Faculdade de Engenharia da Universidade do Porto

Automatic Content Generation for Second Life

António Sérgio Mota Gonçalves

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Doutor Ricardo Morla

Junho de 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"AUTOMATIC CONTENT GENERATION FOR SECOND LIFE"

foi aprovada em provas realizadas em 17/Julho/2009

o júri

Presidente Professor Doutor José Alberto Peixoto Machado da Silva
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Artur Miguel do Amaral Arsénio
Professor Auxiliar do Departamento de Engenharia Informática do Instituto Superior técnico da Universidade Técnica de Lisboa

Professor Doutor Ricardo Santos Morla
Professor Auxiliar Convidado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - **ANTÓNIO SÉRGIO MOTA GONCALVES**

Faculdade de Engenharia da Universidade do Porto

Resumo

Os mundos virtuais têm vindo a desempenhar um papel cada vez mais relevante no armazenamento e partilha de informação. É nesse sentido que este projecto visa automatizar o processo de geração de conteúdo no mundo virtual do *Second Life*.

O trabalho foi dividido em três fases distintas: i) análise de trabalho relacionado; ii) desenvolvimento do *software*; iii) validação do trabalho desenvolvido. Na primeira fase foram pesquisadas algumas formas automáticas de geração de conteúdo e identificadas fontes para a informação do conteúdo a gerar. Na segunda e terceira fases, é desenvolvido e posteriormente validado com alguns testes de desempenho, o *software* para geração automática de conteúdo no *Second Life*.

O *software* desenvolvido permite a criação automática de estradas, percursos de autocarros assim como alguns objectos pré-definidos.

Abstract

Virtual worlds have been performing a role of extreme relevance in storage and share of knowledge. Following that direction, this project aims to make the generation of content to the virtual world of Second Life an easier task.

The work was divided in three distinct phases: i) analysis of related work; ii) development of the software; iii) validation of the work. In the first phase some forms of automatic content generation are reviewed and some information sources for the content to generate are identified. In the second and third phases, it is developed and validated with some performance tests, the software for automatic content generation in Second Life's virtual world.

Developed software allows automatic creation of roads, buses' waypoints as well as some predefined objects.

“Things should be made as simple as possible, but not any simpler.”

Albert Einstein

Agradecimentos

Um trabalho individual, por muito mérito e valor que tenha, pode tirar-nos a vontade de agradecer. Apesar de longos períodos de desenvolvimento e reflexão pessoal, este não foi, de maneira alguma, um desses trabalhos. Assim sendo,

à minha família, em particular aos meus avós, tia, irmão e de um modo muito especial aos meus pais pelo exemplo de vida que me têm transmitido e por todo o apoio incondicional que sempre me deram.

ao meu orientador, Ricardo Morla, por todo o apoio, criticas, sugestões, boa vontade e disponibilidade durante todo o processo.

a todos os amigos e colegas que comigo partilharam ideias e alimentaram discussões, a todos que de uma forma ou de outra já se encontram nas entrelinhas deste texto.

a todos os companheiros do INESC Porto pelo agradável, sempre bem-disposto e felizmente movimentado ambiente de trabalho.

a todos os professores da Faculdade de Engenharia da Universidade do Porto, em especial ao professor Adriano Carvalho, director do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, por toda a paciência, profissionalismo, atenção e boa vontade demonstradas ao longo destes anos.

o meu mais sincero obrigado.

Índice

Lista de figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de abreviaturas:

CAD	*Computer Aided Design*
GE	*Google Earth*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MMORPG	*Massive Multiplayer Online Role Playing Game*
MP	*Multi Player*
MXP	*Metaverse Exchange Protocol*
NPC	*Non-player character*
OS	*OpenSimulator*
OSM	*OpenStreetMap*
REST	*Representational State Transfer*
SL	*Second Life*
SLV	*Second Life Viewer*
SP	*Single Player*
UDP	*User Datagram Protocol*
XBL	*XBoxLive*
XBLA	*Xbox Live Árcade*
XML	*Extensible Markup Language*

xx

Capítulo 1

Introdução

Na actual sociedade da informação em que vivemos cada vez existe menos tempo para desperdiçar na procura ou criação de conteúdo. Cada vez mais é necessário encontrar formas para tornar a informação disponível mais acessível e universal.

Com este projecto pretende-se compilar informação vinda de diferentes fontes e áreas da sociedade num só local, num só mundo: o mundo virtual do *Second Life*. Contudo, irá também haver espaço para a imaginação, podendo cada pessoa criar o seu próprio mundo, misturando conteúdo real com outro fantasiado por si.

Este projecto será a base para a criação destes mundos individuais que poderão posteriormente ser partilhados através da Internet com milhares de pessoas. A maior limitação num projecto ambicioso é o tempo. Visto este ser um projecto bastante limitado nesse aspecto, foi apenas possível criar e integrar uma pequena parte do que se poderia, visto as opções de conteúdo a implementar serem praticamente infinitas: desde pessoas a andar, lançamento de satélites, simulações de corridas de fórmula um, entre muitas outras.

De entre todas as possibilidades, decidiu-se gerar automaticamente as estradas de cidades e o movimento de autocarros nessas mesmas estradas. Para isso, é necessário recolher quer informações relativas a todas as estradas da área que se pretende recriar, bem como, do percurso dos autocarros. Compilando estas informações é possível simular o caminho percorrido pelos diferentes autocarros na cidade e visualizá-lo.

Este documento encontra-se dividido em cinco grandes capítulos em que cada um deles trata diferentes fases de desenvolvimento que decorreram ao longo de todo o projecto. Assim, no primeiro capítulo é feita uma introdução ao tema que será abordado, mencionando o enquadramento do projecto e objectivos que se pretendem atingir. No segundo capítulo é feito um agrupamento de toda a pesquisa efectuada sobre tecnologias que possivelmente serão usadas para o desenvolvimento do protótipo, especificando vantagens de umas em relações às outras. Todos os detalhes de desenvolvimento e implementação do protótipo são descritos no

terceiro capítulo, bem como o planeamento feito para a duração do projecto, enquanto que o quarto trata de testes realizados ao protótipo já desenvolvido, assim como os respectivos resultados, dando ênfase, não só a todas as funcionalidades implementadas mas também a todas as falhas observadas. Por fim, no último capítulo estão presentes todas as conclusões e possíveis soluções para maior escalabilidade e robustez, bem como futuros desenvolvimentos e integração em aplicações.

1.1 Enquadramento

O *Second Life* é um mundo virtual a três dimensões imaginado e criado pelos seus utilizadores. Desde o seu lançamento em 2003, milhões de utilizadores têm usado esta ferramenta quer para lazer quer por motivos profissionais. Na realidade, as aplicações dadas a este mundo virtual são inúmeras, destacando-se o lazer, comércio e educação. Assim, tem-se notado um uso crescente de mundos virtuais em todo o mundo, Portugal incluído. Enquanto algumas pessoas recorrem ao mundo do *Second Life* para se divertirem, conviver com os amigos assim como fazer novas amizades, outras há que o utilizam apenas para fins comerciais, isto é, tirar partido do mundo virtual para a obtenção de lucro, usando-o como suporte para a sua actividade profissional ou mesmo até criando a própria empresa dentro deste mundo. Um grupo mais específico e diminuto de pessoas, exploram este mundo tendo em vista a sua utilização em aplicações mais pedagógicas, ou seja, usando-o para a educação. Desde a criação de salas de aula virtuais para leccionar, quer local quer remotamente, até estudos baseados neste mundo e nos seus residentes, o *Second Life* provou ser um poderoso aliado para as mais distintas tarefas.

Visto que todo o mundo é criado pelos seus utilizadores, o acto de criar é uma das acções mais importantes, senão mesmo a mais importante, que o *Second Life* tem para oferecer. Todos os objectos criados permitem um nível de personalização quase sem limites, dependendo assim da imaginação do utilizador para criar objectos únicos em todo o mundo virtual. Este projecto irá explorar esta funcionalidade para assim criar automaticamente no mundo virtual o conteúdo desejado de uma forma muito mais rápida, fácil e intuitiva para o utilizador, permitindo também alguns níveis de personalização.

1.2 Objectivos

O principal objectivo a atingir é a criação automática de conteúdo para o mundo virtual do *Second Life*. Este conteúdo terá que ser tanto estático como dinâmico.

O conteúdo que se poderia introduzir num mundo virtual com as características do *Second Life* é quase ilimitado. Porém, visto que este projecto é limitado temporalmente, a quantidade de conteúdos diferentes introduzida será também limitada.

Pretende-se também analisar a performance do *software* desenvolvido em termos de consumo de recursos – memória e processador.

Com o objectivo de uma maior facilidade de implementação de futuras adições de conteúdo, o projecto será construído e desenvolvido, tanto quanto possível, em módulos.

1.3 Nome do Software

Como qualquer software é também necessário que este projecto tenha um nome. Assim sendo, e depois de ponderadas várias hipóteses, foi escolhido *2ndComing*.

O nome foi escolhido devido à *Second Coming of Christ*, isto é, o retorno de Jesus Cristo à Terra. Uma vez que este *software* tem o objectivo de ser o criador (gerador) de diferentes tipos de conteúdo foi fácil relacionar ambos. Além disso, a relação com o *"second"* de *Second Life* foi também tida em conta.

As distintas partes do projecto terão por isso que ser denominadas segundo as seguintes regras:

- Interface para o utilizador: *2ndComing*;
- Módulo do OpenSimulator: *2ndComing OpenSim Module*;
- Geradores de conteúdo: *2ndComing 'X'Generator*.

Em que X será o nome em inglês do conteúdo que esse gerador irá ter a responsabilidade de gerar - *BusGenerator*, por exemplo.

Capítulo 2

Estado da Arte

Desde o aparecimento dos videojogos e dos respectivos mundos virtuais que estes têm estado intimamente ligados a momentos de lazer. Na actualidade esta visão está a mudar e começa-se a olhar para o potencial dos mundos virtuais como suporte das mais variadas actividades bem como um meio para fazer chegar um produto ao seu mercado alvo, por parte de empresas.

2.1 Mundos Virtuais

Um mundo virtual é um ambiente simulado por computador que serve de espaço para os seus utilizadores, muitas vezes denominados residentes, habitarem e interagirem com este através dos seus avatares. Um *avatar* é uma representação do utilizador, no mundo virtual, podendo ser uma simples palavra ou uma representação a duas ou três dimensões. Outras formas de representação do utilizador são também possíveis como um simples sinal auditivo ou até uma sensação, embora estes sejam menos utilizados.

O projecto *KissMe* [1] pretende implementar outras formas de interacção entre utilizadores num mundo virtual, transpondo-as para o mundo real. Desta forma, a experiência do mundo virtual torna-se mais real. Para isso, cada utilizador necessita colocar um colar ao pescoço, ligado por *Bluetooth*. Quando, no mundo virtual, um utilizador beija outro, o sinal é enviado pela Internet, estimulando os actuadores no colar que, por sua vez, transmitirão essa mesma sensação ao utilizador que a recebeu no mundo virtual. A Figura 1 mostra o princípio de funcionamento do *KissMe* [1].

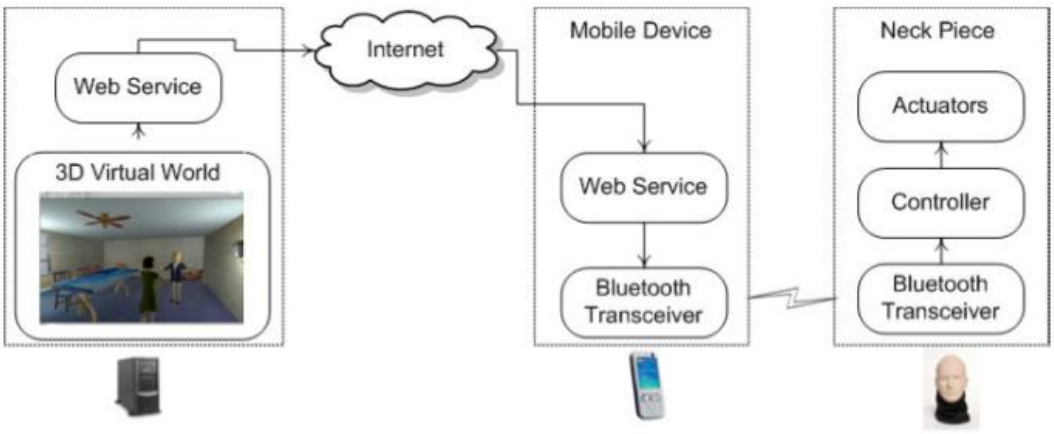

Figura 1 - Princípio de funcionamento do *KissMe* [1]

O utilizador, através do seu *avatar* pode interagir com os elementos modelados no mundo virtual podendo ter alguns objectivos para cumprir ou simplesmente explorá-lo. Enquanto existem mundos virtuais modelados directamente a partir do real, obedecendo a leis físicas reais como é o exemplo da gravidade, locomoção e topografia, outros são pura fantasia ou híbridos, isto é, misturam elementos do mundo real com o da fantasia. Alguns mundos virtuais permitem o acesso simultâneo de vários utilizadores: fóruns, MMORPGs, entre outros. Este facto permite enriquecer e tornar única a experiência de cada pessoa cada vez que entra no mundo, permitindo assim a comunicação e interacção entre utilizadores, quer seja através de texto, imagens gráficas, sons, gestos, ou até mesmo de acções feitas pelos seus avatares.

Os mundos virtuais podem ser divididos em dois grandes tipos: online e off-line. Sendo que os primeiros são acessíveis via Internet com o objectivo de estar em utilização 24horas por dia. Este objectivo não é atingível porque todos os servidores necessitam de manutenção e portanto existem sempre períodos de tempo em que determinado mundo virtual não está disponível. Por outro lado, os mundos virtuais off-line, presentes em alguns jogos *singleplayer* (SP), por exemplo, são criados apenas para serem acedidos por um pequeno grupo de pessoas, localmente. Estes mundos são então povoados por *non-player characters* (NPC) com quem o(s) utilizador(es) pode interagir. Muitas vezes, existe a possibilidade de se guardar o estado actual do mundo virtual para mais tarde se poder voltar a continuar do ponto em que se parou.

2.1.1 Aspectos Sociais

O limite entre mundo virtual e real é mais ténue do que se poderia imaginar. Um utilizador quando se conecta a um mundo virtual e usa um avatar para comunicar, continua com a mesma cultura, ideais, convicções e preconceitos que tem na vida real, fazendo com que o seu avatar seja uma projecção de si próprio no mundo virtual.

Existem vários tipos de mundos virtuais sendo que os mais utilizados são:

- Fóruns de discussão;
- *Blogs;*
- Salas de conversação;
- *Wikis;*
- MMORPGs e outros mundos de videojogos.

Todos estes mundos virtuais em que existe interacção entre utilizadores permitem e estimulam o nascimento de comunidades. Estas comunidades são uma das grandes vantagens dos mundos virtuais, visto que permitem a comunicação e estimulam a troca de conhecimento entre pessoas.

Assim, os mundos virtuais podem ser vistos como uma nova forma de levar as pessoas a socializar mas, como tudo o que é usado e/ou consumido em excesso, pode ser prejudicial. Por vezes, alguns utilizadores tornam-se tão dependentes destes mundos virtuais que lhes começam a dar mais importância que ao mundo real, afectando assim a sua vida social, já que comunicam maioritariamente através do mundo virtual, estes utilizadores criam uma "segunda vida" com um determinado nome, imagem e carisma [2].

Pessoas que, de alguma forma, se encontram incapacitadas podem usar os mundos virtuais para fazer acções tão simples como caminhar, correr e dançar. Devido à liberdade mental e emocional que temporariamente conquistam nesse mundo, ganham grandes benefícios permitindo-lhes esquecer suas incapacidades por breves períodos de tempo. Estas pessoas podem assim ter um meio muito mais fácil para socializar do que teriam apenas no mundo real, onde as suas incapacidades poderiam dificultar o simples acto de conversar com outra pessoa.

Infelizmente, estes mundos podem também ser usados por pessoas com intenções menos boas. Nestes mundos, qualquer pessoa pode ser aquilo que desejar. Muitos utilizadores fazem-se passar por outras pessoas, enganando assim os outros utilizadores para mais tarde poderem beneficiar da situação [3]. Em casos extremos, os mundos virtuais podem servir como base para terroristas planearem e comunicarem sobre possíveis atentados [4].

2.1.2 Economia

Alguns mundos virtuais mais complexos, tipicamente os videojogos, têm a sua própria economia que pode ter reflexos na economia real. Como no mundo real, os residentes de um dado mundo virtual têm um tempo limitado para atingir os objectivos que pretendem atingir. Muitos desses objectivos passam pela obtenção de dinheiro virtual, o qual pode ser adquirido das mais variadíssimas formas.

O valor dos objectos no mundo virtual está intimamente ligado com a sua utilidade e tempo/dificuldade gasto na sua aquisição. Desta forma, o investimento de recursos reais (tempo e mensalidades) na geração de riqueza no mundo virtual reflecte-se na economia

real, já que, existem pessoas a fornecerem serviços de venda de objectos virtuais por dinheiro real [5].

Cada vez mais existem utilizadores que se aproveitam das economias virtuais para obter lucro na economia real, quer criando a sua actividade profissional nestes mundos quer explorando estes e outros utilizadores para benefício próprio [5]. Um exemplo deste último grupo de utilizadores são os chamados *gold sellers* que vendem dinheiro virtual para determinado jogo por dinheiro real. As companhias responsáveis pelos respectivos videojogos têm tentado com pouca eficácia reduzir o impacto dos *gold sellers* na economia virtual.

2.1.3 Educação

Os mundos virtuais são cada vez mais utilizados como suporte para a educação [6] e investigação [6]. O uso destes mundos dá aos professores garantias de uma maior participação por parte dos alunos, permitindo também a realização de tarefas que no mundo real seriam mais difíceis quer por restrições de horário, localização ou custos.

Estes mundos permitem que alunos com algumas restrições possam aceder ao mesmo conteúdo didáctico, a partir de locais remotos, que os alunos que estão fisicamente a assistir à apresentação. Apesar de todas as vantagens que os mundos virtuais introduzem na educação, estes devem ser utilizados como suporte pois não podem substituir completamente as aulas físicas, pelo menos por enquanto. O uso destes mundos tem a desvantagem de se perder linguagem corporal que muitas vezes ajudam à explicação de determinado assunto, bem como a outros aspectos mais pessoais.

Um número cada vez maior de universidades começa a explorar alguns mundos virtuais como suporte para o estudo dos alunos, criando salas de aula virtuais e fóruns de discussão.

Um estudo intitulado *"What will happen to field trips? Beyond classroom"* [8] descreve um possível modelo de aprendizagem experimental com o uso do *Second Life*. Este estudo pode ser esquematizado no diagrama da Figura 2.

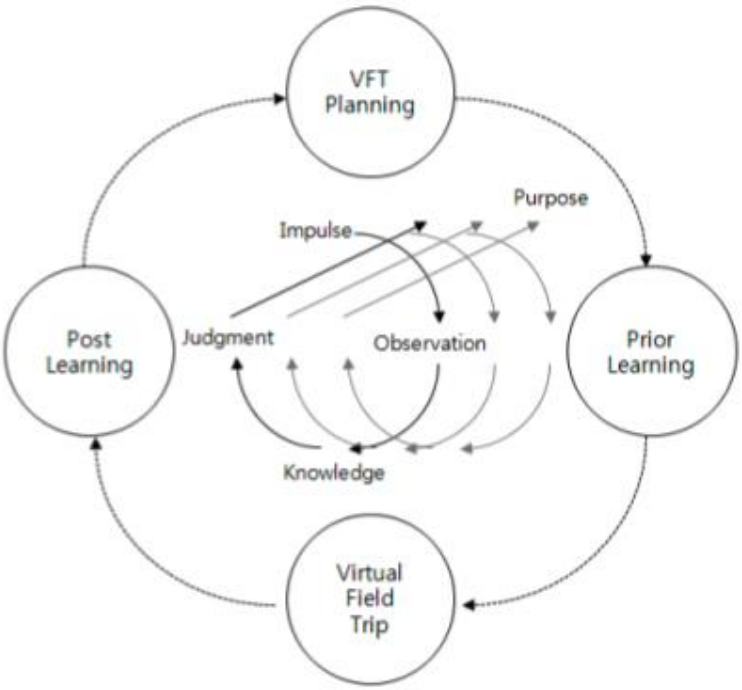

Figura 2 - Modelo de uma visita de estudo para aprendizagem experimental [8]

Na fase de planeamento da visita de estudo (*VFT Planning*) o professor deve dar a conhecer o objectivo e assunto da visita, métodos de actividades e uma lista de possíveis localizações para visitar. Na fase de pré-aprendizagem (*Prior Learning*) os alunos devem recolher informação sobre o assunto a explorar na visita de estudo com vista a escolherem o local a visitar da lista anteriormente divulgada pelo professor. A fase da visita de estudo propriamente dita (*Virtual Field Trip*) envolve observação, investigação e discussão usando funcionalidades do *Second Life*. Na última fase, os alunos devem organizar todos os dados recolhidos para a elaboração de dois relatórios, um final e outro de reflexão [8].

2.1.4 Comercial

Com o aumento de compras e vendas de produtos *online* os comerciantes tradicionais têm que se adaptar às novas tecnologias.

Muitas companhias começam já a utilizar mundos virtuais como forma de publicidade para os seus produtos, diminuindo assim os custos associados a colocar publicidade noutro meio. Ao utilizar os mundos virtuais as companhias recebem também *feedback* por parte dos outros utilizadores sobre os seus produtos. Esta forma de chegar ao público-alvo, começa a tornar-se crucial na fase de desenvolvimento pois, desta forma, a companhia tem informações reais sobre o que o mercado espera dos seus novos produtos.

Da mesma forma, outras companhias criam pontos de venda, ou seja, lojas virtuais, nestes mundos, onde colocam todas as informações relevantes sobre os seus produtos, permitindo assim a que outras empresas tenham também acesso a essas informações. Assim,

empresas geograficamente separadas umas das outras podem fazer negócios facilmente sem custos de viagens e alojamento, aumentando assim a produtividade e poupando ao mesmo tempo. Um exemplo deste tipo de abordagem por parte das companhias são as ilhas criadas pela *Sun Microsystems* e Microsoft no mundo virtual do *Second Life*, de forma a promover a sua linguagem de programação – *C#*.

2.2 Second Life

O mundo virtual do *Second Life* [9] pode ser acedido via Internet e está disponível para qualquer pessoa em qualquer parte do mundo. Cada utilizador começa com uma conta básica, não necessitando do pagamento de qualquer mensalidade. Porém, conforme deseje evoluir no jogo, quer em termos de suporte que recebe por parte da equipa técnica, quer para obter porções de terra para construção, existem outros tipos de contas, com diferentes modalidades de mensalidade.

