elementos.

A base do processo de definição de elementos distintos foi o trabalho desenvolvido por Copper [Copper99]. O sistema descrito por Copper permite a criação de modelos urbanos virtuais de uma forma rápida, começando o processo de modelação de uma cidade por considerar as áreas com pouco interesse, aplicando-lhes um modelo de parâmetros pré-definidos baseado em características reais, tais como o estilo do edifício, a sua idade e o número de janelas. Ao contrário destas zonas, as zonas de interesse são modeladas detalhadamente. O modelo produzido não fica exactamente como a área urbana definida, contudo mantém um sentido de realidade e semelhança.

Tendo em conta o princípio adoptado por Copper, foram definidos os elementos constituintes de um modelo de cidade, respectivas características e relações entre si, de modo a ser possível, na fase de implementação, a sua criação. A definição dos elementos que constituem o modelo consiste numa descrição 2D e um parâmetro que estabelece a altura respectiva.

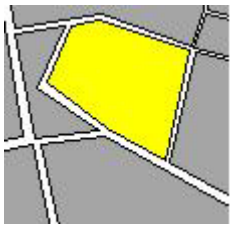
Uma cidade é constituída, principalmente, por ruas, quarteirões, edifícios e jardins. Uma rua é identificada por um nome e pode ser definida como um segmento de recta (Figura 27 (a)) ou um conjunto de segmentos de recta (Figura 27 (b)). Um quarteirão descreve a forma de um polígono, convexo ou não (Figura 28).



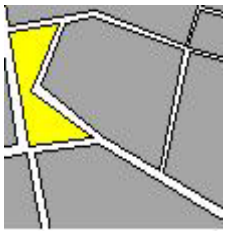
(a) **(b)**

Figura 27 – Rua constituída por um segmento de recta (a) ou por um conjunto de segmentos de recta (b)

Os edifícios representam a área edificada podendo ser de dois tipos: sem interesse público e com interesse público (Figura 29). Os do primeiro tipo podem ser definidos quanto à sua altura, comprimento e largura, de uma forma quase aleatória, visto não terem um peso relevante na caracterização de uma cidade.



(a)

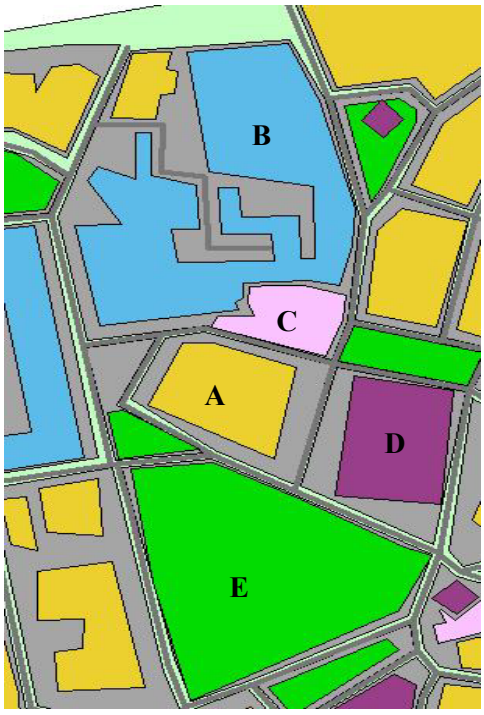


(b)

Figura 28 – Quarteirão constituído por um polígono convexo (a) ou constituído por um não convexo (b)

Os elementos com interesse público, incluem os monumentos e edifícios classificados, como igrejas, hospitais, etc. Estes elementos são identificados por um nome e têm características próprias como área e altura.

A classificação dos edifícios e outros elementos arquitectónicos segundo os dois grupos permite definir e especificar com maior realismo os elementos com interesse público, pois são estes que caracterizam a própria cidade e fornecem ao utilizador a orientação necessária para navegar pela cidade.



Legenda:

A – Área edificada sem interesse público;

B – Área edificada com interesse público: hospital;

C – Área edificada com interesse público: igreja;

D – Área edificada com interesse público: monumento.

Figura 29 – Classificação dos edifícios

4.2 - Segmentação e sua Importância

Os cenários urbanos virtuais implicam grandes desafios aos sistemas de visualização interactiva, visto possuírem uma grande diversidade e quantidade de informação a representar. A sua modelação pressupõe, normalmente, uma grande quantidade de polígonos, a utilização de imagens de alta qualidade para as fachadas dos edifícios e zonas arborizadas e, em alguns casos, a fotografia aérea para as texturas a aplicar no modelo digital de terreno.

Outra dificuldade inerente à reprodução de cidades é a discrepância de elementos a representar enquanto o utilizador se desloca. Isto é, numa determinada posição do observador, é visível um pequeno conjunto de edifícios mas, com orientação diferente, é visível uma outra área panorâmica que pode englobar centenas de outros componentes. A memorização e processamento de todos estes componentes contribui para diminuir o desempenho do sistema.

A segmentação de ambientes virtuais permite minimizar este problema, através da divisão do ambiente original em ambientes menores, de modo a que apenas os objectos próximos sejam carregados para a memória. A movimentação do observador implica no entanto a leitura constante de novos segmentos, de forma a que o mesmo não se aperceba da fragmentação do mundo virtual que está a visitar.

No trabalho desenvolvido por Sillion [Sillion97] a segmentação consiste na divisão do modelo em duas partes correspondentes à vizinhança local próxima e à paisagem mais distante (Figura 30).

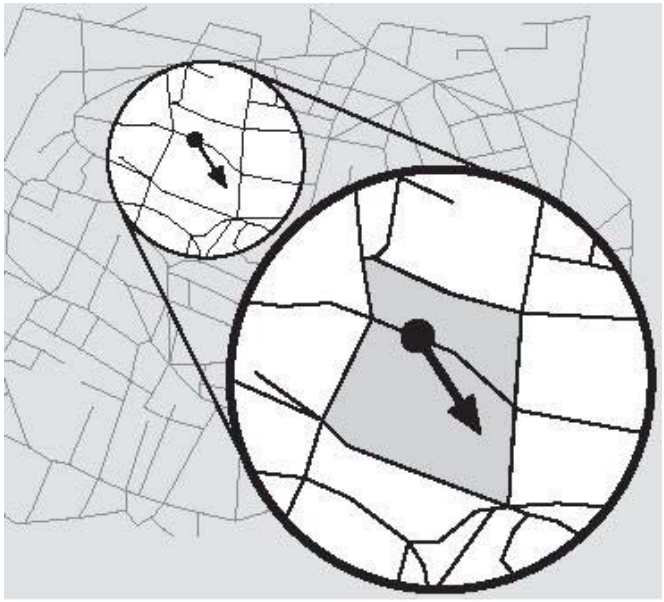


Figura 30 – Os quarteirões adjacentes à rua onde se encontra o observador compõem a vizinhança local próxima

Considerando que, numa cena, os objectos maiores e mais próximos produzem, normalmente, maior impacto, Wimmer [Wimmer99] divide a cena em campo próximo (*Near Field*) e campo distante (*Far Field*). O conjunto de elementos descritos como pertencentes ao campo próximo é determinado a partir de uma distância pré-estabelecida ao observador. Os objectos são considerados no conjunto de elementos do campo próximo através da aplicação de uma malha regular.

Em [Bittner01] e [Kultun00a] a segmentação é obtida a partir do PVS (*Potentially Visible Set*) para uma dada região da cena. No primeiro caso, as ruas são definidas como as células visíveis, independentemente do PVS; o algoritmo determina a visibilidade para um determinado ponto de vista numa cena 2.5D, combinando uma solução exacta para o problema de visibilidade em 2D com uma solução conservativa para a restante "metade" da dimensão. Para isso, são determinados todos os objectos potencialmente visíveis a partir de uma determinada célula, representada por um poliedro convexo feito a partir de faces verticais. Em seguida, são consideradas apenas as faces superiores das entidades da cena: para cada uma destas faces, são processados os oclusores numa ordem frente-para-trás, em função da proximidade, e é construída a estrutura hierárquica em *line-space*[13] que representa as partes visíveis da cena. No trabalho desenvolvido por Kultun, para simplificação, o PVS foi definido como uma grelha regular que divide a cena em células, representadas por cubos alinhados em função de X. É introduzido o conceito de oclusor virtual, representado por um objecto convexo dependente do ponto de vista do utilizador, o qual garante ser um oclusor efectivo para determinado ponto de vista.

4.3 - Metodologia para Segmentação de um Modelo de Cidade

O âmbito desta dissertação abrange o desenvolvimento de uma metodologia de segmentação que potencie a utilização de ambientes urbanos virtuais extensos com base na localização do utilizador. Desta forma, a segmentação deve centrar-se na localização do utilizador e nos objectos visíveis relativamente ao seu campo de visão. A metodologia de segmentação deve basear-se numa perspectiva 2D correspondente à planta da cidade, não descurando a importância da altura dos edifícios e subsequentes problemas.

Visto que o modelo de uma cidade abrange áreas que podem ir até vários quilómetros, requerendo grande capacidade de armazenamento/processamento e largura de banda para

[13] Bittner define o fenómeno da visibilidade por meio de pontos mutuamente oclusos, isto é, dois pontos são visíveis se o segmento que os une não estiver ocluso. Desta forma, a visibilidade de um conjunto de pontos é dada pela visibilidade ao longo das linhas que cruzam estes pontos, sendo o espaço definido pelas linhas 2D orientadas designado de *line-space*.

o envio/recepção do documento de modelação, é conveniente que o modelo virtual seja dividido ou segmentado em modelos mais pequenos. Um aspecto fundamental a ter em conta na metodologia prende-se com o facto de se pretender que a forma de deslocação do observador seja o mais realista possível. A forma como é definida a navegação para o observador influencia o modo de segmentação de um modelo de cidade. A navegação do observador relaciona-se com a maneira como o observador se desloca pelo modelo de cidade, podendo ser em modo de "passeio" e/ou em modo "voar". Sendo a navegação em modo de "passeio" o observador explora o modelo da cidade caminhando pelas ruas e visualizando os elementos contidos num volume com a forma de uma pirâmide quadrangular. Esta pirâmide contém o seu vértice no observador e os restantes lados são limitados por: a superfície do chão no lado inferior da pirâmide, pelos edifícios que se encontram do lado esquerdo e direito do observador respectivamente, pela altura dos edifícios no lado superior da pirâmide. A Figura 31 apresenta os elementos visíveis de uma cidade para um observador que navega em modo de "passeio" [Pimentel01].

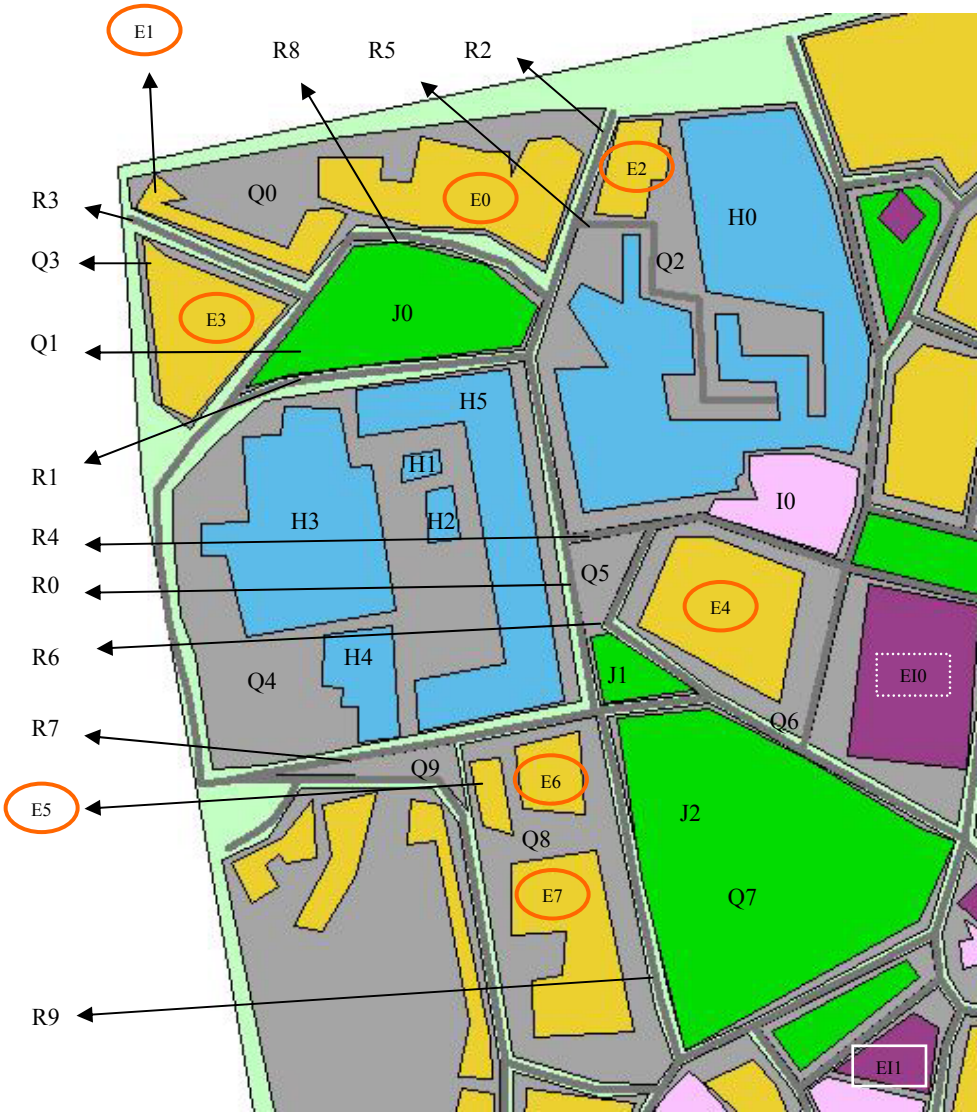
Figura 31 – Elementos visíveis para uma navegação em modo "passeio"

A navegação em modo de "voar" apresenta-se ao observador com uma perspectiva diferente, enquadrada no mesmo tipo de volume piramidal referido para o modo de navegação em "passeio", mas semelhante à visualizada a partir de um avião que paira sobre o modelo da cidade. O observador visualiza os elementos que compõem a cidade com um panorama de cima para baixo. A Figura 32 apresenta os elementos visíveis de uma cidade para um observador que navega no modo de "voar" [3DFunchal].

Figura 32 – Elementos visíveis para uma navegação em modo "voar"

Partindo do princípio que, numa cidade virtual, o utilizador se desloca pelas ruas, como foi referido anteriormente e mantendo-se que a segmentação deve basear-se na planta da cidade, a segmentação pode seguir os limites dos quarteirões, por motivos da oclusão natural que decorre dos prédios imediatos. Esta segmentação adaptativa, implica ter em consideração vários aspectos, como os elementos oclusores que estão naturalmente associados a determinada rua mas sem esquecer, a visualização de edifícios afastados com uma altura superior à média assim como a visibilidade permitida além das áreas verdes, áreas vazias ou áreas com edifícios de pequeno porte.

A determinação dos elementos associados a determinada rua inicia-se com a associação dos quarteirões que a circundam e, a estes, dos edifícios ou jardins considerados dentro dos limites.



Legenda:	
R0: rua Prof. Vicente José de Carvalho	Q0 a Q9: quarteirões
R1: rua do Dr. Tiago Almeida	E0 a E7: edifício sem interesse publico
R2: rua Travessa do Carrega	H0 a H5: edifício com interesse publico – hospital
R3: rua do Prof. Jaime Sousa	I0: edifício com interesse publico – igreja
R4: rua do Carmo	EI0: edifício com interesse público – Faculdade
R5: caminho Hosp. S. António	EI1: edifício com interesse público – Cadeia de Relação
R6: largo do Prof. Abel Salazar	J0, J1, J2: jardins
R7: rua da Restauração	
R8: rua Clemente Meneres	
R9: campo dos Mártires da Pátria	

Figura 33 – Elementos associados a uma determinada rua

Uma relação rua/quarteirão pode ser estabelecida, no caso de este último intersectar o polígono formado pela rua ou se se encontrar num raio pré-definido de um dos pontos que definem a rua. Quando a rua é constituída por um conjunto de segmentos de recta, a relação rua/quarteirão é determinada da mesma forma mas para cada um dos segmentos de recta que constituem a rua. Tendo em conta o exemplo apresentado na Figura 33 e alguns dos elementos identificados, é necessário estabelecer as associações entre os diversos elementos:

$CQ_{R0} = \{Q1, Q2, Q4, Q5, Q7, Q8\}$	Conjunto que contém os quarteirões associados à rua R0
$CQ_{R1} = \{Q1, Q2, Q3, Q4\}$	Conjunto que contém os quarteirões associados à rua R1
$CQ_{R2} = \{Q0, Q1, Q2, Q4\}$	Conjunto que contém os quarteirões associados à rua R2
…	…

Na Figura 33 a rua R1 não intersecta o quarteirão Q3 mas, este faz parte dos elementos que se encontram num raio próximo da rua. Estes elementos devem ser incluídos na lista de elementos visíveis, pois quando o observador se aproxima do limite da rua, é possível visualizar uma continuação do modelo de cidade em vez de um vazio.

Um quarteirão pode encontrar-se vazio (isto é, sem qualquer elemento), ou pode conter um ou vários edifícios, assim como um ou vários jardins. Um edifício é considerado nos limites de um quarteirão se o intersecta. Tendo como exemplo a Figura 33, é necessário definir, para cada quarteirão, os respectivos edifícios e/ou jardins associados:

$C_{Q0} = \{E0, E1\}$	Conjunto de elementos associados ao quarteirão Q0
$C_{Q1} = \{J0\}$	Conjunto de elementos associados ao quarteirão Q1
$C_{Q2} = \{E2, H0, I0\}$	Conjunto de elementos associados ao quarteirão Q2
…	…

E5

Na Figura 33 pode-se identificar o quarteirão Q9 como um quarteirão vazio. O mesmo se pode verificar para o quarteirão assinalado na Figura 34.

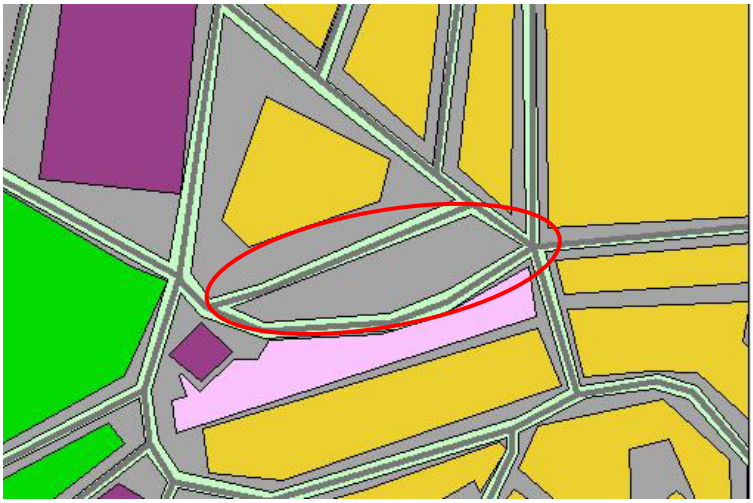


Figura 34 – Quarteirão sem elementos

Associada a cada rua, deverão ainda estar as ruas que a cruzam e os respectivos elementos associados. Desta forma, para o exemplo apresentado na Figura 33, ainda deverá ser criado, para cada rua, um conjunto que contenha as ruas que lhe estão associadas:

$CR_{R0} = \{R1, R2, R4, R7, R9\}$	Conjunto das ruas associadas à rua R0
$CR_{R1} = \{R0, R2, R8\}$	Conjunto das ruas associadas à rua R1
$CR_{R2} = \{R0, R1, R5, R8\}$	Conjunto das ruas associadas à rua R2
...	…

Um aspecto considerado relevante no processo de segmentação, como forma de tornar mais real o ambiente virtual urbano, é a inclusão de objectos que, embora não pertençam aos elementos associados a uma determinada rua, possuem uma altura acima da média e estão contidos num perímetro pré-definido. A inclusão de edifícios com uma altura relevante é importante porque permite ao observador gozar de pontos de referência durante a sua navegação pelo ambiente virtual. Estes edifícios caracterizam a cidade e consequentemente devem ser igualmente realçados no ambiente virtual. Para o exemplo apresentado na Figura 33, os edifícios identificados como EI0 e EI1 encontram-se classificados como edifícios com uma altura acima dos restantes elementos. Desta forma, é necessário verificar para as ruas identificadas as que devem incluir os edifícios EI0 e EI1. Tendo em conta o perímetro pré-definido, as ruas identificadas devem incluir os seguintes edifícios:

$CE_{R0} = \{EI0\}$	Conjunto de edifícios com altura relevante associados à rua R0
$CE_{R4} = \{EI0\}$	Conjunto de edifícios com altura relevante associados à rua R4
$CE_{R5} = \{EI0\}$	Conjunto de edifícios com altura relevante associados à rua R5
...	…

No trabalho desenvolvido por [Sillion97], também são considerados os edifícios que se encontram a uma grande distância do utilizador mas que são importantes na visualização do modelo.

Quando existe um jardim numa determinada rua, o campo de visibilidade estende-se por todo o jardim e componentes que lhe estão associados. Este cuidado recai em áreas verdes, áreas vazias e em áreas com objectos de pequeno porte. No exemplo apresentado na Figura 33, também existem quarteirões que possuem um jardim (quarteirões Q1, Q5 e Q7) obrigando à inclusão dos quarteirões que o circundam e respectivos elementos associados. Se um quarteirão não contém elementos ou existem apenas pequenos monumentos, tais como estátuas, isso significa que todos os quarteirões da sua retaguarda são potencialmente visíveis. Para o exemplo apresentado na Figura 33, existe ainda o quarteirão Q9 que não possui elementos, mas não existe qualquer quarteirão que compreenda o requisito de possuir pequenos monumentos. Mas a Figura 35 apresenta um quarteirão (Q60) com três pequenos monumentos (EI4, EI5 e EI6), os quais devido ao papel insignificante que têm para a oclusão, pelas suas reduzidas dimensões, não permitem ocultar outros elementos. Deste modo, é necessário armazenar, para este tipo de quarteirões, os quarteirões que lhe devem estar associados (C_{Q60c}):

$$C_{Q60c} = \{Q61, Q62, Q63, Q64, Q65, Q66, Q67, Q68, Q69\}$$

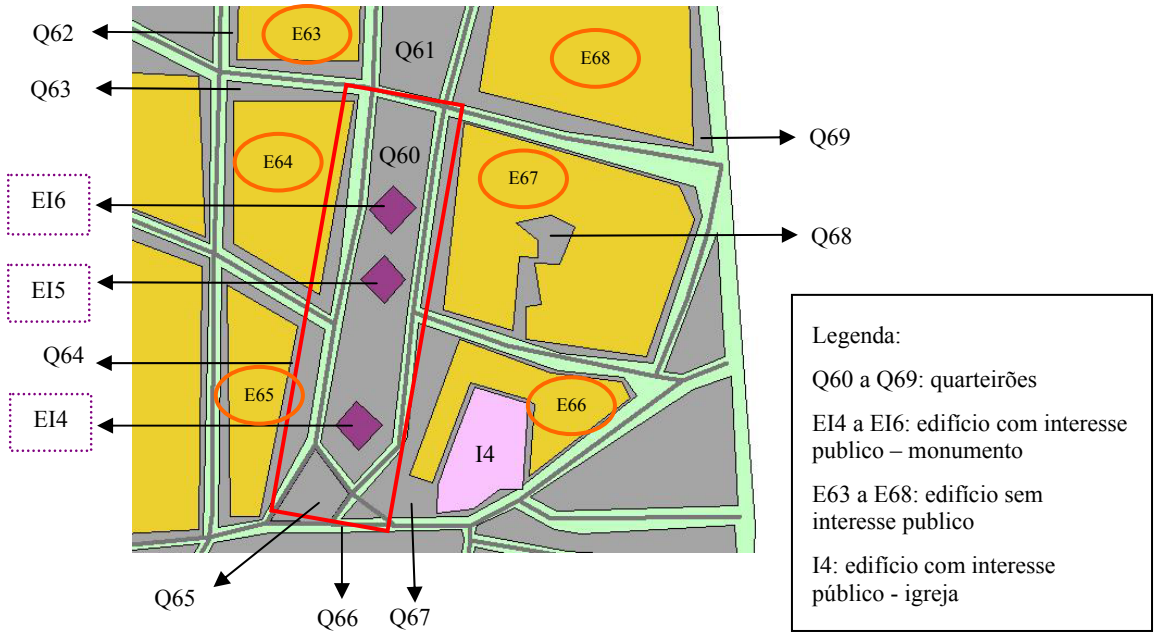


Figura 35 – Quarteirão com pequenos monumentos

Concluída a definição do tipo de segmentação e consequentes características, procede-se à determinação do campo de visibilidade para cada uma das ruas.

4.4 - Campo de Visibilidade

O campo de visibilidade diz respeito à área ou elementos visíveis a partir da localização do utilizador. Tendo em conta a segmentação adaptativa definida na secção 4.3, o campo de visibilidade é determinado em função da rua onde se encontra o utilizador. Isto é, estando o utilizador numa determinada rua R, o campo de visibilidade para esse utilizador corresponde à rua onde este se encontra (R), o conjunto de ruas (A) que formam um cruzamento com R, o conjunto de quarteirões (Q) que circundam R e A e o conjunto de linhas de comboio (L) que se cruzam com R. Dizem ainda respeito ao campo de visibilidade do utilizador todos os elementos associados a cada quarteirão: o conjunto de edifícios sem interesse público (E), o conjunto de jardins (J), o conjunto de monumentos (M), o conjunto de estações de comboio (C), o conjunto de igrejas (I) e o conjunto de hospitais (H):

$$CV = \{R, A, Q, L, E, J, M, C, I, H\}$$

Para o exemplo apresentado na Figura 33 e supondo que se pretende obter o campo de visibilidade para a rua R2 o primeiro passo é introduzir na lista de elementos visíveis a própria rua (R). O segundo passo consiste em verificar para a rua em questão quais as ruas que lhe estão associadas (conjunto A):

$$CV_{R2} = \{R2, R0, R1, R5, R8\}$$

O passo seguinte consiste em verificar para as ruas introduzidas na lista de elementos visíveis os quarteirões que lhe estão associados (conjunto Q):

$$CV_{R2} = \{R2, R0, R1, R5, R8, Q0, Q1, Q2, Q3, Q4, Q5, Q7, Q8\}$$

Segue-se a introdução das linhas de comboio (conjunto L) e de todos os elementos associados a cada quarteirão do CV_{R2} (conjuntos E, J, M, C, I e H):

$$CV_{R2} = \begin{Bmatrix} R2, R0, R1, R5, R8, Q0, Q1, Q2, Q3, Q4, Q5, Q7, Q8, E0, E1, E2, E3, E5, E6, \\ E7, H0, H1, H2, H3, H4, H5, J0, J1, J2 \end{Bmatrix}$$

Como foi referido na secção 4.3, é conveniente a inclusão de elementos que possuem uma altura superior à média tendo em conta um perímetro pré-definido. Desta forma é necessário verificar se existe algum conjunto definido de edifícios com altura relevante para a rua R2 (CE_{R2}) e consequentemente juntar os respectivos elementos ao conjunto CV_{R2}. Para o caso especifico da rua R2 não existe o conjunto CE_{R2}.

Para terminar o cálculo do campo de visibilidade resta verificar, como foi referido na secção 4.3, se a rua onde se encontra o observador possui um jardim, uma área vazia ou uma área com objectos de pequeno porte. Para o exemplo da Figura 33 a rua R2 apenas admite o primeiro requisito através do jardim J0 corresponde ao quarteirão Q1. Visto que o quarteirão Q1 já faz parte do CV_{R2}, logo os elementos que lhe estão associados já se encontram também incluídos no CV_{R2}.

Para o modelo de cidade apresentado na Figura 33 e determinados os elementos constituintes do campo de visibilidade para a localização do observador na rua R2, obteve-se o resultado apresentado na Figura 36.

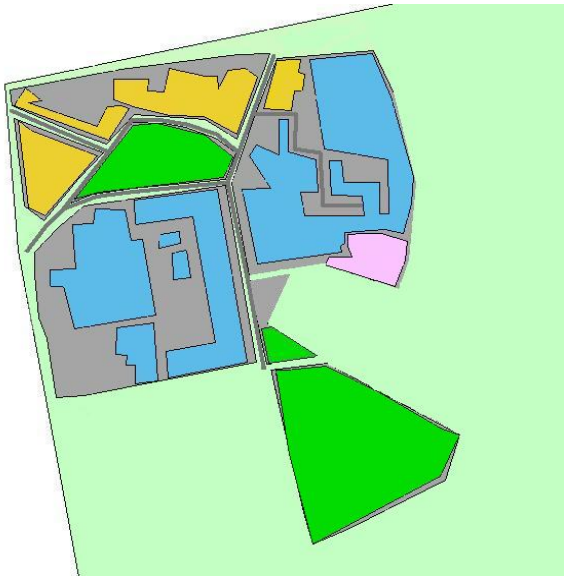


Figura 36 – Elementos constituintes de um modelo de cidade referentes apenas ao campo de visibilidade para a rua R2

4.5 - Cálculo de Oclusão

Tendo-se obtido o campo de visibilidade em função da localização do utilizador, e de forma a tornar mais rápido o processamento e a visualização da cena, a utilização de técnicas de redução da complexidade é essencial, nomeadamente por cálculo de oclusão, como foi descrito no capítulo 2. Estas técnicas são indispensáveis para acelerar os tempos de representação das cenas através da eliminação de partes do modelo que não contribuem significativamente para a imagem final ou através da redução da complexidade geométrica do modelo.

A omissão desta fase resulta em ficheiros bastante grandes e complexos para poderem ser representados, ou origina tempos de espera excessivos. A aplicação de uma determinada técnica, ou conjugação de várias técnicas, deve considerar as características da cena a representar, de modo a conseguir a melhor eficiência do modelo.

Face aos objectivos propostos para esta dissertação, a técnica de remoção rápida de objectos por volumes de sombra, apresentada por Hudson [Hudson97] e a utilização de portais e células para definir o campo de visibilidade [Luebke95], complementam-se e ajustam-se no cálculo de oclusão para o modelo apresentado para esta dissertação.

O cálculo de oclusão para cada rua decorre assim em duas fases. Na primeira fase determinam-se os edifícios que originam maior oclusão, pois são estes que ocultam o maior número de objectos, reduzindo substancialmente a quantidade de edifícios a computar a sua sombra. Para os edifícios classificados como oclusores, numa segunda

fase, é construída a respectiva sombra, de modo a detectar quais os objectos que a intersectam ou que lhe estão contidos. Estes últimos são eliminados da lista de edifícios a representar ou é apenas incluída a parte não intersectada.

4.5.1 - Definição dos Portais e Células

A técnica de cálculo de oclusão definida por Luebke [Luebke95], com as devidas adaptações, adequa-se à metodologia de segmentação definida para esta dissertação porque aplica ao portal e célula conceitos que se enquadram ao tipo de ambiente urbano a analisar nesta dissertação assim como, a filosofia descrita no algoritmo do modo como os objectos são projectados no ambiente.

Na primeira fase do cálculo dos oclusores, os elementos essenciais são os portais e as células, sendo os portais representativos das ruas (elementos na Figura 37 representados em cor cinzento escuro) e as células representativas dos edifícios (elementos na Figura 37 representados em cor amarela).

Figura 37 – Definição dos portais e células

As células visíveis dependem do portal onde se encontra o observador. Para Luebke, as células representam um volume convexo no espaço, limitado por um conjunto de paredes opacas e os portais são a região 2D transparente existente sobre um limite de célula que conecta células adjacentes. O objectivo é determinar, para cada célula, o conjunto de objectos estáticos visíveis a partir do portal. Na segmentação adaptativa, os edifícios representam um polígono que, ao lhe ser projectada uma luz a partir do portal, originam uma sombra (Figura 38).