O *Second Life* é muito mais que um simples videojogo, pois apesar de ser visitado por muitos jogadores, este não é um mundo apenas para se jogar. Como o próprio nome leva a crer, o *Second Life* é uma experiência *online* muito rica e complexa e não um simples local para conversar com os amigos. A principal actividade neste mundo é a construção de objectos interactivos através de uma linguagem implementada para o efeito. Esta linguagem de programação é bastante simples, permite que qualquer pessoa, mesmo sem antecedentes na programação, consiga criar os ditos objectos. Depois de criados, os objectos podem ser doados ou vendidos por *Linden Dollars*. Estes dólares são a unidade monetária deste mundo e podem ser trocados por dinheiro real e vice-versa [9].

O mundo virtual do *Second Life* é totalmente construído pelos seus utilizadores, designados de residentes. Desta forma, pode dizer-se que este mundo é bastante rico em imaginação e criatividade, tornando-se, assim, apelativo a *designers* e a pessoas mais ligadas a áreas tecnológicas do que a indivíduos que preferem, apenas e só, socializar.

Cada utilizador começa por fazer um registo no *site* oficial do jogo [9] e após o descarregamento e instalação do *Second Life Viewer*, pode entrar neste mundo.

A primeira tarefa do novo residente será a de personalizar o seu *avatar*. Para isso, tem à sua disposição um número bastante elevado de opções – desde a cor dos olhos até ao *design* da camisola, tudo é personalizável. Alguns utilizadores tentam criar uma versão computadorizada dos seus corpos reais enquanto outros tentam criá-los o mais estranhamente possível.

Quando se entra no mundo do *Second Life*, o novo residente pode começar desde logo, caso o deseje, por fazer um *tutorial* que lhe vai explicar quais as funcionalidades básicas que este tem à sua disposição no mundo virtual. Apesar deste *tutorial* e de todos os ficheiros de ajuda, quer *in game* quer através do *site*, serem bastante explícitos e de fácil compreensão,

a curva de aprendizagem deste mundo virtual é bastante difícil de ultrapassar. A partir deste momento o novo residente está apto para começar a explorar este novo mundo, conhecer e interagir com outros residentes.

2.3 Google

A Google é uma das maiores empresas mundiais quando se fala em Internet [10]. A sua missão é "organizar toda a informação do Mundo e torná-la universalmente acessível e utilizável. O *Googleplex*, sede da Google, está localizado na Califórnia, *Mountain View* e neste momento conta já com mais de vinte mil empregados. Foi fundada em 1998 por dois estudantes da *Stanford University*: Larry Page e Sergey Brin. É uma empresa que faz lucro com publicidade *online* relacionada com os seus serviços de:

- Pesquisa;
- Correio electrónico;
- Mapas Online;
- Produtividade em escritórios com as suas ferramentas online;
- Redes sociais;
- Partilha de vídeos.

Ou vendendo o mesmo *software* mas sem publicidade envolvida.

Apesar de ter começado apenas como uma pequena empresa privada, depois de uma série de novos programas desenvolvidos, novas aquisições e parcerias, a Google tornou-se uma das maiores empresas do mundo. Foi, por mais que uma vez, considerada pela revista *Fortune Magazine* como o melhor local para se trabalhar [11] e, segundo o *MillwardBrown Group*, como a marca mais poderosa do mundo [12].

A Google conta também com uma das maiores redes sociais activas actualmente: o Orkut.

2.3.1 Lively

O *Lively* [13] é uma rede de avatares e quartos virtuais que podem ser decorados pelos seus utilizadores. Este serviço foi lançado pela *Google* em Julho de 2008 com o objectivo de dotar os seus utilizadores de melhores meios para se expressarem na Internet. Este mundo virtual, tal como o do *Second Life*, é criado pelos seus residentes e cresceu bastante nos primeiros tempos após o seu lançamento. Porém, os servidores fecharam ao público a 1 de Janeiro de 2009, terminando assim esta experiência. O mundo virtual do *Lively* podia ser acedido a partir de um simples *browser* – *Mozilla Firefox* ou *Internet Explorer* – através de *Windows Vista* ou *XP*, após descarregar e instalar um pequeno *plugin*.

Ao contrário do *Second Life*, no mundo do *Lively* não se podia comprar ou vender conteúdo, visto este não poder ser gerado pelos utilizadores. Todo o conteúdo era pré desenhado e estava exposto num catálogo.

2.3.2 Google Earth

O Google Earth [14] é um software que permite observar um modelo tridimensional do globo terrestre, construído a partir de imagens de satélite, imagens aéreas ou imagens a três dimensões.

O Google Earth pode ser usado como um gerador de mapas a duas dimensões utilizando as fotografias de satélite ou como um mostrador das paisagens do nosso planeta: identificando locais, construções, cidades, paisagens entre outras. Recentemente a Google incluiu novas funcionalidades neste programa:

- *Google Mars*;
- *Google Sky*;
- *Google Ocean*;
- *Google Moon*;
- Imagens históricas.

O *Google Mars* [15], permite observar o planeta Marte em alta resolução da mesma forma que se pode observar a Terra. O *Google Sky* [16] permite navegar pelo universo: Vénus, Saturno, entre outros. Da mesma forma, o *Google Ocean* [17] permite a visualização de elementos ligados à superfície e ao fundo dos oceanos, tais como:

- Locais de mergulho;
- Locais indicados para pesca submarina;
- Naufrágios;
- Pontos de *surf;*
- Áreas de protecção marinha;
- Entre outros.

Para comemorar o trigésimo sexto aniversário da aterragem da *Apollo 11* na Lua, a 20 de Julho de 1969, a *Google* lançou uma funcionalidade denominada *Google Moon* [18]. Este novo *update* é constituído por um conjunto de imagens públicas da Lua com algumas características particulares: o ponto de aterragem da *Apollo 11* na Lua, por exemplo, é um desses pontos. Esforços mais recentes com a *NASA Ames Research Center*, permitem que a *Google* actualize e detalhe melhor toda a informação relativa a este *software*.
Com a mais recente actualização é ainda possível fazer a comparação de algumas imagens de locais que inevitavelmente sofreram alterações ao longo do tempo.

Das mais importantes funcionalidades do Google Earth destaca-se ainda a topografia, isto é, dados terrestres recolhidos pela missão SRTM – Missão Topográfica Radar *Shuttle*. Esta missão tem o objectivo de obter um modelo digital do terreno da zona da Terra entre 56º S e 60º N de forma a gerar uma base completa de cartas topográficas digitais terrestres de alta resolução. Isto significa que se pode observar locais como o *Grand Canyon* ou o Monte Evereste a três dimensões. A Google possibilita também que os utilizadores possam usar uma

camada extra: *Layers – 3D buildings* que consiste em usar edifícios modelados a três dimensões de algumas das principais cidades dos Estados Unidos da América.

2.3.3 Google Earth API

O plugin do Google Earth e a sua API [19] permite introduzir um mundo virtual a três dimensões em qualquer *website*. Usando a API é possível desenhar linhas, colocar imagens sobre o terreno, adicionar modelos tridimensionais de edifícios e/ou outros objectos, permitindo aos utilizadores a construção de aplicações de mapeamento 3D com alguma sofisticação.

Utilizando esta API seria possível no futuro, compatibilizar o *2ndComing* com o Google Earth, expandindo assim os horizontes deste projecto e, consequentemente, a sua longevidade e utilidade para mais utilizadores em todo o mundo.

2.3.4 GoogleMaps

O *Google Maps* [20] é um serviço de mapeamento online fornecido gratuitamente pela Google, desde que para usos não comerciais.

Com este serviço, o utilizador tem à sua disposição mapas de estradas, um planeador de rotas para caminhadas, viagens de bicicleta, carro ou mesmo por transportes públicos, podendo também ajudar a encontrar a localização do mais variado tipo de estabelecimentos: comerciais, museus, escolas, entre outros.

Esta aplicação pode ser acedida em qualquer website que inclua o *plugin* da Google Maps API, em telemóveis e dispositivos de outras empresas preparados para o efeito, nomeadamente alguns GPS.

Este projecto da Google está intimamente ligado ao Google Earth visto que em muitas das funcionalidades são idênticos.

2.4 Microsoft

A Microsoft [21] é uma multinacional dos Estados Unidos da América, considerada a maior empresa do mundo no ramo dos *softwares*. Emprega cerca de oitenta e nove mil pessoas em cento e cinco países diferentes. É, actualmente, uma das empresas que mais investe em pesquisa e desenvolvimento no mundo.

2.4.1 Xbox Live

O *XBOX Live* [22] é um serviço online de jogos com suporte para vários jogadores e de distribuição de media digital criado e mantido pela Microsoft. Actualmente, é o único serviço de jogos online que cobra aos seus utilizadores. Apesar de ter sido criado a pensar nas

consolas domésticas de videojogos, primeiro para a XBOX e, mais tarde, adaptado e relançado para a *XBOX 360*, este serviço também pode ser acedido na plataforma *Windows*, com o nome *Games for Windows – Live*. Assim, grande parte das funcionalidades do serviço presentes na consola estão também presentes para utilizadores *Windows*.

A Microsoft anunciou também a intenção de expandir este serviço para outras plataformas, como dispositivos de media portáteis, incluindo telemóveis, como parte do projecto *LiveAnywhere*.

O *Xbox Live* conta com um vasto número de funcionalidades que permitem ao utilizador uma experiência única no mundo de jogos online. De entre todas as funcionalidades destacam-se:

- Criação de avatares;
- Controlo Parental;
- Lista de Amigos;
- *Achievement Points* – ganhos no decorrer de jogos;
- *Gamerscore* – soma de todos os *achievement points*;
- Sistema de queixa de má conduta de outros utilizadores;
- Integração com o *Windows Live Messenger*;
- Acesso ao *Xbox Live Marketplace*;
- Comunicação por texto, voz e vídeo entre utilizadores;
- Salas de conversação;
- Marcação de jogos online.

O *Marketplace* permite o acesso a:

- Novos jogos;
- Complemento para jogos já existentes;
- Música;
- Vídeos de jogos;
- Filmes – *video on demand*;
- Entre outros.

Recentemente a Microsoft anunciou novas adições ao seu serviço de jogos online de entre as quais se destacam:

- Suporte futuro para interligação com algumas das redes sociais mais conhecidas: *Facebook*, *Twitter* e *Last.fm*.
- No Reino Unido, através de uma parceria com a *Sky TV*, será possível aceder a conteúdo *Premium* e assistir a televisão em directo a partir da *Xbox 360*.
- Filmes de alta definição (1080p) com som *surround* 5.1 permitindo que os utilizadores façam *stream* a partir do *Marketplace*.
- *Movie Parties* permite que filmes e programas de televisão sejam vistos em conjunto com amigos através do *Xbox Live*.

- *MusicStore* permite a compra de conteúdo para jogos intimamente ligados ao mundo da música: *Rock Band, Guitar Hero, Lips* e outros.
- *Games on Demand* – sistema de compra de jogos *online*.

2.4.2 Project Natal

O *Project Natal* [23] é o nome de código de um projecto que pretende introduzir uma nova forma de jogar: sem controlador ou, melhor dizendo, o controlador será o corpo de cada jogador.

Baseado num novo acessório para a sua consola doméstica, o *Project Natal* permite que os jogadores interajam com a sua Xbox 360 sem a necessidade de usar um comando de jogo, numa interface natural com o uso de gestos, comandos de voz ou ainda apresentando objectos e/ou imagens.

Este novo acessório apenas estará disponível durante o decorrer de 2010 e é esperado que a Microsoft utilize esta tecnologia como base para a sua próxima consola.

O *Project Natal* constituído por uma câmara de vídeo, um sensor de profundidade, um microfone e ainda um processador criado à medida para o projecto. Este acessório permite:

- Captação de movimentos tridimensionais de todo o corpo;
- Reconhecimento facial;
- Reconhecimento de voz.

O microfone consegue localizar a origem da voz e reduzir o ruído ambiente presente localmente permitindo assim aos utilizadores o uso deste para aplicações muito vastas como dar comandos de voz ou conversações com amigos através do *Xbox Live*.

O sensor de profundidade permite capturar os movimentos tridimensionais dos utilizadores em qualquer ambiente luminoso. O alcance do sensor pode ser regulado, sendo que este se calibra de acordo com o ambiente luminoso e de jogo, como por exemplo a presença de cadeiras e sofás.

Segundo a Microsoft, com o *Project Natal*, o mundo assistirá a uma nova era dos videojogos, atraindo assim novos públicos, nomeadamente aqueles que não se sentem à vontade com um comando de jogo nas mãos e ainda os jogadores mais casuais.

2.5 OpenStreetMap

O *OpenStreetMap* [24] é um projecto com o objectivo de criar um mapa do mundo, livre de custos. Os mapas são criados a partir de informações de aparelhos GPS portáteis, fotografia aérea ou outras formas sem custo ou de conhecimento local.

O mapa inicial foi construído por voluntários que faziam leituras sistemáticas no terreno usando um dispositivo GPS móvel e um computador portátil, armazenando depois a informação recolhida na base de dados da OpenStreetMap. Estes voluntários movimentavam-

se a pé, bicicleta ou carro, apesar de as bicicletas serem o meio preferido para mapearem as zonas urbanas.

Mais recentemente, a disponibilização de fotografias aéreas e de outros tipos de conteúdo quer por parte de instituições comerciais quer governamentais aumentou o ritmo do trabalho.

A informação do OpenStreetMap é publicada segundo uma *open content license* com o objectivo de promover o uso livre e a redistribuição da informação. Assim, torna-se perfeita a utilização do *OpenStreetMap* para gerar estradas no *2ndComing*. É notório que existe bastante trabalho para ser realizado em termos de cobertura de zonas, sendo que Portugal está apenas coberto nas suas duas principais cidades: Lisboa e Porto. Porém, com o passar do tempo, a cobertura será alargada permitindo assim gerar novas cidades com o *2ndComing*. De notar que cada utilizador pode recolher as informações pretendidas de várias formas:

- *OpenStreetMap XML data;*
- *Mapnik Image;*
- *Osmarender Image;*
- *HTML;*

De todas as formas apresentadas, a que mais interessa a este projecto é o ficheiro XML para efeitos de leitura pelo *2ndComing*.

Porém, o *OSM* tem o pequeno problema de não disponibilizar informação acerca do relevo. Para se obter a informação de relevo será necessário pesquisar outro serviço.

A informação sobre outro tipo de conteúdos idênticos disponibilizados por outros *sites* pode ser consultada na Tabela 1.

Tabela 1: Conteúdos semelhantes em vários portais *online*

Site	Descrição de Conteúdo
OpenStreetMap	Estradas.
OpenSeaMap	Mapas náuticos.
FreeMap	Caminhadas.
Tobo	Caminhadas e ciclismo.
OpenCycleMap	Ciclismo.
YourNavigation	Planeamento de Rotas.
OpenRouteService	Planeamento de Rotas.

Visto que estes serviços de mapeamento fornecem informações acerca de outros conteúdos, podem também ser usados no futuro para gerar esses mesmos conteúdos para o *2ndComing*.

2.6 Tipos de Conteúdo

O termo "conteúdo" é muito vasto e, por essa razão, é necessário concretizar um pouco que tipos de conteúdo o *2ndComing* poderá suportar. Ainda mais importante que o tipo, é a forma real como se irá gerar esse conteúdo.

2.6.1 Conteúdo Estático

Este tipo de conteúdo tem uma característica que o distingue dos outros que é o facto de se manter inalterável ao longo do tempo. Casos particulares de conteúdos estáticos são: objectos, edifícios, monumentos, entre outros.

2.6.1.1 Blueprints

Uma planta de um edifício contém toda a informação relativa à estrutura do mesmo. Assim, é possível a partir de uma planta gerar a imagem a três dimensões do respectivo edifício. O maior problema deste método que se torna, em certa medida, fulcral, é a falta de informação do aspecto dos elementos exteriores do edifício – portas, janelas e paredes. Para usar este método na geração de conteúdo seria necessário obter a informação desses elementos a partir de outros dados, como, por exemplo, fotografias do edifício em questão.

Outra limitação deste método é o facto de se aplicar apenas a edifícios – moradias, prédios, apartamentos – e não poder ser usado para gerar outro tipo de conteúdo estático.

2.6.1.2 Fotos Aéreas

É relativamente fácil e acessível a todos, obter fotos aéreas de um determinado edifício e/ou zona, utilizando ferramentas como o *Google Earth*, por exemplo. A partir dessas fotos será fácil mostrar o enquadramento de um determinado edifício na área que o rodeia.

Porém, este método tem várias limitações: tal como o método por *blueprints*, não nos dá qualquer tipo de informações sobre o aspecto exterior nem interior. Assim sendo, pode ser visto como um complemento para o método de geração por plantas de edifícios, dando informações relativas ao enquadramento dos mesmos, isto é, à zona que os rodeia.

2.6.1.3 Sequência de Fotos

Método normalmente aplicado a objectos de menores dimensões que os métodos por *blueprints* e fotos aéreas, visto que permite capturar detalhes que os outros não conseguem. O princípio de operação consiste em tirar uma sequência de fotos de modo a que todo o objecto seja documentado. Posteriormente, a partir das fotos tiradas, é reconstruída a imagem desse objecto, a três dimensões. Alguns algoritmos permitem uma boa aproximação ao objecto real sendo por isso um método a considerar para inclusão no *2ndComing*.

2.6.1.4 3D CAD Browser

O *3D CAD Browser* [25] é um sistema de trocas de modelos 3D *online,* onde se podem descarregar modelos de alta qualidade e em troca colocar modelos criados pelo utilizador. Através de um catálogo disponível no *site*, dividido em categorias, pode facilmente encontrar-se os modelos desejados para o conteúdo que se pretende gerar. Visto que é necessário criar modelos e colocá-los no *site* de forma a poder descarregar outros modelos, não será viável a longo termo o uso deste serviço. Porém, a ideia será tirar partido deste tipo de modelos, disponíveis em variadíssimos formatos de ficheiros, neste ou noutro *site* onde seja mais fácil descarregar os modelos, para a geração de conteúdo.

Dois exemplos de modelos são os que se encontram na Figura 3 e na Figura 4.

Figura 3 - Modelo 3D do Audi S5

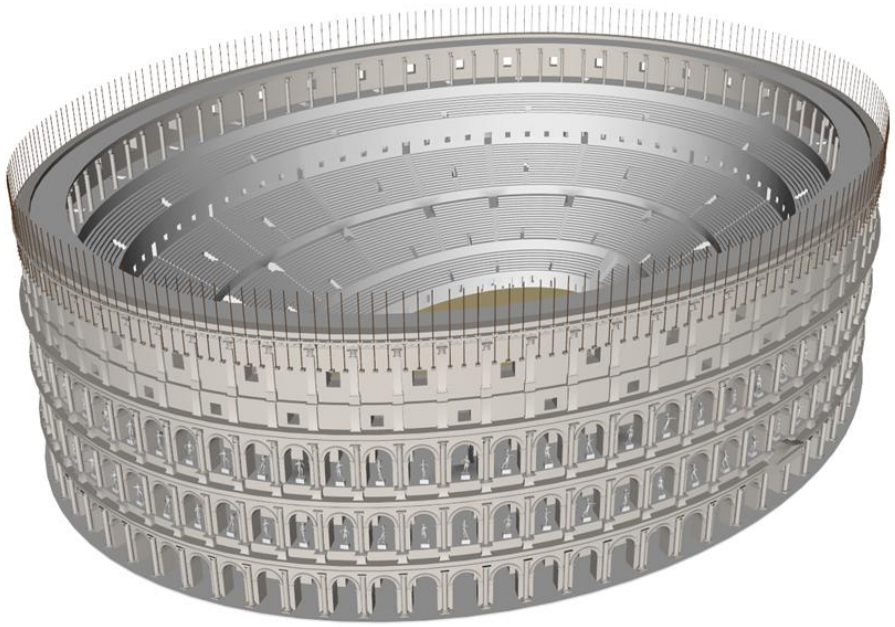

Figura 4 - Modelo 3D do coliseu de Roma

A quantidade de modelos para descarregar é elevada, podendo assim criar-se os mais variadíssimos conteúdos.

2.6.1.5 Automatic Building Generation

O *Automatic Building Generation* [26] é um *software* desenvolvido pelo *Graphics and Gaming Group* do *Institute of Technology Blanchardstown* com o objectivo de gerar modelos tridimensionais de edifícios de forma automática de modo a serem usados em aplicações de tempo real como um videojogo ou uma simulação virtual. Um dos aspectos mais importantes do trabalho é o de poder baixar os custos financeiros e consumo de tempo associados à modelização 3D de edifícios que, normalmente, é bastante demorada e complicada. Assim, este algoritmo gera apenas geometria dinamicamente de forma a que o resultado final se assemelhe a um edifício.

2.6.1.6 Conversão de 2D para 3D

Este método é bastante simples e está a ser estudado para ser o futuro da televisão. O seu princípio de funcionamento consiste em alterar o visionamento de uma dada imagem a duas dimensões de modo a dar a sensação de estar a ser visualizada a três dimensões [27].

Se for tirado um conjunto de imagens sequenciais de um determinado objecto em movimento e posteriormente aplicada esta técnica a todas essas imagens, sequencialmente, será possível ver o objecto em movimento, a três dimensões. Neste caso, tratar-se-ia então de geração de conteúdo dinâmico. É uma técnica interessante mas neste momento ainda não

tem resultados suficientemente credíveis para ser utilizada no desenvolvimento do *2ndComing*.

2.6.2 Conteúdo Dinâmico

Conteúdo dinâmico difere do estático porque vai sofrendo alterações ao longo do tempo, quer seja na forma, estado, comportamento ou localização. O percurso feito por uma pessoa numa determinada zona é um bom exemplo de conteúdo dinâmico que pode ser gerado.

2.6.2.1 Geração de cidades on demand

Este método consiste em criar uma cidade através da geração de vários edifícios geometricamente distintos, aleatoriamente, conforme vão sendo necessários, isto é, enquanto um utilizador vai andando pela cidade virtual, os edifícios vão sendo criados, tornando esta cidade infinita e diferente de cada vez que é visitada.

Os edifícios são gerados com formas geométricas, dependendo do número de planos e do número de iterações que o telhado pode ter, entre outras opções.

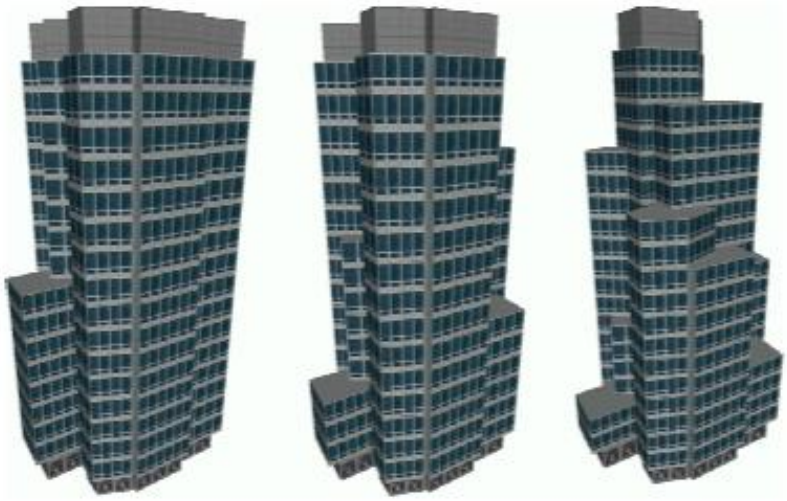

Figura 5 - Exemplos de edifícios com 2, 4 e 8 planos respectivamente [28].

Figura 6 - Exemplos de edifícios com 2, 3 e 5 iterações respectivamente [28][28].

A geração de cidades *on demand* [29] é bastante interessante visto que permite gerar não só uma cidade mas, quando combinado com um simulador de tráfego, também o movimento de pessoas nessa cidade. Nas Figuras 7 e 8 podem ser observados alguns exemplos de cidades

e edifícios criados com este projecto. Estas cidades são diferentes no tamanho, já que a primeira apenas tem cem edifícios enquanto que a segunda tem mil.

Figura 7 - Exemplo de cidade gerada com 100 edifícios [28].

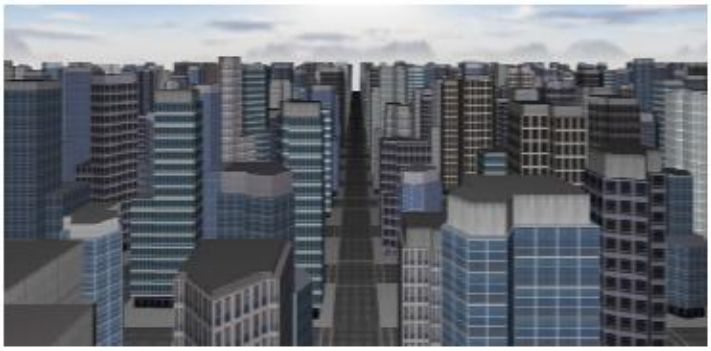

Figura 8 - Exemplo de cidade criada com 1000 edifícios [28].

2.7 Simuladores

Para a geração de conteúdo dinâmico simulando movimentos, quer de pessoas quer de automóveis, a melhor solução é recorrer a um simulador. Todos os simuladores aceitam parâmetros relativos à simulação que se pretende realizar de modo a que esta seja feita dentro do meio e condições pretendidas. Assim, um simulador seria ideal para o projecto pois pode interagir com o *2ndComing* de modo a receber as condições da zona onde se pretende simular o movimento de pessoas e, durante a simulação ou no final desta, comunicar com o *2ndComing* de modo a que este possa inserir no mundo virtual os resultados da simulação.

2.7.1 Simuladores de tráfego

Tal como o próprio nome indica, os simuladores de tráfego são programas que tentam simular, com maior ou menor certeza, os caminhos seguidos por um determinado meio de locomoção: avião, carro, pessoa, pacotes em redes de computadores, entre outros. Uma simples aplicação usando estes simuladores pode ser feita com o objectivo de se perceber o porquê de, a uma determinada hora do dia, os carros circularem normalmente numa estrada e, quando chega à hora de ponta, os carros circularem muito mais lentamente [29].

Este tipo de simuladores é usado para as mais diversas aplicações desde regulação de tempos de semáforos em intersecções ao dimensionamento de pontes e são os desejados para gerar conteúdo dinâmico devido a serem específicos para os dados que se querem introduzir no *2ndComing*.

2.7.1.1 ATC-SIM

O ATC-SIM é um simulador de tráfego aéreo que pode ser acedido directamente de qualquer navegador de Internet. Está implementado numa aplicação de modo a que quem o aceda possa fazer o controlo do tráfego aéreo de uma determinada zona, como se de um simples jogo se tratasse. Aceita vários parâmetros diferentes como o aeroporto, tipo de códigos aéreos, direcção do vento e realismo/grau de dificuldade.

2.7.1.2 SATURN

O SATURN é um simulador de tráfego criado pela *Atkins-ITS*, lançado há 25 anos e está já alojado em mais de trinta países. O tempo que está no mercado pode ser contabilizado como uma vantagem ou desvantagem. Pode ser uma vantagem no sentido em que está já muito explorado e são conhecidas todas as suas capacidades e falhas, sendo que o suporte é muito vasto e é relativamente fácil encontrar soluções para os problemas encontrados. Por outro lado, pode também ser uma desvantagem pois a tecnologia já evoluiu bastante desde o seu lançamento e neste momento existem outras alternativas com novas tecnologias.

2.7.1.3 DRACULA

O *Dynamic Route Assignment Combining User Learning and Microsimulation*, também conhecido como DRACULA, pode ser uma boa alternativa ao SATURN, pois é mais recente. Permite também fazer a conversão de simulações do SATURN para o DRACULA. O DRACULA é um micro simulador de tráfego desenvolvido no *Institute for Transport Studies,* na *University of Leeds* e tem a vantagem de ser compatível com o *SATURN* e de ter animações que torna mais fácil o visionamento da simulação. Este simulador é orientado a aplicações operacionais e dinâmicas, que evoluem ao longo do tempo. Neste tipo de aplicações o DRACULA tem vantagem sobre o SATURN pois este tem mais dificuldades com simulações envolvendo conteúdo dinâmico.

2.8 OpenSimulator

O *OpenSimulator* [29], também conhecido por *OpenSim* (ou OS) é um servidor *open source* para a criação de mundos virtuais. Embora seja principalmente conhecido pela sua compatibilidade com o cliente do *Second Life*, é também capaz de recriar mundos alternativos com diferentes características.

A sua arquitectura permite uma expansão fácil através do uso de *plugin modules* o que permite também que vários interessados no desenvolvimento de *software* para o servidor o possam fazer por conta própria, aumentando assim a comunidade de utilizadores deste servidor *open source*.

Múltiplos servidores podem ser integrados numa grelha, chamada de *OSGrid* [30], e usados em simultâneo para criar mundos virtuais com uma área maior. O OpenSim pode ser usado de duas formas diferentes: *grid mode* e *standalone mode*. Enquanto no modo *standalone* apenas um processo gere toda a simulação, quando operado em *grid mode* são necessários vários processos interligados e, dependendo do tamanho da área que se pretende simular e dos utilizadores que esta deverá suportar, poderá ser necessário o uso de várias máquinas para correr a simulação.

Actualmente, o OpenSimulator usa o protocolo do *Second Life* para comunicações entre cliente e servidor o que lhe permite ser compatível com *o World Viewer* criado pela Linden *Lab* para o *Second Life*. O protocolo do *Second Life* utiliza quer UDP quer XML-RPC. O *Remote Procedure Call* utiliza XML para codificar os seus pedidos e HTTP para fazer o transporte dos mesmos. À medida que novas funcionalidades foram sendo introduzidas, este protocolo começou a ser conhecido como SOAP. Porém, apesar da evolução, muitos programadores continuaram a preferir usar o XML-RPC devido ao seu minimalismo, simplicidade e facilidade de utilização.

No futuro, existe a possibilidade de novos protocolos virem a ser implementados, na medida em que estes aumentariam a compatibilidade do *OpenSim* com outros softwares disponíveis no mercado. Um dos protocolos propostos para implementação é *o Metaverse Exchange Protocol* (MXP) que é um protocolo de comunicação entre clientes e servidores de segunda geração. Internamente, as diversas componentes do OpenSim comunicam entre si através dos protocolos XML-RPC e REST (JSON/HTTP e XML/HTTP).

O OpenSim tem um largo número de opções de entre as quais se destacam:

- Criação e personalização de avatares;
- Conversações com outros utilizadores no mesmo ambiente;
- Ferramentas de construção para a geração de conteúdo 3D no mundo virtual;
- Criação de aplicações 3D personalizadas;
- Criação de regiões de 256 por 256 metros, em um ou vários computadores distintos;
- Obedecer a leis da física como a gravidade;
- Desenvolvimento de aplicações *in game* com recurso às linguagens de programação: LSL, C# e *Javascript*.

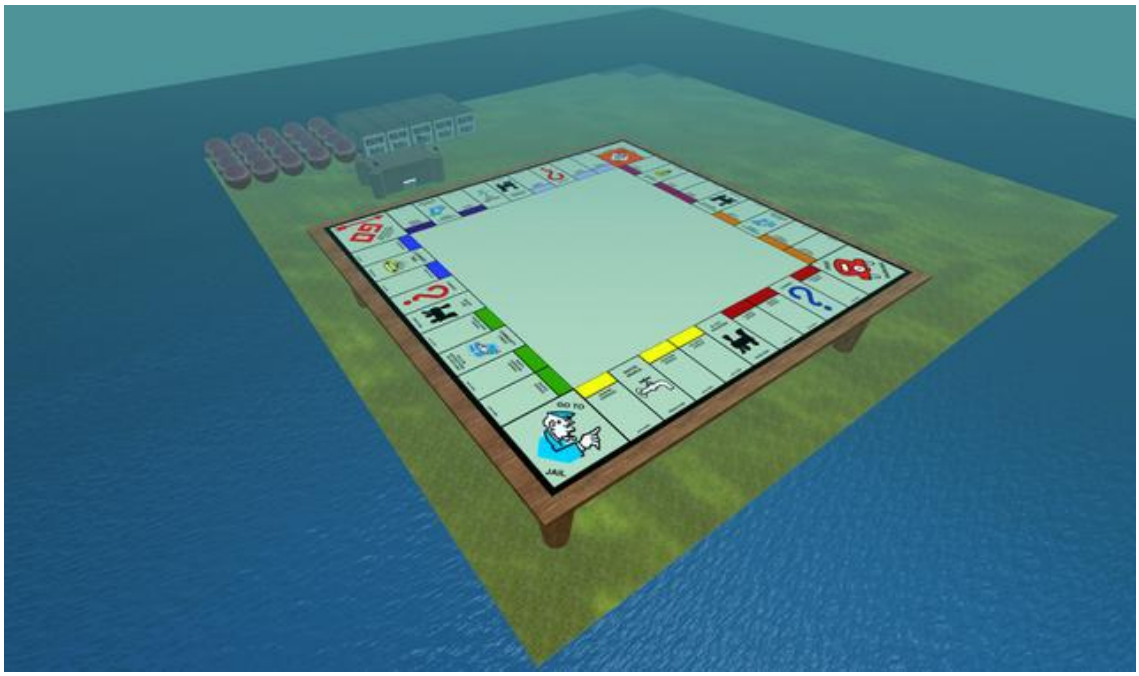

Figura 9 - Exemplo de uma aplicação criada no OpenSimulator

Qualquer utilizador pode colocar o seu mundo virtual, criado no seu servidor privado, na *OSGrid* dando assim acesso a todos os utilizadores da *grid* ao conteúdo presente no seu servidor, bem como a interactividade entre os utilizadores e esse conteúdo.

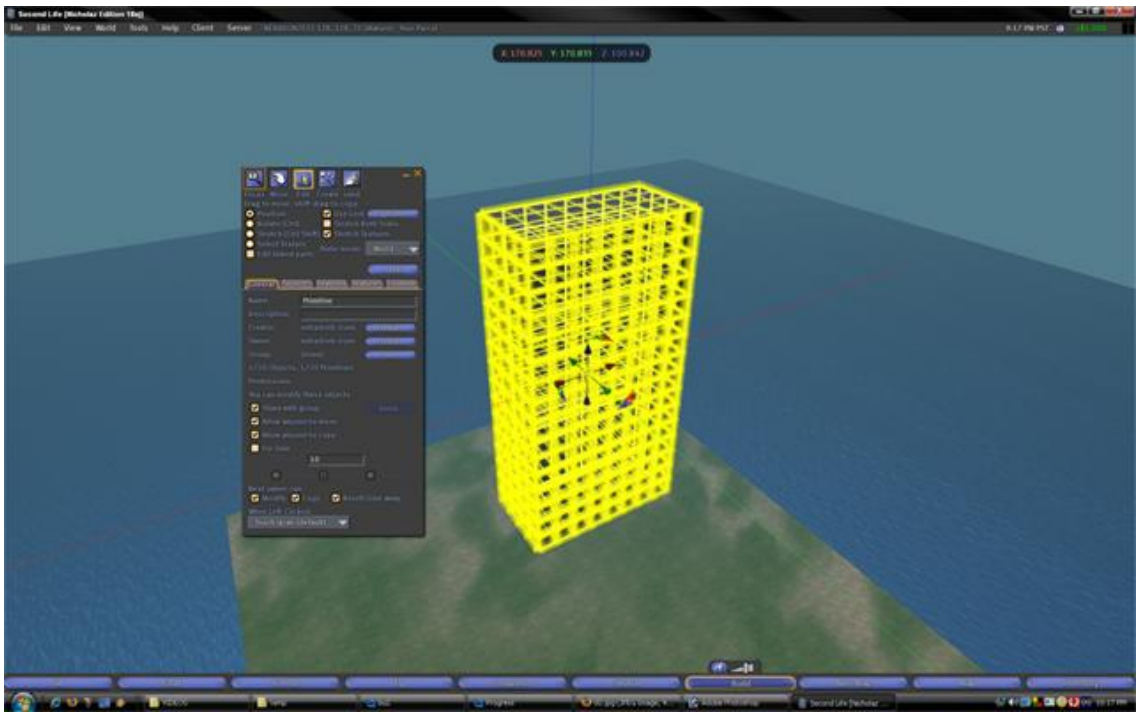

Figura 10 - Exemplo de uma construção no mundo virtual do OpenSimulator

Como se pode observar na Figura 9, Figura 10 e Figura 11, as ferramentas para criar e editar conteúdo são bastante idênticas às usadas no próprio *Second Life* o que, por si só, é outra grande vantagem para o uso deste simulador.

As cidades são também bastante parecidas com as do *Second Life*, dando um sem número de diferentes possibilidades apenas limitadas pela imaginação e criatividade de cada utilizador. Um bom exemplo disso é a zona da cidade que se apresenta na Figura 11.

Figura 11 -Exemplo de cidade criada no mundo do *OpenSimulator*

Capítulo 3

Arquitectura

Neste capítulo será apresentada toda a arquitectura do projecto quer globalmente quer, posteriormente, a um nível mais pormenorizado.

Um dos principais objectivos do projecto é o de não o limitar aos geradores de conteúdo actuais mas sim dotá-lo de meios para que no futuro sejam desenvolvidos mais geradores e a ligação entre estes e o *software* já desenvolvido seja o mais simples possível. Com este princípio em mente, todo o software foi desenvolvido por módulos, bastando assim ligeiras alterações para interligar novos geradores.

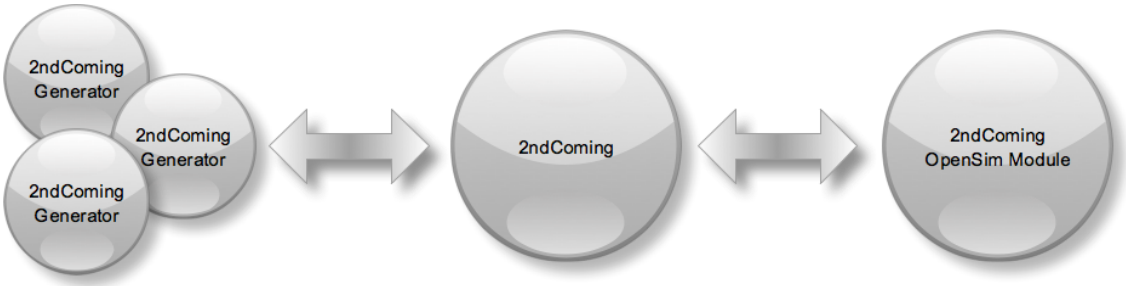

Figura 12 - Arquitectura geral genérica do *2ndComing*

Pela observação da Figura 12, pode-se concluir que o projecto foi dividido em três secções distintas. Esta divisão foi feita devido ao facto das secções terem funções específicas e distintas dentro do projecto. Para fazer a ligação entre as secções, utiliza-se o *Hypertext Transfer Protocol* (conhecido como HTTP). Foi escolhido este protocolo por ser amplamente usado pelo *Second Life*, *OpenSimulator* e outros *softwares* com características e funcionalidades idênticas.

Todos os módulos seguem uma arquitectura tipo (Figura 13), embora na implementação estas possam variar ligeiramente.

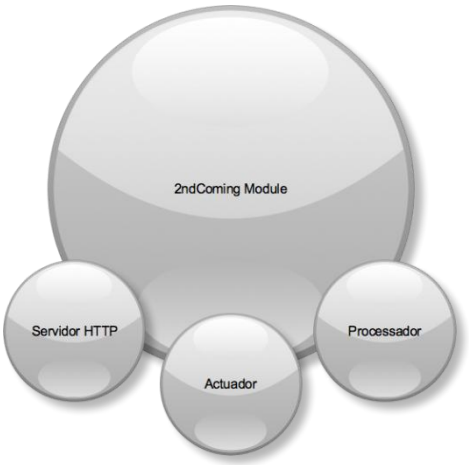

Figura 13 - Arquitectura genérica dos módulos *2ndComing*

Cada módulo tem obrigatoriamente que ser constituído por um Servidor HTTP, com o propósito de receber e enviar mensagens, e um Processador para interpretar as mensagens recebidas e preparar o envio de outras. Alguns dos módulos necessitam de outras funcionalidades pelo que a arquitectura de cada um será detalhada mais à frente neste capítulo.

3.1 2ndComing OpenSimulator Module

Este módulo será responsável pela acção, ou seja, tem a função de transformar toda a informação que recebe em conteúdo no mundo virtual. Toda a informação que recebe tem que ser preparada e/ou convertida para dados que possam ser usados para criar conteúdo no mundo virtual, isto é, algo perceptível pelo *OpenSim*.

Este módulo foi então desenvolvido como sendo um *plugin* adicional para o *OpenSimulator*, sendo, na prática, apenas um ficheiro *.dll* que terá que estar na pasta *bin* do *OpenSim* quando este for iniciado. Assim, quando o *OpenSim* carregar os módulos necessários ao seu funcionamento, carregará também o *2ndComing OpenSimulator Module*.

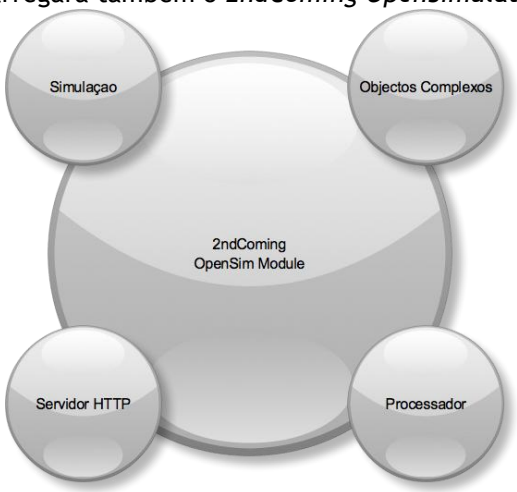

Figura 14 - Arquitectura do *2ndComing OpenSim Module*

Como se pode observar na Figura 14, este módulo é constituído por quatro secções distintas: Servidor HTTP, Processador, Simulador e Objectos Complexos.

O Simulador é a secção que trata exclusivamente do movimento dos autocarros no mundo virtual. Tratando cada autocarro de forma independente, estes fazem o seu percurso de forma cíclica, isto é, ao chegar ao final do percurso, voltam ao inicio e repetem tudo de novo até que o utilizador decida acabar a simulação.

Por outro lado, toda a secção dos Objectos Complexos trata de tudo o que envolve a criação, movimento e remoção dos mesmos.

3.2 2ndComing RoadGenerator

Este módulo tem a função de gerar toda a informação relativa às estradas que se pretendem criar no mundo virtual. A sua arquitectura é bastante semelhante à arquitectura tipo dos módulos apenas com uma pequena diferença, como se pode observar na Figura 15.

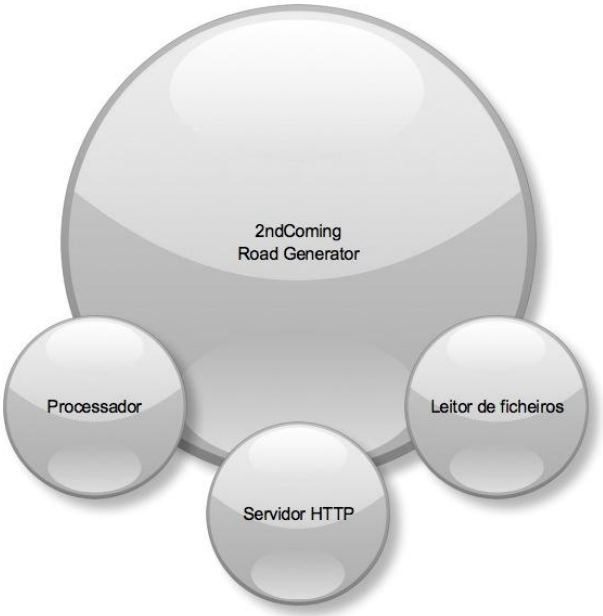

Figura 15 - Arquitectura *2ndComing RoadGenerator*

A única diferença é a de incluir uma secção para a leitura e interpretação dos ficheiros XML que será a secção mais importante e mais específica deste módulo. A leitura do ficheiro demorará mais tempo quanto maior for o tamanho do ficheiro que, por sua vez, será tanto maior quanto mais detalhe tiver a zona que se pretende gerar.

Para conseguir gerar as ruas, este módulo necessita de ter no mesmo directório que o seu executável, os ficheiros relativos às zonas que se pretendem gerar. Os mapas das zonas a criar deverão ser retirados do site do *OpenStreetMap* em formato XML. Cada ficheiro XML contém a informação de nós e de caminhos da respectiva zona da cidade. Enquanto os nós

servem apenas para dar as coordenadas e assinalar algum local com relevância, os caminhos são listas de vários nós, ou seja, cada caminho é constituído por um conjunto de nós.
O *2ndComing RoadGenerator* envia para o cliente toda a informação (nós e caminhos) pois esta será mais tarde usada por outros geradores, como o *BusGenerator* por exemplo, para ajudar a criar outros conteúdos.

3.3 2ndComing BusGenerator

Este módulo é responsável pela criação do movimento de autocarros. Cada zona deverá conter um ficheiro com informação relativa aos nós pelos quais cada autocarro passa no seu percurso.

A arquitectura deste ficheiro é muito semelhante a um ficheiro XML mas, como foi criada apenas para conter as informações vitais a este projecto, é bem mais simples e de fácil interpretação.

A arquitectura do módulo propriamente dita, Figura 16, é, em tudo, igual à do *2ndComing RoadGenerator* visto que também contém uma secção para fazer a leitura de ficheiros relativos às zonas que se pretendem criar.