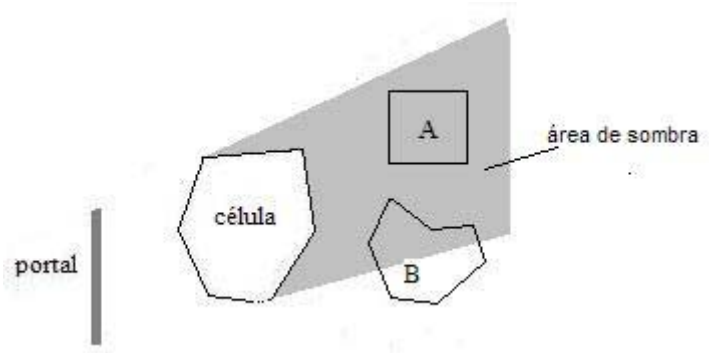


Figura 38 – Representação da sombra originada pela célula

Todos os objectos totalmente incluídos na sombra, são considerados como invisíveis. Os objectos não abrangidos ou parcialmente abrangidos pela sombra são organizados de forma a constituírem novas células.

O processo de caracterização dos portais inicia-se com a definição da sua configuração. Uma rua assume diferentes configurações: 'normal', 'comprida' ou 'circuito' (Figura 39).

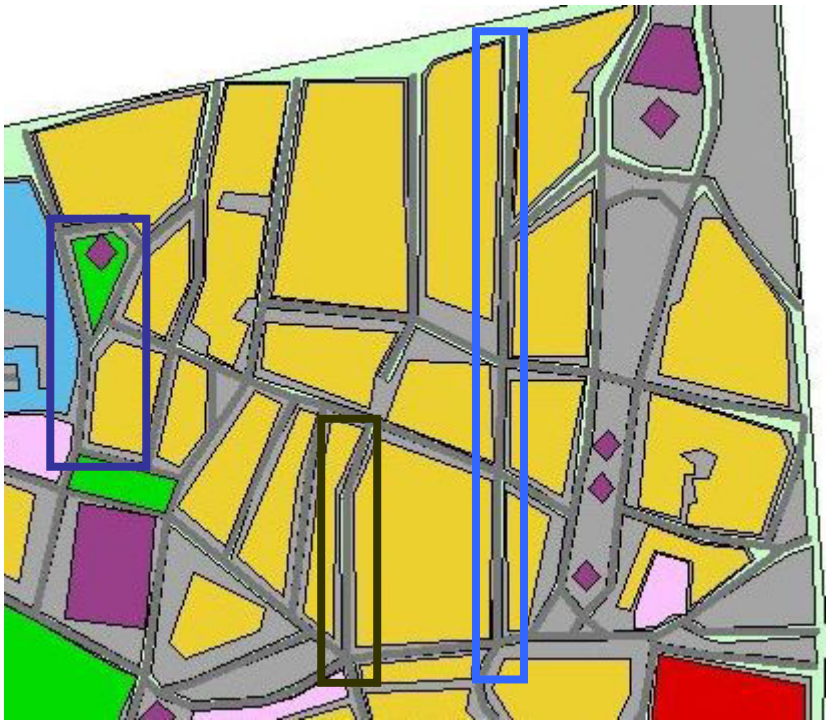


Figura 39 – Exemplo de diferentes configurações de ruas

	Rua definida como 'normal'
	Rua definida como 'circuito'

	Rua definida como 'comprida'

A caracterização de uma rua em diferentes configurações deve-se a dois factos: o primeiro prende-se com a existência de ruas que possuem uma grande extensão e que, consequentemente, possuem um campo de visibilidade também muito extenso. Refira-se no entanto que, neste, se pode reduzir facilmente o número de elementos, visto que uma grande parte se encontra oclusa *à priori*, dada a localização do utilizador e consequente campo de visão, o qual circunda apenas uma área limitada dessa rua. Desta forma, estas ruas são classificadas de 'comprida' de modo a reduzir o número de elementos que constituem o conjunto do campo de visibilidade na segunda fase do cálculo de oclusão.

O segundo facto, deve-se à repetição de coordenadas para as ruas com a forma de rotunda pois, neste caso, o campo de visibilidade não pode ser em função da primeira e última coordenada da rua visto que, na maior parte dos casos, estas coordenadas são idênticas ou não representam o início e o fim de uma rua (Figura 40).

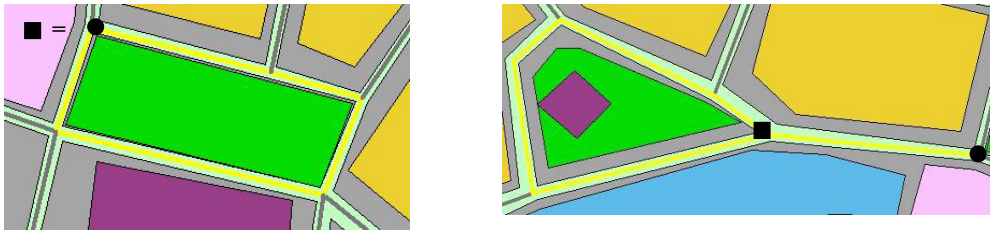


Figura 40 – Exemplo de duas ruas do tipo 'Circuito' e respectivas coordenadas de início e fim da rua

●	**Coordenada de inicio da rua**

■	**Coordenada de fim da rua**

Para estes casos, o campo de visibilidade deve ser calculado em função da coordenada maior e menor da rua relativamente ao plano XY e ao oclusor (Figura 41).

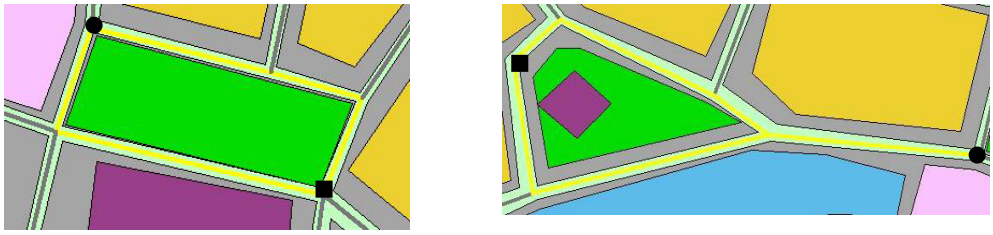


Figura 41 – Alteração das coordenadas de início e fim da rua em ruas do tipo 'Circuito'

●	**Coordenada de inicio da rua**

■	**Coordenada de fim da rua**

A caracterização dos portais engloba, para além da sua configuração, a sua orientação. Um portal pode ter uma orientação horizontal ou vertical (Figura 42). Um portal denomina-se na horizontal quando o ângulo formado pelo portal com o eixo dos X é

inferior a 45º. Quando o ângulo formado pelo portal com o eixo dos Y é superior a 45º, o portal possui uma orientação vertical.

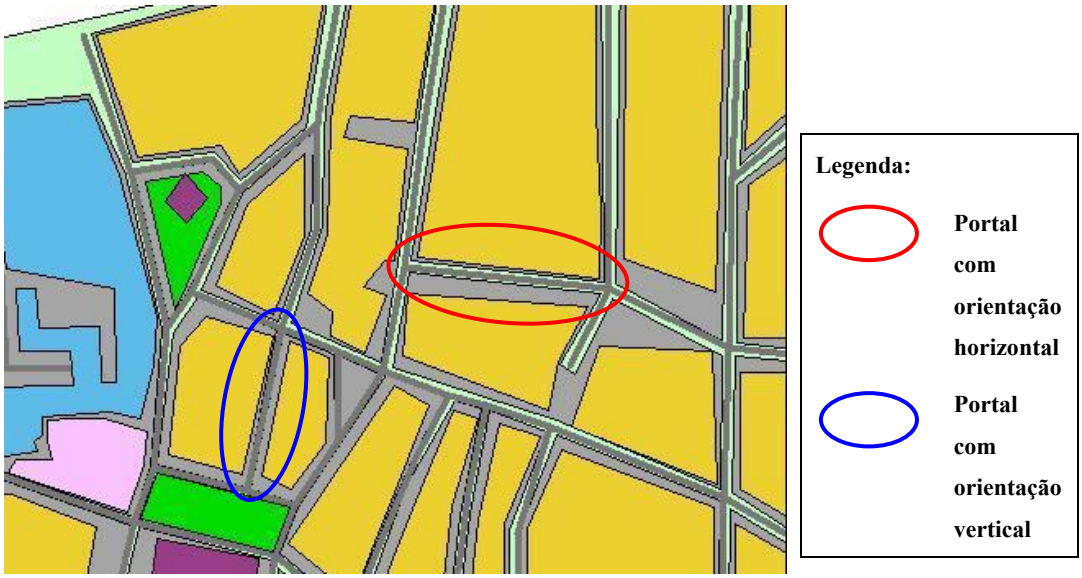
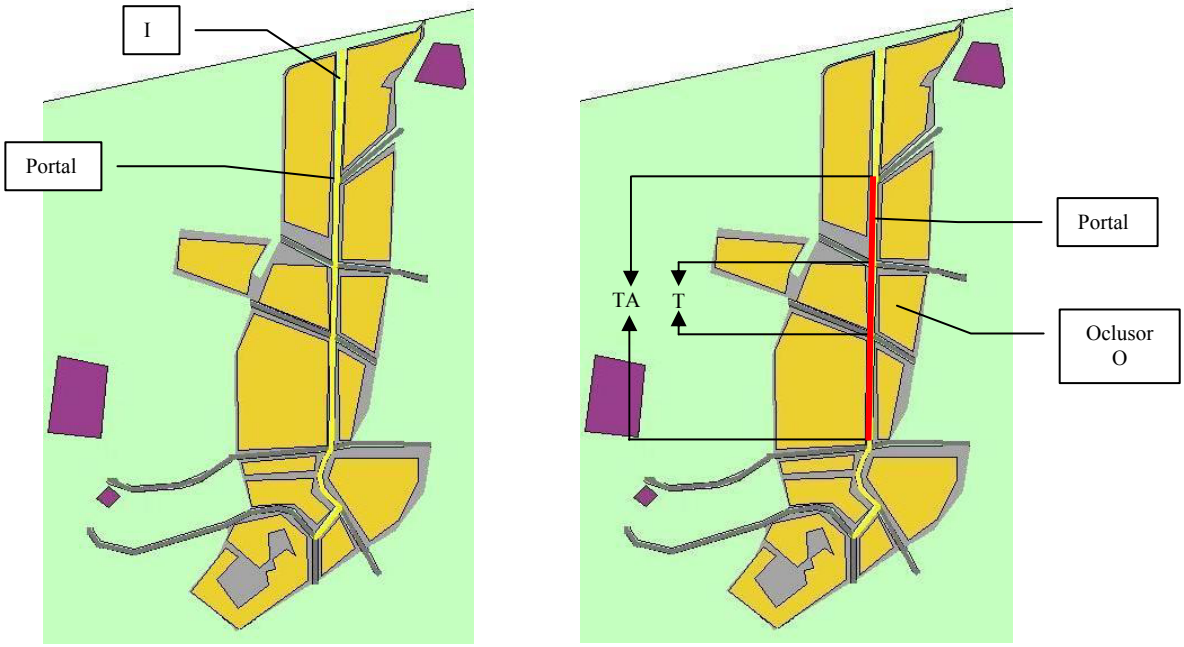


Figura 42 – Orientação de um portal

Do processo de caracterização dos portais resulta a definição das respectivas dimensões, tendo em conta a configuração da rua.

Para as ruas do tipo 'comprida', o portal é restringido em função da divisão da rua pelos cruzamentos existentes com outras ruas e o oclusor. Esta restrição do portal é definida com o intuito de tornar a fase de cálculo da área de sombra mais realista. Na realidade, se o observador estiver localizado por exemplo no ponto I da rua apresentada na Figura 43(a) não consegue visualizar os edifícios que se encontram perto do outro extremo da rua. Nestes casos, quando a rua toda é definida como portal, o resultado final do cálculo de oclusão resulta em poucos edifícios oclusos. Deste modo, para tornar o modelo de cidade mais autêntico, o portal deve ser o mais aproximado possível do campo de visão do observador. A Figura 43 representa o campo de visibilidade para uma rua caracterizada de 'comprida'. A Figura 43(a) apresenta o portal inicialmente definido (portal definido em cor amarela), enquanto que a Figura 43(b) apresenta o portal restringido em função da localização do oclusor (portal definido em cor vermelha). O troço aumentado (TA) define-se em função do troço (pedaço da rua limitado por dois cruzamentos) referente ao oclusor (T), sendo-lhe adicionado, em cada um dos seus sentidos, o troço seguinte.



(a) **(b)**

Figura 43 – Elementos que constituem o campo de visibilidade de uma rua do tipo 'Comprida'

T	**Troço da rua referente ao oclusor O**

TA	**Troço aumentado para o oclusor O**

Para o caso das ruas classificadas como 'Circuito' o portal é definido em função das coordenadas da rua e da posição do oclusor relativamente à rua. Para o exemplo apresentado na Figura 44, o oclusor encontra-se numa posição vertical logo, as coordenadas da rua devem ser a de maior e menor valor em Y. No caso do oclusor ocupar uma posição horizontal, então as coordenadas da rua devem ser a de maior e menor valor em X.

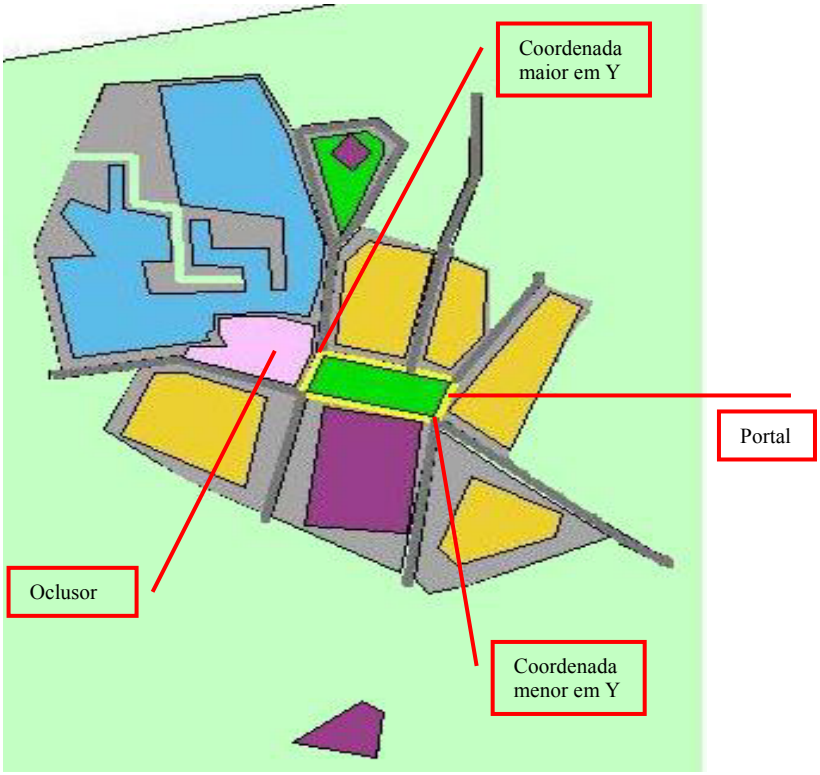


Figura 44 – Definição do portal para uma rua do tipo 'circuito'

Considerando as dimensões do portal e os elementos associados à respectiva rua, determinam-se as células (edifícios oclusores) com base na proximidade: os edifícios mais oclusores são aqueles que se encontram mais perto da rua. Desta forma, é possível eliminar edifícios que se encontram totalmente atrás do oclusor, sem ser necessária a determinação da sombra que projectam. Tendo por base, na Figura 33, o campo de visibilidade da rua R2, definido na secção 4.4, e a caracterização da rua com uma configuração normal e uma orientação vertical determina-se a lista de proximidade para os edifícios. Esta lista de proximidade consiste em ordenar os edifícios do conjunto do campo de visibilidade da rua R2 em função da distância entre a rua e o edifício.

Para terminar a fase de definição dos portais e células, a célula definida como oclusor deve ser caracterizada. A caracterização de uma célula consiste em determinar as coordenadas do oclusor que se encontram "de frente" para a rua (ListFrente) isto é, quais as coordenadas do edifício que serão utilizadas para a construção da área de sombra na segunda fase do cálculo de oclusão. Estas coordenadas são designadas de "de frente" para a rua porque são as coordenadas do oclusor visíveis a partir de cada uma das coordenadas da rua. As Figura 45 eFigura 46 apresentam, para a rua R2, definida anteriormente, as coordenadas do oclusor que se encontram "de frente" para a rua.

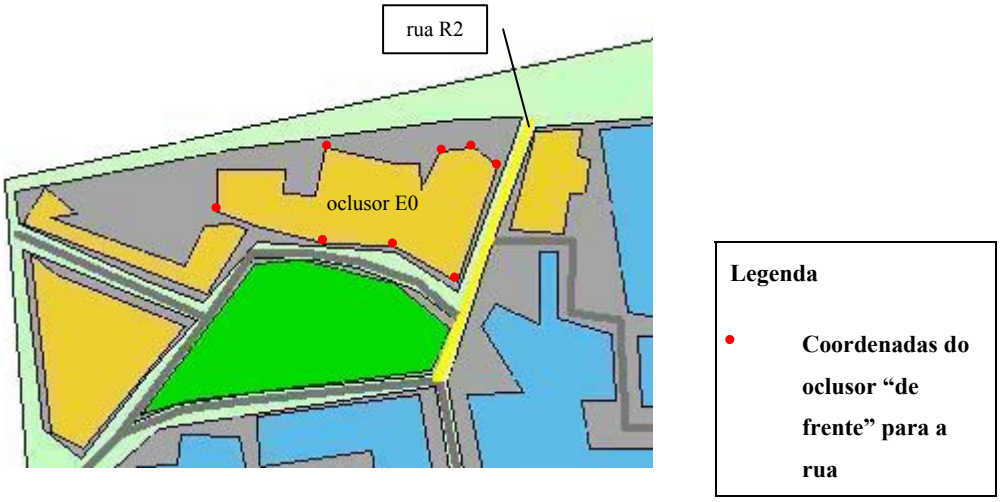


Figura 45 – Definição das coordenadas do oclusor E0 "de frente" para a rua

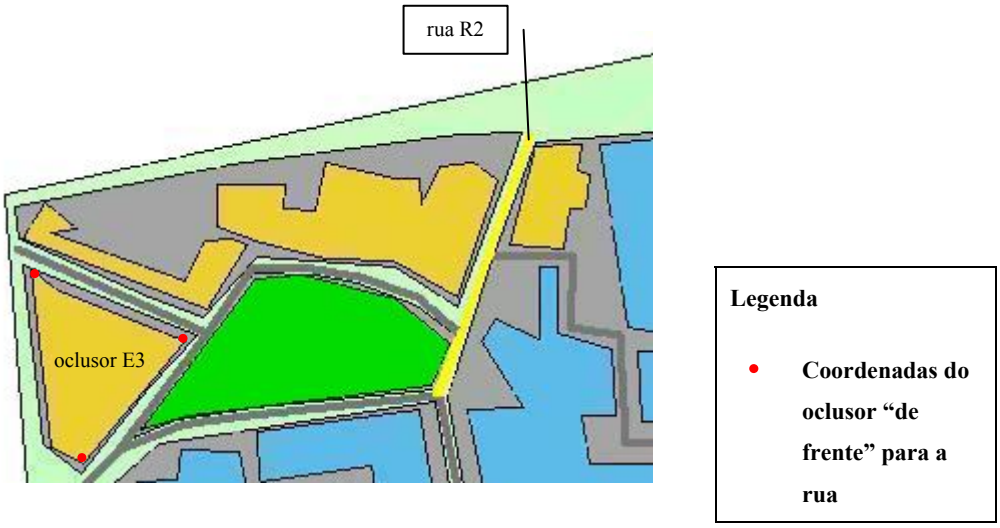


Figura 46 – Definição das coordenadas do oclusor E3 "de frente" para a rua

4.5.2 - Cálculo da Área de Sombra

Na segunda fase e para cada oclusor, calcula-se a respectiva área de sombra em função da rua [Hudson97]. A área de sombra é definida a partir do portal, passando pelo oclusor (Figura 38). Esta área (ASf – Área de Sombra final) é o resultado da união entre as duas áreas de sombra (AS1 e AS2) obtidas em função de cada uma das extremidades da rua, A1 (Figura 49) e A2 (Figura 52) respectivamente, com as coordenadas do oclusor "de frente" para a rua que se encontram armazenadas na lista *ListFrente* (Figura 46). As áreas de sombra AS1 e AS2 são o resultado da união de dois quadriláteros (Q1 e Q2), sendo Q1 constituído pela maior e menor coordenada em X, e a maior e menor coordenada em Y da lista de coordenadas *ListCoordenadas* (Figura 47 referente a AS1 e Figura 50 referente a AS2). A lista *ListCoordenadas* contém a lista *ListFrente* (nas Figura 47, Figura 48,

Figura 50 e Figura 51 as coordenadas F0, F1 e F2) e o conjunto CI constituído pelas coordenadas resultantes da intersecção das semi-rectas que passam na extremidade da rua (A1 ou A2) e na lista *ListFrente* com os segmentos de recta que formam a superfície do modelo de cidade (coordenadas I0, I1 e I2). O quadrilátero Q2 é constituído pela segunda maior e menor coordenada em X, e a segunda maior e menor coordenada em Y da lista de coordenadas *ListCoordenadas*.

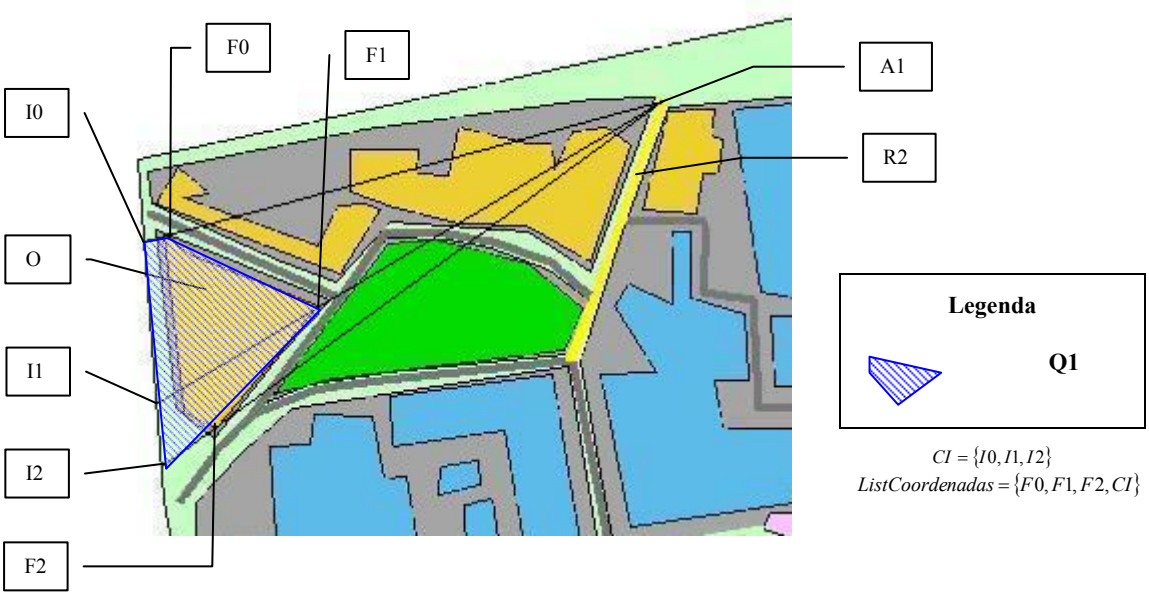


Figura 47 – Quadrilátero Q1 para a rua R2 e o oclusor (O)

Figura 48 - Quadrilátero Q2 para a rua R2 e o oclusor (O)

A reunião do quadrilátero Q1 com o quadrilátero Q2 para a extremidade A1 da rua R2 resulta na área de sombra AS1 apresentada na Figura 49.

Figura 49 - Área de sombra para a coordenada A1 da rua R2 e o oclusor (O)

A Figura 50 apresenta o quadrilátero Q1 resultante do mesmo processo da Figura 47 mas para a extremidade A2 da rua R2 isto é, a maior e menor coordenada em X, e a maior e menor coordenada em Y da lista de coordenadas *ListCoordenadas*.

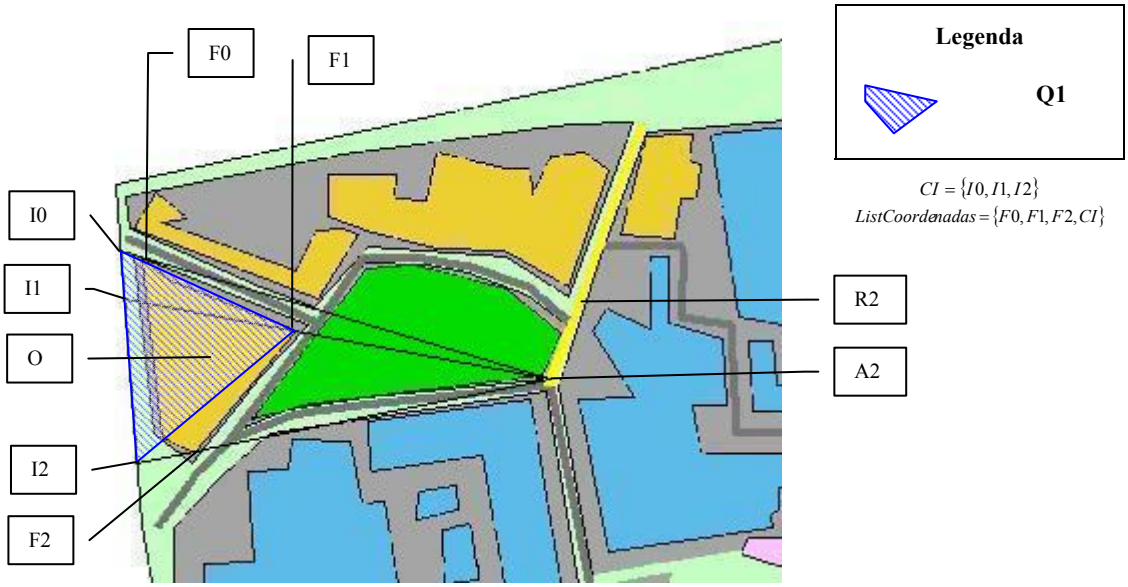


Figura 50 – Quadrilátero Q1 para a rua R2 e o oclusor (O)

O quadrilátero Q2 (Figura 51) para a extremidade A2 da rua R2 é obtido através do mesmo processo da Figura 48 isto é, a segunda maior e menor coordenada em X, e a segunda maior e menor coordenada em Y da lista de coordenadas *ListCoordenadas*.

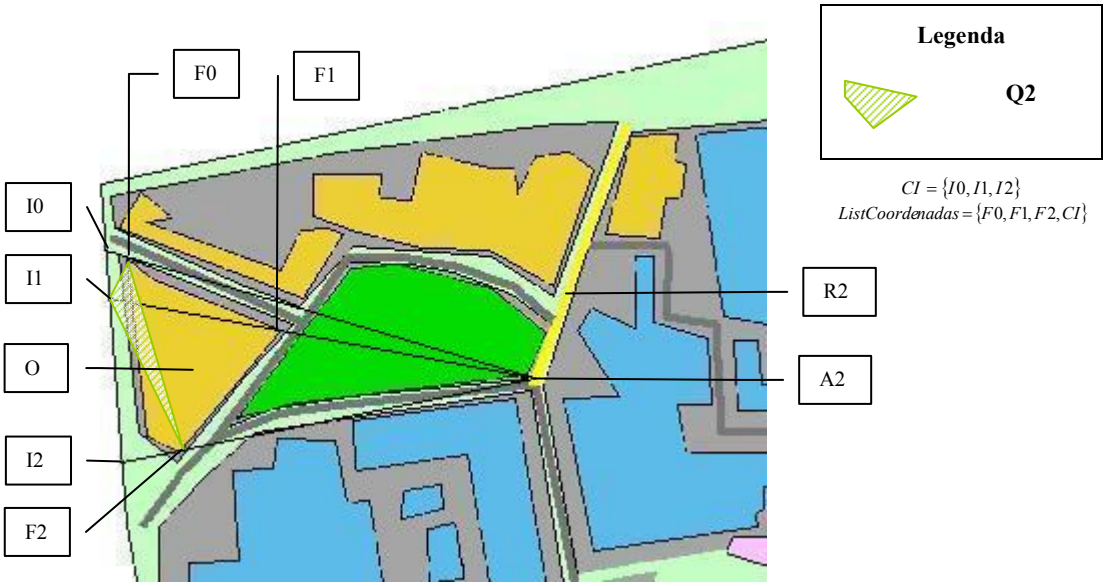


Figura 51 – Quadrilátero Q2 para a rua R2 e o oclusor (O)

A área de sombra AS2 (Figura 52) resulta da reunião do quadrilátero Q1 obtido na Figura 50 com o quadrilátero Q2 obtido na Figura 51.

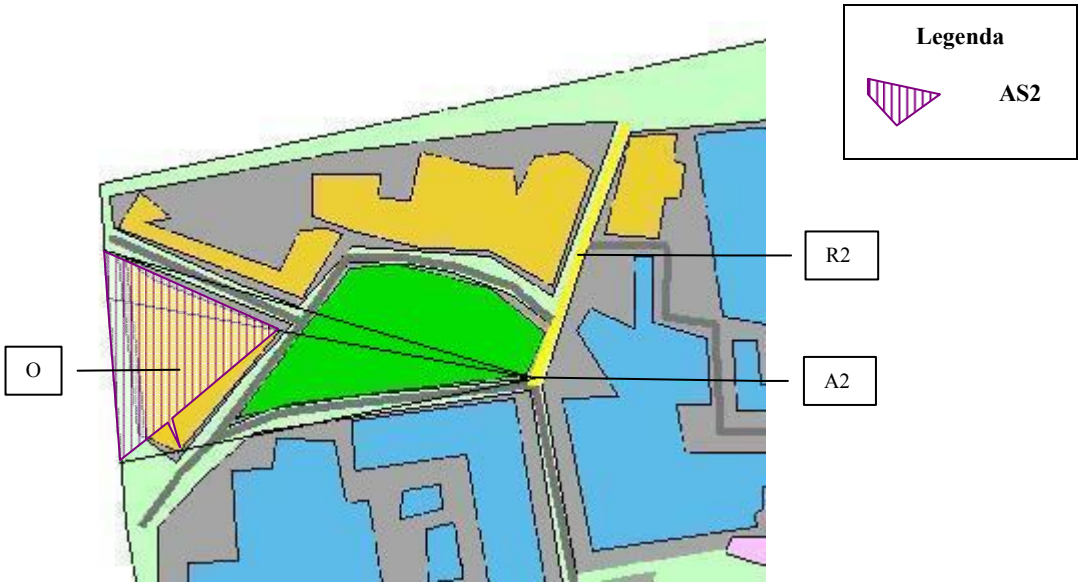


Figura 52 - Área de sombra para a coordenada A2 da rua R2 e o oclusor (O)

A Figura 53 mostra a área de sombra final (ASf): $ASf = AS1 \cup AS2$

Figura 53 - Área de sombra para a rua R2 e o oclusor (O)

O cálculo da área de sombra para uma rua classificada de 'comprida' (Figura 54) é determinado tendo em conta o oclusor (O), para o qual se pretende calcular a área de sombra, e o troço (TA) definido como portal. A determinação da área de sombra para este tipo de rua segue o mesmo processo aplicado a uma rua classificada de "Normal". Inicia-se com o cálculo da área de sombra (AS1) relativamente a uma das extremidades do troço aumentado (TA), de seguida calcula-se a área de sombra (AS2) relativamente à outra extremidade do troço aumentado (TA). A área de sombra final (ASf) resulta da união da área de sombra AS1 com AS2.