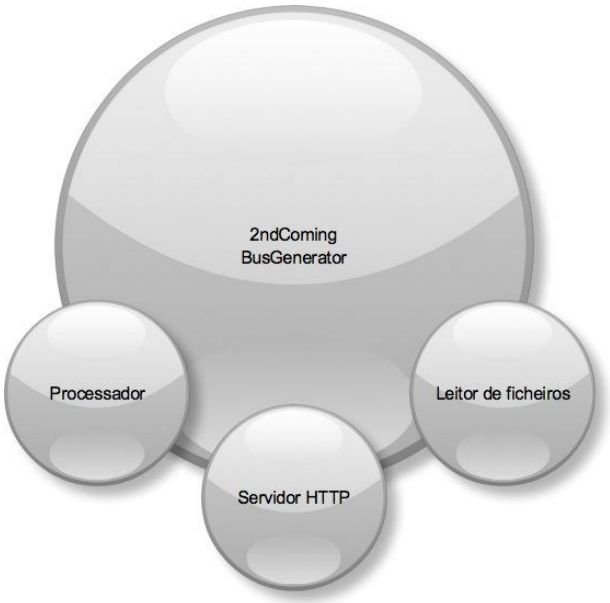

Figura 16 - Arquitectura *2ndComing BusGenerator*

O ficheiro com a informação do percurso dos autocarros de determinada zona contém apenas a referência dos nós por onde cada autocarro passa. Sendo que as coordenadas de cada nó estão já presentes no *2ndComing* e podem ser lidas conforme forem necessárias.
A solução de não enviar de novo informação já disponibilizada anteriormente, além de eliminar muita da redundância vai também diminuir o tráfego de rede e, consequentemente, o número e tamanho de mensagens trocadas.

Como no *RoadGenerator*, os ficheiros com a informação dos percursos dos autocarros deverão estar presentes no mesmo directório que o executável do *BusGenerator*, visto que, caso tal não aconteça, não será possível fazer a leitura do ficheiro.

3.4 2ndComing

Este será o principal módulo do projecto. A partir dele será possível controlar todos os outros e será com este que todos terão que comunicar.

O utilizador do *2ndComing* terá a sua interface de controlo através deste módulo, podendo executar qualquer operação suportada a partir daí.

A arquitectura deste módulo, Figura 17, será então um pouco diferente da dos restantes porém os princípios básicos são os mesmos.

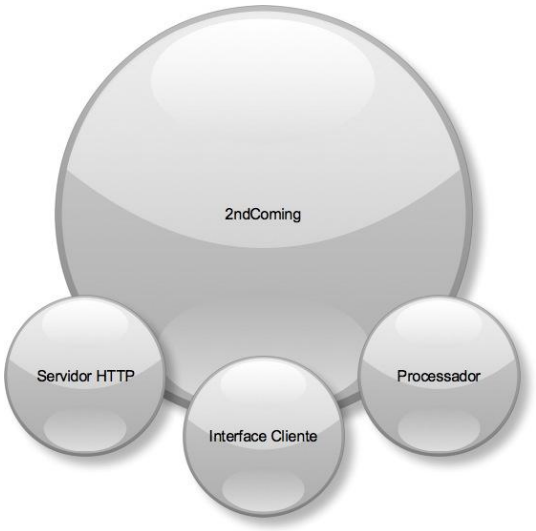

Figura 17 - Arquitectura *2ndComing*

Assim, além dos habituais Servidor HTTP e Processador, este módulo terá também que ter a interface de controlo para o utilizador. Esta interface é de uso bastante simples e fácil apesar de o controlo ser realizado através da linha de comandos. Desta forma cumpre-se mais um dos objectivos do projecto. Depois de vistas todas as arquitecturas individuais pode-se perceber melhor o conceito geral da arquitectura do *2ndComing*, na Figura 18.

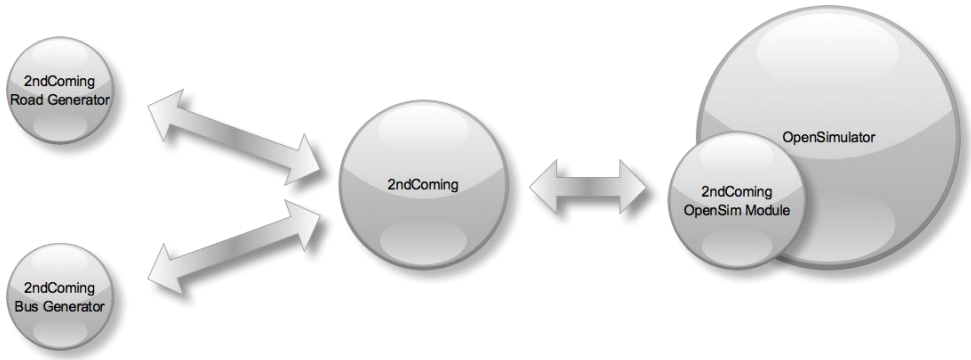

Figura 18 - Arquitectura geral específica do *2ndComing*

Como se pode observar, toda a informação que circula tem obrigatoriamente que passar pelo *2ndComing* fazendo deste o ponto central de todo o *software*. Desta forma consegue-se um controlo mais preciso de tudo o que se passa e de todas as mensagens trocadas e consequente prevenção de erros, tornando todo o processo mais robusto.

Capítulo 4

Implementação

Este capítulo é dedicado à forma como todas as funcionalidades do *2ndComing* foram implementadas e à explicação das mesmas. Numa primeira fase são explicados alguns conceitos para posteriormente ser mais fácil a percepção da implementação.

4.1 IRegion Module

O *IRegion Module* é uma estrutura criada pelos programadores do *OpenSimulator* para facilitar a integração de módulos de regiões com o seu simulador, devendo todos os módulos obedecer à mesma.

Durante o arranque do simulador, é feita uma procura nos directórios */bin* e */ScriptEngines* por ficheiros *.dll* na tentativa de carregar módulos de região contidos nesses ficheiros.

Todos os módulos de regiões devem seguir a seguinte interface *standard*:

```
public interface IRegionModule
{
    void Initialise(Scene scene, IConfigconfig);
    void PostInitialise();
    void Close();
    string Name { get; }
    bool IsSharedModule { get; }
}
```

A função *Initialise()* é chamada imediatamente após o módulo de região ter sido carregado pelo simulador. Nesta fase, é passada ao módulo, a referência da cena presente no simulador e este deverá armazená-la para uso futuro. Visto que o simulador ainda não está

totalmente operacional nesta fase, é vital não invocar funcionalidades que ainda não foram carregadas pelo simulador.

Depois de totalmente operacional e de todos os módulos de região estarem carregados, a função *PostInitialise()* será invocada em todos os módulos e será seguro usar qualquer funcionalidade do simulador nesta fase.

A função *Close()*, como o próprio nome indica, será invocada apenas quando o simulador estiver a ser encerrado.

A função *Name()* é onde se coloca o nome desejado para o módulo que se está a desenvolver. Esta função é invocada quando, no terminal do simulador, se executa o comando *"show modules"*. Este comando retorna o nome de todos os módulos de região a serem executados num determinado momento pelo simulador, como se pode observar na Figura 19.

```
Shared Module: OpenGridProtocolModule
Shared Module: Inventory Archiver Module
Shared Module: 2ndComing OpenSim Module
Shared Module: GroupsModule
Shared Module: LocalInterregionCommsModule
```

Figura 19 - OpenSim: Show Modules

Por último, a função *IsSharedModule()* tem o único objectivo de retornar o valor *true* ou *false* conforme o módulo seja ou não partilhado por todas as regiões. O processo do simulador permite executar múltiplas regiões em simultâneo. Se o módulo for partilhado, a função retorna o valor *true* e este será apenas carregado uma vez para todas as regiões activas no simulador. Por outro lado, caso a função retorne *false*, o módulo será carregado uma vez para cada região diferente a ser executada no simulador quando este é iniciado.

De seguida serão apresentadas algumas características a ter em conta de cada cena presente no simulador. Neste projecto apenas será usada uma cena pelo que os seus limites deverão ser conhecidos de forma a limitar o tamanho do conteúdo a introduzir nesta.

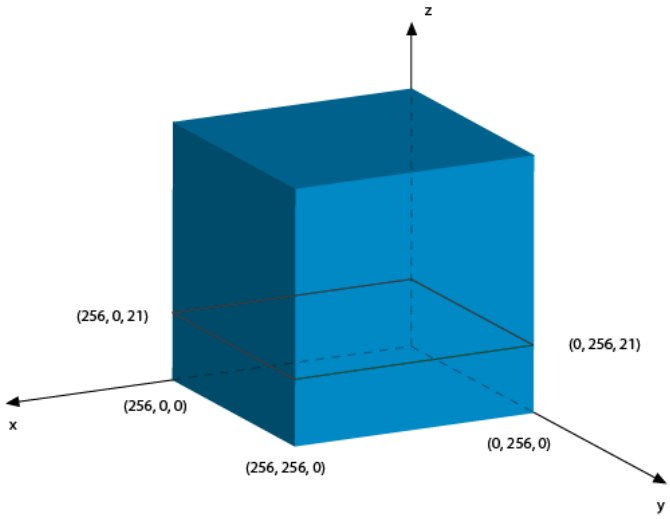

Figura 20 - Ilustração de coordenadas relevantes numa cena

Cada cena, a azul na Figura 20, tem uma área de 256 metros de comprimento por 256 metros de largura. Na mesma figura, podem ser observadas algumas coordenadas importantes. A linha negra no cubo, situada no plano onde z = 21metros, diz respeito ao nível do mar no *OpenSimulator*.

Para se poder criar conteúdo é portanto necessário fazer com que o nível do mar esteja abaixo do de terreno. Para fazer isso é necessário, na consola do *OpenSimulator* recorrer ao comando:

```
terrain fill 21
```

Este comando tem como objectivo fazer com que a superfície seja acima do nível da água, isto é, preencher toda a área da cena com terreno onde se pode construir, como se pode ver na Figura 21.

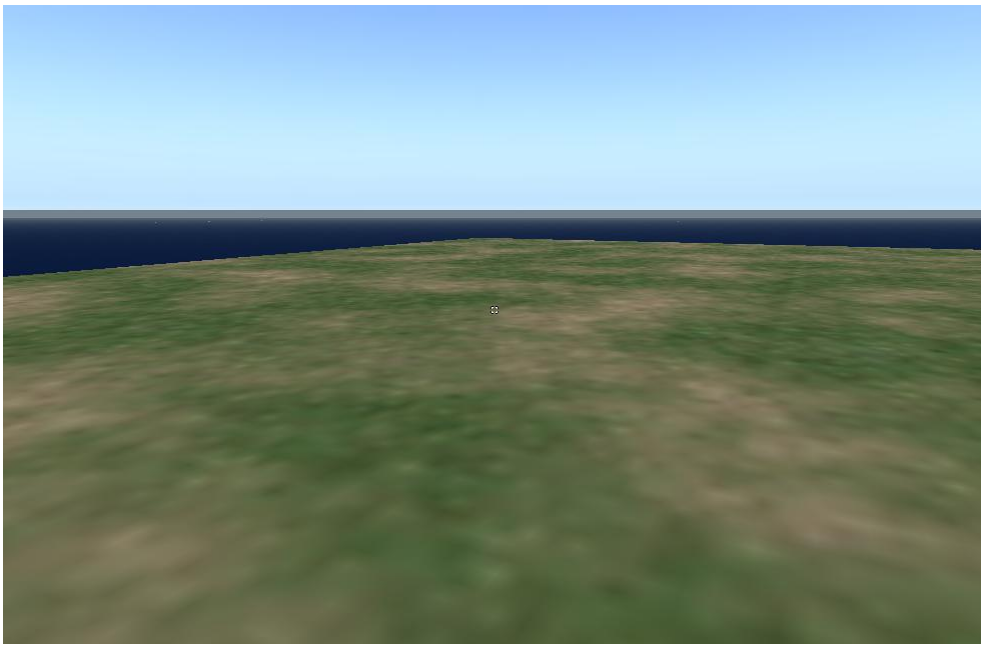

Figura 21 - Imagem da cena após o *fill terrain*

Os *prims* são a unidade de construção mais básica no mundo virtual do *OpenSim*. Podendo tomar várias formas básicas, tamanho, cor, nome, rotação, descrição, entre muitas outras características editáveis, são usados para criar tudo o que existe no mundo virtual.

De modo a ser possível dar rotação aos *prims* presentes na cena, o OpenSimulator utiliza *quaternions* [31]. Os *quaternions* são uma generalização dos números complexos, inventados por William Rowan Hamilton em meados do século XIX.

Visto que uma cena é demasiado curta para a maioria das zonas que se pretendem gerar, foi introduzido um factor multiplicativo pelo qual todo o conteúdo gerado é multiplicado de forma a ser reduzido. Por *default* o factor multiplicativo é de 0.1.

4.2 Mensagens HTTP

Todas as mensagens trocadas entre as diferentes secções do *2ndComing* usam o protocolo HTTP. Este protocolo tem a enorme vantagem de ser largamente usado e reconhecido pelo *OpenSim* e outros softwares com características idênticas.

Para isso, foi desenvolvida uma classe chamada *PostSubmitter* que é usada em todos os módulos do 2ndComing com o objectivo de criar as mensagens a enviar.

Esta classe permite o envio de variáveis em simultâneo nas mensagens HTTP do tipo POST, tendo também a função de codificar e empacotar estas variáveis segundo a estrutura:

```
private void EncodeAndAddItem(ref StringBuilder baseRequest, string key, string dataItem)
{
    if (baseRequest == null)
    {
            baseRequest = new StringBuilder();
    }
    if (baseRequest.Length != 0)
    {
            baseRequest.Append("&");
    }
    baseRequest.Append(key);
    baseRequest.Append("=");
    baseRequest.Append(System.Web.HttpUtility.UrlEncode(dataItem));
}
```

Os parâmetros de entrada desta função são os seguintes:

- *StringBuilder baseRequest*: dados já codificados;
- *String key*: a chave da variável que se pretende enviar;
- *String dataItem*: o valor a enviar para a chave.

O valor presente em *dataItem* é uma *string* que pode conter espaços entre palavras pelo que precisará de ser encriptada usando o método *UrlEncode()*. De seguida apresenta-se um exemplo de como são codificadas nas mensagens as variáveis que se pretendem enviar:

```
variable1=value1&variable2=value2...
```

Para o envio da mensagem final, é usada a seguinte função:

```
public string PostData(string url, string postData)
{
    HttpWebRequest request = null;
    if (m_type == PostTypeEnum.Post)
    {
            Uri uri = new Uri(url);
            request = (HttpWebRequest)WebRequest.Create(uri);
            request.Method = "POST";
            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = postData.Length;
            using (Stream writeStream = request.GetRequestStream())
            {
                    UTF8Encoding encoding = new UTF8Encoding();
                    byte[] bytes = encoding.GetBytes(postData);
                    writeStream.Write(bytes, 0, bytes.Length);
            }
    }
    else
    {
            Uri uri = new Uri(url + "?" + postData);
            request = (HttpWebRequest)WebRequest.Create(uri);
            request.Method = "GET";
    }
    string result = string.Empty;
    using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
    {
            using (Stream responseStream = response.GetResponseStream())
            {
                    using   (StreamReaderreadStream   =   new   StreamReader(responseStream,
                    Encoding.UTF8))
                    {
                            result = readStream.ReadToEnd();
                    }
            }
    }
    return result;
}
```

Esta função aceita como parâmetros o URL para onde irá enviar a mensagem e ainda qual os dados codificados a incluir nessa mensagem. Dependendo do método escolhido quando se faz uma instância desta classe, os dados poderão ser enviados usando o método GET ou POST. Neste projecto todas as mensagens usam o método POST pois assim o cumprimento da mensagem nunca será encarado como um problema já que o POST garante uma maior quantidade de dados enviados na mesma mensagem.

4.3 Objectos Complexos

Quando o *OpenSimulator* é iniciado com o *2ndComing OpenSim Module*, são carregadas para a memória do programa uma série de tabelas correspondendo cada uma a um tipo de objecto complexo.

Cada uma das tabelas contém informação acerca de todos os *prims* que constituem um objecto. Na Tabela 2 pode ser vista a descrição de cada coluna da tabela assim como um exemplo de um valor a dar a essa coluna.

Tabela 2: Colunas das *DataTables* dos Objectos Complexos

Coluna	Descrição	Exemplo
Prims	Nome dado a esse *prim*.	front-right wheel
Shape	Forma do prim.	box
vectorSizeXX	Comprimento do prim.	2
vectorSizeYY	Largura do prim.	2
vectorSizeZZ	Altura do prim.	2
offsetxx	Offset em relaçao ao centro.	1
offsetyy	Offset em relaçao ao centro.	0
offsetzz	Offset em relaçao ao centro.	-2
offsetangle	Offset em relaçao a 0º.	90

Quando o utilizador pede a criação de um novo objecto complexo, toda a informação para a construção deste está na sua tabela.

4.4 2ndComing

Os comandos disponíveis para os utilizadores gerarem conteúdo no seu mundo virtual são os seguintes:

- CreatePrim
- GenRoads
- BuildTerrain
- GenBus
- SimulateBus
- CreateObject
- MoveObject
- DeleteObject
- ShowObjects
- DeleteAll
- RestartScene
- Shutdown
- Help

4.4.1 CreatePrim

Como unidades básicas de construção no mundo virtual do *Second Life*, tornou-se elementar fazer um comando capaz de gerar *prims* com algumas das suas características mais importantes:

- Tamanho;

- Posição na cena;
- Rotação;
- Nome;
- Forma;

O tamanho pode ser um valor decimal e por isso é guardado num vector coluna do tipo *float* denominado por *vectorSize*. Por sua vez, a posição na cena será tratada da mesma forma que o tamanho, ou seja, será usado um vector coluna do tipo *float* para a armazenar com o nome *vectorCoords*. A rotação será a última das três características principais de um *prim* e apenas difere das duas anteriores porque necessita de tratar quatro valores. Porém, será também armazenada num vector coluna - *vectorOrientation*. Por último, o nome do prim é uma característica que apenas vem complementar a informação sobre o mesmo, não acrescentando nenhuma informação vital para a criação deste na cena. A forma de um *prim*, como o próprio nome indica, será influenciada pelo tamanho e decidirá a aparência do *prim*.

De todas as características apresentadas, a rotação e o nome são as únicas características opcionais. Porém, à falta de qualquer uma das outras informações, o *prim* será criado usando os dados *default* da característica em falta. Estes dados *default* são apresentados na Tabela 3.

Tabela 3: Valores *default* para a criação de um *prim*

Variável	Valor
Lengthxx	2
Lengthyy	2
Lengthzz	2
Coordxx	128
Coordyy	128
Coordzz	30
Orientationxx	0
Orientationyy	0
Orientationzz	0
Orientationww	0
Name	
Shape	box

Assim, se for criado um *prim* apenas com os dados *default* será criado um cubo com as dimensões 2x2x2, no centro da cena, isto é, na posição <128, 128, 30>, com rotação e nome nulos, como aliás se pode observar na Figura 22.

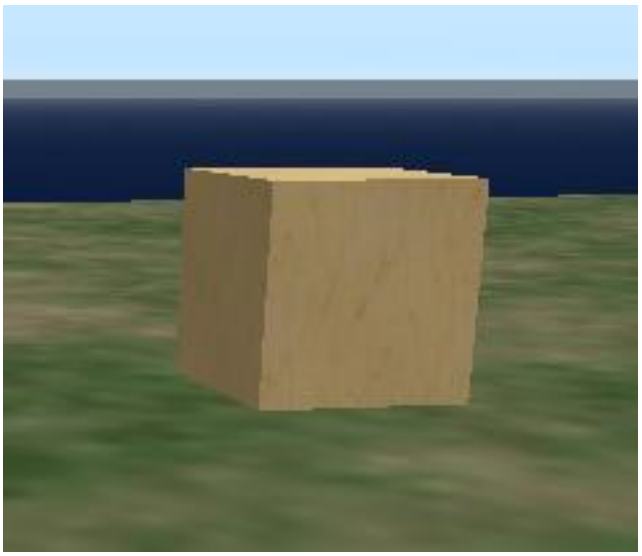

Figura 22 - *Default prim*

Quando um utilizador pede para criar um *prim*, as características definidas acima são-lhe solicitadas. Podendo este optar por definir algumas e deixar outras como *default*.

No *2ndComing* os dados são compilados e preparados para enviar para o *OpenSimulator Module*. Para isso é usada a função:

```
public void PostCreatePrim(float[] vectorSize, float[] vectorCoords, float[]
vectorOrientation, string sName)
{
    PostSubmitter post = new PostSubmitter();
    post.Url = "http://localhost:8080/createPrim";
    post.PostItems.Add("primname", sName);
    post.PostItems.Add("shape", shape);
    post.PostItems.Add("lengthxx", Convert.ToString(vectorSize[0]));
    post.PostItems.Add("lengthyy", Convert.ToString(vectorSize[1]));
    post.PostItems.Add("lengthzz", Convert.ToString(vectorSize[2]));
    post.PostItems.Add("coordxx", Convert.ToString(vectorCoords[0]));
    post.PostItems.Add("coordyy", Convert.ToString(vectorCoords[1]));
    post.PostItems.Add("coordzz", Convert.ToString(vectorCoords[2]));
    post.PostItems.Add("orientationxx", Convert.ToString(vectorOrientation[0]));
    post.PostItems.Add("orientationyy", Convert.ToString(vectorOrientation[1]));
    post.PostItems.Add("orientationzz", Convert.ToString(vectorOrientation[2]));
    post.PostItems.Add("orientationww", Convert.ToString(vectorOrientation[3]));
    post.Type = PostSubmitter.PostTypeEnum.Post;

    post.Post();
}
```

Esta função vai preparar a mensagem HTTP a ser enviada com o objectivo de criar o *prim*. Assim é necessário que seja definida qual a acção que se pretende fazer através do URL da mensagem e todas as variáveis a enviar. Estas variáveis encontram-se definidas na Tabela 4.

Tabela 4: createPrim - variáveis da Mensagem HTTP

Variável	Valor	Descrição

primname	sName	Nome do prim a ser criado.
shape	shape	Forma que o prim deve ter.
lengthxx	vectorSize[0]	Comprimento do prim.
lengthyy	vectorSize[1]	Largura do prim.
lengthzz	vectorSize[2]	Altura do prim.
coordxx	vectorCoords[1]	Coordenada no eixo XX do prim.
coordyy	vectorCoords[2]	Coordenada no eixo YY do prim.
coordzz	vectorCoords[3]	Coordenada no eixo ZZ do prim.
orientationxx	vectorOrientation[0]	Angulo de rotação segundo o eixo XX.
orientationyy	vectorOrientation[1]	Angulo de rotação segundo o eixo YY.
orientationzz	vectorOrientation[2]	Angulo de rotação segundo o eixo ZZ.
orientationww	vectorOrientation[3]	Angulo de rotação.

Apesar de as esferas e os cilindros, normalmente serem caracterizados por medidas diferentes:

- Raio e altura no caso do cilindro;
- Raio no caso da esfera.

No *2ndComing* estes continuarão a ser tratados como descrito acima pois o *OpenSim* permite uma maior liberdade na criação destas formas.

No *OpenSim Module*, as variáveis serão retiradas da mensagem e o *prim* é construído.

```
public void createNewPrim(string shape, float[] vectorSize, float[] vectorCoords,
float[] vectorOrientation, string sName)
{
        Vector3 pos = new Vector3((float)vectorCoords[0], (float)vectorCoords[1],
        (float)vectorCoords[2]);
        SceneObjectGroup sog = new SceneObjectGroup();
        //vectorSize = (length, width, height)
        if(shape == "box")
        {
                sog        =        new        SceneObjectGroup(UUID.Zero,        pos,
                PrimitiveBaseShape.CreateBox());
                sog.RootPart.Scale   =   new   Vector3(vectorSize[0],   vectorSize[1],
                vectorSize[2]);
        }
```

Depois de identificada e criada a forma que se pretende (cúbica, cilíndrica ou esférica) é necessário introduzir o objecto na cena.

```
        sog.Name = sName;
        sog.UpdateGroupRotation(new          Quaternion((float)(vectorOrientation[0]),
        (float)(vectorOrientation[1]),                  (float)(vectorOrientation[2]),
        (float)(vectorOrientation[3])));

        primList.Add(sog);
        actualScene.AddNewSceneObject(sog, true);
        Console.WriteLine("Added a new prim.");
}
```

Com o objectivo de controlar melhor os *prims* que vão sendo gerados, estes são sempre introduzidos na lista *primList*.

4.4.2 GenRoads

Este comando do *2ndComing* tem o objectivo de pedir ao *RoadGenerator* que envie a informação de nós e caminhos relativos à zona que se pretende gerar no mundo virtual. O nome desta zona será pedido ao utilizador imediatamente a seguir a ser invocado este comando, como se pode observar na Figura 23.

```
[2ndComing] Please type a command:
[2ndComing] >> genRoads
[2ndComing]     Area to Load: Porto
```

Figura 23 – Pedido de zona do *genRoads*

Todo o processo para a geração da informação das estradas de uma determinada zona pode ser consultado na Figura 24.

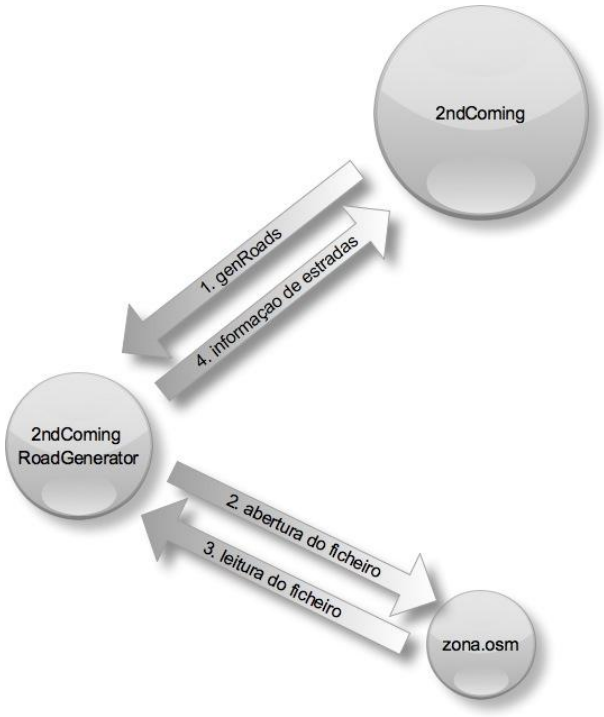

Figura 24 - Diagrama funcional - *genRoads*

4.4.2.1 Ficheiro com informação de zona

Para que a zona introduzida seja gerada com sucesso é necessário que exista um ficheiro com o mesmo nome da zona e todas as informações pretendidas. Estes ficheiros devem ser retirados do site do *OpenStreetMap*, no formato XML, e colocados no mesmo directório do *2ndComing BusGenerator*.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
  <boundsminlat="41.17464" minlon="-8.60342" maxlat="41.18404" maxlon="-8.59035"/>
  <node id="26057762" lat="41.180973" lon="-8.5937145" version="1" />
  <node id="26057764" lat="41.1809157" lon="-8.5941912" version="3"/>

    ...
<way id="4306658" visible="true" timestamp="2008-12-16T21:51:11Z" version="3">
  <nd ref="26122781"/>
  <nd ref="26057797"/>
  <nd ref="26057799"/>
  <nd ref="26057801"/>
  <nd ref="26057803"/>
  <nd ref="26057804"/>
  <nd ref="26057805"/>
  <nd ref="26057813"/>
<tag k="created_by" v="Potlatch 0.5d"/>
<tag k="highway" v="motorway_link"/>
<tag k="oneway" v="yes"/>
</way>
</osm>
```

O ficheiro deverá então ter a seguinte estrutura:

- *Bounds:* variáveis *minlat, minlon, maxlat* e *maxlon;*
- *NodeIDs:* Referência dos nós e respectivas coordenadas *lat* e *lon*;

- *WayIDs*: Nós associados a cada caminho (estrada).

Estas três secções do ficheiro são obrigatórias pelo que o gerador apenas conseguirá gerar o conteúdo se todas estiverem presentes no documento.

4.4.2.2 Leitura do ficheiro

A leitura do ficheiro processa-se em três fases distintas. Primeiro é necessário ler as coordenadas dos limites da zona que se pretende gerar. Após essa leitura, é necessário tratar essas coordenadas de forma a obter-se as coordenadas centrais da região:

```
centerCoordlat = (maxlat - minlat) / 2 + minlat;
centerCoordlon = (maxlon - minlon) / 2 + minlon;
```

As coordenadas centrais da zona são essenciais para se poder fazer coincidir estas com as coordenadas centrais da cena no mundo virtual. Posteriormente, é necessário enviar as coordenadas centrais, quer de latitude quer de longitude, para o *2ndComing* da seguinte forma:

```
PostSubmitter post = new PostSubmitter();
post.Type = PostSubmitter.PostTypeEnum.Post;
post.Url = "http://localhost:8081/centerCoords";
post.PostItems.Add("centerCoordLat", Convert.ToString(centerCoordlat));
post.PostItems.Add("centerCoordLon", Convert.ToString(centerCoordlon));

post.Post();
```

As variáveis enviadas na mensagem anterior têm o significado descrito na Tabela 5.

Tabela 5: Descrição das coordenadas centrais

Variável	Valor	Descrição
centerCoordLat	centerCoordLat	Coordenada da Latitude central da região.
centerCoordLon	centerCoordLon	Coordenada da Longitude central da região.

No *2ndComing*, estas coordenadas necessitam ser armazenadas para uso futuro:

```
if (order == "centerCoords")
{
    Console.WriteLine("\nReceived the center coords of the desired map.");
    centerCoordLat = Convert.ToDouble(varstable["centerCoordLat"]);
    centerCoordLon = Convert.ToDouble(varstable["centerCoordLon"]);
}
```

Depois de enviadas as coordenadas centrais da região, é necessário ler todos os nós presentes nesta e respectivas coordenadas. Enviando esta informação à medida que ela é lida, uma mensagem por cada nó:

```
PostSubmitter post = new PostSubmitter();
post.Type = PostSubmitter.PostTypeEnum.Post;
post.Url = "http://localhost:8081/RoadNode";
post.PostItems.Add("nodeRef", nodeinfoaux[2]);
post.PostItems.Add("nodeLat", nodeinfoaux[4]);
post.PostItems.Add("nodeLon", nodeinfoaux[6]);

post.Post();
```

O significado das variáveis enviadas na mensagem anterior pode ser consultado na Tabela 6.

Tabela 6: Descrição das variáveis com informação dos nós

Variável	Valor	Descrição
nodeRef	nodeinfoaux[2]	Referência do nó.
nodeLat	nodeinfoaux[4]	Latitude do nó.
nodeLon	nodeinfoaux[6]	Longitude do nó.

Em que o valor das variáveis está armazenado no vector de *strings nodeinfoaux[]*.

Do lado do *2ndComing* é necessário armazenar as coordenadas de cada nó. Assim, foram criadas duas *hashtables*, uma para todas as latitudes de todos os nós da região e outra para as correspondentes longitudes: *nodesLat* e *nodesLon* respectivamente:

```
if (order == "RoadNode")
{
    nodesLat.Add(varstable["nodeRef"], OSCoordinates(centerCoordLat,
    Convert.ToDouble(varstable["nodeLat"])));
    nodesLon.Add(varstable["nodeRef"], OSCoordinates(centerCoordLon,
    Convert.ToDouble(varstable["nodeLon"])));
}
```

Porém, não basta adicionar as coordenadas reais dos nós. É necessário primeiro converter estas coordenadas reais para as coordenadas do mundo virtual. Para esse efeito foi escrita a seguinte função:

```
private double OSCoordinates(double centerCoord, double coord)
{
    //0.00001 = 1.11m
    // res = x
    double res = coord - centerCoord;
    //factor multiplicativo = 10
    res = Math.Abs(res) * 1.11 / 0.00001 / 10;

    if(centerCoord>coord)
            return (128 - res);
    else return (128 + res);
}
```

Esta função, determina a distância da coordenada dada ao centro da região e depois transforma-a em metros. Após determinada a distância, é usado o factor de conversão que afirma que um desvio de 0,00001 em coordenadas decimais equivale a 1,11metros. Depois, conforme a coordenada actual seja mais alta ou mais baixa que a coordenada central, é adicionado ou removido o valor resultante a 128. Recorde-se que 128 é o valor para o centro da região, tanto em longitude como em latitude.

O objectivo de guardar toda esta informação no *2ndComing* é para que esta possa ser utilizada por outros geradores de conteúdo, evitando assim o envio de informação previamente enviada.

Por último, faz-se o levantamento de todos os nós constituintes de cada caminho e envia-se toda a informação para o *2ndComing*:

```
PostSubmitter post = new PostSubmitter();
post.Type = PostSubmitter.PostTypeEnum.Post;
post.Url = "http://localhost:8081/RoadWay";
post.PostItems.Add("wayRef", wayref);
post.PostItems.Add("nodeRef", wayinfoaux[1]);
```

O significado das variáveis enviadas nesta mensagem pode ser consultado na Tabela 7.

Tabela 7: Descrição das variáveis com informação sobre caminhos

Variável	Valor	Descrição
wayRef	wayref	Referência do caminho.
nodeRef	wayinfoaux[1]	Referência do novo nó para o caminho.

Sendo que o valor da variável está armazenado no vector de *strings wayinfoaux[]*.

No *2ndComing* é necessário guardar toda a informação recebida para uso posterior. Para o armazenamento desta, é usada uma matriz estruturada da seguinte forma:

[Way 1]	[Ref. do 1º Nó do Way 1]	...	[Ref. do último nó do Way 1]	0	...
[Way 2]	[Ref. do 1º Nó do Way 2]	...	[Ref. do último nó do Way 2]	0	...
...	...	...	...	…	...
[Way N]	[Ref. do 1º Nó do Way N]	...	[Ref. do último nó do Way N]	0	...

A informação é armazenada nesta matriz usando o seguinte código:

```
if (order == "RoadWay")
{
    intwayIndex;

    //search wayNodes for the way ref
    wayIndex = SearchVector(Convert.ToString(varstable["wayRef"]), 400);
    //if the ref is already there -> j++ and add the new node ref to the way
    if (wayIndex>= 0)
    {
        j = 0;
        //search the first position on the matrix's row that is null
        while(wayNodes[wayIndex, j] != 0)
                j++;
        //and adds the new node to that matrix's position
        wayNodes[wayIndex, j] = Convert.ToDouble(varstable["nodeRef"]);
    }
    //if the way ref isnt in the matrix -> add the new way ref and the first
    //way node
    if (wayIndex< 0)
    {
        j = 0;
        wayNodes[i, j] = Convert.ToDouble(varstable["wayRef"]);
        j++;
        wayNodes[i, j] = Convert.ToDouble(varstable["nodeRef"]);
        i++;
    }
}
```

Se o 2ndComing receber a informação de um caminho, é necessário, numa primeira fase, averiguar se esse caminho já se encontra presente na matriz *wayNodes*. Para tal, é usada a função *SearchVector* que será explicada detalhadamente mais à frente. Se a referência do caminho – *wayRef* -já estiver presente na primeira coluna da matriz, significa que pelo menos um nó já está introduzido e é preciso procurar a primeira posição da matriz livre. Depois de encontrada essa posição, o *2ndComing* coloca nessa posição a referência do novo nó.
Se, pelo contrário, a referência do caminho não estiver presente na primeira coluna da matriz, é necessário primeiro adicioná-la e somente depois colocar a referência do nó.

```
privateint SearchVector(string reference, int length)
{
    int res;
    for (int aux2 = 0; aux2 < length; aux2++)
    {
        res = String.Compare(reference, Convert.ToString(wayNodes[aux2, 0]));
        if (res == 0)
                return aux2;
    }
    return -1;
}
```

Esta função é usada para pesquisar a matriz *wayNodes* por referências de caminhos que vão chegando ao *2ndComing*. É necessário passar como parâmetros tanto a referência que se pretende pesquisar na matriz, sob a forma de uma *string*, como o número de linhas, isto é, a altura, da matriz.

Se a referência passada como parâmetro for encontrada na matriz, a função retorna o índice da linha onde essa referência foi encontrada. Por outro lado, se a referência não constar da matriz, a função retorna o valor -1.

Este processo terá que ser repetido até todos os caminhos e respectivos nós estarem presentes na matriz *wayNodes*.

Depois destas três fases, toda a informação necessária para a geração das estradas está presente no *2ndComing*.

4.4.3 BuildTerrain

Este comando tem como objectivo criar todas as estradas anteriormente colocadas em memória no *2ndComing* pelo comando *genRoads*. Para a sua implementação foi criada uma nova classe, denominada de *terrainBuilder*. Esta classe, vai aceder a toda a informação contida na matriz *wayNodes* para realizar a sua função.

```
public void BuildTerrain()
{
        vectorSize[1] = 2;
        vectorSize[2] = Convert.ToSingle(0.1);
        vectorCoords[2] = Convert.ToSingle(21.01);
        vectorOrientation[0] = 0;
        vectorOrientation[1] = 0;
```

Estes valores são atribuídos sempre que esta função é chamada pois são constantes. A

Tabela **8** contém o valor dado e a respectiva descrição às variáveis constantes.

Tabela 8: Variáveis constantes da mensagem *buildTerrain*

Variável	Valor	Descrição
vectorSize[1]	2	Largura de uma estrada.
vectorSize[2]	0.1	Altura de uma estrada.
vectorCoords[2]	21.1	Altura a que o prim vai ser criado na cena.
vectorOrientation[0]	0	Ângulo de rotação do prim em relação ao eixo X.
vectorOrientation[1]	0	Ângulo de rotação do prim em relação ao eixo Y.

Antes de se começar a explorar o código da função, convém demonstrar qual o método usado para determinar o comprimento e a posição da rua que se pretende criar. Para isso, veja-se a Figura 25.

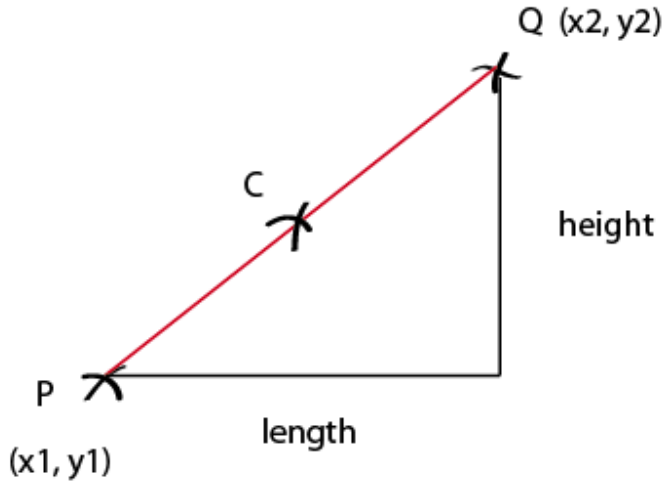

Figura 25 - Método de cálculo - *buildTerrain*

Considere-se que são dadas as coordenadas dos pontos P e Q e pretende-se determinar as coordenadas de C e o comprimento do segmento de recta PQ.

Pela figura, pode-se determinar *Length* e *Height* da seguinte forma:

$$Length = x_2 - x_1 \quad (4.1)$$

$$Height = y_2 - y_1 \quad (4.2)$$

Conhecendo estas duas variáveis, pelo teorema de Pitágoras, determina-se o comprimento do segmento de recta PQ:

$$PQ = \sqrt{Length^2 + Height^2} \quad (4.3)$$

Por fim, as coordenadas de C são dadas pelos pontos:

$$C_x = x_1 + \frac{Length}{2} \tag{4.4}$$

$$Cy = y_1 + \frac{Height}{2} \tag{4.5}$$

Este foi o método usado para determinar todas estas variáveis entre cada dois nós constituintes de um caminho.

Na Tabela 9 estão presentes as variáveis, e respectivos valores, já determinadas por este método.

Tabela 9: Valores atribuídos às diferentes variáveis

Variável	Valor
vectorSize[0]	Comprimento PQ
vectorSize[1]	Constante
vectorSize[2]	Constante
vectorCoords[0]	Cx
vectorCoords[1]	Cy
vectorCoords[2]	Constante
vectorOrientation[0]	Constante
vectorOrientation[1]	Constante
vectorOrientation[2]	Angle
vectorOrientation[3]	Angle

Para se aplicar este método aos nós disponíveis no *2ndComing*, basta ter em mente que cada nó é constituído por duas coordenadas, a latitude e a longitude.

De seguida, é apresentado o diagrama que demonstra o processo de construção das estradas (Figura 26).

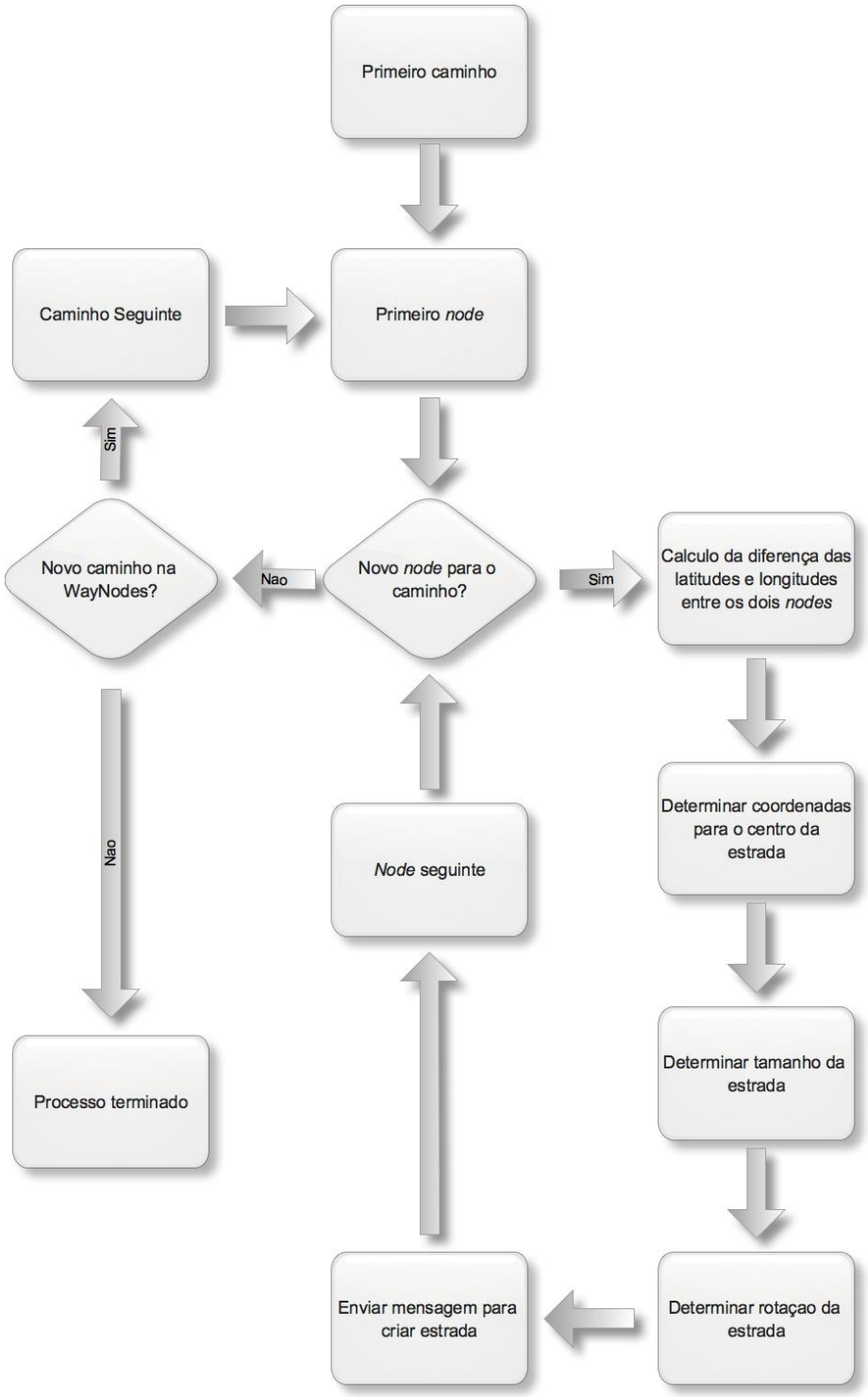

Figura 26 - Diagrama explicativo para o comando *buildTerrain*

```
while (Decider.wayNodes[i, 0] != 0)
{
    h = 1;
    double res;
    while (wayNodes[i, h+1] != 0)
    {
        res = nodesLat[wayNodes[i, h]] - nodesLat[Decider.wayNodes[i, h+1]];
        double length = Math.Abs(res);
        res = res / 2;

        double min = Math.Min(nodesLat[wayNodes[i, h]], nodesLat[wayNodes[i, h+1]]);

        vectorCoords[0] = Convert.ToSingle(Math.Abs(res) + min);

        res = nodesLon[wayNodes[i, h]] - nodesLon[wayNodes[i, h+1]];
        double height = Math.Abs(res);
        res = Math.Abs(res) / 2;