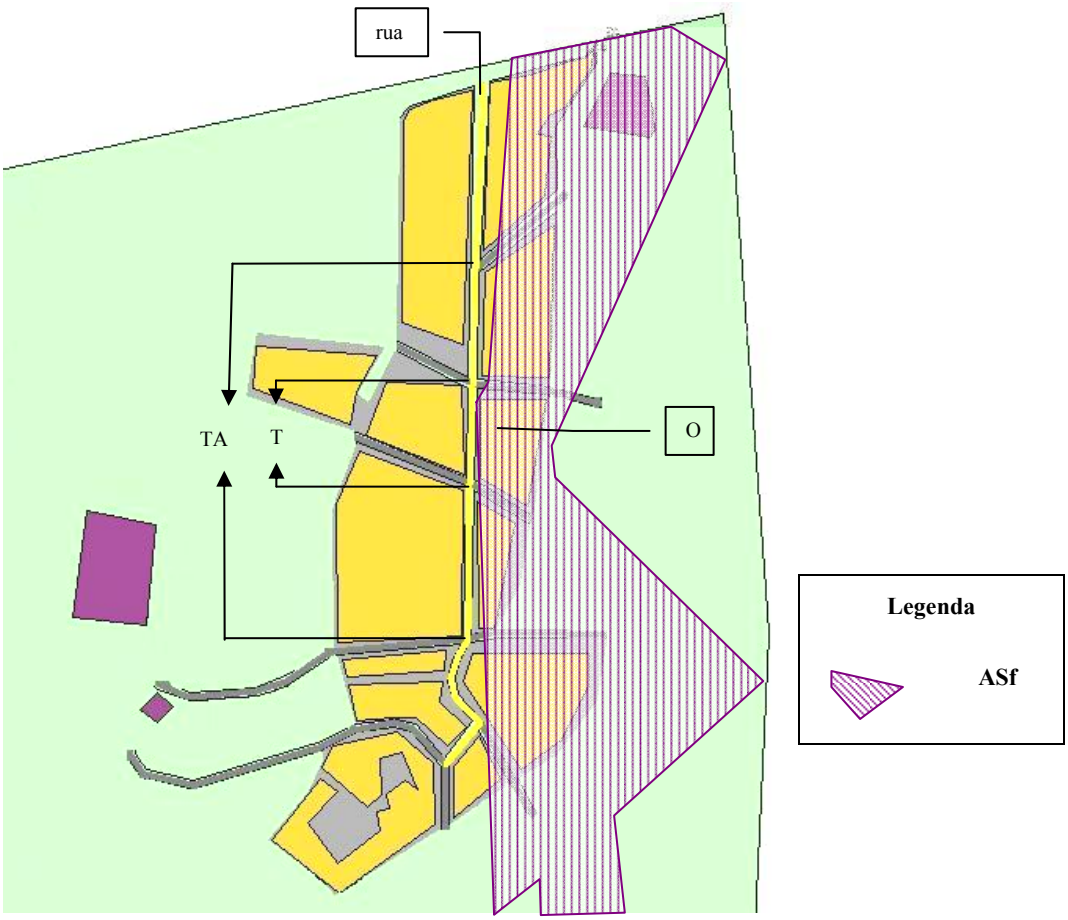


Figura 54 – Área de sombra definida para uma rua do tipo 'comprida' e o oclusor (O)

T	**Troço da rua referente ao oclusor O**

TA	**Troço aumentado para o oclusor O**

Para o caso das ruas classificadas como 'circuito' (Figura 55) o cálculo da área de sombra é determinado em função do oclusor (O), para o qual se pretende calcular o plano de sombra, e das coordenadas da rua, as quais dependem da posição do oclusor relativamente à rua (neste caso as coordenadas da rua são a de maior e menor valor em Y). A determinação da área de sombra para uma rua do tipo 'circuito' é idêntica ao aplicado a uma rua do tipo "normal". Para o exemplo apresentado na Figura 55 calcula-se a área de sombra (AS1) relativamente à coordenada da rua com maior valor em Y, seguindo-se o cálculo da área de sombra (AS2) relativamente à coordenada da rua com menor valor em Y. A área de sombra final (ASf) resulta da união da área de sombra AS1 com AS2.

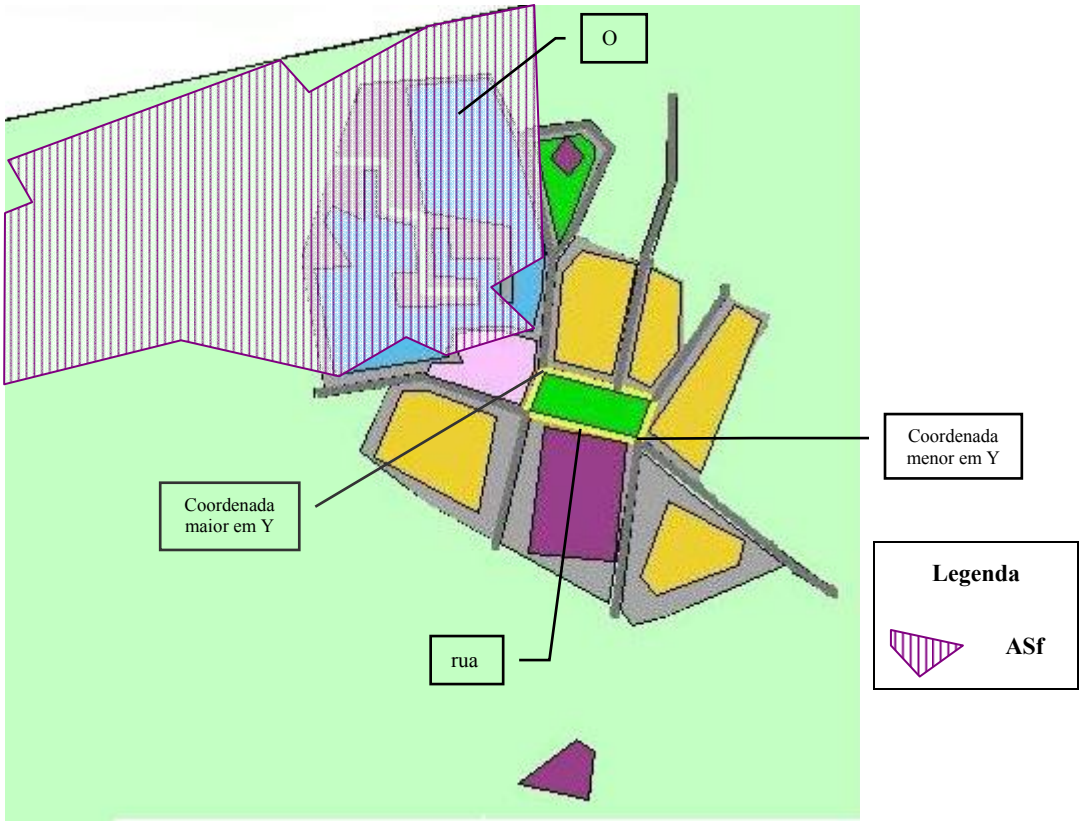


Figura 55 – Área de sombra definida para uma rua do tipo 'Circuito' e o oclusor (O)

Com base na área de sombra final (ASf), removem-se todos os edifícios contidos nesta, dado encontrarem-se fora do campo de visibilidade. Para o exemplo representado na Figura 38, o edifício a ser removido é o A. O edifício B é dividido em dois polígonos: um polígono formado pela área que se encontra dentro da área de sombra, e um segundo polígono constituído pela área que se encontra fora da área de sombra. O primeiro polígono não é visível a partir do portal e, desta forma, não é representado. O segundo polígono fica guardado como polígono visível e ser-lhe-á aplicado o cálculo de oclusão no papel de oclusor, se entretanto, não for removido por outro oclusor.

A remoção de elementos contidos na área de sombra não envolve a averiguação do tipo de elemento nem o respectivo peso na caracterização da cidade. Desta forma, o passo seguinte implica a verificação, nos elementos removidos, da existência de algum edifício com interesse público ou algum edifício com uma altura superior à média de modo, a ser novamente incluído nos elementos visíveis. Relativamente aos edifícios com uma altura superior à média e que foram removidos da lista de elementos visíveis estes passam por um outro processo de modo a garantir a sua visibilidade. Desta forma verifica-se qual a altura mínima (AltMin – Altura Mínima) do conjunto de edifícios (edifícios E4 e EI0 da Figura 56) que se encontram posicionados entre o edifício com uma altura superior à média (EAlt – Edifício Alto: edifício EI3 da Figura 56) e a rua (rua R4 da Figura 56). O

edifício EAlt é incluído na lista de elementos visíveis mas a sua representação inicia-se em AltMin até à sua altura máxima.

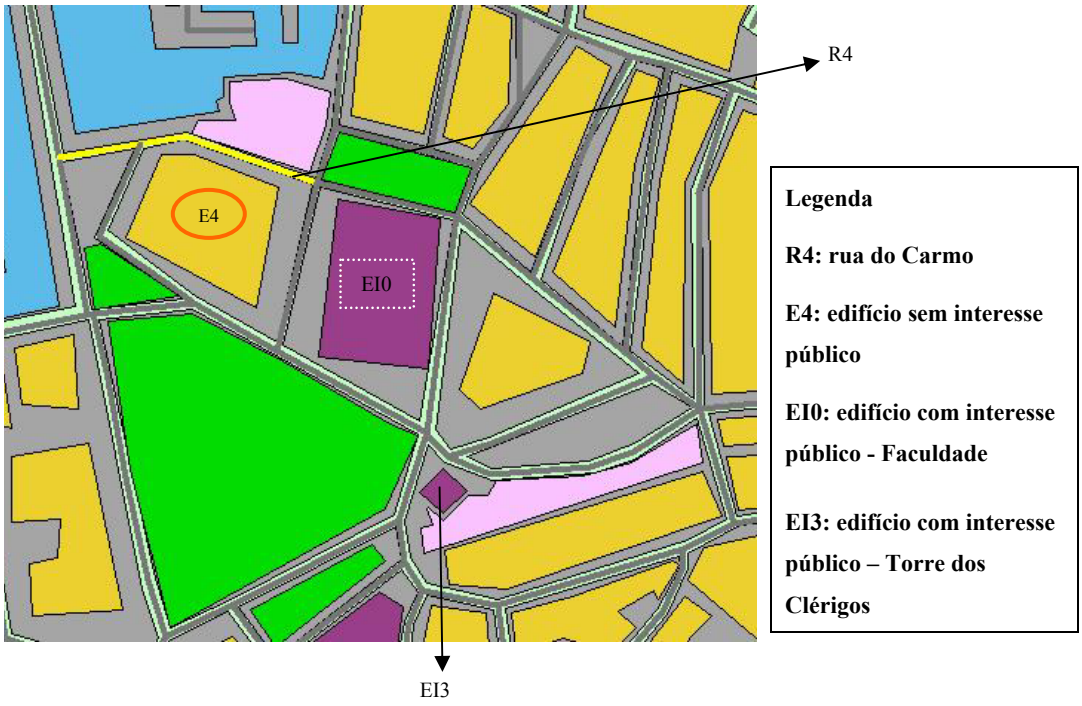


Figura 56 – Edifícios posicionados entre a rua R4 e o edifício EI3 com uma altura superior à média

No final do cálculo de visibilidade para o portal são obtidos os polígonos visíveis e que, consequentemente são incluídos para posterior visualização.

4.6 - Síntese do Capítulo

A segmentação de ambientes virtuais urbanos é de vital importância pois permite a simplificação da complexidade da sua representação, através da divisão do ambiente original em ambientes menores. Existem vários tipos de segmentação, sendo necessário observar o tipo de modelo e navegação que se pretende, assim como as características relativas à rede na qual é feita a transmissão da informação.

A partir dos trabalhos desenvolvidos por vários autores, e considerando as características de um ambiente urbano, foi definida uma metodologia para a segmentação de uma cidade em função da localização e campo de visão do utilizador, utilizando uma navegação em modo "passeio". A determinação dos objectos visíveis para uma determinada localização do utilizador implica várias fases. Uma primeira fase que inclui a determinação dos edifícios que originam maior oclusão e uma segunda fase onde é construída a respectiva sombra, permitindo desta forma eliminar os objectos que lhe estão contidos.

CAPÍTULO 5

SEGMENTAÇÃO DE UM MODELO CONCRETO: A CIDADE DO PORTO

Índice do Capítulo

5 - Segmentação de um Modelo Concreto: a cidade do Porto

A metodologia desenvolvida para a segmentação de ambientes urbanos virtuais que se descreve no capítulo 4 levou ao desenvolvimento de um protótipo com vista à sua avaliação.

Neste capítulo são descritos os requisitos definidos para o protótipo (secção 5.1), a respectiva arquitectura (secção 5.2), e as tecnologias utilizadas (secção 5.3). A classificação dos dados em temas num *software* de GIS, é descrita na secção 5.4 e, sendo a segmentação e cálculo de oclusão um aspecto fundamental em todo o processo de desenvolvimento desta dissertação, a secção 5.5 descreve o seu modo de implementação. Como resultado de todo este processo são obtidos os ficheiros VRML para posterior conversão em X3D (secção 5.6), e a última secção engloba o método utilizado na construção da *interface* e interacção com o utilizador.

5.1 - Especificação

O objectivo desta dissertação, como já foi referido no capítulo 1, é desenvolver uma metodologia de segmentação que, de uma forma dinâmica, possibilite a visualização de ambientes urbanos virtuais extensos, em que a informação apresentada seja gerada automaticamente e em função da localização do utilizador, para uma navegação feita em modo de "passeio". Com base neste objectivo e como forma de concretização para efeitos de teste e avaliação, foi necessário criar um ambiente virtual de um espaço urbano específico, no caso correspondente a uma área limitada da cidade do Porto (protótipo). Este protótipo assenta numa segmentação dinâmica, em função da localização do utilizador, de forma a que o ambiente virtual possa ser dividido em diversos ficheiros, possibilitando o rápido envio e recepção destes. O protótipo permite ainda uma navegação 3D interactiva na forma de "passeio", no qual todos os elementos visíveis são

definidos dinamicamente, permitindo a remoção de objectos que estão potencialmente ocultos, resultando em ficheiros que ocupam menos espaço. Desta forma o utilizador pode navegar pelo ambiente virtual 3D sem ter que esperar que este seja totalmente carregado. À medida que o utilizador entra numa nova área, um novo ficheiro é lido, substituindo o anterior.

A primeira fase para a construção do protótipo consiste em definir uma área da cidade do Porto para ser modelada, em virtude desta cidade conter uma grande superfície e consequentemente tornar-se bastante moroso, não trazendo benefícios concretos para esta dissertação. A razão da zona seleccionada deve-se à grande diversidade de elementos como edifícios, jardins e monumentos, mas também à existência de espaços abertos, como é o caso da Avenida da Liberdade e da Praça da República. Na Figura 57 apresenta-se a região da cidade do Porto sobre a qual se desenvolveu o protótipo.

Figura 57 – Área da cidade do Porto seleccionada para o protótipo

5.2 - Arquitectura do Modelo Implementado

Em função da especificação descrita na secção anterior, a arquitectura do protótipo é apresentada na Figura 58.

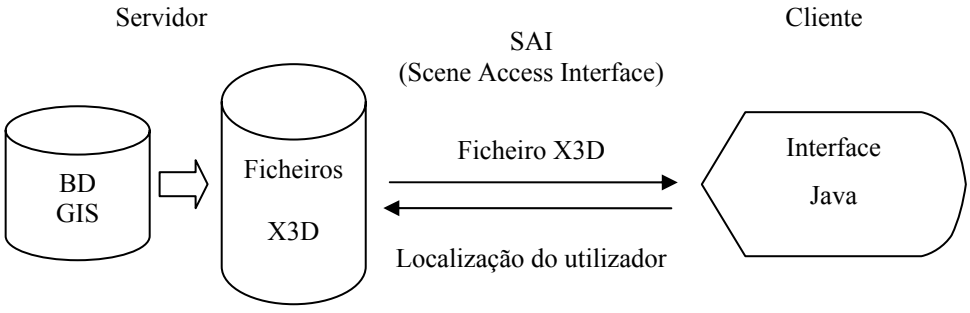


Figura 58 – Arquitectura do protótipo desenvolvido

O modelo implementado baseia-se numa arquitectura cliente-servidor. A base de dados armazena a informação geográfica utilizada para a modelação onde, por exemplo, cada edifício é representado pelo polígono projectado no terreno, um valor com a sua altura e com uma textura pré-definida. Os ficheiros X3D produzidos dinamicamente e específicos para cada rua são invocados à medida que o cliente navega pelo ambiente virtual urbano e consequentemente enviados ao cliente.

A *interface* do cliente é uma parte essencial. Dado que a aplicação se destina a qualquer utilizador, a *interface* tem de ser intuitiva e simples, de modo a que o utilizador seja capaz de interagir naturalmente. A *interface* (Figura 59) é composta por duas partes: uma janela 3D onde se visualiza a cena tridimensional e uma outra onde o utilizador constata, sobre um mapa, qual a rua em que se encontra.

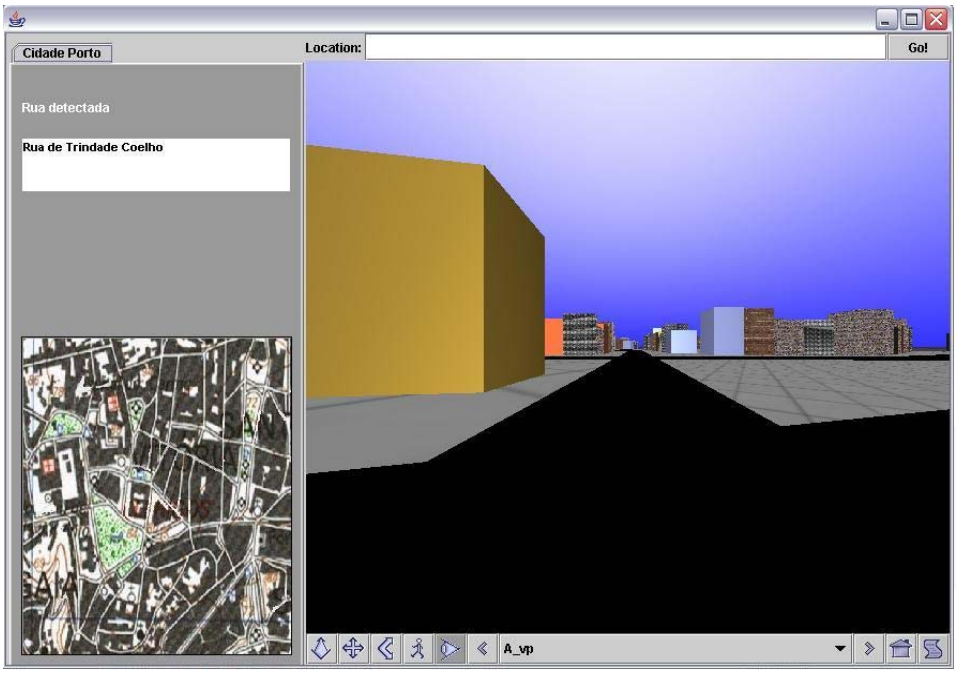


Figura 59 – *Interface* do protótipo

5.3 - Tecnologias

Como foi referido na secção 5.2, o protótipo possui uma base de dados para armazenar toda a informação geográfica utilizada para a modelação. O *software* utilizado na construção da base dados foi o *ArcView GIS* versão 3.1 [ArcView], visto ser necessário armazenar e analisar informação geográfica referente a cada um dos componentes. Este programa permite a gestão, criação e organização de dados geográficos e alfanuméricos, para além da visualização de dados, sua inquirição e análise.

A partir do *software* de GIS são gerados ficheiros com os modelos do ambiente virtual em VRML, que são posteriormente convertidos em X3D utilizando o programa Vizx3D Web3D versão 1.2.1 [Vizx3D], o qual permite criar e visualizar modelos 3D mas também aplicar animações.

A tecnologia utilizada na comunicação cliente-servidor para o envio e recepção dos ficheiros X3D é a SAI (*Scene Access Interface*).

A *interface* com o utilizador foi implementada com base na tecnologia Java, utilizando o programa Eclipse [Eclipse] para ser visualizada com o *browser* Xj3D [Xj3D].

5.4 - Classificação dos Dados em Temas

Tendo por base a área definida da cidade do Porto, foram criados num GIS diversos temas em função da categoria em que se insere o componente: Superfície, Cidade_ruas, Cidade_quarteirão, Edifícios, Cidade_jardins, Cidade_edificiosimp, Cidade_estacao, Cidade_linhacomboio, Cidade_igrejas e Cidade_hospital, dando origem cada um deles à respectiva tabela (Figura 60). Foi necessário definir temas distintos para os edifícios (Edifícios e Cidade_edificiosimp), visto que o primeiro tema diz respeito aos edifícios comuns que existem na cidade, enquanto que o segundo tema armazena os monumentos e restante património cultural.

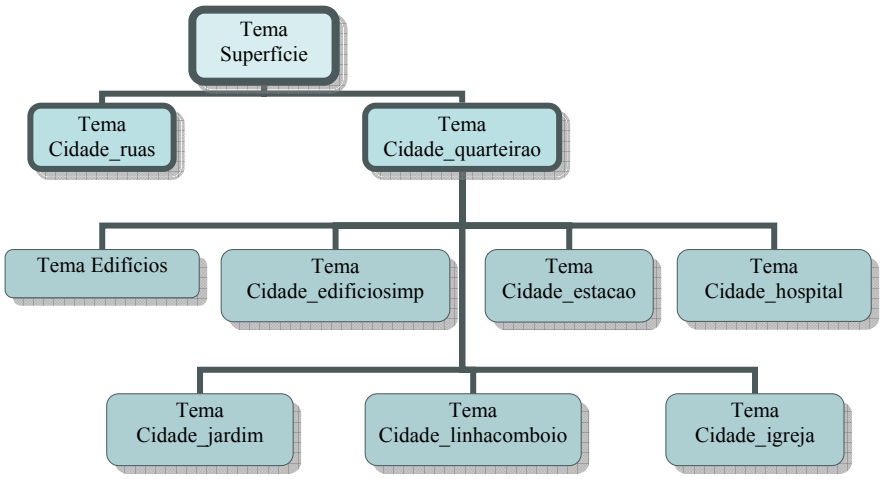


Figura 60 – Hierarquia dos temas definidos no protótipo

O tema Superfície define a área do projecto, construída através de um polígono, assim como os temas Cidade_quarteirão, Edifícios, Cidade_jardins, Cidade_edificiosimp, Cidade_estacao, Cidade_igrejas e Cidade_hospital. O tema referente às ruas e linha de

comboio utilizam como tipo a linha. Na Figura 61 são apresentados os diversos temas e um exemplo da representação de alguns deles.

Figura 61 – Temas definidos no protótipo e exemplo da sua representação

O tema Edifícios foi delineado em função da área edificada para cada um dos quarteirões, sendo esta área posteriormente definida em prédios e/ou casas (Figura 62).

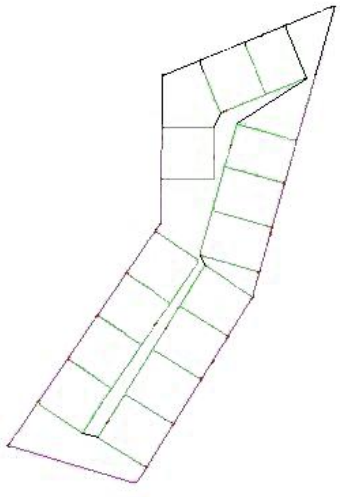


Figura 62 – Exemplo de, no tema Edifícios, uma área dividida em diversos prédios/casas

5.5 - Segmentação e Cálculo de Oclusão

Após a definição da informação geográfica essencial ao protótipo, como foi referido na secção 5.4, é necessário ter em consideração a segmentação a aplicar (Figura 63) e consequente oclusão.

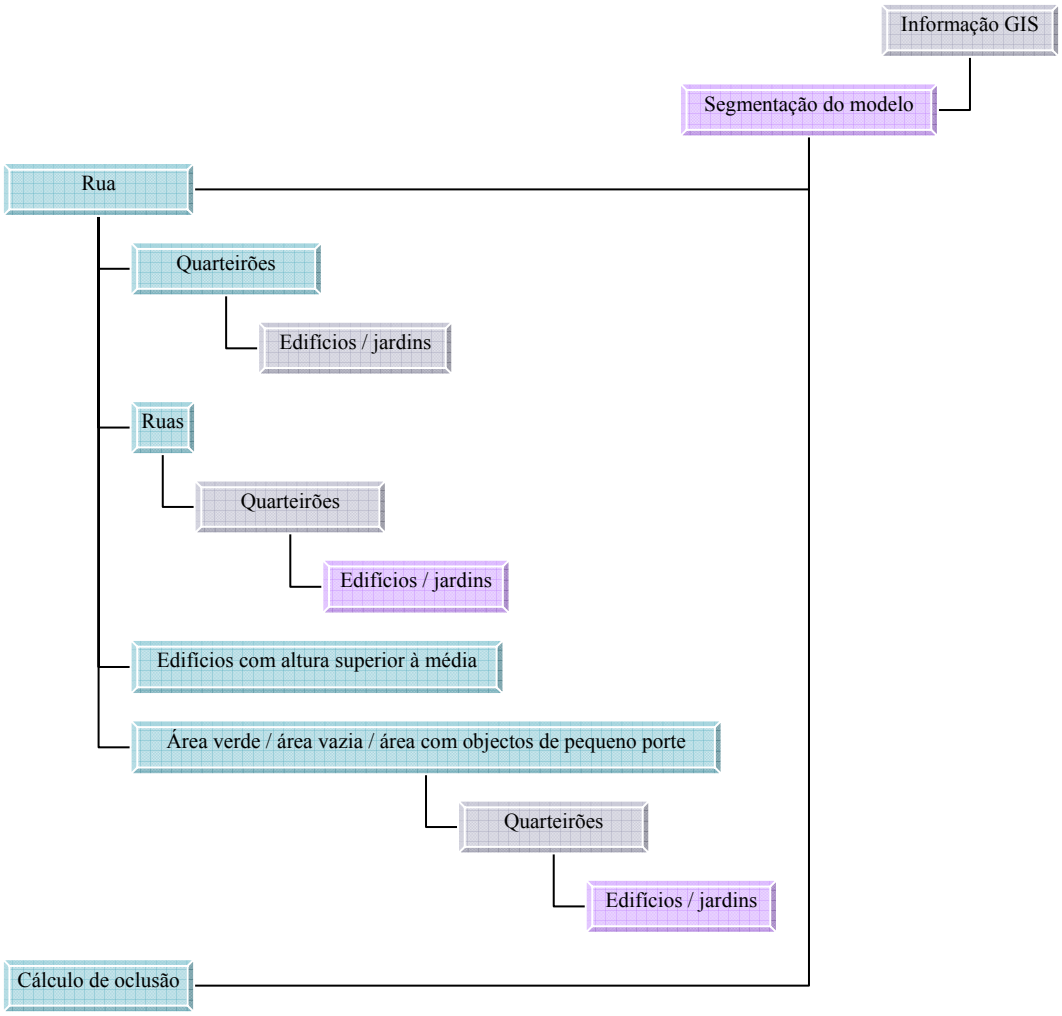


Figura 63 – Esquematização da segmentação de um modelo de cidade

Como foi referido na secção 4.3, a segmentação aplicada ao modelo foi estabelecida em função das ruas, visto o utilizador se deslocar através delas. Inicialmente é determinado, para cada rua, quais os quarteirões que a circundam e, para estes, os edifícios ou jardins que lhe pertencem. O cálculo dos quarteirões que pertencem a cada rua pressupõe a verificação da rua definida como um segmento de recta ou um conjunto de segmentos de recta. Para o primeiro caso verifica-se quais os quarteirões que intersectam o segmento de recta que constitui a rua (Listagem 1) e para cada extremidade quais os quarteirões que intersectam num raio pré-definido (Listagem 2). Para o segundo caso o processo é idêntico mas ocorre para cada segmento de recta que constitui a rua.

```
Variáveis:
   i é uma variável auxiliar
   vectUx, vectUy é o vector em X e Y da rua
   pCX, pCY, uCX, uCY é a primeira e última coordenada da rua
   Mx, My, Nx, Ny, Rx, Ry, Sx, Sy coordenadas auxiliares para a rua
   poliRua é um polígono

seleccionar 1ª (pCX,pCY) e última (uCX,uCY) coordenada da rua
{determinar vector U formado por essa recta}
   vectUx = uCX - pCX
   vectUy = uCY - pCY
determinar vector V perpendicular ao U com norma de 20metros
determinar para a 1ª coordenada (P) da rua e direcção do vector V, as coordenadas
em X e Y para os pontos M e N
{formar os pontos M e N}
   M = Mx@My
   N = Nx@Ny
determinar para a última coordenada (U) da rua e direcção do vector V, as
coordenadas em X e Y para os pontos R e S
{formar os pontos R e S}
   R = Rx@Ry
   S = Sx@Sy
formar polígono com as coordenadas (M,P,N,S,U,R): poliRua
{verificar se o polígono intersecta algum quarteirão}
for i = 0 to número de quarteirões
   if poliRua intersecta quarteirão then
      armazenar quarteirão
```

Listagem 1 – Cálculo dos quarteirões que intersectam a rua

```
Variáveis:
   i é uma variável auxiliar
   P, U é a primeira e última coordenada da rua
   circuloP, circuloU é o círculo para a 1ª e última coordenada da rua

{determinar círculo para cada extremidade da rua}
   circuloP = Circle.Make (P, 40)
   circuloU = Circle.Make (U, 40)
{verificar se o círculo intersecta algum quarteirão}
   for i = 0 to número de quarteirões
      if circuloP intersecta quarteirão then
         armazenar quarteirão
      if circuloU intersecta quarteirão then
         armazena quarteirão
```

Listagem 2 – Cálculo dos quarteirões que intersectam cada extremidade da rua num raio pré-definido

A verificação para cada quarteirão dos elementos que o intersectam inclui o controlo às seguintes tabelas: Edifícios, Cidade_edificiosimp, Cidade_estação, Cidade_hospital, Cidade_igrejas, Cidade_jardins e Cidade_LinhaComboio (Listagem 3).

```
Varáveis
   i, j são variáveis auxiliares

{verificar para cada quarteirão que intersecta a rua quais os elementos que lhe
pertencem{
for i = 0 to número de quarteirões
   {verificar se o quarteirão intersecta algum edifício sem interesse público}
   for j = 0 to número de edifícios sem interesse público
      if edifício intersecta quarteirão then
         armazena edifício
   {verificar se o quarteirão intersecta algum edifício com interesse público}
   for j = 0 to número de edifícios com interesse público
      if edificioIntPublico intersecta quarteirão then
         armazena edificioIntPublico
   verificar se o quarteirão intersecta alguma estação de comboio
   verificar se o quarteirão intersecta algum hospital
   verificar se o quarteirão intersecta alguma igreja
   verificar se o quarteirão intersecta algum jardim
   verificar se o quarteirão intersecta alguma linha de comboio
```

Listagem 3 – Cálculo dos elementos que intersectam um quarteirão

O processo de segmentação engloba ainda a inclusão de objectos que não pertencem aos elementos associados a uma determinada rua, mas que possuem uma altura superior à média e que se encontram localizados num perímetro pré-definido. Desta forma, é necessário verificar, nas tabelas referentes aos edifícios com interesse público, das estações de comboio, dos hospitais e das igrejas, se existe algum elemento com uma altura superior à média dos restantes elementos. Para estes edifícios verifica-se, para uma área pré-definida, quais as ruas que intersectam essa área, sendo imediatamente associadas ao respectivo edifício (Listagem 4).

```
Varáveis
   i, j são variáveis auxiliares
   altura é a altura de um edifício
   centro é a coordenada do centro do edifício
   círculo é uma área pré-definida

{determinar se existe algum edifício com interesse público com uma altura superior
à média}
for i = 0 to número de edificios com interesse público
   if altura >= 30 then
      {determinar uma área pré-definida para o edifício encontrado}
      circulo = Circle.Make (centro, 500)
      {verificar quais as ruas que intersectam a área pré-definida}
      for j = 0 to número de ruas
         if rua intersecta área then
            associar rua encontrada ao edifício com altura superior à média
determinar se existe alguma estação de comboio com uma altura superior à média e
para a estação encontrada determinar as ruas que intersectam a respectiva área
pré-definida
determinar se existe algum hospital com uma altura superior à média e para o
hospital encontrado determinar as ruas que intersectam a respectiva área pré-
definida
determinar se existe alguma igreja com uma altura superior à média e para a igreja
encontrada determinar as ruas que intersectam a respectiva área pré-definida
```

Listagem 4 – Cálculo dos edifícios com uma altura superior à média e respectivas ruas associadas

Quando na segmentação existe uma área verde, uma área vazia ou uma área com objectos de pequeno porte, o campo de visibilidade estende-se para além dessas áreas, visto que não existe qualquer objecto que oculte a visibilidade do observador. Para estes casos

verifica-se quais os quarteirões que circundam a área verde, a área vazia ou a área com objectos de pequeno porte. Estes quarteirões são associados a cada uma das respectivas áreas.