        //h2 = c2+c2
        vectorSize[0] = Convert.ToSingle(Math.Sqrt(length * length + height * height) + 1);
        double angle = Math.Atan2(height, length)*180/Math.PI;

        if(nodesLon[wayNodes[i, h]] >nodesLon[wayNodes[i, h+1]])
                angle = -angle;
        if(nodesLat[wayNodes[i, h]] >nodesLat[wayNodes[i, h+1]])
                angle = -angle;

        min = Math.Min(nodesLon[wayNodes[i, h+1]], nodesLon[wayNodes[i, h]]);
        vectorCoords[1] = Convert.ToSingle(Math.Abs(res)+min);
        vectorOrientation[2] = Convert.ToSingle(Math.Sin((angle*Math.PI/180)/2));
        vectorOrientation[3] = Convert.ToSingle(Math.Cos((angle*Math.PI/180)/2));

        if(vectorCoords[0] >= 0 && vectorCoords[0] <= 256 && vectorCoords[1] >= 0 &&
        vectorCoords[1] <= 256)
        {
                PostFunctions post = new PostFunctions();
                post.PostCreatePrim("box",  vectorSize,  vectorCoords,  vectorOrientation,
                "Way");
        }
        h++;
    }
    i++;
}
```

Enquanto existirem novos caminhos na matriz *wayNodes* e, para esse caminho, existirem novos nós, será calculada a diferença entre a latitude do primeiro nó com a latitude do segundo nó. O mesmo será feito posteriormente com as longitudes de ambos os nós.

O valor dado à variável *Length* será então o valor absoluto da diferença de latitudes enquanto o valor dado a *Height* será o valor absoluto da diferença de longitudes. Para determinar as coordenadas centrais, é preciso determinar primeiro o mínimo quer das latitudes quer das longitudes e, a esses mínimos, somar metade do valor de *Length* e de *Height*, respectivamente.

Depois de determinadas todas as coordenadas, é necessário calcular o tamanho do *prim* que se pretende criar usando o teorema de Pitágoras. Por fim, calcula-se o ângulo de rotação do *prim*, apenas relativamente ao eixo Z pois o *2ndComing* não suporta relevo.

Por fim, se todas as coordenadas se encontrarem dentro da área da cena presente no *OpenSimulator,* será enviada a ordem de criação de novo *prim*. Na Figura 27 podem ser observadas as estradas geradas de uma zona do Porto.

Figura 27 - Estradas da cidade do Porto

4.4.4 GenBus

De forma a serem criados autocarros, é necessário interagir com o *2ndComing BusGenerator*. Com esse fim, é preparada a seguinte mensagem:

```
public void PostGenerateBus(string area)
{
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8091/doMagic";
        post.PostItems.Add("area", area);

        result = post.Post();
}
```

A área que se pretende criar deve ser a mesma definida, anteriormente, para a criação das estradas e será necessário ter um ficheiro com esse mesmo nome no directório do *BusGenerator*.

4.4.4.1 2ndComing BusGenerator

Quando o *2ndComing BusGenerator* recebe uma mensagem para gerar autocarros, a primeira coisa que irá processar será o ficheiro respectivo de forma a adquirir informações sobre todos os autocarros da zona.

4.4.4.1.1 Ficheiro com informação dos autocarros

Este ficheiro além de estar no mesmo directório *do 2ndComing BusGenerator*, necessita ter exactamente o mesmo nome da zona a que está associado e que o utilizador introduz na interface do *2ndComing*. A estrutura deste ficheiro, altamente baseada em XML, foi construída de forma a adaptar-se a este projecto e com o objectivo de conter apenas a informação mínima essencial para o mesmo. Assim sendo, os ficheiros com informação de autocarros devem obedecer à seguinte estrutura:

```
<?xml version="1.0" encoding="UTF-8"?>
<bg version="1.0" generator="Auto Content Bus Generator">
<busbusid="1000"/>
  <nd ref="137970173"/>
  <nd ref="25632467"/>
  <nd ref="138249077"/>
  <nd ref="138194473"/>
  <nd ref="126638962"/>
  <nd ref="299614241"/>
  <nd ref="299614078"/>
  <nd ref="138195760"/>
  <nd ref="138196470"/>
  <nd ref="299613573"/>
  <nd ref="299613373"/>
  <nd ref="126638963"/>
  <nd ref="122480050"/>
</bus>
```

Para guardar toda a informação lida do ficheiro, é criado um dicionário constituído por uma *string*, que funciona como identificador do autocarro, associando cada uma destas a uma lista de *strings*, isto é, a lista de todos os nós constituintes do percurso do autocarro.

Depois de terminar a leitura do ficheiro e de toda a informação estar no dicionário, é necessário enviá-la para o *2ndComing*. Para isso, existem três tipos de mensagens com objectivos distintos:

```
private void PostNewBus(string busID)
{
    PostSubmitter post = new PostSubmitter();
    post.Type = PostSubmitter.PostTypeEnum.Post;
    post.Url = "http://localhost:8081/createBus";
    post.PostItems.Add("busID", busID);
    post.PostItems.Add("path", "starting");

    string result = post.Post();
    Console.WriteLine("Message Received: " + result);
}
```

Esta mensagem tem como único objectivo anunciar a criação de um novo autocarro indicando também o identificador do mesmo e que o caminho vai ser iniciado nas mensagens seguintes.

```
private void PostNewNode(string busID, string nodeNumber, string nodeRef)
{
    PostSubmitter post = new PostSubmitter();
    post.Type = PostSubmitter.PostTypeEnum.Post;
    post.Url = "http://localhost:8081/createBus";
    post.PostItems.Add("busID", busID);
    post.PostItems.Add("nodeNumber", nodeNumber);
    post.PostItems.Add("nodeRef", nodeRef);
    post.PostItems.Add("path", "incomplete");

    string result = post.Post();
    Console.WriteLine("Message Received: " + result);
}
```

Por cada novo nó no percurso do autocarro, o BusGenerator necessita de enviar uma mensagem de *NewNode*. Estas mensagens devem conter:

- Identificador do autocarro;
- Número de sequência do nó;
- Referência do nó;
- Informação de caminho incompleto.

```
    private void PostEndPath(string busID)
    {
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8081/createBus";
        post.PostItems.Add("busID", busID);
        post.PostItems.Add("path", "completed");

        string result = post.Post();
        Console.WriteLine("Message Received: " + result);
    }
```

Por fim, esta será a última mensagem a ser enviada por cada autocarro gerado. Deve conter simplesmente o identificador do autocarro e a informação de que o percurso terminou.

4.4.4.1.2 2ndComing

Ao contrário do que se pretendia com o *RoadGenerator*, desta vez não há a necessidade do *2ndComing* guardar toda a informação que recebe do *BusGenerator*. Assim, todas as mensagens recebidas são tratadas e enviadas imediatamente para o *OpenSim Module* que aguardará pela ordem de início de simulação para a iniciar.

À semelhança do que acontece no *BusGenerator*, também o *2ndComing* precisa de três mensagens diferentes para enviar a informação dos autocarros:

```
    public void PostPathStarting(string busID)
    {
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/createBus";
        post.PostItems.Add("busID", busID);
        post.PostItems.Add("path", "starting");

        result = post.Post();
        Console.WriteLine("Message Received: " + result);
    }
```

Esta mensagem, à semelhança da mensagem respectiva do BusGenerator tem como objectivo indicar apenas que o caminho do autocarro irá iniciar-se.

```
public void PostCreateBus(string busID, string nodeNumber, float[] vectorCoords)
{
    PostSubmitter post = new PostSubmitter();
    post.Type = PostSubmitter.PostTypeEnum.Post;
    post.Url = "http://localhost:8080/createBus";
    post.PostItems.Add("busID", busID);
    post.PostItems.Add("nodeNumber", nodeNumber);
    post.PostItems.Add("nodeX", Convert.ToString(vectorCoords[0]));
    post.PostItems.Add("nodeY", Convert.ToString(vectorCoords[1]));
    post.PostItems.Add("nodeZ", Convert.ToString(vectorCoords[2]));
    post.PostItems.Add("path", "incomplete");

    result = post.Post();
    Console.WriteLine("Message Received: " + result);
}
```

As mensagens *createBus* contêm toda a informação que se pretende enviar sobre o percurso do autocarro, isto é, os diferentes nós por onde este terá que passar.

```
    public void PostPathCompleted(string busID)
    {
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/createBus";
        post.PostItems.Add("busID", busID);
        post.PostItems.Add("path", "completed");

        result = post.Post();
        Console.WriteLine("Message Received: " + result);
    }
```

A função desta mensagem é indicar que o percurso do autocarro chegou ao fim e será esta a última mensagem enviada por cada autocarro gerado.

Quando recebe uma mensagem do *BusGenerator*, o *2ndComing* tem de a processar e decidir qual das mensagens anteriores enviar para o módulo do OpenSimulator.

O processamento das mensagens de criação de autocarros é feito segundo o seguinte código:

```
if (order == "createBus")
{
        if(Convert.ToString(varstable["path"]) == "starting")
        {
                Console.WriteLine("\nReceivedtherequest to create a new Bus!");
                busList.Add(varstable["busID"], "created");
                PostFunctionspost = newPostFunctions();
                post.PostPathStarting(varstable["busID"]);
        }
        if(busList.ContainsKey(varstable["busID"]))
        {
                if(varstable["path"] == "completed")
                {
                        PostFunctionspost = newPostFunctions();
                        post.PostPathCompleted(varstable["busID"]);
                }
                if(Convert.ToString(varstable["path"]) == "incomplete")
                {
                        vectorCoords[0] = nodesLat[varstable["nodeRef"]];
                        vectorCoords[1] = nodesLon[varstable["nodeRef"]];
                        vectorCoords[2] = 22;

                        PostFunctionspost = newPostFunctions();
                        post.PostCreateBus(varstable["busID"],
                        varstable["nodeNumber"]), vectorCoords);
                }
        }
}
```

4.4.4.1.3 2ndComing OpenSim Module

No *2ndComing OpenSim Module*, o percurso do autocarro é recolhido e transformado numa lista de nós que, à semelhança do que se tinha no *BusGenerator*, irá ser associada ao identificador do autocarro num novo dicionário: *AllBus*.

Será a este dicionário que todas as *threads* terão que aceder para saber qual o percurso a seguir pelo seu autocarro quando o utilizador der ordem para o início da simulação.

4.4.5 SimulateBus

Quando o utilizador introduz este comando, o *2ndComing* envia uma mensagem para o *OpenSim Module* com a ordem de início da simulação e todo o processamento desta funcionalidade é executado aí. No diagrama da Figura 28 pode ser visualizado todo o processo necessário para iniciar uma simulação.

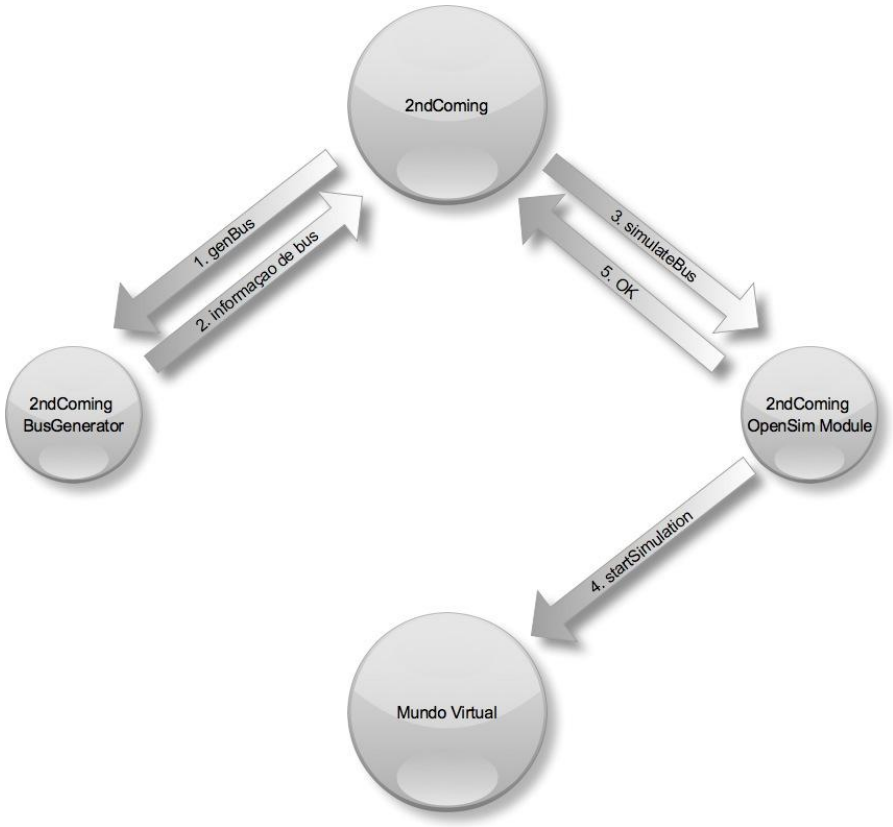

Figura 28 - Diagrama sequencial para o comando *simulateBus*

Por cada autocarro gerado será lançada uma *thread* que trata da criação e manutenção do respectivo autocarro na cena.

O diagrama da Figura 29 demonstra como cada *thread* realiza o percurso do seu autocarro.

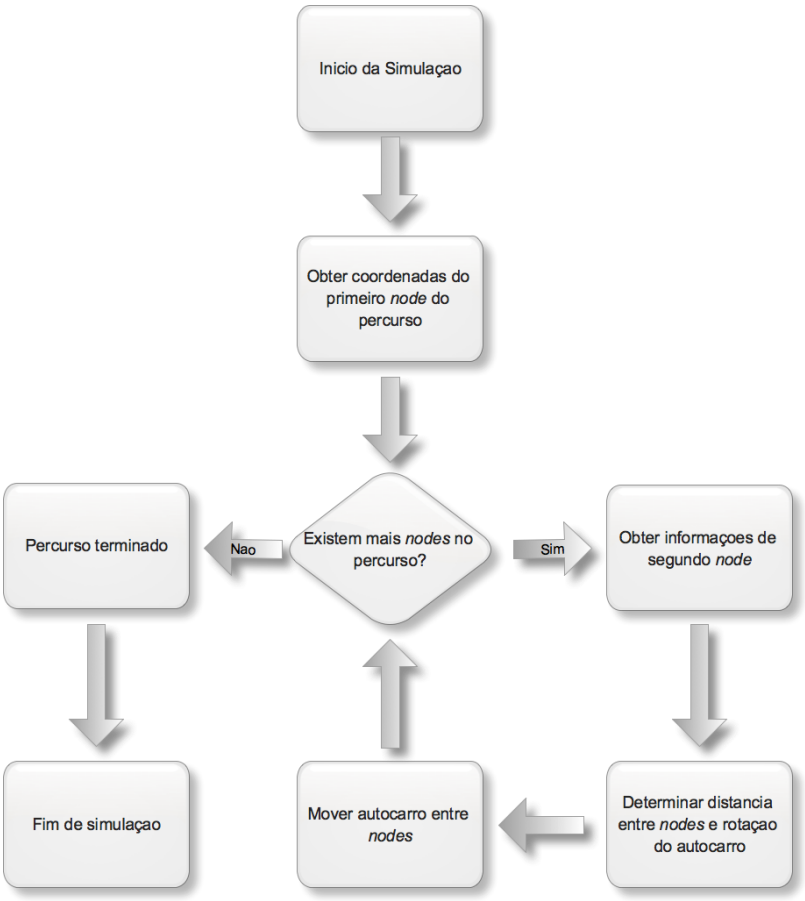

Figura 29 – Diagrama explicativo para o comando *simulateBus*

Quando o utilizador dá ordem para se iniciar a simulação é enviada uma mensagem do *2ndComing* para o *2ndComing OpenSim Module*:

```
public void PostSimulateBus()
{
    PostSubmitter post = new PostSubmitter();
    post.Type = PostSubmitter.PostTypeEnum.Post;
    post.Url = "http://localhost:8080/simulateBus";
    post.PostItems.Add("do", "magic");

    result = post.Post();
}
```

Esta mensagem irá iniciar uma *thread* por cada autocarro já criado no *2ndComing OpenSim Module*.

Mal se inicia a simulação, cada autocarro irá obter informações sobre o seu percurso, começando pelos dois primeiros nós. Depois de se ter a informação destes nós, será determinada a distância entre eles e o ângulo de rotação que o autocarro deve ter para seguir o trajecto correcto.

Só depois de determinadas estes dados é que é criado o autocarro, no primeiro nó, já com a devida rotação, ou seja, na direcção do segundo nó do percurso.

O movimento é simulado da seguinte forma:

```
while(distanceWalked < distance)
{
    distanceWalked = distanceWalked + d;
    System.Threading.Thread.Sleep(100);
    foreach(SceneObjectGroup prim in busPrims)
    {
        prim.AbsolutePosition         =         prim.AbsolutePosition         +
        offset2/BusRefreshFrequency;
        prim.ScheduleGroupForTerseUpdate();
    }
}
```

em que o *BusRefreshFrequency* é a frequência de actualização do autocarro, definida por *default* como cem milissegundos. Ou seja, a posição do autocarro é actualizada a cada cem milissegundos. Aumentando esta variável significaria que o movimento do autocarro tornar-se-ia ainda mais fluído pois a distância que este percorreria nos cem milissegundos seria menor. Por outro lado, diminuindo esta variável o movimento do autocarro seria menos fluído.

O procedimento anterior é aplicado a todos os nós do percurso até este estar concluído.

4.4.6 CreateObject

Para a criação de objectos complexos é necessário que o utilizador forneça um conjunto de dados que serão introduzidos pela interface do *2ndComing*, nomeadamente:

- Referência;
- Nome;
- Coordenadas;
- Orientação.

A recolha destes dados pela interface do *2ndComing* pode ser observada na Figura 30.

```
[2ndComing] >> createObject
[2ndComing] Received the request to create a complex object!
[2ndComing] What object would you like to create?
[2ndComing]     ::bus
[2ndComing]     ::chair
[2ndComing]     ::table
[2ndComing]     ::plane
[2ndComing]     ::house
[2ndComing]     ::building
[2ndComing]     ::car
[2ndComing] >> bus
[2ndComing] What is the desired object's location?
[2ndComing] In the X axis: 130
[2ndComing] In the Y axis: 130
[2ndComing] In the Z axis: 22
[2ndComing] Rotation angle (in degrees): 0
Message Received: HTTP/1.1 200 OK
[2ndComing] Order to create a new object sent!
[2ndComing] Please type a command:
[2ndComing] >>
```

Figura 30 - *2ndComing* Interface - *createObject*

Após reunidas todas as informações necessárias, estas serão enviadas para o *2ndComing OpenSim Module* para o objecto pretendido seja gerado no mundo virtual:

```
public void PostCreateObject(string identifier, string reference, float[] vectorCoords,
float[] vectorOrientation)
{
    PostSubmitter post = new PostSubmitter();
    post.Type = PostSubmitter.PostTypeEnum.Post;
    post.Url = "http://localhost:8080/createObject";

    post.PostItems.Add("ID", identifier);
    post.PostItems.Add("reference", reference);
    post.PostItems.Add("coordxx", Convert.ToString(vectorCoords[0]));
    post.PostItems.Add("coordyy", Convert.ToString(vectorCoords[1]));
    post.PostItems.Add("coordzz", Convert.ToString(vectorCoords[2]));
    post.PostItems.Add("orientationxx", Convert.ToString(vectorOrientation[0]));
    post.PostItems.Add("orientationyy", Convert.ToString(vectorOrientation[1]));
    post.PostItems.Add("orientationzz", Convert.ToString(vectorOrientation[2]));
    post.PostItems.Add("orientationww", Convert.ToString(vectorOrientation[3]));

    result = post.Post();
}
```

Ao receber a mensagem de *createObject* no *2ndComing OpenSim Module*, este vai-lhe retirar as informações das variáveis e tratá-las para a poder executar. Depois de ter todas as variáveis tratadas é necessário criar o objecto na cena. Para isso é usada a seguinte função *CreateObject* que será agora analisada:

```
public void CreateObject(string ID, string reference, float[] vectorCoords,
float[] vectorOrientation)
{
    List<SceneObjectGroup> objectPrims = new List<SceneObjectGroup>();
    SceneObjectGroup prim = new SceneObjectGroup();
    DataTable co = new DataTable();

    if(reference == "bus")
        co = ComplexObjects.bus;
    else if(reference == "chair")
        co = ComplexObjects.chair;
    else if(reference == "plane")
        co = ComplexObjects.plane;
    else if(reference == "table")
        co = ComplexObjects.table;
    else if(reference == "car")
        co = ComplexObjects.car;
```

Numa primeira fase, a função tem que identificar a referência do objecto que se pretende criar e carregar para uma tabela local todos os *prims* que constituem esse objecto.

```
foreach(DataRow row in co.Rows)
{
        Vector3 pos = new Vector3(vectorCoords[0] + row["offsetXX"],
        vectorCoords[1] + row["offsetYY"], vectorCoords[2] + row["offsetZZ"]);

        if(row["Shape"] == "box")
        {
                prim = new SceneObjectGroup(UUID.Zero, pos,
                PrimitiveBaseShape.CreateBox());
        }
        else if(row["Shape"] == "cylinder")
        {
                prim = new SceneObjectGroup(UUID.Zero, pos,
                PrimitiveBaseShape.CreateCylinder());
        }
        elseifConvert.ToString(row["Shape"] == "sphere")
        {
                prim = new SceneObjectGroup(UUID.Zero, pos,
                PrimitiveBaseShape.CreateSphere());
        }
        else
        {
                Console.WriteLine("ERROR: NO SHAPE DEFINED!");
                break;
        }
```

Após identificar a referência do objecto é preciso percorrer a tabela correspondente criando os *prims* que a constituem, um a um de acordo com a sua forma, coordenadas e rotação na cena. Para determinar as coordenadas individuais de cada *prim* na cena, existem na tabela as variáveis:

- *Offsetxx*;
- *Offsetyy*;
- *Offsetzz*.

Somando os valores contidos nestas variáveis ao valor das coordenadas presentes no *vectorCoords*, cuja informação foi introduzida pelo utilizador, ter-se-á a posição de cada objecto na cena.

```
        prim.RootPart.Scale = new Vector3(row["vectorSize XX"], row["vectorSize
        YY"], row["vectorSize ZZ"]);

        prim.UpdateGroupRotation(new Quaternion(0, 0,
        (float)row["offsetangle"], (float)row["offsetangle"]));

        objectPrims.Add(prim);
}
AllObjects.Add(ID, objectPrims);
ObjectPositions.Add(ID, vectorCoords);
```

Após definida a forma e a posição na cena, é necessário obter a informação do tamanho e da rotação de cada *prim*.

Depois de criados, cada *prim* é introduzido na lista *objectPrims* que, posteriormente, é associada ao identificador do objecto complexo no dicionário *AllObjects*. A posição de cada objecto complexo, isto é, do centro desse objecto, será armazenada no dicionário *ObjectPositions* associada ao identificador do mesmo.

Por último, falta apenas adicionar todos os *prims* constituintes do objecto à cena:

```
        foreach(SceneObjectGroupsog in objectPrims)
        {
                Quaternion defaultRotation = sog.GroupRotation;
                Quaternion userRotation = new
                Quaternion((float)vectorOrientation[1],
                (float)vectorOrientation[1], (float)vectorOrientation[2],
                (float)vectorOrientation[3]);

                sog.UpdateGroupRotation(defaultRotation + userRotation);
                actualScene.AddNewSceneObject(sog, true);
        }
}
```

A Figura 31 mostra um objecto complexo do tipo *bus* criado no centro da cena do mundo virtual com rotação nula.

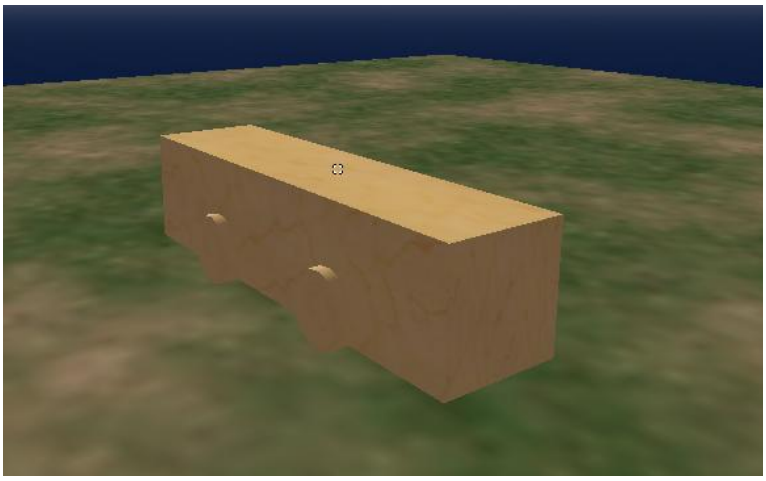

Figura 31 - Objecto Complexo do tipo *bus*

4.4.7 MoveObject

Esta funcionalidade tem o objectivo de movimentar objectos no mundo virtual. Para isso necessita apenas de conhecer o identificador do objecto, a nova posição e orientação do objecto que se pretende mover.

A Figura 32 mostra a forma como o *2ndComing* faz a recolha das informações necessárias ao *moveObject*.

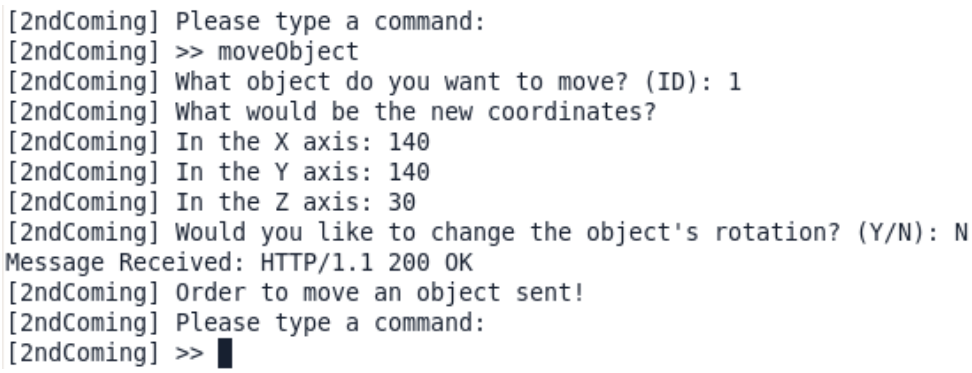

Figura 32 - *2ndComing* Interface - *MoveObject*

Depois da recolha destas informações pelo *2ndComing*, Figura 32,estas são enviadas encapsulando toda a informação na mensagem HTTP da seguinte forma:

```
public void PostMoveObject(string identifier, float[] vectorCoords, float[]
vectorOrientation)
{
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/moveObject";
        post.PostItems.Add("ID", identifier);
        post.PostItems.Add("coordxx", Convert.ToString(vectorCoords[0]));
        post.PostItems.Add("coordyy", Convert.ToString(vectorCoords[1]));
        post.PostItems.Add("coordzz", Convert.ToString(vectorCoords[2]));
        post.PostItems.Add("orientationxx", Convert.ToString(vectorOrientation[0]));
        post.PostItems.Add("orientationyy", Convert.ToString(vectorOrientation[1]));
        post.PostItems.Add("orientationzz", Convert.ToString(vectorOrientation[2]));
        post.PostItems.Add("orientationww", Convert.ToString(vectorOrientation[3]));

        post.Post();
}
```

Após recolha dos dados no *2ndComing OpenSim Module*, é preciso trata-los um pouco antes de se realizar o movimento do objecto.

```
public void MoveObject(string IDgiven, float[] vectorCoords, float[] vectorOrientation)
{
        List<SceneObjectGroup> objectPrims = new List<SceneObjectGroup>();
        objectPrims = AllObjects[IDgiven];
        float[] currentCoords = new float[3];
        currentCoords = ObjectPositions[IDgiven];
        float[] aux = new float[3];
        aux[0] = vectorCoords[0] - currentCoords[0];
        aux[1] = vectorCoords[1] - currentCoords[1];
        aux[2] = vectorCoords[2] - currentCoords[2];

        Vector3 offset = new Vector3(aux[0], aux[1], aux[2]);

        foreach(SceneObjectGroupsog in objectPrims)
        {
                sog.AbsolutePosition = sog.AbsolutePosition + offset;
                sog.UpdateGroupRotation(new Quaternion(0, 0,vectorOrientation[1]),
                vectorOrientation[2]));
                sog.ScheduleGroupForTerseUpdate();
        }
        ObjectPositions.Remove(IDgiven);
        ObjectPositions.Add(IDgiven, vectorCoords);
}
```

Primeiramente, é necessário calcular a distância do centro do objecto à nova posição. Esta distância servirá depois para somar à actual posição de cada um dos *prims* constituintes do objecto em causa, de forma a que todos se movam para o mesmo local mantendo os *offsets* que cada um tem em relação ao centro do objecto. Antes de se fazer o *update* a cada *prim,* é ainda actualizada a rotação deste em relação à cena.

4.4.8 DeleteObject

Como o próprio nome indica, esta funcionalidade elimina o objecto desejado. Para chamar esta função o utilizador necessita de saber o identificador do objecto que pretende eliminar. Para isso, há a possibilidade de utilizar o comando *ShowObjects* para obter informações sobre todos os objectos criados até aí, nomeadamente o identificador e o tipo de cada objecto.

Depois da escolha do objecto a eliminar, é criada a mensagem para ser enviada ao *2ndComing OpenSim Module*:

```
public void PostDeleteObject(string identifier)
{
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/deleteObject";
        post.PostItems.Add("ID", identifier);

        post.Post();
}
```

Como se pode observar pelo código apresentado em cima, apenas é introduzido nesta mensagem a funcionalidade pretendida – *deleteObject* – e o identificador desse objecto.

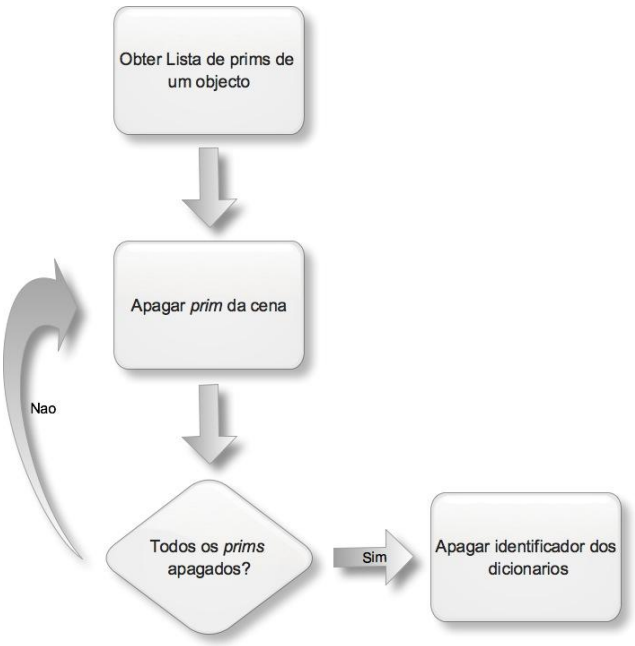

Figura 33 - Diagrama explicativo do comando *DeleteObject*

O fluxograma da Figura 33 demonstra como esta funcionalidade é implementada no *2ndComing OpenSim Module*.

```
public void DeleteObject(string IDgiven)
{
        List<SceneObjectGroup> objectPrims = new List<SceneObjectGroup>();
        objectPrims = AllObjects[IDgiven];

        foreach(SceneObjectGroupsog in objectPrims)
        {
                sog.DeleteGroup(false);
        }

        AllObjects.Remove(IDgiven);
        ObjectPositions.Remove(IDgiven);
}
```

O código acima escrito equivale a apagar do mundo virtual todos os *prims* constituintes do objecto, cujo identificador foi retirado da mensagem de *deleteObject*.

É criada uma lista local que é preenchida com a informação dos *prims* que constituem o objecto. A informação destes *prims* está armazenada no dicionário *AllObjects*. Posteriormente, cada *prim* nessa lista será retirado da cena. Para completar o processo, é necessário retirar dos dicionários todas as entradas relativas ao identificador em questão.

Depois de todo este processo estar concluído, o objecto foi apagado da cena e da memória do programa com sucesso.

4.4.9 ShowObjects

Este comando tem o objectivo de mostrar quais os objectos que foram criados pelo utilizador mostrando na interface o identificador e a referência de cada um, isto é, o tipo de cada objecto.

Este comando é útil quando o utilizador pretende mover ou eliminar um objecto necessitando assim de ver qual o identificador deste antes de o conseguir fazer. Este comando produz uma mensagem como a que pode ser vista na Figura 34.

```
[2ndComing] Please type a command:
[2ndComing] >> showObjects
[2ndComing] Listing all the current complex objects in the scene:
[2ndComing] Object ID    Reference
[2ndComing] 1            bus
[2ndComing] 2            car
[2ndComing] 3            bus
[2ndComing] 4            bus
[2ndComing] Please type a command:
[2ndComing] >>
```

Figura 34 - Interface *2ndComing* - *ShowObjects*

Para implementar esta funcionalidade, basta percorrer e mostrar o dicionário *ObjectsInfo* que contém informação de todos os objectos criados:

```
if (order == "showObjects")
{
    SCWrite("Listing all the current complex objects in the scene: \n");
    SCWrite("Object ID    Reference\n");
    foreach(KeyValuePair<string, string> pair in ObjectsInfo)
    {
            SCWrite(Convert.ToString(pair.Key) + "          " +
            Convert.ToString(pair.Value) + "\n");
    }
}
```

Esta funcionalidade é bastante simples de implementar pois consiste apenas na apresentação de alguns dados na consola, porém é de uma utilidade extrema visto que os identificadores dos objectos são atribuídos automaticamente pelo *2ndComing* e o utilizador necessita de os saber para executar acções como mover ou apagar objectos.

4.4.10 DeleteAll

Como o próprio nome indica, este comando tem como principal objectivo eliminar todo o conteúdo presente na cena, deixando-a completamente vazia.

Para enviar este comando para o *2ndComing OpenSim Module* é usada a seguinte função para a criação da mensagem HTTP:

```
    public void PostDeleteAll()
    {
        PostSubmitter post = new PostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/deleteAll";
        post.PostItems.Add("do", "magic");

        post.Post();
    }
```

O *2ndComing OpenSim Module*, ao receber esta mensagem vai interpretá-la e executar a seguinte acção:

```
    public void DeleteAll()
    {
        Console.WriteLine("[2ndComing] All content will be wiped. Please wait a
second, thank you.");

        actualScene.DeleteAllSceneObjects();

        //code to wipe all dictionaries
    }
```

A primeira acção será a de enviar uma mensagem para a interface avisando que todo o conteúdo presente na cena será apagado. Posteriormente, todo o conteúdo é apagado usando o método *DeleteAllSceneObjects* do *OpenSimulator*. Por fim, é apenas necessário apagar todas as entradas de todos os dicionários do *2ndComing* para que não exista conflito quando for criado novo conteúdo.

4.4.11 RestartScene

Esta funcionalidade tem o único objectivo de reiniciar a cena actual no simulador. Para a implementar basta criar uma mensagem HTTP com a seguinte informação:

```
    Public void PostRestartScene()
    {
        PostSubmitterpost = newPostSubmitter();
        post.Type = PostSubmitter.PostTypeEnum.Post;
        post.Url = "http://localhost:8080/restartScene";
        post.PostItems.Add("do", "Magic");

        result = post.Post();
    }
```

Como se pode observar apenas é necessário definir o URL da mensagem com a acção que se deseja executar. Depois do *2ndComing OpenSim Module* receber esta mensagem vai executar a seguinte acção:

```
Public void RestartScene()
{
    Console.WriteLine("Restartingthescene.   Please   login   againin   a
    fewseconds. ThankYou!");
    actualScene.RestartNow();
}
```

Pode visualizar-se que o método utilizado pela função para realizar esta acção vem já definido de origem pelo *OpenSimulator*, bastando que o *2ndComing* invoque este método para a cena reiniciar.

4.4.12 Help

Esta funcionalidade pode ser acedida através de três comandos diferentes:

- *?*
- *Help*
- *-h*

sendo que o resultado em todos eles será invariavelmente o mesmo: um menu que tem o objectivo de ajudar os utilizadores fornecendo informação sobre todas as funcionalidades presentes no *2ndComing* (Figura 35).

```
[2ndComing] Please type a command:
[2ndComing] >> ?
[2ndComing] Available commands:
[2ndComing]     ::createPrim :: creates a new prim in the scene.
[2ndComing]     ::genRoads :: generates the desired zone's streets.
[2ndComing]     ::terrainBuild :: creates the streets in the scene.
[2ndComing]     ::genBus :: generates the zone's bus.
[2ndComing]     ::simulateBus :: starts the bus simulation.
[2ndComing]     ::createObject :: creates an object.
[2ndComing]     ::moveObject :: moves the object of your choice.
[2ndComing]     ::deleteObject :: deletes the object of your choice.
[2ndComing]     ::deleteAll :: deletes all prims in the scene.
[2ndComing]     ::showObjects :: shows currently created objects.
[2ndComing] Please type a command:
[2ndComing] >>
```

Figura 35 - *2ndComing* Interface - *Help*

4.4.13 Shutdown

Com o objectivo de encerrar o *2ndComing* foi criado a função de *Shutdown*. Para realizar esta acção o utilizador pode executar diferentes comandos:

- *q*
- *quit*
- *shut*
- *shutdown*
- *exit*

sendo que o resultado será sempre o mesmo: o encerramento (Figura 36). Optou-se por introduzir esta opção visto que diferentes *softwares* utilizam diferentes comandos para o seu encerramento e assim, ao compilar alguns desses comandos de encerramento mais comuns, utilizadores com hábitos diferentes podem adaptar-se melhor ao *2ndComing*.