Desta forma fica concluída a fase da segmentação, seguindo-se o cálculo do campo de visibilidade para cada uma das ruas. Esta fase consiste em determinar para a rua onde se encontra o observador, todos os elementos que lhe foram associados durante a fase da segmentação. Desta forma cada rua existente no ambiente virtual tem uma lista de elementos que lhe estão associados como por exemplo as ruas, os quarteirões, os edifícios sem interesse público, os jardins, os edifícios com interesse público, as igrejas, os hospitais, as estações de comboio e as linhas de comboio (Figura 64).

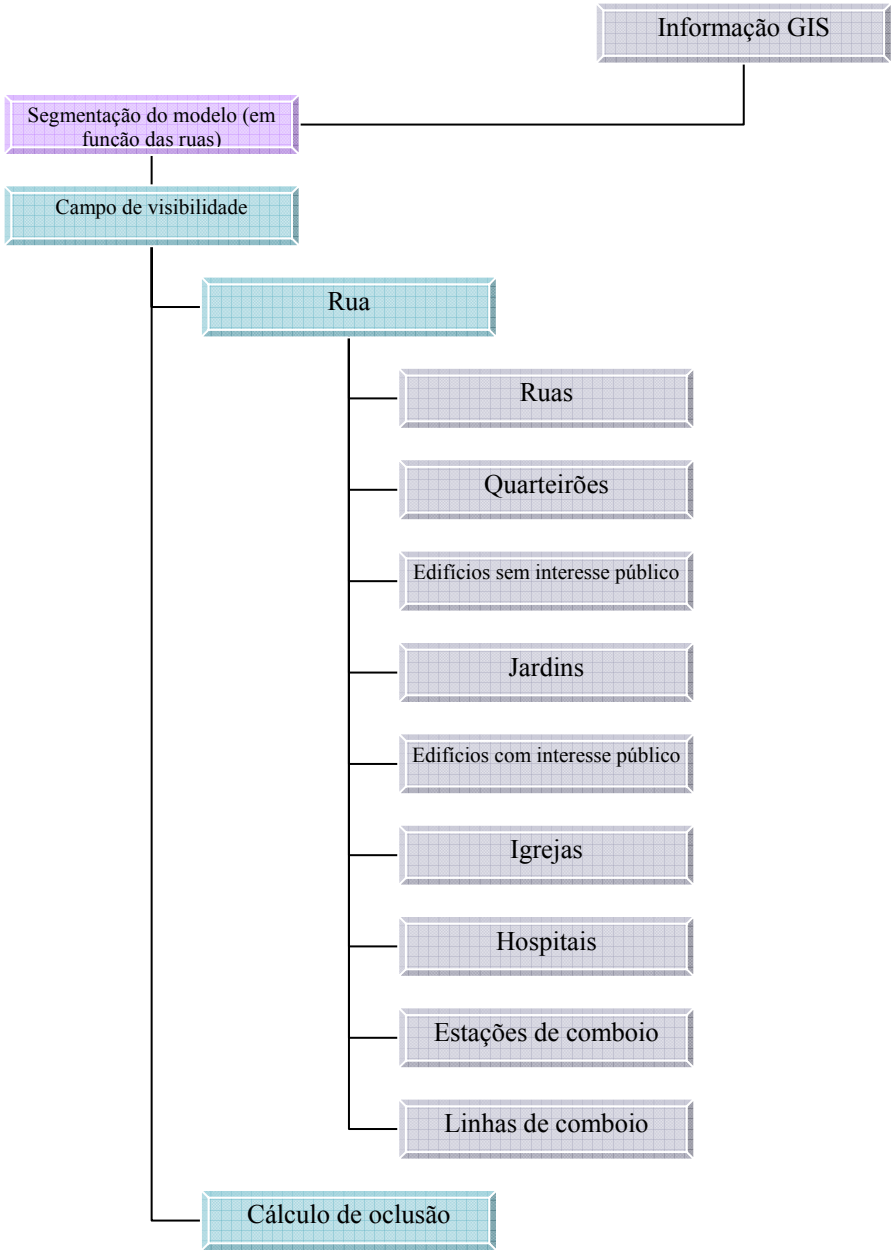


Figura 64 – Esquematização do campo de visibilidade de um modelo de cidade

A primeira fase do cálculo de oclusão consiste na determinação dos edifícios que originam maior oclusão. Desta forma, reduz-se substancialmente a quantidade de edifícios para os quais é necessário calcular a respectiva sombra.

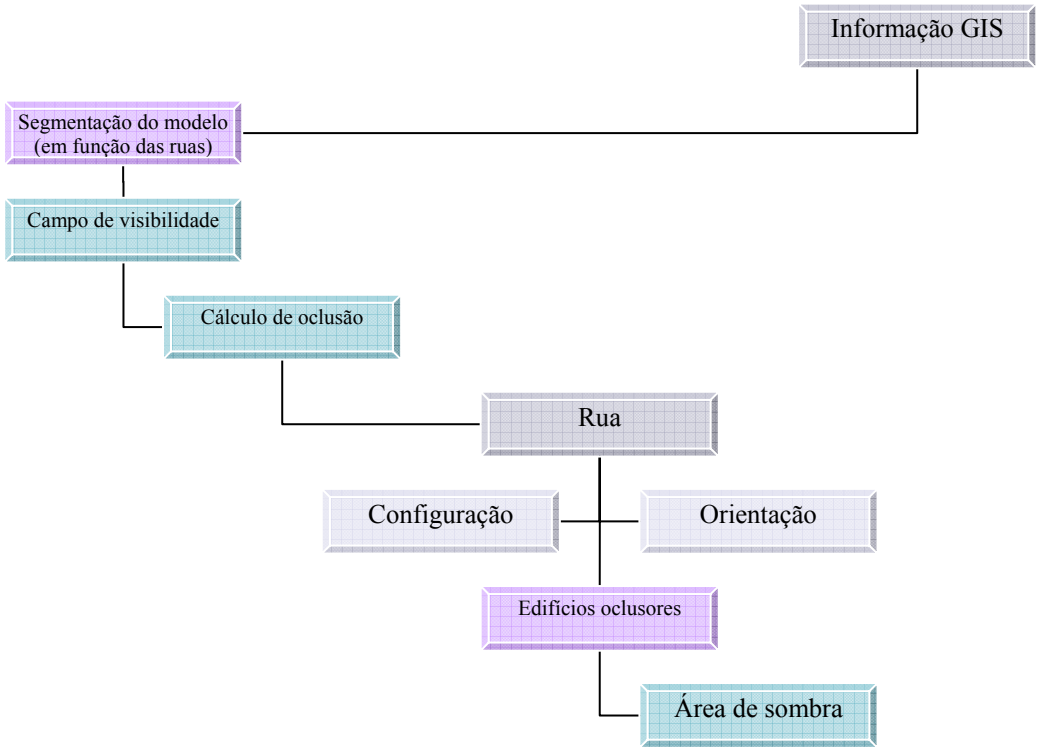


Figura 65 – Esquematização do cálculo de oclusão de um modelo de cidade

Sendo as ruas, nesta primeira fase, o elemento essencial, é definida a sua configuração, (secção 4.5.1). Independentemente do seu aspecto, é apurada a sua orientação relativamente ao eixo da abcissa e são determinadas as coordenadas com maior e menor valor.

A necessidade de definir diferentes configurações para as ruas surgiu na construção da área de sombra para cada edifício, como foi referido na secção 4.5.1, visto ser necessária a coordenada de início e fim da rua. Numa rua do tipo ‘circuito’, as coordenadas de início e fim normalmente coincidem ou estas coordenadas não representam o início e o fim da respectiva rua (Figura 40). Uma vez que o oclusor desempenha um papel relevante no cálculo de oclusão devido à área de sombra que projecta, para as ruas em ‘circuito’, as coordenadas com maior e menor valor são determinadas em função da parte da rua que se encontra de frente para o edifício que representa o papel de oclusor (Listagem 5).

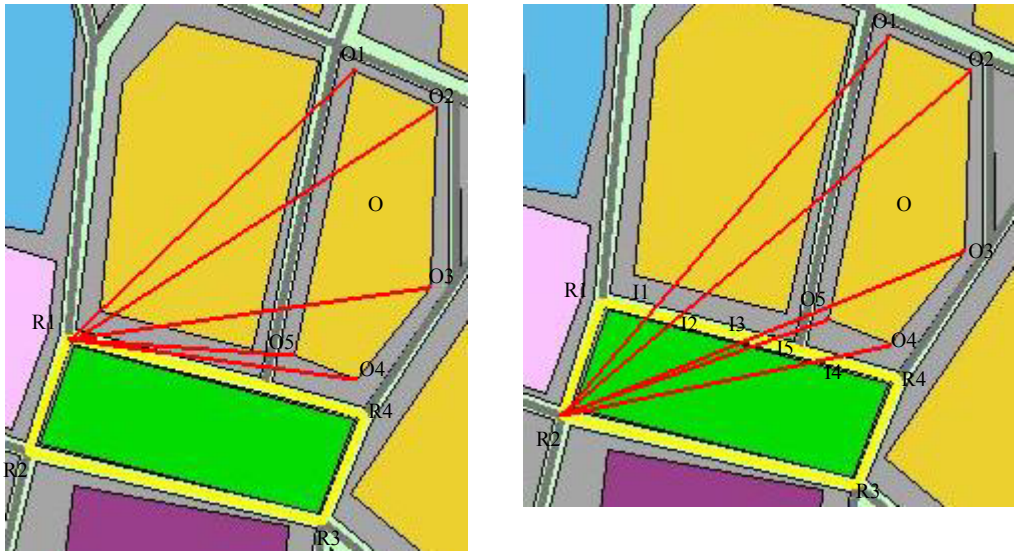
```
Variáveis:
   i, j são variáveis auxiliares
   CoordR é uma coordenada da rua
   CoordP é uma coordenada do edifício
   ListCoordR é uma lista com as coordenadas da rua
   ListCoordP é uma lista com as coordenadas do edifício
   Recta representa duas coordenadas de uma recta

{determinar uma recta desde cada coordenada da rua até cada coordenada do polígono
formado pelo edifício oclusor}
for i = 1 to número de coordenadas da rua
   retirar da lista ListCoordR uma das coordenadas da rua (CoordR)
   for j = 0 to número de coordenadas do edifício
      retirar da lista ListCoordP uma das coordenadas do edifício (CoordP)
      Recta = linha formada pela CoordR e CoordP
      if Recta não intersecta polígono da rua (PoliRua) then
         armazenar CoordR
```

Listagem 5 – Cálculo das coordenadas de uma rua em 'circuito'

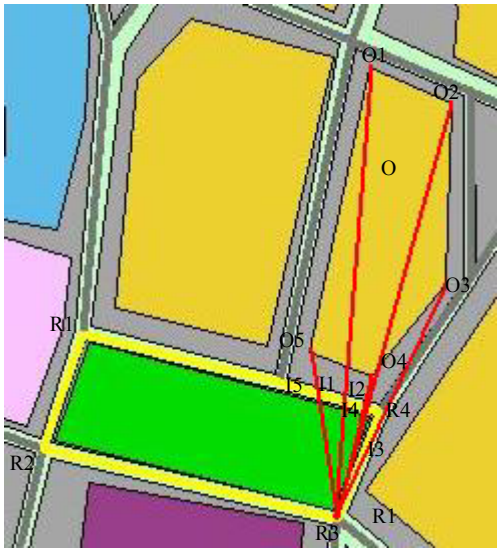
A Listagem 5 determina os segmentos de recta que passam em cada coordenada da rua (R1, R2, R3, e R4) e em cada coordenada do edifício (O1, O2, O3, O4, O5). Para cada segmento de recta é verificado se este intersecta o polígono formado pelas coordenadas da rua (*PoliRua*). Caso o segmento de recta não intersecte o polígono (*PoliRua*), é armazenada a respectiva coordenada da rua pois esta vai constituir a parte que se encontra de frente para o oclusor (Figura 66).

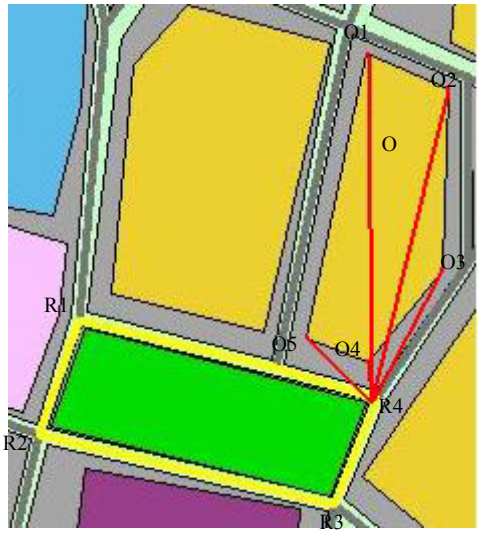
$$PoliRua = \{R1, R2, R3, R4\}$$

$$Edificio = \{O1, O2, O3, O4, O5\}$$



$$R1O1 \cap PoliRua = \varnothing \qquad R2O1 \cap PoliRua = I1$$
$$R1O2 \cap PoliRua = \varnothing \qquad R2O2 \cap PoliRua = I2$$
$$R1O3 \cap PoliRua = \varnothing \qquad R2O3 \cap PoliRua = I3$$
$$R1O4 \cap PoliRua = \varnothing \qquad R2O4 \cap PoliRua = I4$$
$$R1O5 \cap PoliRua = \varnothing \qquad R2O5 \cap PoliRua = I5$$





$$R3O1 \cap PoliRua = I1 \qquad R4O1 \cap PoliRua = \varnothing$$
$$R3O2 \cap PoliRua = I2 \qquad R4O2 \cap PoliRua = \varnothing$$
$$R3O3 \cap PoliRua = I3 \qquad R4O3 \cap PoliRua = \varnothing$$
$$R3O4 \cap PoliRua = I4 \qquad R4O4 \cap PoliRua = \varnothing$$
$$R3O5 \cap PoliRua = I5 \qquad R4O5 \cap PoliRua = \varnothing$$

$FrenteRua = \{R1, R4\}$ para o oclusor O

Figura 66 – Determinação da coordenada com maior e menor valor para uma rua em 'circuito'

Numa rua do tipo 'comprida' o campo de visibilidade abrange normalmente uma área bastante extensa, como se pode verificar pela comparação entre o conjunto CV_{21} e o conjunto CV_{84}, em que o primeiro representa o campo de visibilidade de uma rua do tipo 'comprida' e o segundo representa o campo de visibilidade de uma rua do tipo 'normal':

$$CV_{21} = \left\{ \begin{array}{l} R5, R12, R20, R21, R22, R29, R35, R66, R67, R78, Q7, Q12, Q13, Q14, Q26, \\ Q28, Q29, Q30, Q31, Q32, Q33, E13, E16, E17, E19, E20, E21, E26, E27, \\ E28, E29, E30, E45, E46, EI0, EI1, EI2, EI3 \end{array} \right\}$$

$$CV_{84} = \{R17, R33, R58, R68, R84, Q25, Q44, Q45, Q46, E9, E11, E37, EI1, EI3\}$$

Aquando da aplicação do cálculo de oclusão num oclusor em função de toda a rua, o número de elementos do campo de visibilidade CV_{21} que passa para a lista de elementos visíveis é quase idêntico (Figura 67) comparativamente ao obtido se a rua for adequada à localização do oclusor (Figura 43). Desta forma, o número de elementos definidos como visíveis torna-se pouco realista.

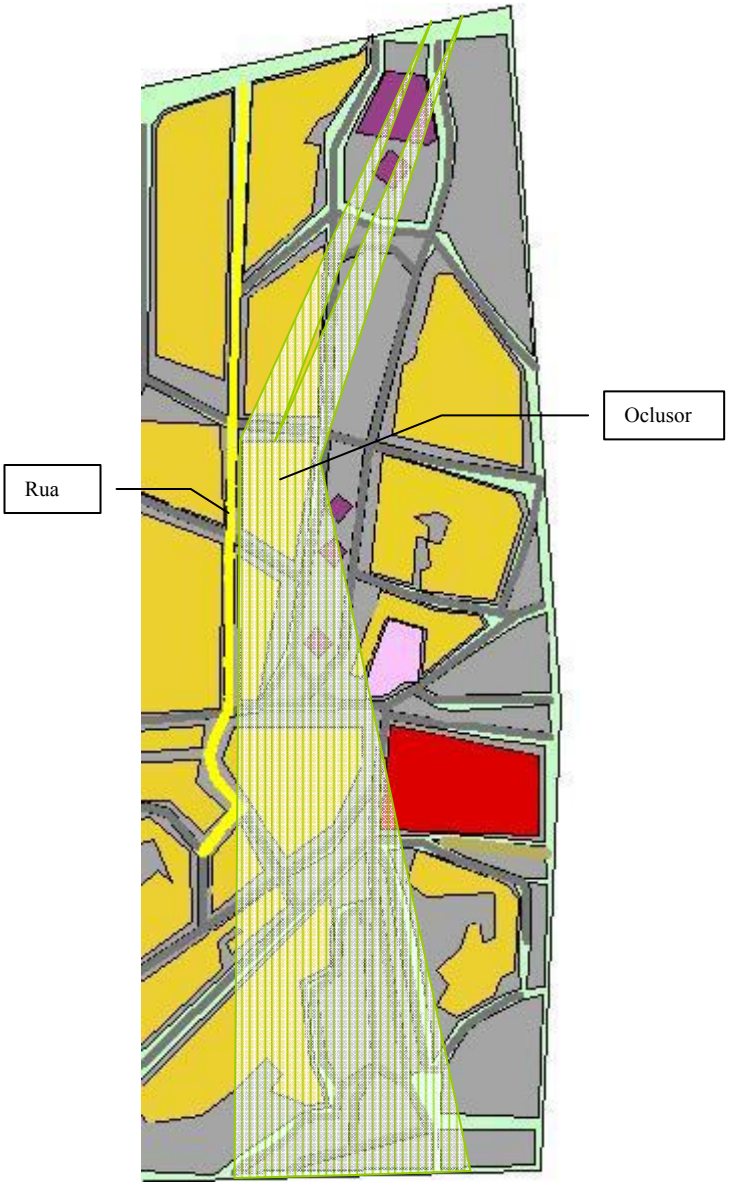
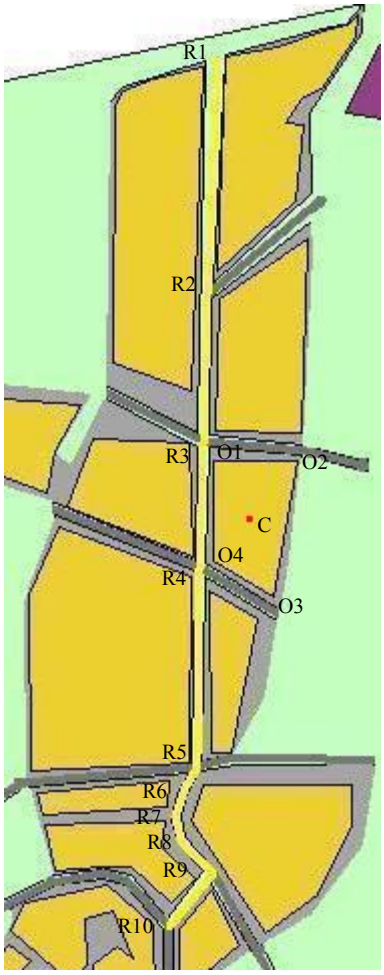


Figura 67 – Área de sombra para uma rua do tipo 'comprida' e o respectivo oclusor

Como forma de aproximar à realidade o número de elementos definidos como visíveis para uma rua classificada como "comprida", é determinado o centro do polígono que representa o edifício (*CentroPoli*), e a distância entre *CentroPoli* e todas as coordenadas da rua (Figura 68). Para as distâncias calculadas, começando da menor para a maior distância, selecciona-se a respectiva coordenada da rua e compara-se a posição desta

coordenada relativamente à coordenada com maior e menor valor do edifício. Caso a coordenada da rua esteja compreendida entre as coordenadas de maior e menor valor do edifício, então é armazenada (Listagem 6).



$$Edificio = \{O1, O2, O3, O4\}$$
$$MaiorEdificio = \{O1\}$$
$$MenorEdificio = \{O3\}$$

$$dR1C = \sqrt{(R1x - Cx)^2 + (R1y - Cy)^2}$$
$$dR2C = \sqrt{(R2x - Cx)^2 + (R2y - Cy)^2}$$
$$dR3C = \sqrt{(R3x - Cx)^2 + (R3y - Cy)^2}$$
…

dR4C < dR3C < dR5C < dR2C < dR6C < dR7C < dR8C < dR9C < dR1C < dR10C

dR4C: O3 < R4 < O1 → armazena R4
dR3C: O3 < R3 > O1 → armazena R3
dR5C: R5 < O3 E R5 < O1
dR2C: R2 > O3 E R2 > O1
dR6C: R6 < O3 E R6 < O1
dR7C: R7 < O3 E R7 < O1
dR8C: R8 < O3 E R8 < O1
dR9C: R9 < O3 E R9 < O1
dR1C: R1 > O3 E R1 > O1
dR10C: R10 < O3 E R10 < O1

Figura 68 – Determinação da coordenada com maior e menor valor para uma rua do tipo 'comprida'

Este tipo de análise permite que, nas ruas classificadas de compridas, o campo de visibilidade seja determinado em função de pequenos troços da rua, obtendo assim oclusores e ocludidos mais coerentes e fiáveis.

```
Variáveis:
   i é uma variável auxiliar
   dist é o valor da distância
   CoordRx, CoordRy são as coordenadas da rua em ordem a X e a Y
   CoordPx, CoordPy são as coordenadas do edifício em ordem X e a Y
   ListDist é uma lista com todas as distâncias calculadas
   CoordR é a coordenada da rua

{determinar distância entre centro polígono (CentroPoli) e todas as coordenadas da
rua}
for i = 1 to número de coordenadas da rua
   dist = √((CoordRx − CentroPx)² + (CoordRy − CentroPy)²)
   guardar distancia na lista ListDist
{encontrar a distancia menor na lista ListDist}
for i = 1 to número de elementos da lista
   if (dist < distMenor) then
      distMenor = dist
   determinar para a menor distância encontrada a coordenada da respectiva rua
{comparar se a coordenada da rua está situada entre a coordenada de maior e menor
valor do edifício}
   Maior = 1ª coordenada da rua
   Menor = 1ª coordenada da rua
   if (CoordR < Maior) E (CoordR > Menor)) then
      guardar CoordR
   else
      substituir na lista com distâncias (ListDist) a distância pelo valor 99999
```

Listagem 6 – Cálculo das coordenadas de uma rua 'comprida'

Na fase do cálculo dos oclusores, é necessário, para cada rua, criar uma lista de proximidade em função dos edifícios que lhe estão associados. Os maiores oclusores são aqueles que se encontram mais perto da rua, visto que todos os edifícios situados na área definida como plano de sombra do oclusor não são visíveis.

Definido o oclusor, limita-se as suas coordenadas a um polígono que contém apenas as coordenadas de frente para a rua, através de segmentos de recta com início em cada uma das coordenadas da rua e fim na coordenada do oclusor. Caso o segmento de recta não intersecte o oclusor então a sua coordenada do segmento de recta pertence às coordenadas de frente.

5.5.1 - Construção da Área de Sombra

Na segunda fase calcula-se para cada oclusor, a respectiva área de sombra em função da rua (secção 4.5.2). A área de sombra é determinada em função das semi-rectas que têm origem na primeira e na última coordenadas da rua e passam no oclusor, obtendo-se as coordenadas do elemento Superfície. A intersecção entre as semi-rectas formadas pelas coordenadas da rua e do oclusor, com os segmentos de recta da Superfície, resultam numa lista de pontos (Listagem 7).

```
Variáveis:
   CoordSup é a coordenada do elemento da Superfície a guardar
   topo, direita, esquerda e fundo são as coordenadas referentes ao elemento
Superfície
   Sentido é o valor do sentido em que se encontra o vector

determinar sentido (Sentido) do vector definido pela coordenada da rua e
coordenada da frente do edifício
{em função da variável Sentido, determinar coordenadas do elemento Superfície}
if (Sentido = 1) then
   CoordSup = topo e direita
else
   if (Sentido = 2) then
      CoordSup = topo e esquerda
   else
      if (Sentido = 3) then
         CoordSup = esquerda e fundo
      else
         if (Sentido = 4) then
            CoordSup = fundo e direita
determinar ponto de intersecção entre a recta (rectaRF), formada pelas coordenadas
da rua e as coordenadas da frente do edifício, com a recta (rectaS), formada pelas
coordenadas encontradas do elemento Superfície
```

Listagem 7 – Cálculo das coordenadas a integrar a área de sombra

A área de sombra de um oclusor é obtida, de um modo simplista, a partir do polígono resultante das coordenadas da rua, das coordenadas do oclusor definidas como "de frente" para a rua e das coordenadas do elemento Superfície. A área de sombra final (ASf), como referido na secção 4.5.2, requer a determinação da área de sombra (AS1) referente à coordenada de maior valor da rua e a área de sombra (AS2) referente à coordenada de menor valor da rua. A determinação da área de sombra AS1 resulta dos seguintes passos (Figura 69):

- Para a coordenada de maior valor da rua (R1):
  - Determinar pontos de intersecção (I1, I2, I3 e I4) entre os segmentos de recta que iniciam em R1, passam em cada uma das coordenadas do oclusor que se encontram "de frente" para a rua (O1, O2, O3 e O4) e terminam num dos segmentos de recta do elemento Superfície;
  - Construir quadrilátero com maior coordenada em X e Y, e menor coordenada em X e Y, a partir do conjunto de coordenadas {I1,I2,I3,I4,O1,O2,O3,O4}: [I3I1O4I3];
  - Construir quadrilátero com a segunda maior coordenada em X e Y, e a segunda menor coordenada em X e Y, a partir do conjunto de coordenadas {I1,I2,I3,I4,O1,O2,O3,O4}: [I4O1O3I4];
  - Determinar polígono resultante da união de [I3I1O4I3] com [I4O1O3I4] (Listagem 8).

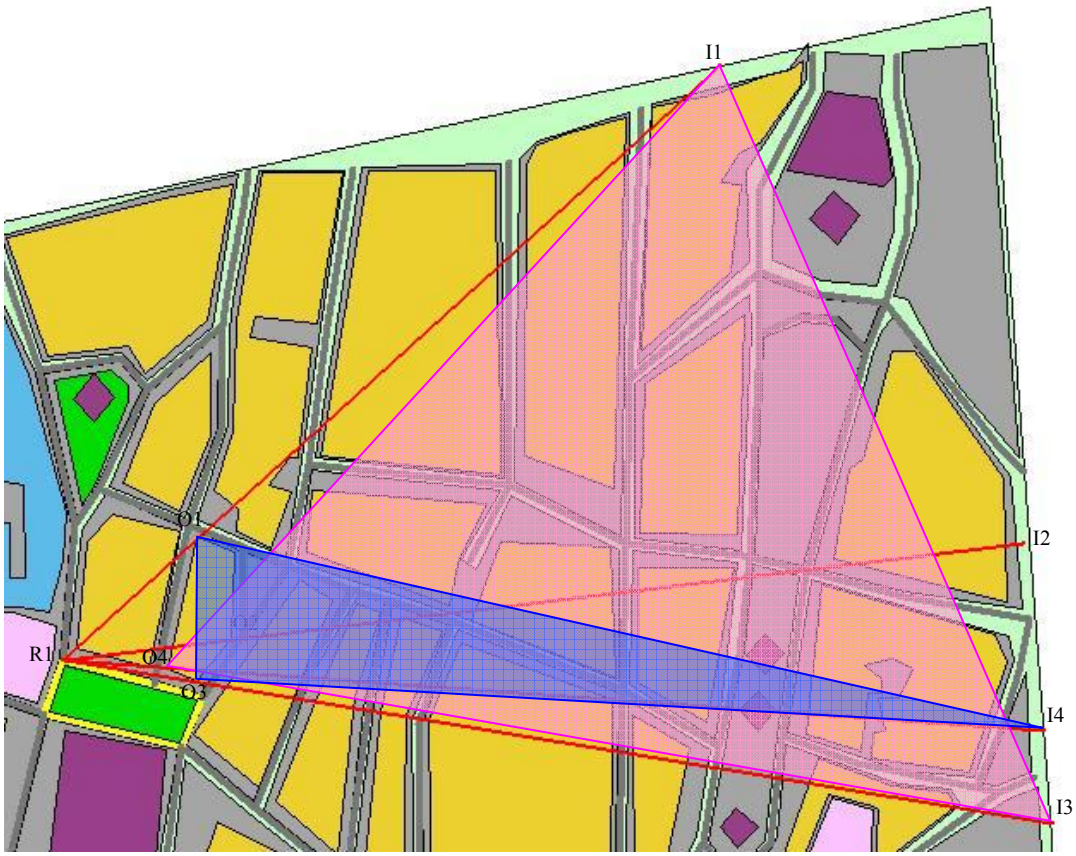


Figura 69 – Área de sombra para a coordenada de maior valor de uma rua e o oclusor

A área de sombra AS2 resulta dos seguintes passos (Figura 70):

- Para a coordenada de menor valor da rua (R2):
  - Determinar pontos de intersecção (I1, I2, I3 e I4) entre os segmentos de recta que iniciam em R2, passam em cada uma das coordenadas do oclusor que se encontram "de frente" para a rua (O1, O2, O3 e O4) e terminam num dos segmentos de recta do elemento Superfície;
  - Construir quadrilátero com maior coordenada em X e Y, e menor coordenada em X e Y, a partir do conjunto de coordenadas {I1,I2,I3,I4,O1,O2,O3,O4}: [I2I2I4O3];
  - Construir quadrilátero com a segunda maior coordenada em X e Y, e a segunda menor coordenada em X e Y, a partir do conjunto de coordenadas {I1,I2,I3,I4,O1,O2,O3,O4}: [O2I1I3O4];
  - Determinar polígono resultante da união de [I2I2I4O3] com [O2I1I3O4] (Listagem 8).

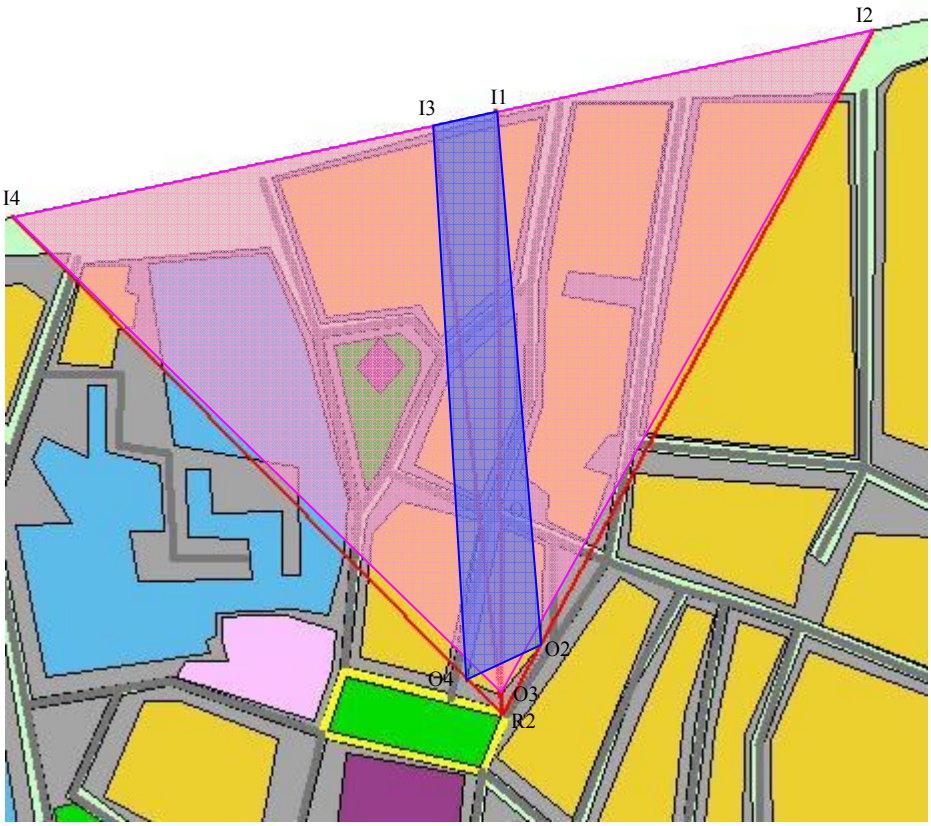


Figura 70 – Área de sombra para a coordenada de menor valor de uma rua e o oclusor

A união da área de sombra AS1 com AS2 resulta na área de sombra final (ASf).

```
Variáveis:
   i, j são variáveis auxiliares
   rectaRF é a recta que passa em uma das coordenada da rua e em uma das
coordenadas da frente do edifício
   pontosInter é o ponto de intersecção entre a recta rectaRF e o elemento
Superfície
   listaP é o conjunto de coordenadas da frente do edifício e dos pontos
pontoInter
   Pt1Maior, Pt1Menor são o maior e o menor ponto
   Pt2Maior, Pt2Menor são o 2° maior e menor ponto
   poliExterior, poliInterior são as coordenadas que formam um polígono
   areaSombra é as coordenadas de um polígono

determinar recta (rectaRF) com coordenadas da frente do edifício e com coordenadas
da rua
guardar pontos (pontosInter) de intersecção entre a recta (rectaRF) e o elemento
Superfície
for i = 0 to coordenada da frente do edifício E j = 0 to pontoInter
   {determinar a maior e menor coordenada, do conjunto (listaP) de coordenadas da
frente do edifício e dos pontos (pontoInter) resultantes da intersecção}
   Pt1Maior = maior ponto do conjunto listaP
   Pt1Menor = menor ponto do conjunto listaP
   {determinar as 2ª maior e menor coordenadas, do conjunto (listaP) de
coordenadas da frente do edifício e dos pontos (pontoInter) resultantes da
intersecção}
   Pt2Maior = 2° maior ponto do conjunto listaP
   Pt2Menor = 2° menor ponto do conjunto listaP
{formar polígono (poliExterior) com Pt1Maior, Pt1Menor, e coordenadas da frente do
edifício}
poliExterior = Polygon.Make(Pt1Maior, Pt1Menor, coordenadas da frente do edifício)
{formar polígono (poliInterior) com Pt2Maior, Pt2Menor, e coordenadas da frente do
edifício}
poliInterior = Polygon.Make(Pt2Maior, Pt2Menor, coordenadas da frente do edifício)
{determinar polígono que resulta da união de poliExterior e poliInterior}
areaSombra = poliExterior.ReturnUnion (poliInterior)
```

Listagem 8 – Cálculo da área de sombra para um oclusor

As coordenadas que constituem a frente do oclusor para a rua, a identificação da rua e a identificação do oclusor são armazenados numa tabela, denominada Segmentação, de forma a que apenas as coordenadas visíveis do oclusor sejam representadas.

A partir da área de sombra final (ASf) do oclusor são verificados todos os edifícios que estão associados à rua, determinando os que estão contidos ou intersectam a área de sombra final. Os edifícios contidos na área de sombra final são eliminados pois, encontrando-se na sombra do oclusor, não são visíveis. Quando um determinado edifício intersecta apenas a área de sombra final, a parte contida no plano de sombra é eliminada e é constituído um novo polígono com as coordenadas fora do plano (Listagem 9).

Aquando do processo de constituição de um novo polígono devido à intersecção do polígono original com a área de sombra final é necessário verificar se o novo polígono representa uma figura geométrica. Caso contrário, esse polígono é automaticamente configurado ou, em caso impraticável, é eliminado (Figura 71).

```
Variáveis
   planoSombra é as coordenadas de um polígono (Listagem 8)
   Edif contem as coordenadas de um edifício
   poliPartido são as coordenadas de um polígono

{verificar se polígono planoSombra intersecta ou contém restantes edifícios
associados à rua (Edif)}
if (planoSombra.Intersects (Edif)) then
   if (planoSombra.Contains (Edif)) then
      elimina edificio
   else
      poliPartido = Edif.ReturnDifference (planoSombra)
```

Listagem 9 – Verificação da intersecção ou inclusão total do edifício na área de sombra

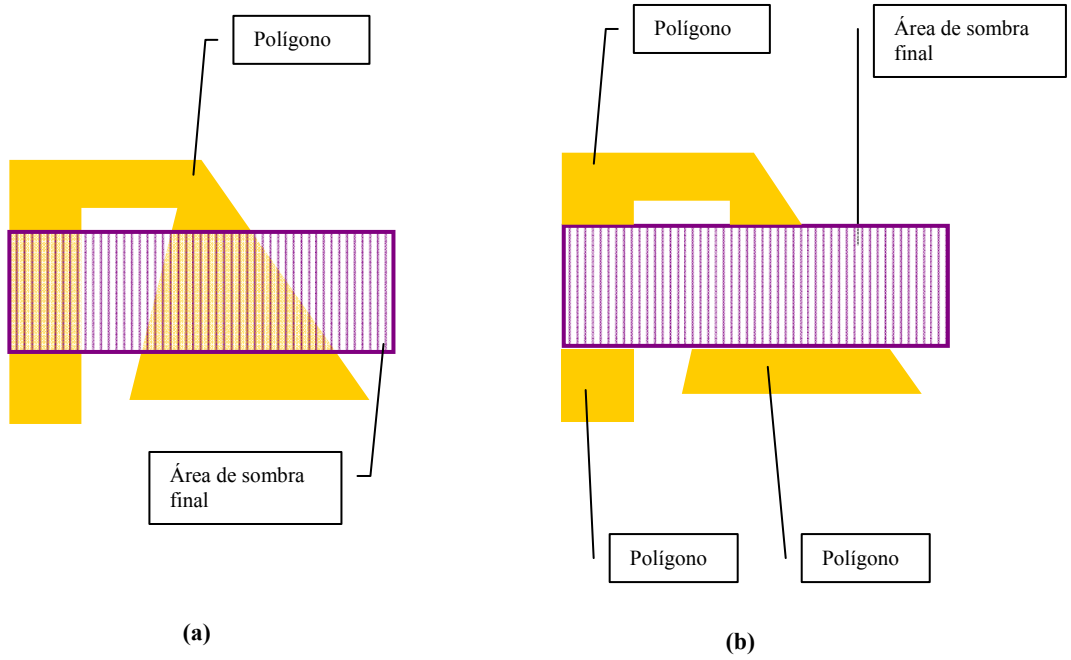


Figura 71 – Intersecção da área de sombra final com um polígono (a) e novos polígonos resultantes (b)

Durante o processo do cálculo de visibilidade, os edifícios com uma altura relevante são examinados tendo em conta a altura dos edifícios que se encontram entre a rua e o edifício alto. Para isso, é determinado o ponto médio (*Pmedio*) do edifício alto e calculadas rectas (*rectaER*) que passam nas coordenadas da rua e no ponto *Pmedio*. Dos edifícios que estão associados à rua, é verificado se algum deles intersecta as rectas *rectaER*. Do conjunto de edifícios encontrados, a altura mínima serve de referência como mínimo de representação do edifício alto. Desta forma, o edifício alto é considerado visível, cabendo ao algoritmo de cálculo de visibilidade (*Z-buffer*) representá-lo, entre a sua altura máxima e a altura do edifício mais baixo (Listagem 10).

```
Variáveis:
   Pmedio é o ponto que representa o ponto médio de um polígono
   rectaER representa as coordenadas de uma recta
   i é uma variável auxiliar
   ListAltura é uma lista que armazena os valores das alturas de edifícios
   MenorAltura representa o valor da altura menor de um edifício

{determinar ponto médio do edifício alto}
Pmedio =ponto centro do polígono
{determinar recta desde ponto médio (Pmedio) até cada coordenada da rua}
rectaER = Line.Make (Pmedio, coordenadas da rua)
{verificar se recta intersecta algum edifício}
for i = 1 to total de edifícios associados à rua
   if (rectaER.Intersects (edificio asssociado à rua) then
      determinar altura do edifício intersectado
      guardar atura encontrada (ListAltura)
{determinar da lista (ListAltura) a altura menor}
ListAltura.Sort(True)
MenorAltura = 1° elemento da lista ListAltura
```

Listagem 10 – Cálculo da altura dos edifícios entre a rua e edifício alto

Devido à inexistência de dados e ao objectivo do caso de estudo, a altura dos edifícios comuns é gerada aleatoriamente assim como os respectivos materiais. Cada um dos restantes temas tem definida uma altura e um material específicos.

Como é referido na secção 5.4, os temas referentes às ruas e linhas de comboio são representados como uma recta, tendo de ser configurados para polígonos aquando da transformação das coordenadas, de forma a possuírem uma largura real.

Durante o processo de mudança do sistema de coordenadas em milhas do *software* de GIS para o sistema de coordenadas em metros, pode surgir o mesmo problema da intersecção da área de sombra final com um polígono. Desta forma, sempre que um elemento não possua uma área mínima representativa de um polígono é automaticamente configurado, ou em caso impraticável, é eliminado.

5.6 - Codificação em Linguagens de Anotação

Concluída a segmentação de todas as ruas existentes no projecto, é necessário codificar os modelos em linguagens de anotação. Neste contexto, foi decidido utilizar, num passo inicial, a linguagem VRML, convertendo os ficheiros obtidos, posteriormente e por recurso a uma ferramenta comercial, para X3D.

Cada rua possui um sensor para cada cruzamento com outra rua de modo a que, aquando da navegação do utilizador pelo mundo, sejam detectadas as mudanças de rua, sendo o seu nome lido do servidor e actualizado no cliente. Para que uma rua não seja carregada logo que o sensor é encontrado, quando, por exemplo, o utilizador apenas se aproxima da nova rua, o sensor pertencente à nova rua é posicionado a 20 metros dessa extremidade. (Listagem 11).

```
Variáveis:
   vectEV representa as coordenadas de um vector
   a, b e c são os parâmetros da equação geral de uma recta
   CoordEixo é a coordenada referente ao eixo da via
   valorX_pri, valorX_seg, valorY_pri, valorY_seg são os valores em X e Y
resultante da equação geral da recta
   pontoSensA, pontoSenB são pontos onde serão localizados os sensores

{determinar vector (vectEV) formado pelas duas coordenadas que formam o eixo da
via}
vectEV = 1ª coordenada - 2ª coordenada
{determinar equação geral da recta que passa nas coordenadas que formam o eixo da
via}
a = (1 + (vectEv)^2)
b = -2 * CoordEixo * (1 + (vectEv)^2)
c = (CoordEixo)^2 * (1 + (vectEv)^2) - (20)
valor_pri = (((-1)*b) - sqrt(((b*b) - (4*a*c))))/(2*a)
valor_seg = (((-1)*b) + sqrt(((b*b) - (4*a*c))))/(2*a)
{determinar valores de Y}
valorY_pri = (vectEv * valorX_pri) + CoordEixo - (vectEV * CoordEixo)
valorY_seg = (vectEv * valorX_seg) + CoordEixo - (vectEv * CoordEixo)
{criar pontos}
pontoSensA = Point.Make (valorX_pri, valorY_pri)
pontoSensB = Point.Make (valorX_seg, valorY_seg)
```

Listagem 11 – Posicionamento dos sensores nos extremos de uma rua

O pseudo-código apresentado na Listagem 11 descreve o processo utilizado para determinar, no eixo da via, as coordenadas onde são inseridos os sensores. Inicialmente, determina-se o vector formado pelas duas coordenadas de início e fim da rua pertencentes ao eixo da via (R0i, R0f), seguindo-se o respectivo cálculo da equação geral da recta que passa nas coordenadas R0i e R0f. A partir desta equação da recta determinam-se dois pontos a uma distância de 20 metros (Sensor A, Sensor B) das coordenadas R0i e R0f e que pertençam à rua r0 (Figura 72).

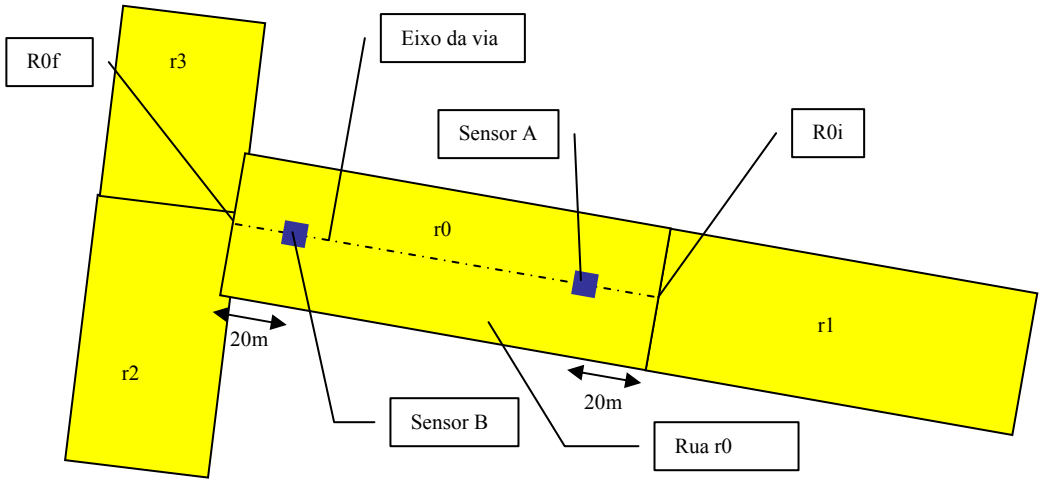


Figura 72 – Posição dos sensores na rua r0

Cada sensor especifica assim uma orientação para o observador que é armazenada na tabela respectiva. Desta forma, para cada cruzamento ou entroncamento que uma rua possui existe, no mínimo, a correspondência a dois ficheiros VRML idênticos mas cada um deles possuindo diferente orientação para a localização do observador, tendo em conta o sensor. No exemplo apresentado na Figura 72, se o observador navega no ambiente virtual vindo da rua r1 para a rua r0 é detectado o sensor A, permitindo assim que a nova rua detectada (r0) seja actualizada com os respectivos elementos que a compõem e o observador posicionado na posição do sensor detectado (sensor A) com uma orientação no sentido do sensor A para o sensor B. No caso do observador navegar vindo da rua r2 ou r3 para a rua r0, o sensor detectado será o sensor B, sendo consequentemente actualizada a nova rua (r0) e posicionado o observado no sensor B com uma orientação no sentido do sensor B para o sensor A. Conclui-se deste modo que uma rua tem sempre, no mínimo, dois ficheiros VRML idênticos mas possuindo cada um uma orientação diferente para o observador. Um primeiro ficheiro (r0_a) tem uma orientação para o observador no sentido do sensor A para o sensor B, enquanto que um segundo ficheiro (r0_b) tem uma orientação para o observador no sentido do sensor B para o sensor A. A Figura 73 apresenta uma rua (r81 a amarelo) que possui sete entroncamentos e um cruzamento. Estes entroncamentos e cruzamentos devem-se a outras ruas que estão ligadas à rua r81. Desta forma, quando o observador navega vindo dessas ruas para a rua r81, ao ser detectado o sensor referente à nova rua, o observador tem de ser posicionado na rua r81 tendo em conta a rua de onde vinha. Logo, para cada entroncamento que a rua r81 tem, é necessário existir um ficheiro VRML com a respectiva localização do observador e para cada cruzamento que a rua r81 possui é necessário fazer existir dois ficheiros VRML, cada um com sentido diferente na localização do observador.

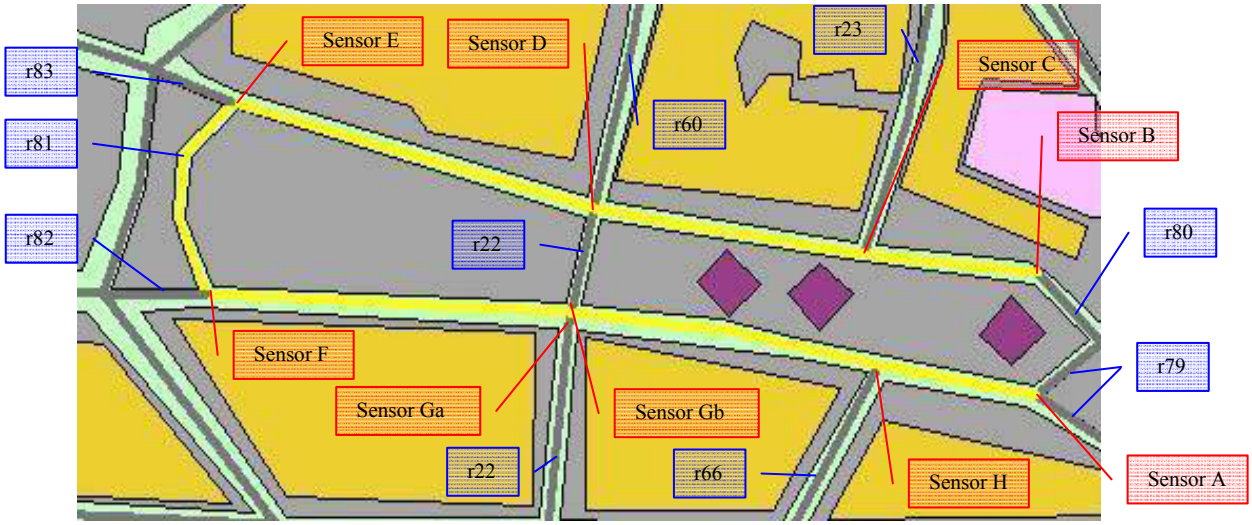


Figura 73 – Posição dos sensores na rua r81

A rua r81 possui portanto, nove ficheiros que serão carregados do servidor para o cliente em função da rua de origem (Tabela 7).

Tabela 7 – Nome dos ficheiros para a rua r81 e respectiva descrição

r81_a	Quando o observador vem da rua r79
rR81_b	Quando o observador vem da rua r80
r81_c	Quando o observador vem da rua r22
r81_d	Quando o observador vem da rua r60
r81_e	Quando o observador vem da rua r80
r81_f	Quando o observador vem da rua r82
r81_ga	Quando o observador vem do lado oeste da rua r22
r81_gb	Quando o observador vem do lado este da rua r22
r81_h	Quando o observador vem da rua r66

A Figura 74 apresenta outras ruas que sofrem o mesmo processo de cálculo dos sensores visto que, possuem entroncamentos/cruzamentos com outras ruas.

Figura 74 – Exemplos de ruas com entroncamentos/cruzamentos

Aos ficheiros VRML gerados em função das tabelas obtidas da segmentação (secção 5.5.1), é atribuído, a cada material, as respectivas propriedades e texturas. Para a iluminação do mundo virtual são criados sete pontos de luz, sendo o primeiro posicionado na localização do utilizador, e os restantes distribuídos por toda a superfície do modelo virtual. O ponto de luz posicionado na localização do observador permite uma iluminação mais directa com o intuito do observador poder visualizar facilmente os objectos representados. Os restantes pontos de luz foram definidos como sendo necessário seis devido ao elemento Superfície possuir a forma de um rectângulo, resultando numa distribuição mais uniforme dos pontos de luz por esta mesma Superfície.

A área da cidade do Porto seleccionada para este protótipo contém 84 ruas, tendo gerado 255 ficheiros VRML, os quais foram, posteriormente, convertidos em ficheiros X3D. A conversão foi realizada por meio de um *software* comercial mas, apesar disso, alguns dos ficheiros resultantes contêm anomalias, o que torna necessária a sua correcção manual.

5.7 - Interface

Pretendeu-se conceber uma *interface* amigável e funcional para os utilizadores com menos experiência neste tipo de novas tecnologias. Como o sistema será consultado remotamente, a *interface* deve ser visualizada em qualquer máquina, independentemente da sua plataforma, utilizando um *browser* comum.

A *interface* do projecto é composta por duas partes (Figura 75):

- Do lado esquerdo é possível visualizar o nome da rua onde se encontra localizado o utilizador. Quando o observador se situa num cruzamento de ruas, é detectada uma rua com duas orientações, sendo ambas as orientações descritas na caixa de texto. Neste caso, é adicionado um botão 'Ok' por baixo da caixa de texto, para que o utilizador possa seleccionar o sentido da rua para onde pretende ir e depois pressionar o botão para a activar (Figura 76).

A imagem situada por baixo da caixa de texto representa, na forma de mapa 2D, a área modelada do protótipo, de modo a que o utilizador se possa situar mais facilmente.

- Do lado direito encontra-se o *browser* Xj3D para a navegação 3D do ambiente criado.

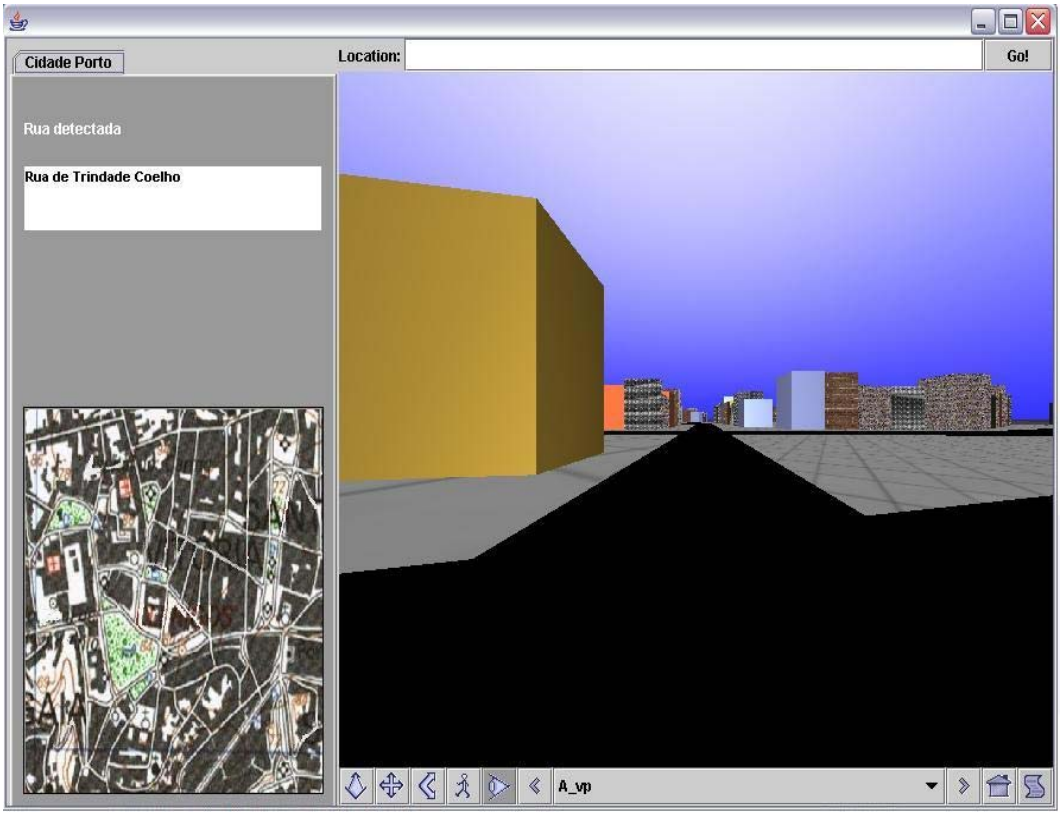


Figura 75 – *Interface* do protótipo

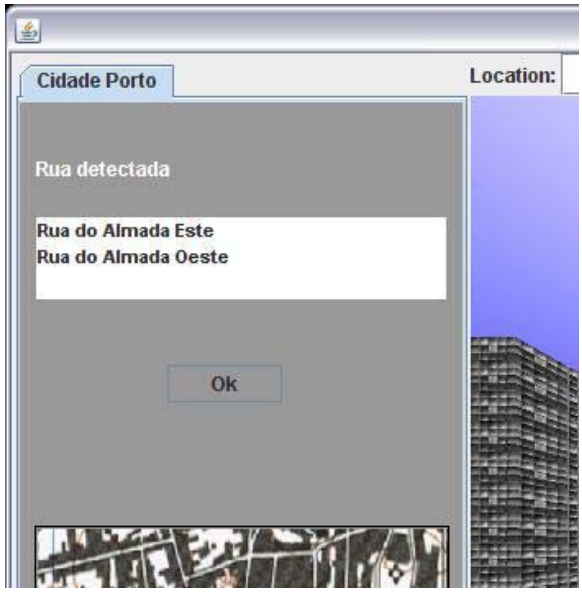


Figura 76 – *Interface* do protótipo quando a rua tem dois sentidos

A *interface* é implementada numa classe Java, carregada no *browser* Xj3D e utilizando a tecnologia *SAI*, secção3.2.2.

Para permitir a visualização do nome das ruas na *interface* durante a navegação pelo ambiente virtual, e a detecção dos respectivos sensores (secção 5.6) é armazenada, em vectores, a informação respectiva (Listagem 12).

```
public class CidadePorto_interface extends javax.swing.JFrame implements
X3DFieldEventListener {
    String[] RuasNomes = {"Rua do Prof. Vicente José de Carvalho", "Rua do Dr.
Tiago Almeida", "Travessa do Carregal","Rua do Prof. Jaime Rios de Sousa","Rua
Mouzinho da Silveira", "Rua de Trindade Coelho", "Rua do Carmo", "Rua de Trás",
"Rua das Carmelitas", "Praça de Carlos Alberto",”Rua de Cedofeita", "Rua das
Oliveiras", "Rua de Ramalho Ortigão", …};
    String[] r0 = {"sensorA_elemento_R1", "sensorB_elemento_R1",
"sensorA_elemento_R2", "sensorB_elemento_R2", "sensorC_elemento_R2",
"sensorD elemento R2", "sensorA elemento R6", "sensorB elemento_R6", …};
String[] r1 = {"sensorA elemento R0", "sensorB elemento R0",
"sensorC elemento R0", "sensorA_elemento_R2", "sensorB_elemento_R2",
"sensorC_elemento_R2", …};
```

Listagem 12 – Vectores com o nome das ruas e respectivos sensores

A Listagem 12 mostra, no ficheiro Java, a classe *CidadePorto_interface*, onde são definidos três vectores *RuasNomes*, *r0* e *r1*. O primeiro vector representa os nomes das ruas do ambiente virtual, enquanto o segundo e terceiro vectores armazenam os nomes dos sensores existentes para a rua identificada com o nome de *r0* e a rua identificada com o nome de *r1*. Desta forma, sempre que o observador entra numa nova rua, em função do sensor detectado e respectiva rua, é possível determinar no vector *RuasNomes* o nome da respectiva rua. Por exemplo, se o sensor detectado se refere à rua r0, então é lido do vector *RuasNomes* o nome da rua referente à posição 1 do vector: Rua do Prof. Vicente José de Carvalho. De seguida, para a nova rua (r0) é lido o vector *r0* que contém todos os sensores existentes no ambiente virtual que podem ser detectados a partir da rua r0.

A comunicação entre o mundo X3D e o utilizador efectua-se através da tecnologia SAI, efectuando ligações ao *browser* e aos nós fundamentais do ficheiro X3D. Quando o protótipo é carregado pela primeira vez, é carregada a *interface* com as respectivas formatações (Listagem 13) e estabelecida a rua a ser carregada (Listagem 14): rua identificada com o nome de r0 a qual tem o nome de Rua do Prof. Vicente José de Carvalho.

```
public CidadePorto_interface () {
   [inicializar componentes]
   X3DComponent x3dComp = BrowserFactory.createX3DComponent(null);
   JComponent x3dPanel = (JComponent)x3dComp.getImplementation();
   contentPane.add(x3dPanel, BorderLayout.CENTER);
   x3dBrowser = x3dComp.getBrowser();
   setSize(1000,750);
   show();
```

Listagem 13 – Leitura da *interface*

```
cena3D = x3dBrowser.createX3DFromURL(new String[]
{"ficheiros_x3d/cidadeporto_r0_a.x3d"});
x3dBrowser.replaceWorld(cena3D);
```

Listagem 14 – Leitura da rua pré-definida como primeira

O passo seguinte é carregar todos os sensores referentes à rua e activá-los (Listagem 15).

```
for(i=0; i<numElem; i++) {
   String sensor = r0[i];
   X3DNode Sensor = (X3DNode) cena3D.getNamedNode(sensor);
   SFBool PSactivo = (SFBool) Sensor.getField("isActive");
   PSactivo.addX3DEventListener(this);
```

Listagem 15 – Leitura de todos os sensores referentes à rua

Assim que um sensor é detectado (método *readableFieldChanged*), verifica a que rua pertence. Compara a nova rua a carregar com o conjunto de ruas que possuem um cruzamento duplo, de modo a inserir o nome na lista de ruas detectadas (Listagem 16).

```
public void readableFieldChanged(X3DFieldEvent evt){
   Object id = evt.getData();
   String NewS = id.toString();
   String StrIndiceR = nomeFx.substring(13,compStr-2);
   int indiceR = Integer.valueOf(StrIndiceR).intValue();
   [verifica rua dividida]
```

Listagem 16 – Detecção de um sensor

Caso a rua pertença ao conjunto de ruas com um cruzamento duplo, é necessário determinar quais as duas orientações que essa rua possui e apresentá-las na lista de ruas, de modo a que o utilizador possa seleccionar qual a direcção da rua que pretende seguir (Listagem 17).

```
If (RuaPrincipal.equals("R5")) then
   RuaDividida = True;
   orientacao = "NS";
if RuaDividida = True then
   if orientação = "N" then
      RuaNome= "Norte";
   if orientação = "S" then
      RuaNome = "Sul";
   if orientação = "E" then
      RuaNome = "Este";
   if orientação = "O" then
      RuaNome = "Oeste";
   [introduzir nome da rua na lista de ruas]
   ListaRuas.setListData(RuaNome);
```

Listagem 17 – Detecção das orientações da rua

5.8 - Síntese do Capítulo

A segmentação de ambientes virtuais urbanos é fundamental para a sua visualização, dados estes serem modelos muito complexos. O processo de segmentação desenvolvido é adaptativo dado que faz a divisão da área a modelar em função das ruas, que o utilizador percorre ao navegar no ambiente virtual. A segmentação realiza-se em duas fases:

inicialmente são determinados, para cada rua, os oclusores, sendo posteriormente calculadas as respectivas áreas de sombra.

A primeira fase permite seleccionar os edifícios que desempenham mais eficazmente o papel de oclusores para que, na segunda fase seja possível reduzir substancialmente o número de polígonos considerados para o cálculo de visibilidade.

Os ficheiros VRML são obtidos a partir das tabelas resultantes da segmentação e são posteriormente convertidos para X3D através do programa Vizx3D.

Para uma *interface* simples, intuitiva e que permita a interacção com o utilizador, é utilizado o *browser* Xj3D integrado com uma janela de navegação. Desta forma, uma janela é destinada a informar o utilizador do local onde se encontra, enquanto a outra possibilita a navegação no mundo virtual. A classe que suporta a interface utiliza a linguagem Java e a tecnologia SAI.

CAPÍTULO 6

AVALIAÇÃO DOS RESULTADOS OBTIDOS

Índice do Capítulo

6 - Avaliação dos Resultados Obtidos

Em resultado do trabalho produzido nesta dissertação e como forma de o validar, foi desenvolvido o protótipo CidadePorto Virtual, descrito no capítulo 5, restrito a uma área da cidade do Porto. Com este protótipo realizaram-se diversos testes de avaliação que se reportam neste capítulo.

O capítulo apresenta os resultados obtidos em dois testes diferentes desenvolvidos ao protótipo. O primeiro teste consistiu numa avaliação comparativa entre o protótipo CidadePorto Virtual e uma abordagem mais convencional na qual foi adoptada uma metodologia de segmentação em malha regular e sem cálculo de visibilidade (Modelo MRsCV). O segundo teste teve como objectivo avaliar o desempenho global da metodologia desenvolvida, do ponto de vista do utilizador, tendo sido realizado um questionário a um universo de 49 alunos do 3º ano do curso de Licenciatura em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

Neste capítulo são descritos, na secção 6.1, os resultados obtidos da análise comparativa entre os protótipos CidadePorto Virtual e o Modelo MRsCV, e a secção 6.2 apresenta os resultados apurados no questionário.

6.1 - Análise Comparativa

De modo a realizar-se uma análise comparativa entre a metodologia desenvolvida e uma metodologia convencional com segmentação em malha regular, foi desenvolvido um outro modelo, designado por CidadePorto Virtual MRsCV, referido no texto seguinte como modelo MRsCV.

O modelo MRsCV foi desenvolvido com o mesmo conteúdo da CidadePorto Virtual. A segmentação, como factor determinante na representação do modelo virtual e redução da

sua complexidade, foi desenvolvida de forma diferente em cada aplicação com o intuito de avaliar os benefícios/inconvenientes da metodologia desenvolvida nesta dissertação.

A segmentação em CidadePorto Virtual foi estabelecida em função das ruas, visto a navegação do utilizador ser do tipo "passeio", procedendo-se depois ao cálculo de oclusão, tendo em conta a metodologia desenvolvida nesta dissertação (Capítulo 4).

O modelo MRsCV foi dividido numa malha regular quadrangular, à semelhança do modelo desenvolvido para a Lisboa Virtual [Pimentel01], e sem a implementação de qualquer das técnicas de cálculo de oclusão. A Figura 77 mostra os temas incluídos no modelo, assim como a segmentação aplicada.

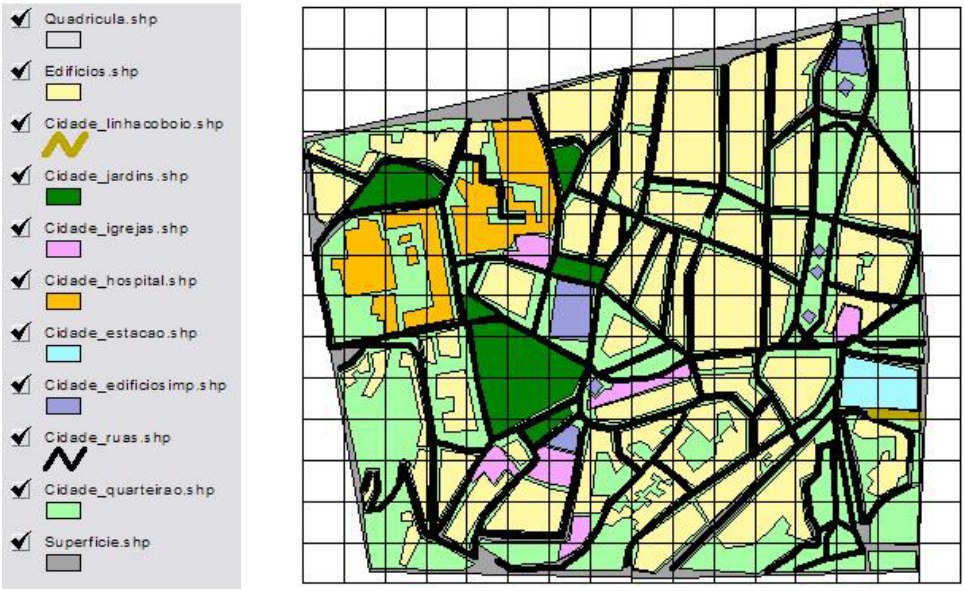


Figura 77 – Segmentação em forma de malha regular quadriculada aplicada no modelo MRsCV

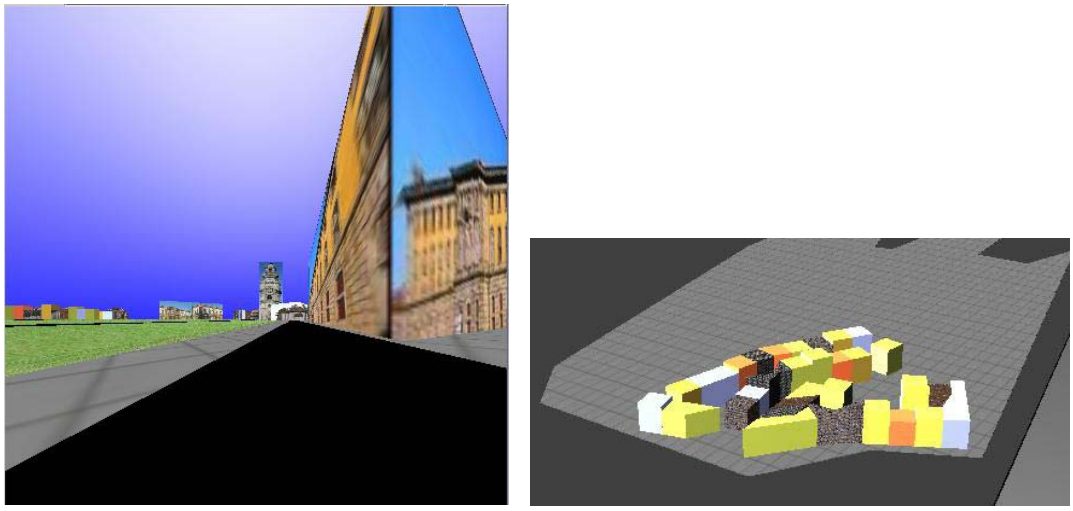
Em seguida apresenta-se uma avaliação feita aos dois modelos, tendo por base um conjunto de critérios como, navegação, segmentação e técnicas de cálculo de oclusão.

6.1.1 - Navegação

Para simular a imersão do utilizador num mundo tridimensional é necessário desenvolver ferramentas de interacção que lhe permitam ultrapassar a diferença entre a navegação no mundo real e a do mundo virtual. A navegação num ambiente urbano virtual pode ser de dois tipos: na forma de "passeio" ou "voo" (secção 4.3). Num ambiente urbano, a navegação em modo "passeio", consiste na imitação da forma convencional utilizada pelas pessoas para explorar uma cidade, que é caminhando ao longo das ruas. A navegação em modo de "voo", baseia-se na forma como uma pessoa observa uma cidade quando voa em avião, fornecendo uma perspectiva de cima para baixo da cidade.

Na CidadePorto Virtual, o utilizador navega em modo de "passeio" (Figura 78.a)). Este tipo de navegação permite ao utilizador uma sensação de pertencer ao mundo onde se encontra a navegar, podendo ter um contacto mais próximo com todos os elementos que compõem a cidade.

A navegação no modelo MRsCV pode ser efectuada em modo de "voar" ou de "passeio", uma vez que o utilizador pode sobrevoar pela quadrícula visualizada (Figura 78.b)) ou caminhar por entre os elementos representados no ambiente. Um dos inconvenientes desta solução é o utilizador visualizar o modelo em função dos elementos que compõem uma quadrícula, enquanto que na CidadePorto Virtual o utilizador observa a cidade como se estivesse a percorrer as ruas, visualiza os edifícios e jardins na direcção do seu olhar. Quando o ambiente representado depende da quadrícula seleccionada, o utilizador deixa de se sentir integrado no ambiente e tem a noção de que a cidade é limitada a uma determinada área. A selecção da quadrícula a visualizar é definida pelo utilizador ao iniciar a aplicação (Figura 79).



a) **b)**

Figura 78 – Avaliação do tipo de navegação em CidadePorto Virtual (a) e no modelo MRsCV

Para ambos os modelos em avaliação, em modo "passeio", a detecção da rua onde se encontra o utilizador é feita através de sensores, resultando no seu carregamento automático. Deste modo, o utilizador pode navegar por todos os segmentos, mesmo que dispersos em ficheiros independentes, de forma automática.

Figura 79 – Grelha de selecção no modelo MRsCV

No trabalho desenvolvido por Pimentel [Pimentel01], baseado na cidade de Lisboa e designado por Lisboa Virtual, a visualização do ambiente depende da área seleccionada pelo utilizador (Figura 80). Carregado o ficheiro correspondente à área seleccionada, o utilizador navega pelo ambiente em modo de "voar". Uma das desvantagens inerentes a este ambiente, relacionado principalmente com o tipo de segmentação aplicada prende-se com o facto de o utilizador não conseguir navegar continuamente pelas diversas áreas que constituem a cidade e, desta forma, reduzir o interesse da visita.

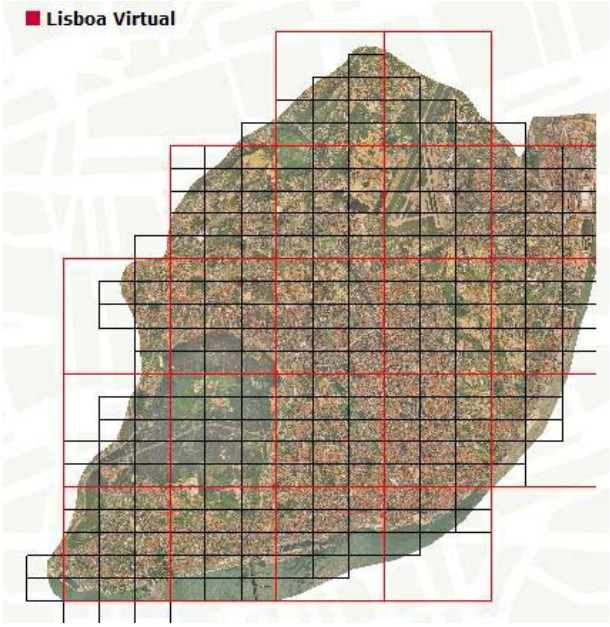


Figura 80 – Grelha de selecção do ambiente virtual Lisboa Virtual

Desta forma conclui-se que o modelo CidadePorto Virtual permite uma melhor integração do utilizador e um maior realismo comparativamente a um passeio de uma pessoa por uma cidade.

6.1.2 - Segmentação

A representação de cidades virtuais engloba uma grande quantidade e variedade de elementos como ruas, quarteirões, edifícios e jardins, dificultando o processo quando a informação a visualizar é lida de um servidor, sabendo à partida das limitações da largura de banda das redes. Desta forma é conveniente segmentar o modelo a representar com o intuito de possibilitar a qualquer utilizador o acesso à informação independentemente do dispositivo e serviço de acesso à rede que possui.

A segmentação consiste na divisão do ambiente virtual em partes menores permitindo, desta forma, melhorar o desempenho do sistema de visualização. Como foi referido na secção 4.2, existem diversas formas de segmentar um modelo. No presente caso foi utilizada uma segmentação adaptativa, isto é, uma grelha irregular definida com base nas ruas, enquanto que no modelo MRsCV foi aplicada uma malha regular quadrangular.

A segmentação aplicada no protótipo CidadePorto Virtual tem por base a malha dos arruamentos, sendo-lhe associadas as ruas que a intersectam, assim como os quarteirões e edifícios respectivos. A ideia subjacente é, de acordo com a posição do observador detectada a partir de sensores previamente definidos no modelo, carregar uma parte do modelo com os edifícios que ladeiam a rua em questão e omitir, por efeito da oclusão, outros edifícios mais afastados. Excepções a este efeito, como edifícios demasiado altos ou visíveis por efeito de falta de outros (jardins, espaços livres) são também tidos em conta. Esta forma de segmentar possibilita que o utilizador possa navegar de uma rua para outra sem ter obrigatoriamente que seleccionar a área que pretende visualizar. Sempre que é detectada a entrada do utilizador numa nova rua (Figura 81), a aplicação lê e actualiza, automaticamente, o ambiente. Esta é uma das vantagens inerentes à segmentação adaptativa, pois o utilizador pode navegar pelo ambiente virtual caminhando de umas ruas para as outras sem ter de estar constantemente a seleccionar a área que pretende visualizar e consequentemente esperar pelo seu carregamento. É o que acontece com o ambiente apresentado por Pimentel [Pimentel01], no qual o utilizador sempre que pretende visualizar uma nova área da cidade, tem de voltar a seleccionar a zona pretendida e esperar pelo carregamento do novo ficheiro.

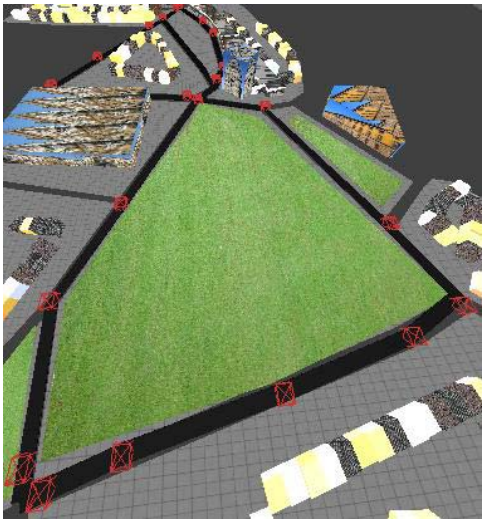


Figura 81 – Ambiente CidadePorto Virtual com os sensores visíveis

No caso do modelo MRsCV, visto a segmentação ser regular, são determinadas as ruas, os quarteirões e os edifícios que cada quadrícula contém ou intersecta. Um problema relativo a este tipo de solução diz respeito à ausência de elementos aquando da aproximação do utilizador às margens da quadrícula. Realmente, antes da detecção de aproximação do utilizador pelo sensor de fim de quadrícula, é visível o limite desta e a ausência do modelo para além do mesmo. Como exemplo deste problema de visualização é apresentada na Figura 82, uma imagem do ambiente virtual desenvolvido por Pimentel [Pimentel01]. Neste trabalho a segmentação aplicada baseia-se numa malha regular pré-definida (Figura 80), no qual é bastante perceptível o fim da quadrícula e a visualização de apenas alguns edifícios relevantes que, porventura, serão ainda referidos pela quadrícula corrente.

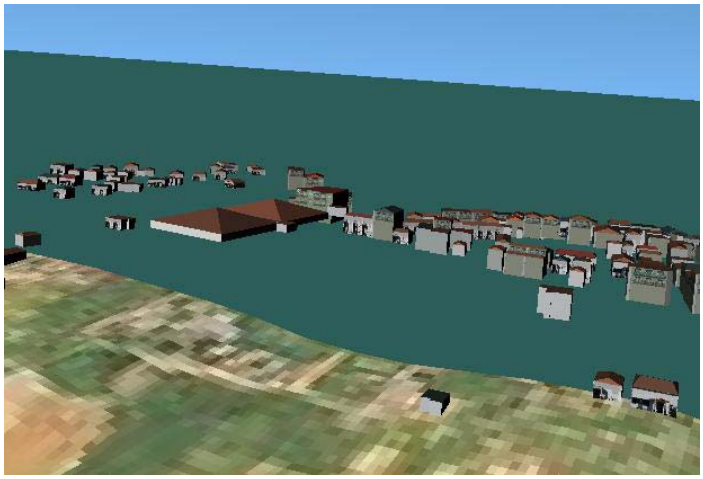


Figura 82 – Exemplo da ausência de elementos além do limite de uma quadrícula

A malha regular aplicada no modelo MRsCV foi do tipo malha fina possuindo cada quadrícula aproximadamente 12,69mx12,69m, sendo o mais comum a utilização de quadrículas com uma dimensão muito superior, como é o caso do trabalho desenvolvido por Pimentel [Pimentel01] que possui quadrículas de 800mx500m. No entanto, a utilização de quadrículas de grande dimensão origina um número elevado de objectos pertencentes a cada quadrícula. Desta forma, foi aplicado ao modelo MRsCV quadrículas de pequena dimensão com o objectivo de verificar se ainda assim existem diferenças significativas entre o número de objectos a representar para uma quadrícula do modelo MRsCV e uma rua no protótipo CidadePorto Virtual.

As Tabela 8 a 11 apresentam os diversos elementos, como ruas, quarteirões, edifícios e edifícios importantes, representados em ambos os protótipos, com o utilizador situado na Rua de Santa Teresa. Das tabelas apresentadas verifica-se que o número de ruas a representar é o mesmo em ambos os protótipos mas, relativamente ao número de quarteirões, de edifícios e edifícios importantes, o protótipo CidadePorto Virtual possuiu uma maior quantidade comparativamente ao protótipo modelo MRsCV. Esta diferença pouco significativa entre os dois protótipos e além disso desfavorável para o protótipo CidadePorto Virtual comprova que só utilizando uma malha fina no modelo MRsCV é que o número de elementos a representar para cada quadrícula se aproxima do número de elementos a representar para uma rua do protótipo CidadePorto Virtual. Não esquecendo que o ambiente CidadePorto Virtual inclui, ainda, edifícios importantes com altura relevante para um perímetro pré-definido e elementos (quarteirões, jardins, edifícios e edifícios importantes) associados a uma área verde, a uma área vazia e a uma área com objectos de pequeno porte, devido ao aumento da área visível ao observador, não analisados no Modelo MRsCV.

Tabela 8 – Ruas representadas a partir da R. de Santa Teresa em ambos os protótipos

	Ruas representadas
CidadePorto Virtual	R17, R30, R31, R32, R33, R58, R59, R67, R68, R84
Modelo MRsCV	R17, R30, R31, R32, R33, R58, R59, R67, R68, R84

Tabela 9 – Quarteirões representados a partir da R. de Santa Teresa em ambos os protótipos

	Quarteirões representados
CidadePorto Virtual	Q5, Q6, Q7, Q25, Q26, Q44, Q45, Q46
Modelo MRsCV	Q5, Q6, Q7, Q25, Q26, Q45, Q46

Tabela 10 – Edifícios representados a partir da R. de Santa Teresa em ambos os protótipos

	Edifícios representados
CidadePorto Virtual	E9, E11, E13, E14, E15, E16, E17, E37
Modelo MRsCV	E11, E13, E14, E15, E16, E17

Tabela 11 – Edifícios importantes representados a partir da R. de Santa Teresa em ambos os protótipos

	Edifícios importantes representados
CidadePorto Virtual	EI1, EI3
Modelo MRsCV	

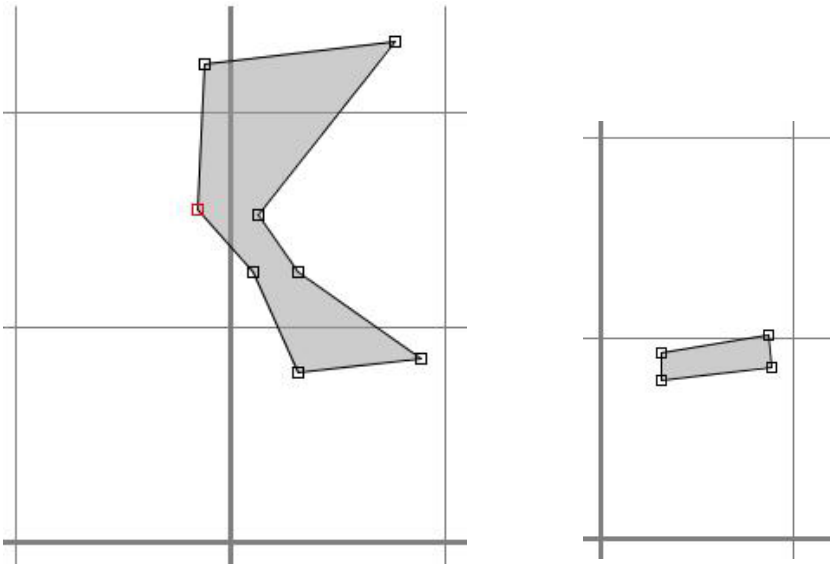
6.1.3 - Técnicas de Cálculo de Oclusão

Com o intuito de reduzir o número de objectos a representar num modelo, utiliza-se um ou mais métodos de cálculo de oclusão (Capítulo 2). O modelo definido para o protótipo CidadePorto Virtual teve por base duas técnicas de oclusão: a utilização de portais e células e a técnica de remoção rápida de objectos por volumes de sombra. A primeira técnica permite determinar, para os elementos que constituem o campo de visibilidade e com o observador numa determinada localização, os edifícios que originam maior oclusão (secção 4.5.1) isto é, os edifícios susceptíveis de serem considerados grandes oclusores. Consequentemente é construída a respectiva sombra, detectando-se os edifícios que intersectam esta ou que nela estão contidos (secção 4.5.2). Os elementos contidos na área de sombra são eliminados da lista de edifícios a representar, enquanto que os objectos que a intersectam são reduzidos à parte que se encontra fora da área. Analisando a Tabela 12 verifica-se que, após a aplicação das técnicas de cálculo de oclusão na Rua Professor José de Carvalho, por exemplo, alguns edifícios são substituídos por outros polígonos que representam uma área menor, como o edifício H4 com uma área de 0.669632 m^2 (Figura 83a)) é substituído por PP1 com uma área de 0.084159 m^2 (Figura 83b)). Com o cálculo de oclusão, é possível concluir que, do ponto de vista do utilizador, o polígono que constitui a base do edifício H4 (Tabela 13) só é visível numa pequena parte, sendo desnecessário representar a parte do edifício que não é visível e consequentemente obter uma diminuição no número de polígonos e tamanho do ficheiro respectivo. Pode-se concluir ainda que, após a aplicação das técnicas de cálculo de oclusão, dois dos edifícios definidos como visíveis foram eliminados (H1 e H3) por se encontrarem completamente oclusos por outros edifícios.

No Modelo MRsCV não se aplica qualquer tipo de técnica de cálculo de oclusão.

Tabela 12 – Edifícios a representar para a Rua Professor José de Carvalho

	Edifícios a representar
Antes da aplicação da técnica de cálculo de oclusão	E4, E5, E6, E7, E12, E18, E22, E23, E24, E31, E32, E52, EI0, EI1, EI3, H0, H1, H2, H3, H4, H5, I0, I1, I5
Após a aplicação da técnica de cálculo de oclusão	E4, E5, PP5, E7, PP4, PP6, PP14, E23, PP13, E31, E32, E52, EI0, EI1, PP2, PP0, 0, H2, 0, PP1, PP3, I0, PP12, PP11
Os elementos identificados na segunda linha da tabela com o sufixo 0 (zero) significam a eliminação do respectivo edifício.	



a) b)

Figura 83 – Representação do edifício H4 antes e depois de aplicada a técnica de cálculo de oclusão

Tabela 13 – Coordenadas do polígono H4 e PP1

	Coordenadas de um edifício
H4	[88.7068, 86.0837; 31.5767, 79.4534; 31.5767, 125.93; 10.3339, 125.93; 13.0698, 152.483; -14.9321, 155.139; -12.1963, 222.858; 76.798, 233.496]
PP1	[88.7068, 86.0837; 31.5767, 79.4534; 31.5767, 93.3899; 87.4194, 101.774]

Após uma análise comparativa entre o número de edifícios definidos como visíveis antes da aplicação do cálculo de oclusão e o número de edifícios considerados visíveis após o cálculo de oclusão, foram elaborados os Gráfico 1 e 2, respectivamente, com uma pequena amostra do número total de elementos avaliados. O Gráfico 2 apresenta para cada rua o total de edifícios a representar sendo que, a parte da barra horizontal de cor amarela representa o número de edifícios a representar que não sofrem qualquer tipo de alteração na sua configuração original enquanto que, a parte da barra horizontal de cor verde representa o número de edifícios a representar mas com uma área inferior ao edifício original.

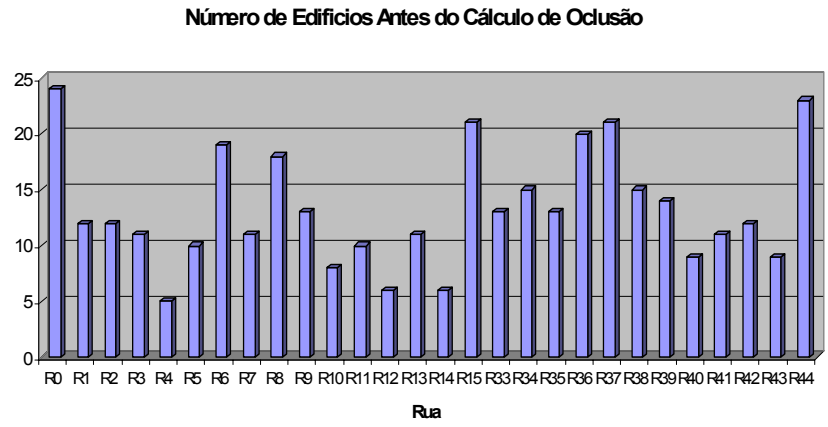


Gráfico 1 – Número de edifícios visíveis para cada rua antes da aplicação do cálculo de oclusão

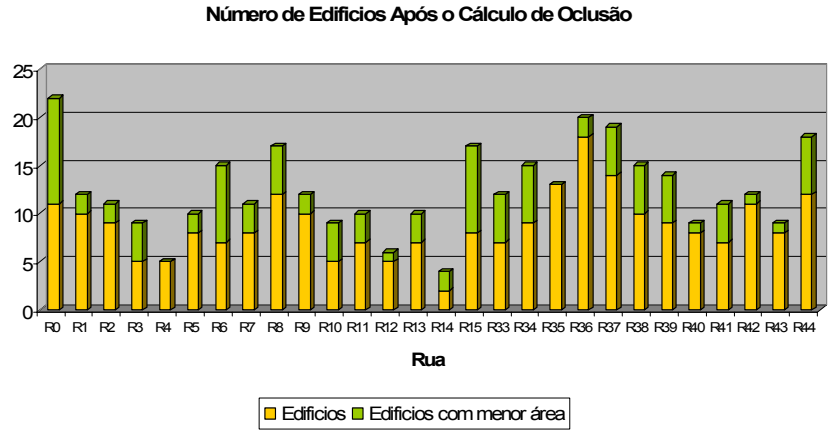


Gráfico 2 – Número de edifícios visíveis para cada rua após o cálculo de oclusão

Através dos Gráfico 1 e 2 pode verificar-se que, para algumas ruas, após o cálculo de oclusão, o número total de edifícios considerados visíveis diminui. Mais ainda, é grande a percentagem de edifícios visíveis que não serão representados na sua totalidade mas

numa área menor, permitindo assim, uma diminuição no número de polígonos e no tamanho no respectivo ficheiro. A Tabela 14 disponibiliza um resumo comparativo, para todas as ruas existentes no ambiente CidadePorto Virtual, entre o número e percentagem de edifícios considerados visíveis antes da aplicação das técnicas de cálculo de oclusão e consequente resultado após o cálculo de oclusão. Conclui-se que na globalidade 4% dos edifícios são eliminados com o cálculo de oclusão e que 27% dos edifícios considerados visíveis não são representados na sua totalidade mas apenas por uma pequena área.

Tabela 14 – Edifícios antes e após o cálculo de oclusão

		Número de edifícios	Percentagem de edifícios
Antes do cálculo de oclusão		868	
Após o cálculo de oclusão	**Edifícios**	601	69%
	Edifícios com menor área	234	27%
		Total	**96%**

6.2 - Avaliação CidadePorto Virtual

Com o objectivo do protótipo ser avaliado por um público-alvo, foi elaborado um inquérito (anexo C) a ser preenchido durante e após a navegação pelo ambiente. O universo de utilizadores para esta avaliação foi definido tendo em conta os conhecimentos na área da Informática e da Internet. Considerou-se pouco relevante que este grupo de utilizadores fosse heterogéneo, o comum neste tipo de inquéritos, visto que se pretendia que estes fossem bastantes exigentes e precisos durante a avaliação.

Durante a avaliação, os utilizadores foram confrontados com dois protótipos: o CidadePorto Virtual (referido como protótipo A no texto seguinte) e CidadePorto Virtual Completa (referido como protótipo B no texto seguinte). A diferença entre os dois protótipos é na ausência de segmentação e da aplicação de técnicas de cálculo de oclusão no protótipo B. Desta forma, o protótipo B apresenta toda a área da cidade do Porto modelada num único ficheiro.

A existência de dois protótipos não foi exposta aos utilizadores com o intuito de consolidar respostas do inquérito. Assim, ao utilizador foi apresentado um protótipo inicial para percorrer um passeio pré-definido, e durante o qual e após o seu término, respondeu a duas secções do inquérito. Após uma breve pausa, o utilizador foi convidado a percorrer o mesmo passeio pré-definido e a responder a duas novas secções do inquérito, uma primeira secção a responder durante o percurso e uma segunda secção a

responder após o seu término. Nesta segunda visita o protótipo apresentado foi diferente sem que o utilizador fosse avisado, referindo-se apenas que a nova sessão tinha a finalidade de validar as respostas anteriores.

O inquérito foi dividido em 7 secções, com objectivos concretos:

1ª secção (Identificação do Utilizador):	- obter informações relativas ao utilizador - definir o perfil de utilização em contexto 3D
2ª secção (Navegação livre):	- permitir ao utilizador um primeiro contacto com o protótipo B
3ª secção (Navegação Orientada):	- obter informações relativas à importância da visualização de objectos que se encontram do lado oposto ao observador com um jardim ou uma área aberta na sua frente - o papel desempenhado por um oclusor assim como a função relevante desempenhada por um edifício com uma altura superior à média.
4ª secção (Conclusão da Navegação Orientada):	
5ª secção (Novas impressões – Navegação Orientada):	- idênticos aos examinados nas duas secções anteriores sendo apenas alterado a sequência do propósito a avaliar
6ª secção (Conclusão da Navegação Orientada):	
7ª secção (Questões Finais):	- análise global onde o utilizador pode ainda escrever sugestões e comentários

Na terceira e quarta secção o utilizador executa um percurso pré-definido no protótipo B, sendo a primeira secção respondida durante a navegação enquanto que a segunda é respondida após concluída a tarefa e terminada a aplicação. As questões das secções cinco e seis dizem respeito ao protótipo A, seguindo a mesma sequência das duas secções anteriores.

6.2.1 - Caracterização da Amostra

O inquérito foi respondido por um universo de 49 alunos do 3ºano do curso de Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto, caracterizado por: 96% do sexo masculino e 4% do sexo feminino.

Dos resultados apurados pode concluir-se que, no que diz respeito ao perfil de utilização em contexto 3D, 74% dos alunos nunca teve, anteriormente, experiências de navegação em linguagem VRML ou equivalente, 6% experimentou uma única vez e 20% experimentou várias vezes, como se constata no Gráfico 3. Em virtude da maioria dos utilizadores nunca ter tido experiências em linguagem VRML e apesar de se integrarem numa população naturalmente hábil na utilização de ferramentas computacionais e da Internet, é compreensível que tenham alguma dificuldade na realização deste teste. Por outro lado, esta situação torna-se uma vantagem pois, sendo um utilizador inexperiente, acaba por realizar erros como um utilizador comum mas com uma capacidade crítica que lhe permite emitir sugestões interessantes de forma a tornar o protótipo acessível a qualquer utilizador.

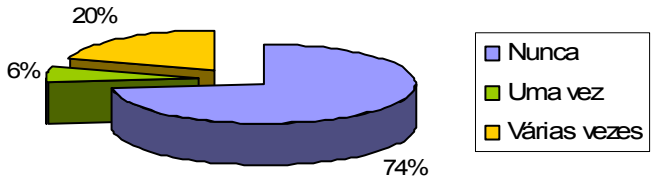


Gráfico 3 – Resultado da questão 'Perfil de utilização em contexto 3D'

6.2.2 - Navegação Livre

Na secção dois, Navegação livre, foi definida uma duração de cerca de 10 minutos para o utilizador navegar livremente pelo protótipo, de modo a estabelecer um primeiro contacto. Esta primeira navegação no ambiente permitiu ao utilizador conhecer as teclas de controlo no computador, de modo a conseguir caminhar pelas ruas e, para os mais experientes nesta tecnologia, tomar conhecimento que o modo de navegação se limita ao modo em 'passeio'.

6.2.3 - Navegação Orientada

A navegação orientada diz respeito às secções três, quatro, cinco e seis do inquérito sendo que, nas duas primeiras, o utilizador navega no protótipo B, e nas duas últimas navega no protótipo A. A navegação para estas secções foi definida como orientada, visto que o utilizador tem de percorrer um trajecto pré-definido de modo a que todos os utilizadores transitem pelos mesmos locais, tendo estes aspectos essenciais de avaliação. Como referido anteriormente, as questões das secções três e cinco são respondidas durante a navegação do utilizador pelo ambiente enquanto que as questões das secções quatro e seis

são respondidas depois de terminada a visita e finalizada a aplicação. O objectivo da separação das secções quatro e seis das restantes deve-se ao tipo de questões e área que se pretende analisar.

Esta secção divide-se em cinco subsecções tendo em conta o objectivo a analisar no inquérito durante a navegação orientada: análise das áreas verdes, áreas vazias e áreas com objectos de pequeno porte; análise do papel de oclusor; análise de edifícios com uma altura superior à média; interface e navegabilidade; questões finais. Recomenda-se, para o efeito a consulta do inquérito realizado, que consta no Anexo C.

6.2.3.1. Análise das Áreas Verdes, Áreas Vazias e Áreas com Objectos de Pequeno Porte

As alíneas 3.1, 3.3 e 5.2 visam avaliar a importância da inclusão, na fase de segmentação (secção 4.3), de todos os elementos associados às zonas que circundam as áreas verdes, áreas vazias ou áreas com objectos de pequeno porte, com vista a uma perfeita identificação do local. A resposta a estas alíneas é feita através de uma escala numérica quantitativa em que o valor 1 corresponde a uma fraca apreciação e o valor 5 representa a melhor avaliação possível. Quando inquiridos relativamente à visibilidade da Faculdade de Ciências na Travessa de S. Bento e os monumentos existentes na Avenida dos Aliados, a maioria atribuiu uma apreciação de valor 4 (Gráfico 4).

De acordo com estes resultados, pode constatar-se que a inclusão da Faculdade de Ciências e dos monumentos existentes na Avenida dos Aliados é fundamental para a identificação do local devido à facilidade com que os identificaram. A título de exemplo, caso na fase de segmentação (secção 4.3) não tivesse sido prevista esta situação das áreas verdes, áreas vazias ou áreas com objectos de pequeno porte, os utilizadores seriam confrontados com uma área vazia isto é, ao se aproximarem de, por exemplo, um jardim, podiam constatar a inexistência de outros elementos do outro lado do jardim, problema semelhante ao encontrado em [Pimentel01].

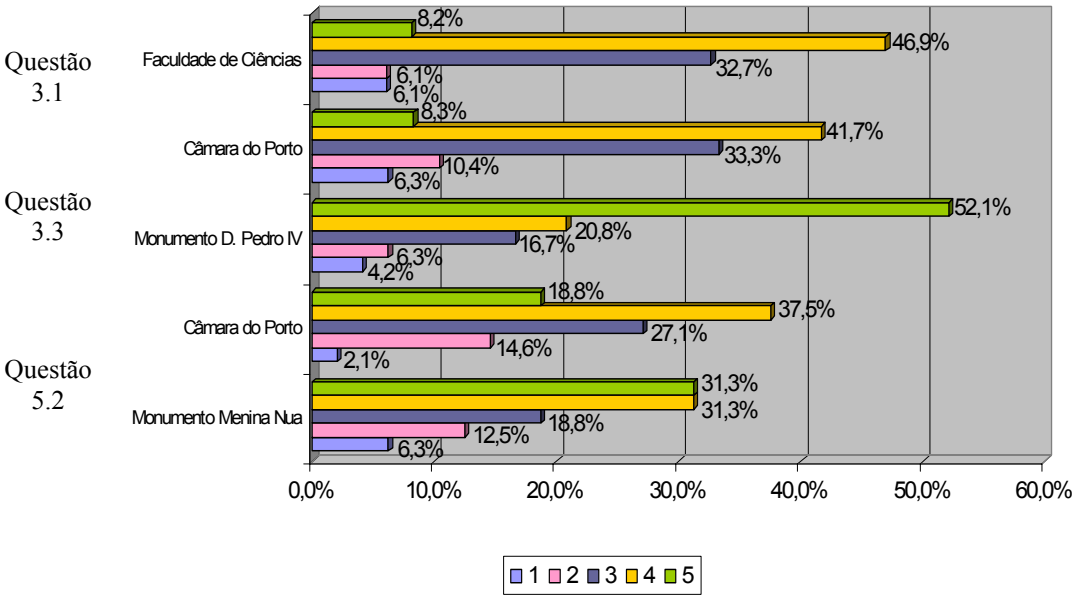


Gráfico 4 – Resultado das questões 3.1, 3.3 e 5.2

6.2.3.2. Análise do Papel de Oclusor

A fase do cálculo de oclusão (secção 4.5) é extremamente importante porque, com a eliminação de partes do ambiente que não contribuem significativamente para a imagem final, é possível uma aceleração nos tempos de representação das cenas. Pretendia-se com as questões 3.2 e 6.2 verificar qual a importância da representação dos diversos elementos que se encontram por trás de um oclusor. Desta forma, para a questão 3.2, a Igreja do Carmo desempenhou o papel de oclusor, tendo sido calculada a área visível do Hospital de S^{to}. António. Para a questão 6.2, referente ao protótipo B, no qual não foi aplicada a técnica de cálculo de oclusão, a Igreja dos Clérigos foi totalmente modelada. Nos Gráfico 5 e 6, é possível perceber que a representação total da Igreja dos Clérigos não torna a sua identificação mais fácil em comparação com o Hospital de S^{to}. António. Contudo, o protótipo B origina um ficheiro de dimensão superior ao protótipo A.

Verifica-se assim que, quando existe um elemento com o papel de oclusor e se pretende reduzir ao máximo a dimensão do ficheiro, é vantajosa a aplicação da técnica de cálculo de oclusão descrita na secção 4.5, dado que não altera significativamente a percepção do modelo.

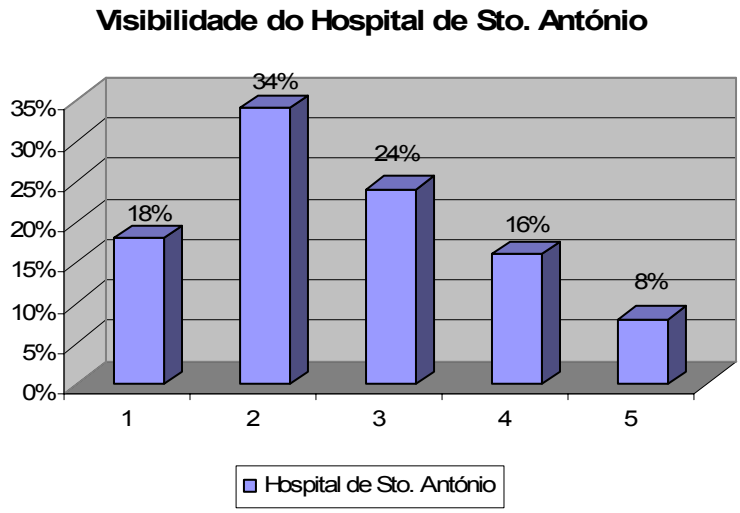


Gráfico 5 – Resultado da questão 3.2

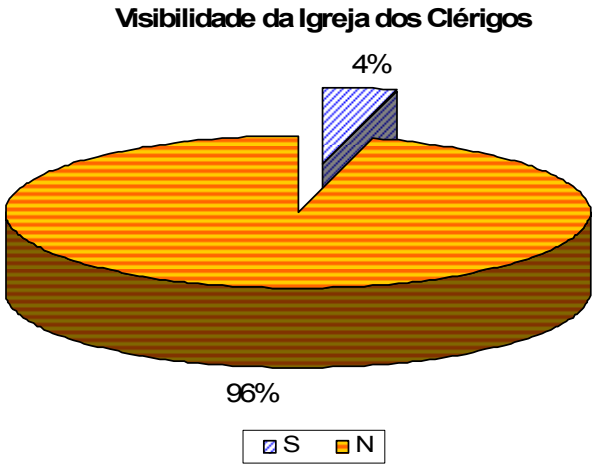


Gráfico 6 – Resultado da questão 6.2

6.2.3.3. Análise de Edifícios com uma Altura Superior à Média

A segmentação de um ambiente virtual urbano obriga à consideração de vários aspectos relacionados com a visualização de edifícios que, embora se situem dentro da área de sombra, são parcialmente visíveis devido à sua maior altura (secção 4.3). As questões 4.2 e 5.1 pretendem averiguar a relevância da representação de edifícios altos, pois estes caracterizam a cidade, permitindo ao utilizador obter pontos de referência durante a sua navegação no ambiente virtual. Desta forma, estes edifícios devem ser incluídos no ambiente de forma a que o utilizador os possa identificar mesmo que se encontre a uma certa distância. Conforme se verifica nos Gráfico 7 e 8, a Torre dos Clérigos, situada

dentro de uma área de sombra para a localização do utilizador, mas visível devido à sua altura ser superior à média, foi facilmente identificada, apresentando uma percentagem de 67% para o protótipo A e um total de 77,6% para o protótipo B.

Constata-se assim que, durante a fase de segmentação, a inclusão de edifícios com a característica de estarem dentro da área de sombra mas com uma altura superior à média, é obrigatória, visto que estes edifícios podem apresentar um peso significativo na visualização do ambiente virtual urbano.

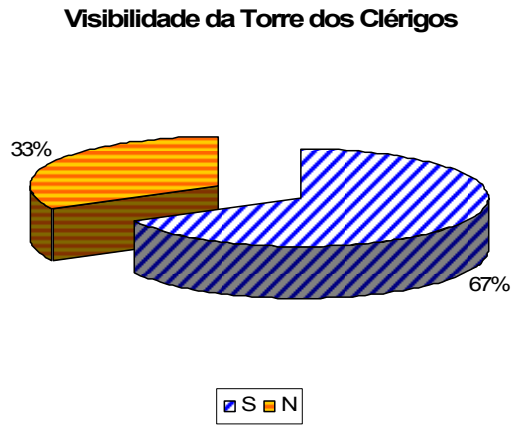


Gráfico 7 – Resultado da questão 4.2

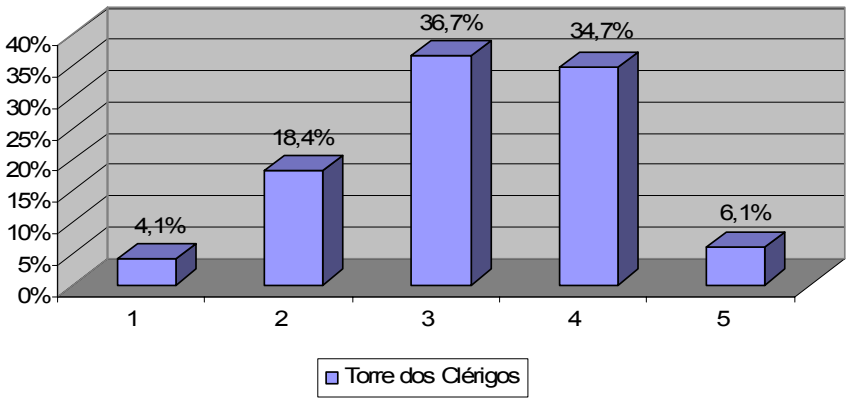


Gráfico 8 – Resultado da questão 5.1

6.2.3.4. *Interface* e Navegabilidade

Como referido na secção 5.8, a *interface* deve ser amigável e funcional para qualquer tipo de utilizador e independente da plataforma. Desta forma, foi necessário dividir o ambiente de trabalho em duas áreas: uma área do lado esquerdo que contém o nome da

rua onde se encontra posicionado o utilizador e uma área do lado direito com o ambiente virtual urbano propriamente dito. No que concerne à navegabilidade, a *interface* do protótipo A dispõe de dois pontos-chave diferentes do protótipo B. Um primeiro ponto é relativo ao tempo de espera na actualização da nova rua quando o utilizador passa de uma rua para outra; o segundo ponto refere-se à orientação do utilizador quando é posicionado na nova rua em que entrou, após a respectiva actualização. Visto que o protótipo B contém todo o ambiente virtual urbano modelado num único ficheiro, os pontos-chave referidos para o protótipo A não existem. Tendo em conta a importância da *interface* e da navegação num ambiente virtual urbano e com o intuito de não centralizar a avaliação do protótipo apenas na respectiva segmentação e cálculo de oclusão, achou-se conveniente, considerar estes dois aspectos. Dos resultados obtidos (Gráfico 9), acabou por se concluir que foi fácil a execução do percurso pedido. Uma maioria (65%) considerou importante a identificação da rua na *interface*. Relativamente ao tempo de espera de apresentação de cada rua, uma proporção elevada de utilizadores (39,6 + 25 = 64,6%) expressou uma satisfação geral, bem como à orientação da posição do utilizador na nova rua.

Analisando os resultados obtidos, pode concluir-se que a navegação pelo ambiente urbano virtual não foi tão fácil como se previa, possivelmente devido à inexperiência por parte dos utilizadores, como foi salientado em virtude dos resultados obtidos na secção 1 do inquérito.

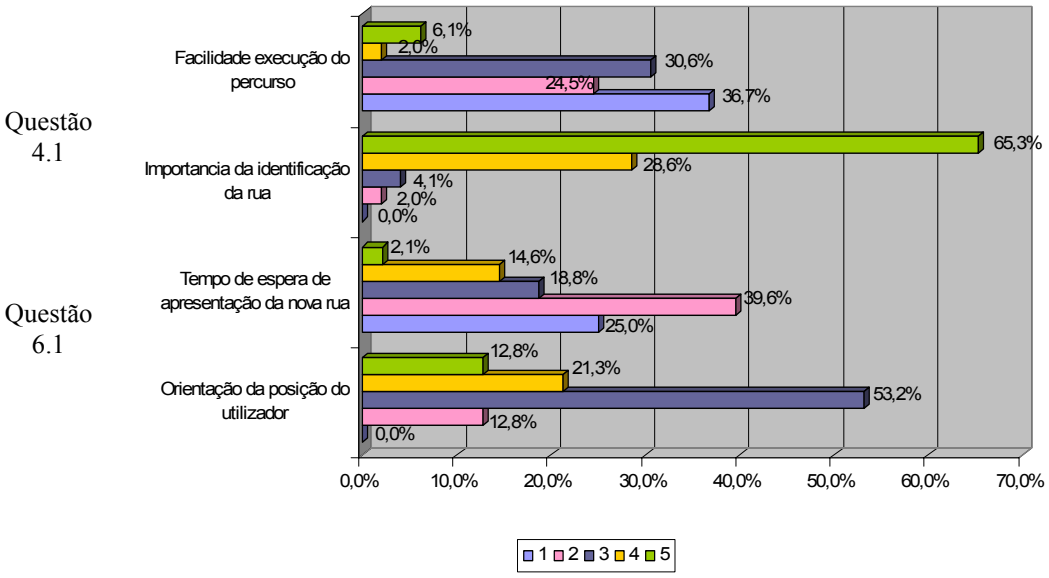


Gráfico 9 – Resultado da questão 4.1 e 6.1

6.2.3.5. Questões Finais

A secção sete do inquérito, como referido na secção 6.2, destina-se a fomentar os utilizadores a proporem sugestões e comentários à aplicação, assim como obter uma apreciação global e verificar se, de alguma forma, foi detectada a mudança de um protótipo para outro. Desta forma foram definidas três questões que abrangessem estes aspectos. Quando inquiridos relativamente às diferenças encontradas nos objectos representados em ambos os protótipos (Gráfico 10), 73,3% não encontrou nenhuma diferença. Mostra-se assim a vantagem da metodologia desenvolvida para esta dissertação (capítulo 4), visto permitir a visualização sem prejuízos graves de ambientes virtuais urbanos com ficheiros de dimensão reduzida e, portanto, mais facilmente transmitidos através da rede e mais eficientemente visualizáveis em *hardware* de recursos limitados.

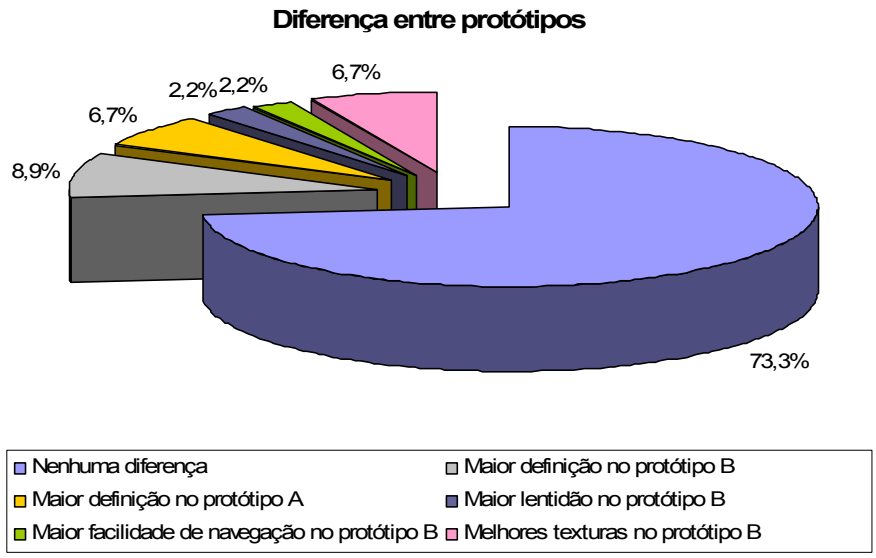


Gráfico 10 – Resultado da questão 7.1

As questões 7.2. e 7.3 têm como objectivo proporcionar a referida área de sugestões e comentários, avaliando ainda o trabalho desenvolvido na globalidade.

No Gráfico 11, é possível perceber que a avaliação global do trabalho desenvolvido é bastante boa, tendo a maioria considerado como úteis os recursos e funcionalidades disponibilizados pelo protótipo em avaliação. No Gráfico 12 identificam-se as principais sugestões indicadas pelos utilizadores: a possibilidade de navegação em diferentes modos (27,1%); a aplicação e tratamento de texturas com melhor qualidade (27,1%); a capacidade de localização interactiva do utilizador aquando da navegação no ambiente virtual relativamente ao mapa que se encontrava na *interface* (16,9%). As sugestões apresentadas ficam fora do âmbito desta dissertação, mas abordam aspectos que poderão vir a ser implementados em trabalhos futuros.

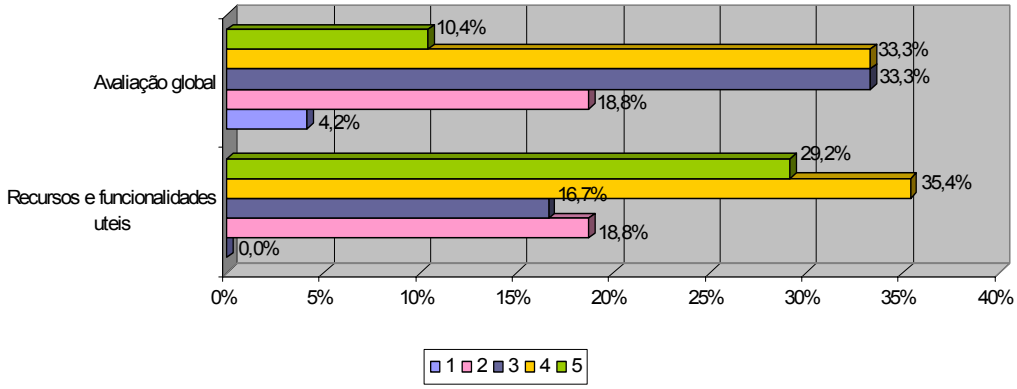


Gráfico 11 – Resultado da questão 7.2

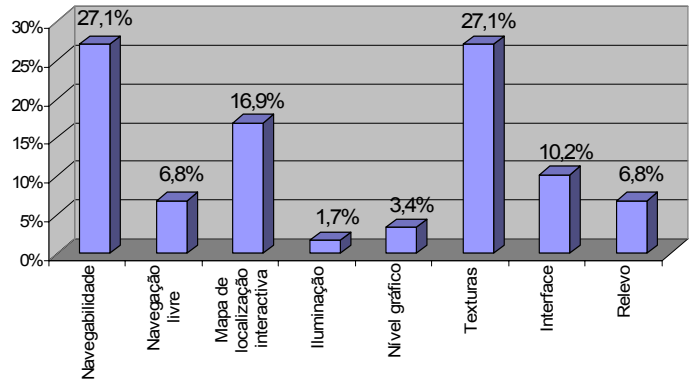


Gráfico 12 – Resultado da questão 7.3

6.3 - Síntese do Capítulo

O protótipo CidadePorto Virtual foi desenvolvido de forma a gerar um ambiente virtual urbano, com modo de navegação do tipo "passeio", sendo a informação automaticamente apresentada em função da localização do utilizador.

Para efeitos de avaliação da metodologia desenvolvida, este protótipo foi sujeito a uma comparação com o protótipo CidadePorto Virtual MRsCV (secção 6.1) e a um teste de avaliação (secção 6.2)

Na secção 6.1, foi feita uma avaliação comparativa entre o protótipo CidadePorto Virtual e o protótipo CidadePorto Virtual MRsCV (Modelo MRsCV), com base em diversos

critérios, tais como: navegação, segmentação e técnicas de cálculo de oclusão. O Modelo MRsCV diferencia-se da CidadePorto Virtual por possuir uma segmentação em malha regular quadrangular e não implementar qualquer técnica de cálculo de oclusão. Relativamente ao critério de navegação submetido na avaliação comparativa, o protótipo CidadePorto Virtual permite que o utilizador se sinta envolvido no ambiente onde se encontra a navegar relativamente ao Modelo MRsCV. A segmentação implementada na CidadePorto Virtual permite que o utilizador navegue livremente de uma rua para a outra, sendo o ambiente virtual actualizado automaticamente. De acordo com o posicionamento do utilizador numa rua, define-se um campo de visão a partir do qual se apresentam os edifícios potencialmente visíveis, tendo em atenção a transparência dos quarteirões visíveis ou preenchidos por espaços ajardinados. Por outro lado, o Modelo MRsCV apresenta um comportamento pouco satisfatório aquando da aproximação do utilizador às margens da quadrícula, sendo perceptível o fim da quadrícula e a visualização de apenas alguns edifícios relevantes. Também, neste protótipo, os elementos utilizados não são em função do campo de visão do utilizador mas com base na quadrícula previamente seleccionada. Em relação à técnica de cálculo de oclusão, apenas o protótipo CidadePorto Virtual usufruiu desta, permitindo que edifícios que se encontrem fora do campo de visão do utilizador sejam removidos, possibilitando uma aceleração nos tempos de representação da cena e no tamanho do ficheiro.

A outra forma utilizada para a avaliação da metodologia foi a realização de um inquérito a um público-alvo que expôs a sua avaliação e sugestões enquanto navegava com dois protótipos diferentes, um dos quais, somente, dotado da metodologia em causa. Pelos resultados obtidos, julga-se que a metodologia desenvolvida dá uma resposta adequada às principais funcionalidades e requisitos do sistema proposto nesta dissertação.

CAPÍTULO 7

CONCLUSÕES

Índice do Capítulo

7 - Conclusões

Este capítulo apresenta as conclusões obtidas com esta dissertação demonstrando as contribuições feitas no estudo e desenvolvimento do projecto e as propostas para trabalhos futuros.

O objectivo para esta dissertação foi desenvolver uma metodologia que permitisse, de uma forma dinâmica, a visualização interactiva de ambientes urbanos virtuais extensos em dispositivos de recursos limitados e que possuem um serviço de acesso limitado à rede.

A segmentação desempenha um papel vital na metodologia desenvolvida, tendo sido definida de modo dinâmico e em função do utilizador. Desta forma, foi concebida uma segmentação adaptativa, isto é, baseada numa grelha irregular definida com base nas ruas. Este tipo de segmentação implicou a apreciação de vários aspectos como a definição para cada rua dos seus elementos associados, mas de forma a permitir a visualização de edifícios com uma altura superior à média colocados fora do campo de visão do utilizador, assim como de outras áreas quando o utilizador se encontra numa zona vazia, numa zona verde ou numa área com monumentos de pequeno porte (secção 4.3). Concluída a fase de segmentação, foi necessário determinar o campo de visibilidade em função do utilizador (secção 4.4). Esta fase consiste no cálculo dos elementos visíveis a partir da localização do utilizador, isto é, em função da rua onde se encontra posicionado o utilizador. Dos elementos visíveis faz parte a rua onde está posicionado o utilizador, o conjunto de ruas que formam um cruzamento com a rua do utilizador, o conjunto de quarteirões que circundam as ruas referidas anteriormente, as linhas de comboio que cruzam a rua principal e todos os elementos associados a cada quarteirão como edifícios sem interesse público, os jardins, os monumentos, as estações de comboio, as igrejas e os hospitais.

A utilização de técnicas de redução da complexidade tem a finalidade de tornar mais rápido o processamento e a visualização de uma cena permitindo obter ficheiros bastante

mais pequenos e simples para poderem ser representados, resultando ainda em tempos de espera significativamente menores (secção 4.5). Tendo em conta os objectivos para esta dissertação, implementou-se, com as devidas adaptações, a técnica de remoção rápida de objectos através das respectivas sombras de Hudson [Hudson97] e a técnica de portais e células apresentada por Luebke [Luebke95]. O cálculo de oclusão foi dividido em duas etapas: Numa primeira etapa, são definidos os edifícios que originam uma maior oclusão, de forma a reduzir substancialmente a quantidade de edifícios a calcular a respectiva sombra. Posteriormente, numa segunda etapa, para cada oclusor resultante da fase anterior, é construída a respectiva sombra e determinados os objectos que a intersectam ou que lhe estam contidos. Todos os edifícios contidos na área de sombra são eliminados da lista de elementos a representar e os que apenas intersectam a área de sombra são redefinidos para a parte não contida na área de sombra.

Comparativamente às soluções com uma abordagem convencional, o protótipo CidadePorto Virtual, desenvolvido por efeitos de confirmação da metodologia anterior, possibilita ao utilizador uma navegação livre entre as diversas ruas, pois sempre que o utilizador entra numa nova rua, o protótipo automaticamente lê e actualiza o ambiente.

Para a representação de ambientes virtuais na Internet e interacção com o utilizador foi seleccionada, inicialmente, a linguagem VRML [VRML97] e a tecnologia EAI. Esta linguagem possibilita a visualização de ambientes virtuais por qualquer pessoa que tenha acesso à Internet, independentemente da plataforma que esteja a usar. No entanto, devido aos diversos problemas existentes no VRML [VRML97], o Web3D *Consortium* evoluiu esta especificação desenvolvendo e adoptando o X3D [X3D]. A comunicação com a cena X3D é feita utilizando a *interface* SAI, a qual elimina a incompatibilidade entre as diversas linguagens de programação ligadas ao X3D e possibilita implementações para todos os *browsers* em qualquer plataforma.

Os resultados obtidos através da análise comparativa entre o protótipo CidadePorto Virtual e a CidadePorto Virtual MRsCV e, do inquérito realizado aos alunos, confirmam as potencialidades da metodologia desenvolvida e o útil contributo dos recursos disponibilizados.

7.1 - Perspectivas de Trabalho Futuro

O desenvolvimento futuro desta dissertação foca-se na segmentação e cálculo de oclusão, podendo ser explorado, também, o aspecto gráfico, a *interface* e a navegabilidade.

Os processos de segmentação e cálculo de oclusão ainda não incluem o relevo do terreno. As formas de relevo do terreno exercem um papel decisivo no campo de visão do utilizador visto que, se o utilizador estiver localizado num ponto alto do ambiente virtual, o seu campo de visão se estende por uma área superior. Por outro lado, se o utilizador estiver situado num ponto baixo do ambiente virtual, o seu campo de visão será reduzido.

Desta forma, a definição dos elementos que estão associados a uma determinada rua deve ter em conta o aspecto relevo.

Num segundo momento, poderão ser desenvolvidos alguns aspectos relacionados com a implementação resultante da metodologia aplicada como o aspecto gráfico, a *interface* e a navegabilidade. O aspecto gráfico do ambiente virtual poderá sofrer um melhoramento na aplicação de texturas e relevo de melhor qualidade de forma a obter uma elevada qualidade visual.

O outro aspecto a ser explorado refere-se ao mapa introduzido na *interface* do protótipo CidadePorto Virtual, o qual poderia indicar interactivamente a posição do utilizador à medida que este navega pelo mundo virtual, de forma a que o utilizador conheça a sua localização.

Também poderá ser levado em consideração para aperfeiçoamento, a possibilidade de o utilizador poder navegar pelos passeios da rua e não apenas pela própria rua.

Por último, considera-se necessário o melhoramento na localização e modo de funcionamento dos sensores no ambiente com o objectivo de permitir uma maior automatização na actualização das ruas durante a navegação do utilizador.

CAPÍTULO 8

REFERÊNCIAS BIBLIOGRÁFICAS

8 - Referências Bibliográficas

[3DFunchal] 3D Funchal
http://www.3dfunchal.com

[Agarwal01] Agarwal, Pankaj K.; Har-Peled, Sariel e Wang, Yusu (2001), "Occlusion Culling for Fast Walkthrough in Urban Areas". Eurographics 2001.

[Andersson06] Andersson, Mattias; Gudmundsson, Joachim e Levcopoulos, Christos (2006), "Restricted Mesh Simplification Using Edge Contractions". European Workshop on Computational Geometry, Delphi.
http://delaunay.tem.uoc.gr/~mkaravel/ewcg06/papers/29.pdf

[ArcView] ESRI ArcView
http://www.esri.com/software/arcgis/arcview/index.html

[Arnold99] Arnold, D.B. e Day, A.M. (1999), "The Norwich Virtual City Project". Eurographics' 99, Short Papers and Demos.

[Assarsson00] Assarsson, Ulf; Möller, Tomas (2000), "Optimized View Frustum Culling Algorithms for Bounding Boxes". ACM Journal of Graphics Tools, 5(1):9-22. http://www.ce.chalmers.se/~uffe/

[Bitmanagement] Bitmanagement Software GmbH
http://www.bitmanagement.de/

[Bittner99] Bittner, Jiri; Havran, Vlastimil e Slavik, Pavel (1999), "Hierarchical Visibility Culling with Occlusion Trees". Department of Computer Science and Engineering Czech Technical University.

[Bittner01] Bittner, Jiri; Wonka, Peter e Wimmer, Michael (2001), "Visibility Preprocessing for Urban Scenes using Line Space Subdivision". Proceedings of Pacific Graphics, volume 19, páginas 276-284.

[Bittner03] Bittner, Jiri e Wonka, Peter (2003), "Visibility in Computer Graphics". Journal of Environment and Planning B: Planning and Design volume 5, número 30, páginas 729-756, Pion Ltd.
http://citeseer.ist.psu.edu/bittner03visibility.html

[Campbell03] Campbell, Matthew (2003), "Ray Tracing". Computer Science Department, Virginia.
http://pixel.cs.vt.edu/courses/cs4204.html

[CaR] CaR, Software de Geometria Dinâmica
http://www.professores.uff.br/hjbortol/index.html

[Carey97] Carey, Rikk; Bell, Gavin (1997), "The Annotated VRML 2.0 Reference Manual". 4ª edição, Estados Unidos da América, Addison-Wesley, 1997.

[Chamberlain96] Chamberlain, Bradford; DeRose, Tony; Lischinski, Dani; Salesin, David e Snyder, John (1996), "Fast rendering of complex environments using a special hierarchy". Proceedings of Graphics Interface '96, páginas 132-141.
http://grail.cs.washington.edu/pub/abstracts.html

[Chierici04] Chierici, Carlos; Ribeiro, Hebert; Rocca, Renato (2004), "VRML e X3D". Seminário de Hipermídia, ICMC
http://catuaba.icmc.usp.br/medias/1o_2004/Hipermidia/VRML&X3D_12_08/background/X3D_Seminario_Apresenta%3Fao%20final.ppt

[Cohen98] Cohen-Or, Daniel; Fibich, Gadi; Halperin, Dan e Zadicario, Eyal (1998), "Conservative Visibility and Strong Occlusion for Viewspace Partitioning of Densely Occluded Scenes". Eurographics' 98, volume 17, número 3.

[Cohen03] Cohen-Or, Daniel; Chrysanthou, Yiorgos; Silva, Cláudio T. e Durand, Frédo (2003), "A Survey of Visibility for Walkthrough Applications". IEEE TVCG.
http://www.cs.tau.ac.il/~dcor/online_papers/index.html

[Constantinescu01] Constantinescu, Zoran (2001), "Levels of detail: An overview". Norwegian University of Science and Technology.
http://csgsc.idi.ntnu.no/2001/pages/papers/zoran.pdf

[Copper99] Cooper, J. W.; Arnorld, D. B. e Day, A. M. (1999), "Rapid Urban Modelling". Eurographics' 99, Short Papers and Demos.

[Crispen00] Crispen, Bob (2000), "VRML Works".
http://vrmlworks.crispen.org/

[Decoret99] Decoret, Xavier; Schaufler, Gernot; Sillion, François e Dorsey, Julie (1999), "Multi-layered impostors for accelerated rendering". Eurographics' 99, volume 18, número 3.

[Decoret02] Décoret, Xavier; Durand, Frédo; Sillion,, François e Dorsey, Julie (2002), "Billboard Clouds". Project iMAGIS. Relatório de investigação nº 4485, Junho 2002, 20 páginas.

[Durand00] Durand, Frédo; Drettakis, George; Thollot, Joëlle e Puech, Claude (2000), "Conservative Visibility Preprocessing using Extended Projections". Proceedgins of Siggraph 2000.
http://people.csail.mit.edu/fredo/sig2000/

[Eclipse] Eclipse – an open development platform
http://www.eclipse.org/

[Esperança06] Esperança, Cláudio e Cavalcanti, Paulo (2006), "Introdução à Computação Gráfica – Visibilidade e Recorte".
http://www.lcg.ufrj.br/Cursos/COS-751/visibilidade-ppt

[Fernandes06] Fernandes, António (2006), "Fundamentos de Computação Gráfica".
http://sim.di.uminho.pt/disciplinas/fcg/curr/aula5/a5.pdf

[Fernandes06a] Fernandes, António (2006), "Computação gráfica".
http://sim.di.uminho.pt/disciplinas/mm/0607/t10/t10.pdf

[Ferreira06] Ferreira, Vasco (2006), "Os SIG e a Administração Local". Dossiês Especiais Informática – Sistemas de Informação Geográfica. Ferramentas de Aplicação Universal, edição nº 1735 do Expresso de 28 de Janeiro de 2006.

[Fleischmann99] Fleischmann, Peter (1999), "Mesh Generation for Technology CAD in Three Dimensions". Tese de mestrado pela Faculdade Electrónica da Universidade de Wien. Wien.
http://www.iue.tuwien.ac.at/phd/fleischmann/node68.html

[Foley97] Foley, James; Dam, Andries; Feiner, Steven; Hughes, John e Philips, Richard (1997), "Introduction to Computer Graphics". Reading, MA. Addison-Wesley Publishing Company.

[Funkhouser93] Funkhouser, Thomas A. e Séquin, Carlo H. (1993), "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments". Computer Graphics (Siggraph' 93), Los Angeles, páginas 247-254.
http://www.cs.princeton.edu/~funk/sig93.pdf

[Garland99] Garland, Michael (1999), “Survey of Multiresolution Modeling”. http://www.cs.cmu.edu/afs/cs/user/garland/www/multires/survey.html

[Shen06] Shen, Han-Wei (2006), “Spacial Data Structures and Culling Techniques”, CSE 781.

[Haumont03] Haumont, Denis; Debeir, Olivier e François, Sillion (2003), “Volumetric cell-and-portal generation”. Computer Graphics Forum, volume 22, número 3, páginas 303 – 312. http://artis.imag.fr/Publications/2003/HDS03/mcp.pdf

[Hudson97] Hudson, T.; Manocha, D.; Cohen, J.; Lin, M.; Hoff, K. e Zhang, H. (1997), “Accelerated Occlusion Culling Using Shadow Frusta”. Simposium on Interactive 3D Graphics, paginas 1-10.

[Jeschke05] Jeschke, Stefan; Wimmer, Michael, Schumann, Heidrun e Purgathofer, Werner (2005), “Automatic Impostor Placement for Guaranteed Frame Rates and Low Memory Requirements”. ACM Press. http://www.cg.tuwien.ac.at/research/ publications/2005/jeschke-05-AIP/jeschke-05-AIP-Paper.pdf

[Jiménez00] Jiménez, W.F.H.; Esperança, C.; Oliveira, A.A.F. (2000), “Efficient Algorithms for Computing Conservative Portal Visibility Information”. Eurographics 2000, volume 19, número 3.

[Kultun00a] Kultun, Vladlen; Chrysanthou, Yiorgos e Cohen-Or, Daniel (2000), “Virtual Occluders: an Efficient Internediate PVS Representation”. 11th Eurographics Workshop on Rendering, páginas 59-70.

[Kultun00] Koltun, Vladlen e Cohen-Or, Daniel (2000), “Selecting Effective Occluders for Visibility Culling”. Eurographics 2000, Short Presentations.

[Kultun01] Kultun, Vladlen; Chrysanthou, Yiorgos e Cohen-Or, Daniel (2001), “Hardware-accelerated from-region visibility using a dual ray space”. 12th Eurographics Workshop on Rendering, páginas 204-214.

[Lefebvre03] Lefebvre, Sylvain e Hornus, Samuel (2003), “Automatic cell-and-portal decomposition”. INRIA, número 4898. http://artis.imag.fr/Publications/2003/LH03/lefebvre-hornus-portals.pdf

[Luebke95] Luebke, David; Georges, Chris (1995), “Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets”. Symposium on Interactive 3D Graphics

[Luebke05] Luebke, David (2005), "Levels-of-detail". http://www.cs.virginia.edu/~gfx/Courses/2005/Intro.String.05/lecture24.lod.ppt

[Mann97] Mann, Yair e Cohen-Or, Daniel (1997), "Selective Pixel Transmission for Navigating in Remote Virtual Environments". Eurographics 1997, volume 16, número 3.

[Maciel95] Maciel, Paulo; Shirley, Peter (1995), "Visual Navigation of Large Environments Using Textured Clusters". Symposium on Interactive 3D Graphics, páginas 95-102. http://www.cs.utah.edu/~shirley/papers/interactive95.pdf

[Martin00] Martin, Ioana M.; Klosowski, James T. e Horn, William P. (2000), "Interactive 3D Rendering and Visualization in Networked Environments". Eurographics 2000.

[Martins] Martins, Nuno e Miranda, José (s.d.), "Scene Access Interface". http://www.sal.ipg.pt/user/estg/martins/x3d.htm

[Miranda99] Miranda, José (1999), "Urbanismo e Espaços Virtuais – Divulgação e Discussão na Comunidade". Tese de mestrado em Tecnologia Multimédia. Porto: Faculdade de Engenharia da Universidade do Porto.

[Muniz05] Muniz, André; Augusto, Cícero; Monteiro, Ivan; Nunes, Leandro e Vieira, Luiza (2005), "Algoritmos e estrutura de dados para jogos". http://twiki.im.ufba.br/pub/MAT0537€Seminarios/BspA-estrela.pdf

[Parallel] Parallel Graphics http://www.parallelgraphics.com/

[Peccini03] Peccini, Grasiela e d'Ornellas, Marcos (2003), "Segmentação de Imagens por Watersheds: Uma Implementação Utilizando a Linguagem Java". Centro de Tecnologia, Universidade Federal de Santa Maria, Brasil http://www.sbc.org.br/bibliotecadigital/download.php?paper=140

[Pecis] Pecis, Emiliano (s.d.), "Interazione trai l Vrml e Java com l'interfaccia Eai". http://eureka.lucia.it/vrml/tutorial/eai/home.html

[Pimentel01] Pimentel, João; Batista, Nuno; Goês, Luís; Dionísio, José (2001), "Construção e gestão da complexidade de cenários urbanos 3D em ambientes virtuais imersivos". http://ulisses.cm-lisboa.pt/data/003/004/

[Pires01] Pires, Pedro (2001), "Dynamic Algorithm Binding for Virtual Walkthroughs". Tese de mestrado em Engenharia Informática e Computadores. Porto: Faculdade de Engenharia da Universidade do Porto.

[Pollo97] Pollo, Luís (1997), "Software para a Geração Automática de Modelos 3D em VRML". Trabalho de Graduação em Informática. Brasil: Departamento de Electrónica e Computação da Universidade Federal de Santa Maria.
http://www-usr.inf.ufsm.br/~pollo/TG/

[Raposo04] Raposo, Alberto (2004), "Computação Gráfica Interactiva".
http://www.tecgraf.puc-rio.br/~abroposo/INF1366

[Rodrigues05] Rodrigues, José (2005), "Detecção de colisões".
http://mega.ist.utl.pt/~ic-cv/tecnicas aceleracao.pdf

[Rossignac93] Rossignac, Jarek e Borrel, Paul (1993), "Multi-resolution 3D approximations for rendering complex scenes". Computer Graphics 1993, Italia, páginas 455-465.

[Schaufler00] Schaufler, Gernot; Dorsey, Julie; Decoret, Xavier e Sillion, François (2000), "Conservative volumetric visibility with occluder fusion". Siggraph 2000, páginas 229-238.

[Schroeder92] Schroeder, William; Zarge, Jonathan e Lorensen, William (1992), "Decimation of Triangle Meshes". Siggraph 92, paginas 65-70.

[SGI] SGI
http://www.sgi.com

[Sillion97] Sillion, François; Drettakis, George e Bodelet, Benoit (1997), "Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery". Eurographics' 97, volume 16, número 3.

[Silva] Silva, Fernando (s.d.) "Introdução ao Ray Tracing".
http://w3.impa.br/~nando/publ/rt/

[Slater97] Slater, Mel e Chrysanthou, Yiorgos (1997), "View Volume Culling Using a Probabilistic Casting Scheme". ACM Virtual Reality Software and Technology VRST'97, 71-78, Lausanne, Switzerland.
http://www2.cs.ucy.ac.cy/~yiorgos/

[Sousa91] Sousa, António e Ferreira, Fernando (1991), "Computação Gráfica. Breves Considerações sobre 'Rendering'". Apontamentos da Faculdade de Engenharia da Universidade do Porto.

[Sullivan06] Sullivan, Carol (2006), "OpenGL Overview". Interaction, Simulation and Graphics Lab., Trinity College Dublin
http://isg.cs.tcd.ie/cosulliv/4BA6/notes/OpenGL_6pp.pdf

[Sutherland74] Sutherland, Ivan; Sproull, Robert e Schumacker, Robert (1974), "A Characterization of Ten Hidden-Surface Algorithms". Computing Surveys, volume 6, número 1.

[TechTarget06] SearchTechTarget.com
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci211625,00.html

[Teler01] Teler, Eyal e Lischinski, Dani (2001), "Streaming of Complex 3D Scenes for Remote Walkthroughs". Eurographics 2001, volume 20, número 3.

[Teller92] Teller, Seth Jared (1992), "Visibility Computations in Densely Occluded Polyhedral Enviroments". Tese de doutoramento em Computer Science. University of Califórnia at Berkeley.

[Valente04] Valente, Luís (2004), "Representação de Cenas Tridimensionais: Grafo de Cenas". Mestrado em Computação. Brasil: Universidade Federal Fluminense.
http://www.ic.uff.br/~lvalente/pt/scenegraph.html

[Velho00] Velho, Luís (2000), "Radiosidade".
http://w3.impa.br/~lvelho/i3d00/demos/rad/index.html

[Vizx3D] Virtock Technologies Inc., Vizx3D
http://www.vizx3d.com

[VRCreator] Platium VRCreator – Editor de VRML
http://www.freedownloadscenter.com/

[VRML97] The VRML Consortium Inc.; "The Virtual Reality Modeling Language - International Standard ISO/IEC 14772 1:1997"
http://www.web3d.org/x3d/specifications/vrml/

[Wang98] Wang, Yigang; Bao, Hujun e Peng, Qunsheng (1998), "Accelerated Walkthroughs of Virtual Environments Based on Visibility Preprocessing and Simplification". Eurographics' 98, volume 17, número 3.

[Wikipedia07] Wikipedia contributors (2007), "Painter's algorithm2". Wikipedia, The Free Encyclopedia.
http://en.wikipedia.org/wiki/Painter%27s_algorithm

[Wimmer99] Wimmer, Michael; Giegl, Markus e Schmalstieg, Dieter (1999), "Fast Walkthroughs with Image Caches and Ray Casting". Eurographics' 99, páginas 73-84.

[Wimmer01] Wimmer, Michael; Wonka, Peter e Sillion, François (2001), "Point-Based Impostors for Real-Time Visualization". Eurographics 2001 http://artis.imag.fr/Publications/2001/WWS01b/Wimmer_egrw2001.pdf

[Wonka99] Wonka, Peter e Schmalstieg, Dieter (1999), "Occluder Shadows for Fast Walkthroughs of Urban Environments". Eurographics' 99, volume 18, número 3.

[Wonka00] Wonka, Peter; Wimmer, Michael e Schmalstieg, Dieter (2000), "Visibility Prepocessing with Occluder Fusion for Urban Walkthroughs". Technical Report TR-186-2-00-06, Institute of Computer Graphics, Vienna University of Technology.

[X3D] Web 3D Consortium, "X3D: The Virtual Reality Modeling Language – International Standard ISO/IEC 14772:200x". http://www.web3d.org/

[Xj3D] Web 3D Consortium, Xj3D http://www.xj3d.org

[XML] WWW Consortium, Extensible Markup Language (XML) http://www.w3.org/XML

[Zhang97] Zhang, Hansong e Hoff III, Kenneth (1997), "Fast backface culling using normal masks". Siggraph 1997, páginas 103-106.

ANEXOS

9 - Anexos

9.1 - Anexo A

A comunicação entre a *applet* Java e o mundo VRML é feita através da *interface* EAI, sendo para isso necessário as seguintes quatro classes que se encontram no *package* *vrml.external.**: *vrml.external.Browser*, a qual representa o mundo VRML na *applet* Java; *vrml.external.Node*, esta classe representa um nó VRML; *vrml.external.field.**, classe que representa todos os tipos de dados existentes na linguagem VRML; *vrml.external.exception.**, a qual representa as classes de excepção para tratamento de erros.

Para "correr" o mundo VRML e a *applet* em Java, o ficheiro HTML tem de conter as seguintes *tags*:

```
<EMBED SRC="FicheiroVRML.wrl">
<APPLET code="NomeApplet.class" MAYSCRIPT> </APPLET>
```

Quando a *applet* é executada, as classes da EAI Java API têm de estar referenciadas no início do ficheiro. No mínimo, tem de ser importada a seguinte classe *import* *vrml.external.Browser*; as restantes classes dependem do tipo de nó e evento que é utilizado na *applet*.

A referência ao *browser* é feita através da instrução: `browser = Browser.getBrowser(this);`

De seguida, é feito o pedido para receber a referência a um nó particular da hierarquia VRML, podendo a *applet*, desta forma, manipular o nó, enviando mensagens. Esta interacção é limitada pois é apenas possível o acesso a campos que são públicos.

A

Tabela 15 apresenta o conjunto de instruções em Java que permitem a recepção e envio de mensagens a partir de um nó específico.

Tabela 15 – Instruções em Java referentes à recepção e envio de eventos, e respectiva descrição

`browser.getNode("Mat"):`	este método obtém a referência ao nó material ao qual foi atribuído o nome de Mat no ambiente VRML
`material.getEventIn("set diffuseColor"):`	este método obtém a referência ao campo `eventIn` no mundo VRML
`set diffuseColor.setValue( val):`	este método envia o valor na *string* `val` para o campo `set_diffuseColor`
Node [] transform = browser. createVrmlFromString("Transform {}");	utilizando a instância *Browser* é possível criar nós e adicioná-los à cena através do método `createVrmlFromString()`

Em [Crispen00] pode-se encontrar a informação completa sobre a especificação desta *interface*.

9.2 - Anexo B

Para utilizar a SAI é necessário que o computador esteja preparado de modo a executar as aplicações do X3D integrado com a Java. Desta forma, é necessário instalar o *X3D installer for Windows*, para ser instalado o *browser* Xj3D e o Java2 *Runtime enviroment* SE V1.4.2_07. Segue-se a instalação do *Netbeans* (plataforma para o desenvolvimento de aplicações em Java e um ambiente integrado de desenvolvimento (IDE), concebido totalmente em Java, a partir da plataforma *NetBeans*. A plataforma *NetBeans* permite que as aplicações sejam desenvolvidas a partir de um conjunto de módulos designados de *modules*). O passo seguinte é a definição do *classpath* da Java, sendo necessário copiar os ficheiros *jars* da directoria "C:\Programas\Xj3D\jars" (local de instalação do Xj3D por defeito) para a directoria "C:\j2sdk1.4.2_04\jre\lib\ext" (local de instalação do *Netbeans* por defeito). Por último, todas as aplicações desenvolvidas deverão ser colocadas dentro da directoria "C:\Programas\Xj3D\examples".

Concluída a preparação do computador, pode-se desenvolver a *interface* de comunicação entre o mundo X3D e o utilizador, tendo em conta o tipo de acesso necessário: interno ou externo. Em ambos os acessos, o *package* para carregar a cena X3D na classe Java tem de ser importado. Os parágrafos seguintes descrevem a filosofia de programação para cada um dos tipos de acesso.

Acesso interno

Inicialmente, no acesso interno o ficheiro X3D deve conter os nós *Script* e *Route* (Listagem 18). O primeiro nó permite que a classe Java seja executada e o campo de saída calculado quando o campo de entrada for gerado. O segundo nó faz a ligação entre os objectos da cena X3D e os respectivos campos de entrada e saída definidos anteriormente.

```
<Script DEF="S" url= "ObjectoMoveu.class">
   <field accessType= "InputOnly" name= "pulse" type= "SFBool"/>
…
</Script>
…
<ROUTE fromField="pulse" fromNome= "Sensor" toField= "pulse" toNode= "S"/>
```

Listagem 18 – Definição do nó *Script* e *Route*

Para ser possível o acesso interno a classe Java precisa de implementar a *interface X3DScriptImplementation*, assim como o *package org.web3d.x3d.sai* (Listagem 19).

```
import org.web3d.x3d.sai.*;
public class Cidade implements X3DScriptImplementation {
```

Listagem 19 – Importação de um *package* e de uma *interface*

Para o acesso aos campos definidos na *interface* do nó *Script* utiliza-se o método *setFields* (Listagem 20).

```
public void setFields(X3DScriptNode externalView, Map fields) {
```

Listagem 20 – Método *setFields*

Adiciona-se *Listeners* (objecto com o intuito de "ouvir" se foi executado algum evento) para a Java ser notificada da ocorrência dos eventos de "entrada" (Listagem 21).

```
isOver.addX3DEventListener(this);
```

Listagem 21 – *Interface X3DFieldEventListener*

Para cada mudança associado ao *Listener* a função *readableFieldChanged* é executada de forma a atribuir os valores de saída para a cena 3D com o método *setValue*() (Listagem 22).

```
public void readableFieldChanged (X3DFieldEvent evt) {
```

Listagem 22 – Função *readableFieldChanged*

Acesso externo

Inicialmente, no acesso externo, para executar a SAI é necessário importar o *package org.web3d.x3d.sai*: `Import org.web3d.x3d.sai.*;`

Para carregar a cena X3D é necessário criar um componente para servir de *browser* X3D e adicionar uma janela 3D à aplicação Java através da classe *SAI BrowserFactory* (Listagem 23).

```
X3DComponent x3DComp = BrowserFactory. createX3DComponent(null);
JComponent x3dPanel = (JComponent) x3dComp.getImplementation();
        contentPane.add(x3dPanel, BorderLayout.CENTER);
```

Listagem 23 – Criação de um componente e aplicação de uma janela 3D

A visualização da cena X3D é feita através do *browser*, o qual recebe a cena através da especificação do seu nome e localização (Listagem 24).

```
x3dBrowser = x3dComp.getBrowser();
cena3D = x3dBrowser.createX3DFromURL (new String[] {"ficheiros-x3d.x3d"});
```

Listagem 24 – Aplicação da cena X3D para dentro do *browser*

Por fim actualiza-se o *browser* com a cena criada: `x3dBrowser.replaceWorld(cena3D);`

O acesso, por exemplo, ao nó *Mat* na cena X3D obtém-se com o método *getNamedNode()*: `X3DNode mat = cena3D.getNamedNode("Mat");`

O acesso, por exemplo, ao campo *diffuseColor* do nó *Mat* na cena X3D obtém-se com o método *getField*(): `SFColor cor = (SFColor) mat.getField("diffuseColor");`

A atribuição, por exemplo, da cor azul à variável local *CorAzul* é feita através do método *setField()*:

```
cor.setValue(CorAzul);
```

O método *createNode()* permite criar nós na cena X3D que de seguida têm de ser inseridos no mundo (Listagem 25).

```
X3DNode shape = mainScene.createNode ("Shape");
SFNode shape_geoemetry = (SFNode) (shape.getField("geoemetry"));
X3DNode box = mainScene.createNode("Box");
…
mainScene.addRootNode(shape);
```

Listagem 25 – Método *createNode()*

9.3 - Anexo C

O inquérito apresentado neste anexo tem como objectivo avaliar o protótipo desenvolvido nesta dissertação por um público-alvo, durante e após a navegação pelo ambiente. O inquérito foi desenvolvido tendo em conta o universo de utilizadores e os objectivos a atingir nesta dissertação. Desta forma, dividiu-se o inquérito por secções e concebeu-se as perguntas de modo a não polarizar as respostas. Na primeira secção são pedidas informações relativas ao utilizador e relativas ao seu perfil de utilização em contexto 3D. Com o intuito de permitir um primeiro contacto com o ambiente, a segunda secção, Navegação Livre, possibilita uma navegação durante cerca de 10 minutos. As secções três a seis têm como objectivo avaliar os seguintes aspectos:

- importância da representação dos elementos associados às zonas que circundam as áreas verdes, áreas vazias ou áreas com objectos de pequeno porte
- importância da representação dos diversos elementos que se encontram por trás de um oclusor
- relevância da representação de edifícios altos que caracterizam a cidade e consequentemente permitem ao utilizador obter pontos de referência durante a sua navegação
- navegação e *interface*

Desta forma, o público alvo foi confrontado com dois protótipos (protótipo A e protótipo B) idênticos à excepção da ausência de segmentação e da aplicação de técnicas de cálculo de oclusão no segundo protótipo, não tendo sido exposta a existência deste protótipo com o intuito de consolidar respostas do inquérito. O utilizador foi confrontado nas secções três e quatro com o protótipo B enquanto que, nas secções cinco e seis navegou no protótipo A. Como foi referido anteriormente, estas quatro secções têm objectivos idênticos mas foi alterado a sequência do propósito a avaliar resultando em perguntas com a mesma finalidade mas focando o assunto por um aspecto diferente.

A última secção refere-se a uma avaliação global onde o utilizador pode escrever sugestões e comentários.

Avaliação - CidadePorto Virtual

O presente questionário destina-se a recolher a opinião de todos os utilizadores e têm por objectivo avaliar as funcionalidades, a interface e a navegabilidade da **CidadePorto Virtual**.

Pretende-se que o utilizador navegue por um percurso com vista a identificar determinados elementos existentes, assim como experimentar a navegabilidade da interface.

1. Identificação do Utilizador

1.1 - Dados pessoais

De forma a poder ser considerada a sua contribuição, peço-lhe, que forneça alguns dados pessoais e profissionais. Estes dados serão tratados confidencialmente e exclusivamente no âmbito do estudo supracitado.

Nome: _____

Email: _____

Sexo: ☐ Feminino ☐ Masculino

1.2 - Perfil de utilização em contexto 3D

Já teve experiências de navegação em VRML ou equivalente?

Nunca ☐

Uma única vez ☐

Várias vezes ☐

2. Navegação Livre

Pretende-se que o utilizador navegue pela **CidadePorto Virtual** cerca de 10 minutos, com o objectivo de estabelecer um primeiro contacto com a aplicação.

3. Navegação Orientada

O trajecto que se pretende que o utilizador percorra inicia-se na Travessa S. Bento e termina na Avenida dos Aliados (Figura 1). Este percurso permite que o utilizador transite por ruas com elementos diversificados, como edifícios, igrejas, jardins e monumentos.

Siga a trajectória definida (itens assinalados por uma seta "→") e responda às questões apresentadas.

Figura 1 – Imagem do trajecto a percorrer

→ Ponto de partida: Travessa S. Bento.

3.1 - Em jardins, a visibilidade pode ser possível para além do próprio jardim. Na Travessa de S. Bento, refira o grau de facilidade na identificação da Faculdade de Ciências:

1	2	3	4	5
(Difícil)				(Muito fácil)
☐	☐	☐	☐	☐

→ Siga sempre em frente de forma a contornar o jardim.

→ No cruzamento com a R. das Taipas, vire à esquerda para se manter na mesma rua.

→ Siga em direcção ao jardim João Chagas.

→ No cruzamento, contorne o jardim pela sua direita (encontra-se no Campo dos Mártires da Pátria).

→ Surge um novo cruzamento com a R. do Dr. Ferreira da Silva (rua em frente) e com a R. S. Filipe de Nery (lado direito). Siga pela R. do Dr. Ferreira da Silva.

→ Na Praça de Gomes Teixeira encontra-se o jardim com o mesmo nome, a qual intersecta a R. das Carmelitas (lado direito).

3.2 - Quando um edifício desempenha o papel de oclusor, os elementos que se encontram atrás dele, ficam parcialmente ocultos. Na Praça Gomes Teixeira, a Igreja do Carmo desempenha o papel de oclusor. Refira o grau de facilidade na identificação do Hospital de S. António:

1	2	3	4	5
(Difícil)				(Muito fácil)
☐	☐	☐	☐	☐

→ Siga pela Praça, isto é, contorne o jardim pela direita.

→ No cruzamento da Praça de Gomes Teixeira com a Praça de Guilherme Gomes Fernandes (lado direito), siga por esta última.

→ Surge uma rua do lado esquerdo, mantenha-se sempre na rua onde está.

→ No fim da rua, encontra-se o cruzamento com a R. do actor João Guedes (lado esquerdo), a R. José Falcão (em frente) e a R. de Santa Teresa (lado direito). Vire para a R. de Santa Teresa.

→ A R. de Santa Teresa cruza com três ruas, R. da Galeria de Paris, R. Cândido dos Reis e a R. do Conde de Vizela. Permaneça na primeira rua.

→ A R. da Fábrica surge logo a seguir à R. de Santa Teresa, continue por essa rua. Esta rua termina no cruzamento com a R. do Almada (lado direito e esquerdo) e com a R. do Dr. Artur de Magalhães Basto (em frente). Siga pela rua em frente.

→ No fim da R. do Dr. Artur de Magalhães encontra-se a Avenida dos Aliados (fim do percurso).

3.3 - Como nos jardins, em áreas abertas o campo de visão é bastante extenso. Na Avenida dos Aliados, refira o grau de facilidade na identificação de:

	1	2	3	4	5
	(Difícil)				(Muito fácil)
Monumento a D. Pedro IV	☐	☐	☐	☐	☐
Câmara do Porto	☐	☐	☐	☐	☐

4. Conclusão da Navegação Orientada

Terminada a trajectória e finalizada a aplicação, responda às questões seguintes.

4.1 - Atendendo aos diferentes aspectos da navegação e *interface* classifique cada um dos itens abaixo referenciados.

	1 (Pouco)	2	3	4	5 (Muito)
Facilidade na execução do percurso pedido					
Importante a identificação da rua na parte esquerda da *interface*					

4.2 - Conseguiu identificar o animal que se encontra pousado no topo da Torre dos Clérigos?

5. Novas Impressões – Navegação Orientada

Pretende-se que o utilizador percorra novamente o trajecto definido anteriormente para consolidação dos dados obtidos. Siga a trajectória definida (itens assinalados por uma seta "→") e responda às questões apresentadas.

→ Ponto de partida: Travessa S. Bento.

→ Siga sempre em frente de forma a contornar o jardim.

→ No cruzamento com a R. das Taipas, vire à esquerda para se manter na mesma rua.

→ Siga em direcção ao jardim João Chagas.

→ No cruzamento, contorne o jardim pela sua direita (encontra-se no Campo dos Mártires da Pátria).

→ Surge um novo cruzamento com a R. do Dr. Ferreira da Silva (rua em frente) e com a R. S. Filipe de Nery (lado direito). Siga pela R. do Dr. Ferreira da Silva.

→ Na Praça de Gomes Teixeira encontra-se o jardim com o mesmo nome, a qual intersecta a R. das Carmelitas (lado direito).

→ Siga pela Praça, isto é, contorne o jardim pela direita.

→ No cruzamento da Praça de Gomes Teixeira com a Praça de Guilherme Gomes Fernandes (lado direito), siga por esta última.

→ Surge uma rua do lado esquerdo, mantenha-se sempre na rua onde está.

→ No fim da rua, encontra-se o cruzamento com a R. do actor João Guedes (lado esquerdo), a R. José Falcão (em frente) e a R. de Santa Teresa (lado direito). Vire para a R. de Santa Teresa.

→ A R. de Santa Teresa cruza com três ruas, R. da Galeria de Paris, R. Cândido dos Reis e a R. do Conde de Vizela. Permaneça na primeira rua.

5.1 - Numa cidade, edifícios altos podem ser vistos a longas distâncias, eventualmente por sobre outros edifícios. Na R. de Santa Teresa, cruzamento com a R. da Galeria de Paris, refira o grau de facilidade na identificação da Torre dos Clérigos:

1	2	3	4	5
(Difícil)				(Muito fácil)
☐	☐	☐	☐	☐

→ A R. da Fábrica surge logo a seguir à R. de Santa Teresa, continue por essa rua. Esta rua termina no cruzamento com a R. do Almada (lado direito e esquerdo) e com a R. do Dr. Artur de Magalhães Basto (em frente). Siga pela rua em frente.

→ No fim da R. do Dr. Artur de Magalhães encontra-se a Avenida dos Aliados (fim do percurso).

5.2 - Como nos jardins, em áreas abertas o campo de visão é bastante extenso. Na Avenida dos Aliados, refira o grau de facilidade na identificação de:

	1	2	3	4	5
	(Difícil)				(Muito fácil)
Monumento 'Menina Nua'	☐	☐	☐	☐	☐
Câmara do Porto	☐	☐	☐	☐	☐

6. Conclusão da Navegação Orientada

Terminada a trajectória e finalizada a aplicação, responda às questões seguintes.

6.1 - Atendendo aos diferentes aspectos da navegação e *interface* classifique cada um dos itens abaixo referenciados.

	1 (Pouco)	2	3	4	5 (Muito)
Orientação adequada da posição do utilizador na nova rua					
Tempo de espera para apresentação de uma nova rua					

6.2 - Apercebeu-se de qual era a igreja que se consegue visualizar no cruzamento da R. de Santa Teresa com a R. Cândido dos Reis?

7. Questões Finais

7.1 - Tendo em conta o percurso efectuado em ambas as aplicações, que diferenças encontra nos objectos representados entre ambas as viagens?

7.2 - Considere que 1 representa "Pouco importante" e o 5 representa "Muito importante".

	1	2	3	4	5
Faça uma avaliação global do trabalho desenvolvido					
Considera os recursos e funcionalidades disponibilizados pela **CidadePorto Virtual** úteis					

7.3 - Que outros aspectos gostaria de ver melhorados nesta **CidadePorto Virtual**?

Muito obrigada por ter colaborado no preenchimento do inquérito.