```
[2ndComing] Please type a command:
[2ndComing] >> q
[2ndComing]
[2ndComing] Thank you for using 2ndComing. Hope to see you again soon!
```

Figura 36 - *2ndComing* Interface - *shutdown*

Capítulo 5

Testes e Validação

Com o objectivo de avaliar tanto quantitativamente como qualitativamente o desempenho do *2ndComing* é necessário testá-lo. Para isso, foram realizados alguns testes, centrados nas áreas fundamentais do projecto:

- Geração e manuseamento de *prims*;
- Geração e manuseamento de objectos complexos;
- Geração de estradas.

Para cada uma das áreas apontadas acima, os testes realizados têm o objectivo de quantificar o tempo que o *2ndComing* demora a concretizar o pedido, bem como o desempenho do mesmo ao nível de performance, isto é, de consumo de recursos da máquina. Assim sendo, foi monitorizado tanto o processador como o consumo de memória em cada um dos testes.

É do conhecimento geral que o desempenho do *software* está intimamente ligado à capacidade de processamento da máquina onde o dito está a ser executado. O *2ndComing* não é excepção.

Por sua vez, o processador é a peça de *hardware* fundamental para o desempenho de uma máquina pelo que é indispensável saber como este se comporta quando usado para executar o *2ndComing*.

O computador utilizado para desenvolver todo o projecto, incluindo a fase de testes foi um MacBook 4.1 a 2.4GHz. O ambiente de desenvolvimento foi em Ubuntu 8.04 acedido a partir de uma máquina virtual usando o programa *VMWare Fusion*. Algumas das características de ambas as máquinas, quer física quer virtual, podem ser vistas na

Tabela **10**.

Tabela 10: Características das máquinas usadas

	Máquina	**Máquina Virtual**
Processador	Intel Core 2 Duo 2.400GHz	1 Processador Virtual
Memória RAM	2 GB	1 GB
Placa Gráfica	Intel GMA X3100	3D Disabled
Sistema Operativo	Mac OS X 10.5.6	Ubuntu 8.0.4

5.1 Teste A: Criar 1000 cubos

O teste consiste em gerar mil *prims* com a forma de um cubo com tamanho e posição na cena aleatórios. Assim, para cada *prim* criado é necessário enviar uma mensagem para o *2ndComingOpenSim Module*. Para este teste serão avaliados quer o tempo total quer o estado da utilização do processador e memória.

5.1.1 Tempo de criação:

Este teste tem o objectivo de determinar qual o tempo que demora a ser gerado cada *prim*. Como será praticamente impossível determinar este tempo criando apenas um *prim*, resolveu-se determinar o tempo que o *2ndComing* leva a gerar 1000 *prims* e depois determinar a média de tempo para cada um deles. Para se ter um resultado mais apurado, fizeram-se três testes, sendo o resultado final a média dos três.

Tabela 11: Tempo de execução do teste A

Teste	**Tempo**	**Tempo por cubo**
A.1	1minuto e 45segundos	105 milissegundos
A.2	1minuto e 45segundos	105 milissegundos
A.3	1minuto e 46segundos	106 milissegundos

Pode-se então concluir que cada prim demora em média cerca de 105 milissegundos a ser gerado pelo *2ndComing*. Este número pode aumentar se o processador da máquina que está a executar o *2ndComing* estiver sujeito a uma maior carga do que a presente nos testes anteriores.

5.1.2 Utilização da memória e processador

No decorrer do teste A.1, foi monitorizado quer a ocupação de memória quer os recursos de processador necessários pelo *2ndComing* para concretizar os pedidos.

5.1.2.1 Memória:

A Tabela 12, mostra o resultado com um espaçamento temporal de cinco segundos entre amostras. A tabela completa, com um espaçamento de apenas um segundo entre amostras, pode ser consultada no apêndice deste documento.

Tabela 12: Teste A - resultados de 5 em 5 segundos

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	35504	999860	96.57
0	32760	1002604	96.84
5	31672	1003692	96.94
10	28952	1006412	97.20
15	24928	1010436	97.59
20	23148	1012216	97.76
25	21960	1013404	97.88
30	21504	1013860	97.92
35	19772	1015592	98.09
40	19744	1015620	98.09
45	19776	1015588	98.09
50	19664	1015700	98.10
55	19604	1015760	98.11
60	19484	1015880	98.12
65	19504	1015860	98.12
70	19552	1015812	98.11
75	19568	1015796	98.11
80	24636	1010728	97.62
85	23252	1012112	97.75
90	20528	1014836	98.02
95	17084	1018280	98.35
100	21656	1013708	97.91
105	18628	1016736	98.20
110	18688	1016676	98.20

Pode-se observar o comportamento da máquina em termos de consumo de memória, cinco segundos antes e durante todo o período em que o teste decorreu.

À medida que o teste evolui, observa-se que a memória consumida vai aumentando o que, obrigatoriamente, leva a que a memória disponível vá diminuindo. No gráfico da Figura 37, pode-se observar como varia a memória consumida ao longo de todo o período de tempo em que decorreu o teste.

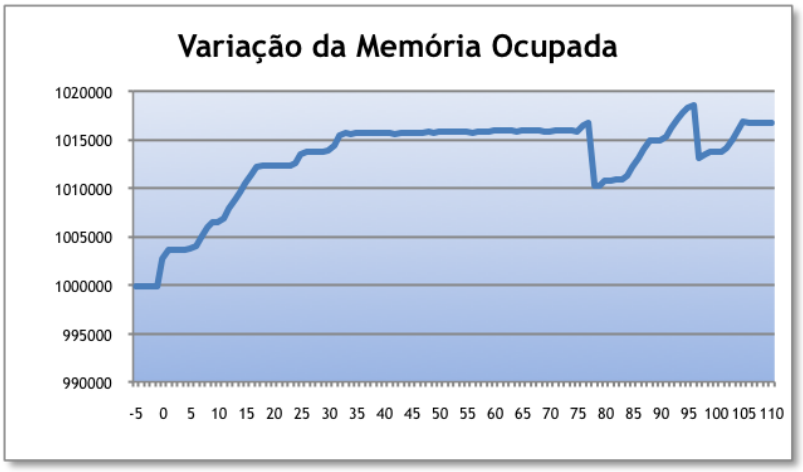

Figura 37 - Teste A: Variação da memória ocupada

Pela visualização da Figura 37, pode concluir-se que a utilização do *2ndComing* aumenta ligeiramente o consumo de memória, como aliás seria de esperar. Porém, pode considerar-se este aumento aceitável tendo em vista que um aumento médio de quinze Megabytes, com o estado da tecnologia actual é praticamente insignificante.

O gráfico seguinte, mostra a quantidade de memória, em percentagem, que a máquina tinha ocupada quando o teste foi realizado.

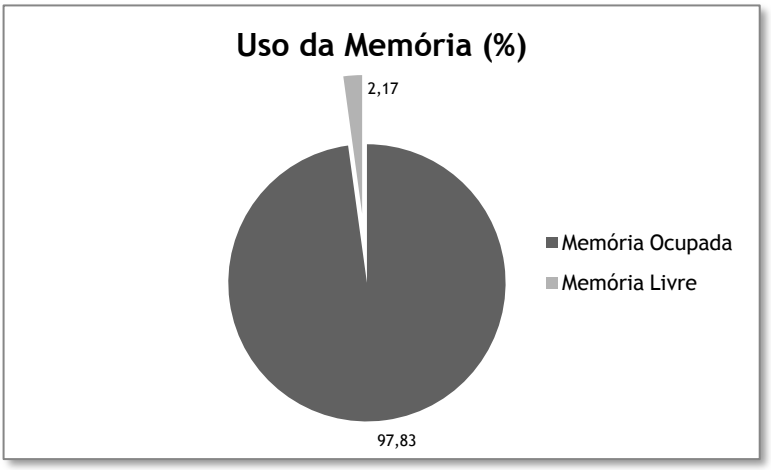

Figura 38 - Teste A: Percentagem do uso global de memória

Conclui-se assim, que praticamente toda a memória disponível estava a ser utilizada pela máquina. De facto, a quantidade de memória disponível, apenas 3%, é muito pouca, contudo, é importante referir que antes do inicio do teste, esta memória estava quase toda em uso, como se vê na Figura 38.

5.1.2.2 Processador:

Na Tabela 13, pode-se observar os dois principais consumidores de recursos do processador: processos executados pelo utilizador e processos necessários ao sistema. A terceira coluna corresponde a outras chamadas do processador ou à não utilização do mesmo.

Tabela 13: Teste A - resultados de consumo dos recursos do processador

Tempo (s)	Utilizador (%)	Sistema (%)	Não Usado ou Outros (%)
-5	3	0	97.00
0	16	5	79.10
5	7	5	88.10
10	8	5	87.00
15	8	4	88.00
20	12	2	86.00
25	11	3	86.00
30	12	5	83.00
35	8	4	88.00
40	13	4	83.00
45	8	2	90.00
50	12	5	83.00
55	11	3	86.00
60	13	4	83.00
65	9	4	87.00
70	10	8	82.00
75	10	4	86.00
80	65	27	8.00
85	12	4	84.00
90	11	6	83.00
95	32	11	57.00
100	15	5	80.00
105	12	2	86.00
110	4	0	96.00

Pela análise da Tabela 13, que pode ser consultada com mais resolução – menor espaçamento temporal entre amostras, no apêndice deste documento - conclui-se que aconteceu um pequeno incremento do uso do processador, principalmente por parte de processos executados pelo utilizador, visto que o uso por parte do sistema manteve-se praticamente inalterado, isto é, o acréscimo foi praticamente insignificante.

Compilando os dados da Tabela 13 em alguns gráficos, pode-se observar o uso do processador pelos diferentes tipos de processos ao longo de todo o teste.

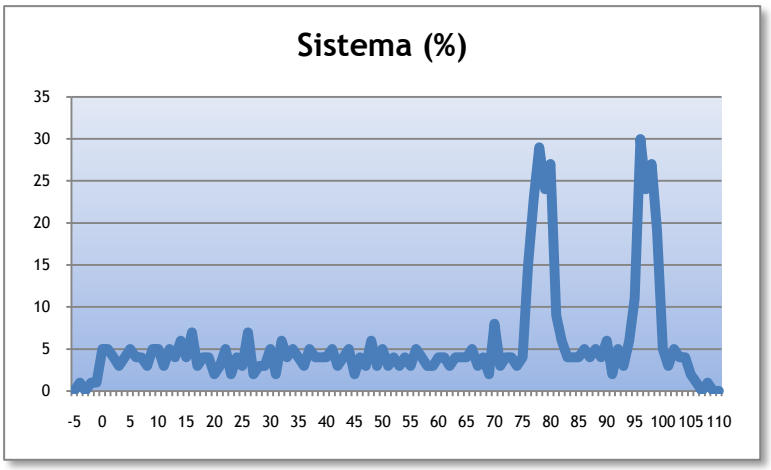

Figura 39 - Teste A: Gráfico do consumo de recursos do processador pelo sistema

No gráfico da Figura 39, observa-se que o uso do processador por parte de processos relativos ao sistema sofreu apenas um ligeiro aumento. No entanto, perto dos 75 e dos 95 segundos aconteceu um brusco acréscimo do uso do processador. De notar que estes dois picos devem-se ao facto de, após algum período de tempo de ser gerado conteúdo, o *OpenSimulator* guardar esse conteúdo na sua base de dados. Ora estas operações de escrita na base de dados consomem bastante mais recursos. De referir também, que mesmo após a conclusão do teste, outros picos semelhantes, relativos a novas escritas na base de dados, ocorreram. Esta solução tem em vista aumentar a performance do *OpenSimulator*, já que era mais prejudicial em termos de eficiência estar constantemente a escrever na base de dados *prim* a *prim*.

A Figura 40, mostra o gráfico da utilização dos recursos do processador por processos relativos ao utilizador.

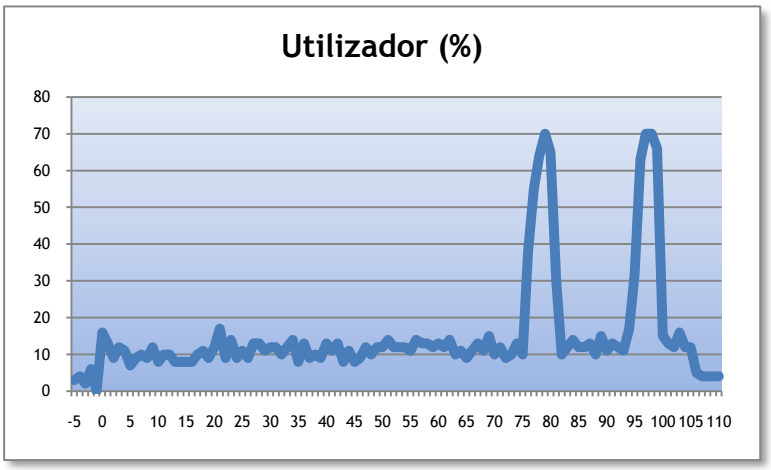

Figura 40 - Teste A: Gráfico do consumo de recursos do processador pelo utilizador

De notar, o aumento sofrido no inicio do teste, mais notório que no gráfico da Figura 39, aos cinco segundos. Os dois picos de utilização, ocorridos exactamente no mesmo instante – aos 75 e aos 95 segundos. Porém, a utilização neste caso chega perto dos 70%, ou seja, muito mais elevada do que nos processos executados pelo sistema.

Nestes picos, somando a utilização dos recursos do processador tanto pelo sistema como pelo utilizador, obtém-se um uso de quase 100% dos recursos disponíveis. No gráfico da Figura 41 pode-se observar isso mesmo, a quantidade de recursos livres do processador, ou em uso por outros processos. De notar as quebras que chegam praticamente aos 0% de recursos livres do processador.

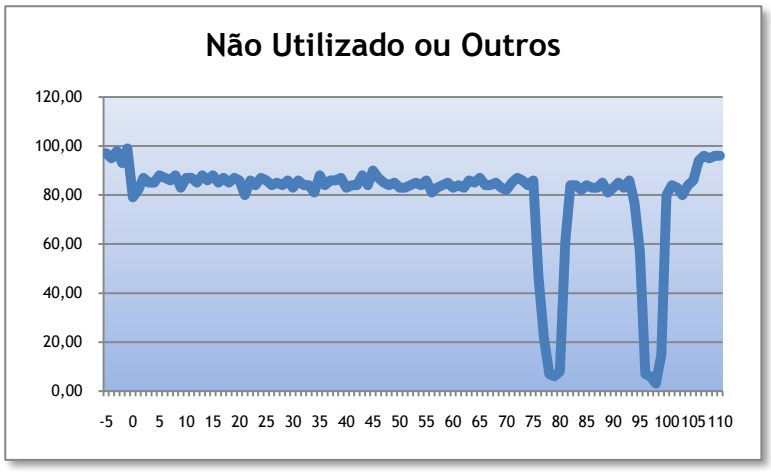

Figura 41 - Teste A: Gráfico da percentagem de recursos do consumidor não utilizados

Conclui-se assim que este teste consumiu cerca de 20% dos recursos do processador ao longo do seu período de execução, chegando a gastar perto de 100% dos recursos deste quando se deram os picos de consumo, relativos à escrita na base de dados por parte do *OpenSimulator*.

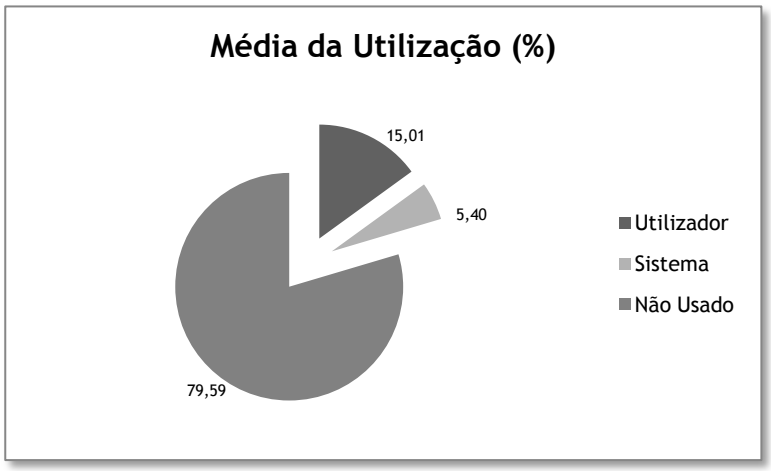

Figura 42 - Teste A: Gráfico da média da utilização dos recursos do processador

No gráfico da Figura 42, observa-se a média dos valores de uso dos recursos do processador no decorrer do teste A.

5.2 Teste B: Mover1000 cubos

Este teste consiste em enviar uma ordem para que todos os *prims* anteriormente criados no teste A se movimentem. Isto é, todos os cubos criados vão-se movimentar aleatoriamente alguns metros. Na segunda vez que se movimenta, cada cubo volta à sua posição original pelo que este teste tem o objectivo de apenas testar a performance do *2ndComing* quando vários *prims* se estão a movimentar.

5.2.1 Utilização da memória e processador

5.2.1.1 Memória

Na Tabela 14, pode-se observar o estado da memória do sistema um pouco antes e durante o decorrer de todo o teste. Uma tabela com maior detalhe foi colocada no apêndice deste documento.

Tabela 14: Teste B - Consumo de memória a cada 5segundos

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	33828	1001536	96.73
0	33608	1001756	96.75
5	36264	999100	96.50
10	35104	1000260	96.61
15	33480	1001884	96.77
20	31532	1003832	96.95
25	29392	1005972	97.16
30	27532	1007832	97.34
35	25576	1009788	97.53
40	22844	1012520	97.79
45	20824	1014540	97.99
50	18712	1016652	98.19
55	16376	1018988	98.42
60	14648	1020716	98.59

Pode-se concluir pela análise da Tabela 14 que a memória livre diminuiu de cerca de trinta megabytes para metade, ou seja, quinze megabytes. Este valor é bastante idêntico ao obtido no teste anterior.

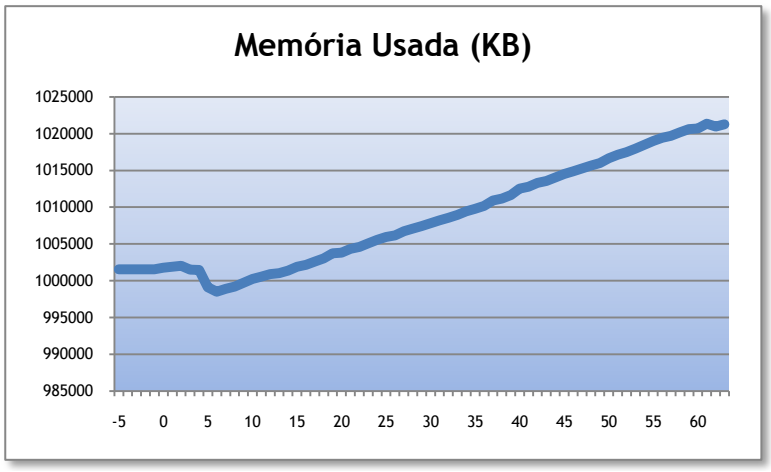

Figura 43 - Teste B: Gráfico da variação da memória usada ao longo do teste

Pelo gráfico da Figura 43, pode-se observar que o consumo de memória neste caso foi mais linear que no caso anterior. Apesar de um pequeno decréscimo cinco segundos após o início do teste, que pode dever-se aos mais variados motivos externos ao teste, o consumo de memória foi gradualmente subindo até à conclusão do mesmo.

A média de utilização da memória pode ser observada na Figura 44.

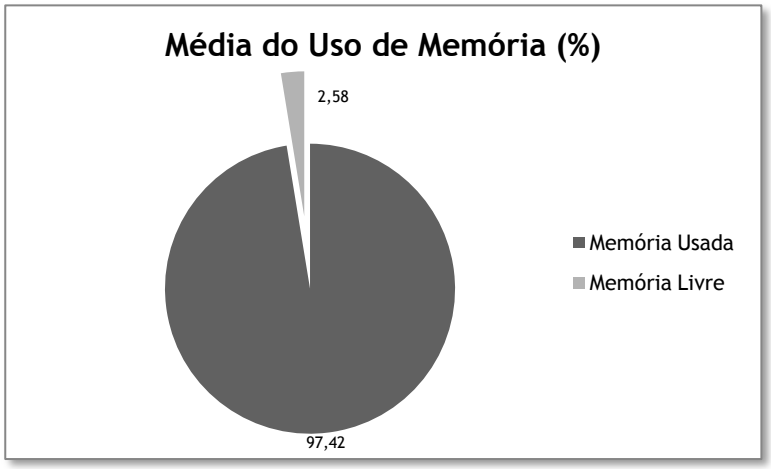

Figura 44 - Teste B: Média do uso de memória

À semelhança do teste anterior, a média da memória usada pelo sistema situa-se nos 97%.

5.2.1.2 Processador

Tabela 15: Teste B - utilização do processador de 5 em 5 segundos

Tempo	Utilizador (%)	Sistema (%)	Não Utilizado ou Outros (%)
-5	5	0	95
0	46	15	39
5	70	24	6
10	71	23	6
15	72	24	4
20	71	21	8
25	80	18	2
30	75	24	1
35	67	22	11
40	72	21	7
45	73	23	4
50	84	15	1
55	75	21	4
60	76	16	8

Pode verificar-se, pelos valores apresentados na Tabela 15, o consumo dos recursos do processador neste teste é bastante mais alto que no anterior, tanto por parte do sistema como por parte do utilizador. De notar também, a subida que ocorre no consumo dos recursos no início do teste, passando de cerca de 5% para 45% no caso do utilizador e de 0% para 15% no caso do sistema.

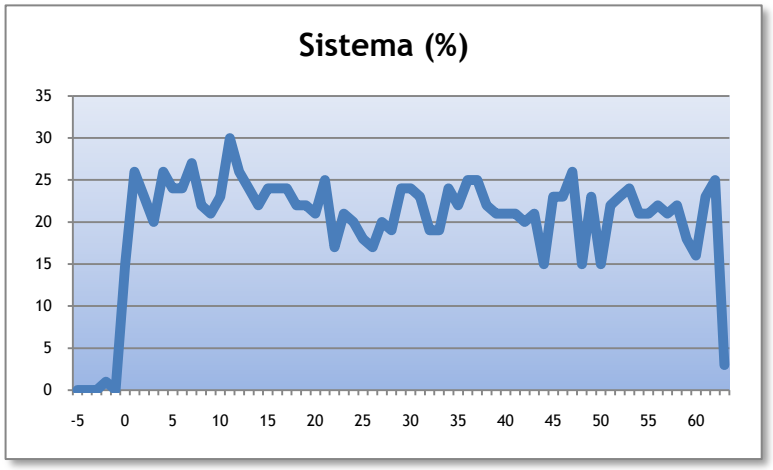

Figura 45 -Teste B: Consumo do processador pelo Sistema

Como se pode observar no gráfico da Figura 45, o consumo dos recursos do processador por parte de processos relativos ao sistema situa-se num valor médio entre os 20% e os 25%. Este valor é portanto bastante superior ao valor obtido no teste anterior para este tipo de processos.

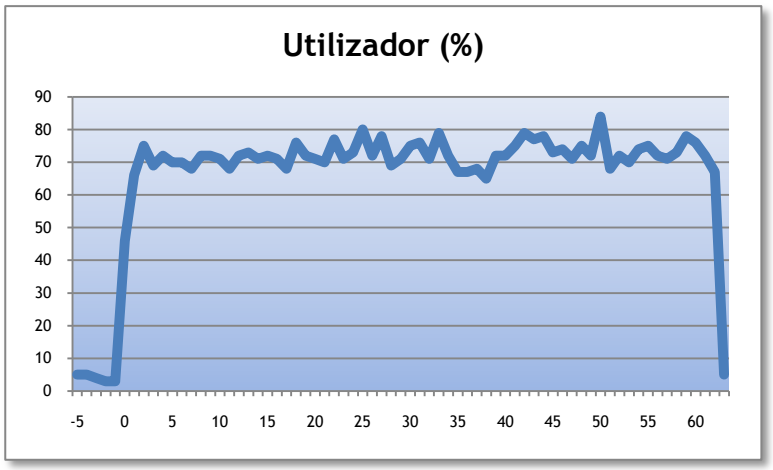

Figura 46 - Teste B: consumo do processador pelo utilizador

A percentagem de uso dos recursos no gráfico da Figura 46, relativo ao consumo de recursos do processador por parte de processos associados ao utilizador, situa-se entre os 70 e 75%. Um valor bastante alto quando comparado com o teste A fazendo prever uma utilização quase completa dos recursos disponíveis.

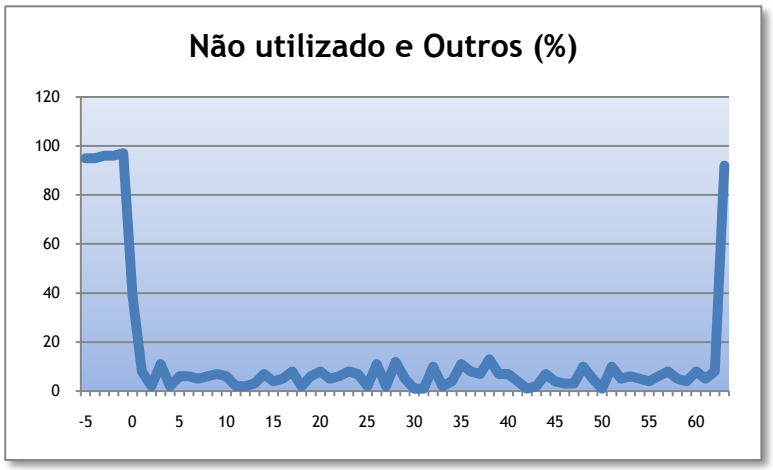

Figura 47 - Teste B: Gráfico da percentagem de recursos do processador não usados

Como era previsível, pela observação dos gráficos anteriores, os recursos do processador não utilizados foram praticamente nulos. É exactamente este fenómeno que se pode observar no gráfico da Figura 47: valores de não utilização do processador a rondar os 5%.

Este teste serviu apenas para monitorizar a actividade do processador e pode concluir-se que este foi usado praticamente na sua capacidade máxima. O aumento do uso do processador do teste B em relação ao teste A pode ser explicado muito facilmente. No teste A, cada acção necessitava de uma mensagem HTTP para dar a ordem, o que fazia com que todo o processo demorasse mais tempo, daí o consumo mais distribuído dos recursos. No teste

B, pelo contrário, apenas é enviada uma mensagem e depois todo o teste é baseado em processamento, daí o alto consumo dos recursos do processador.

Para concluir este teste, observe-se o gráfico da Figura 48 que mostra a média da percentagem de uso dos recursos.

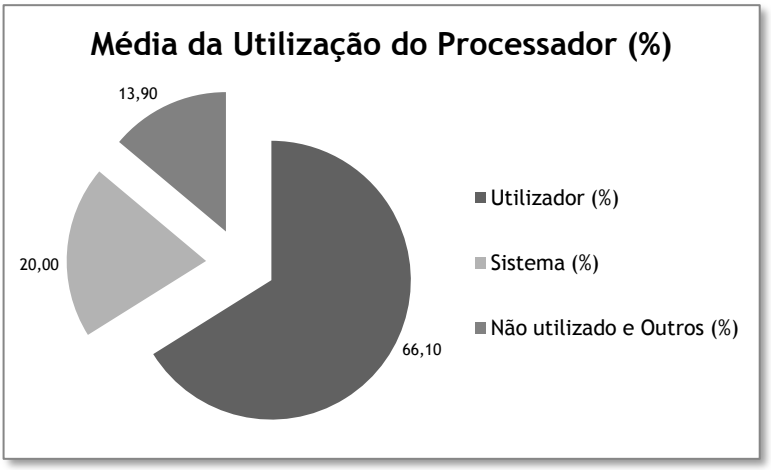

Figura 48 -Teste B: Gráfico da média da utilização do processador

Os processos relativos ao utilizador têm uma parte bastante expressiva dos recursos com 66%. A não utilização do processador ou a sua utilização por outros processos está apenas nos 14%.

5.3 Teste C: Apagar 1000 cubos

Este teste consiste em apagar todos os cubos previamente gerados. Para isso será apenas enviada uma mensagem HTTP para ordenar esta acção pelo que a troca em termos de mensagens de rede será muito pouca. Espera-se contudo que o processamento seja elevado e que o tempo de duração do teste seja consideravelmente inferior ao do teste A.

5.3.1 Tempo de remoção dos cubos

Consiste em determinar o tempo que o *2ndComing* leva para apagar os cubos gerados nos testes anteriores, ou seja, os 1000 cubos. Os resultados dos três ensaios realizados neste teste estão presentes na Tabela 16.

Tabela 16: Tempo de execução do teste C

Teste	Tempo	Tempo por cubo
C.1	51 segundos	51 milissegundos
C.2	51 segundos	51 milissegundos
C.3	53 segundos	53 milissegundos

5.3.2 Utilização da memória e processador

Durante a realização do teste C.1 foi monitorizado quer o consumo de memória (Tabela 17) quer o consumo dos recursos disponíveis do processador (Tabela 18).

5.3.2.1 Memória

Tabela 17 - Teste C: monitorização da memória do sistema

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	39252	996112	96,21
0	39008	996356	96,23
5	38980	996384	96,24
10	38820	996544	96,25
15	38836	996528	96,25
20	38756	996608	96,26
25	38624	996740	96,27
30	38484	996880	96,28
35	38128	997236	96,32
40	38196	997168	96,31
45	37880	997484	96,34
50	37872	997492	96,34
55	37864	997500	96,34

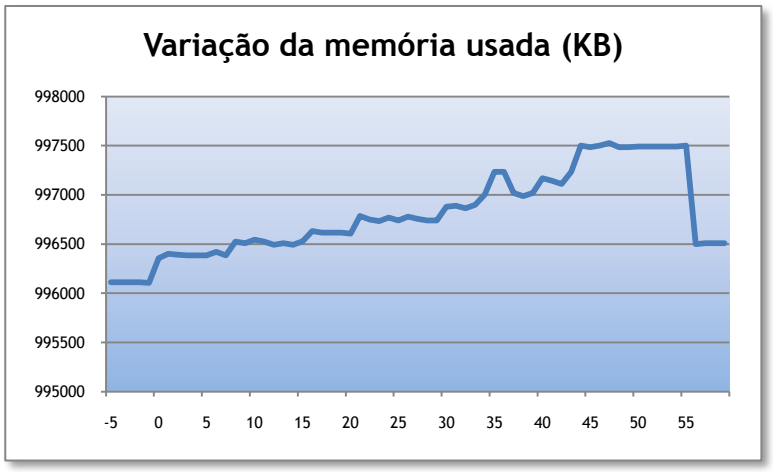

Figura 49 - Teste C: Gráfico da variação da memória usada

A utilização da memória vai aumentando de uma forma pouco linear durante o decorrer do teste, como se pode observar no gráfico da Figura 49. No final deste, observa-se uma pequena descida no valor da memória ocupada.

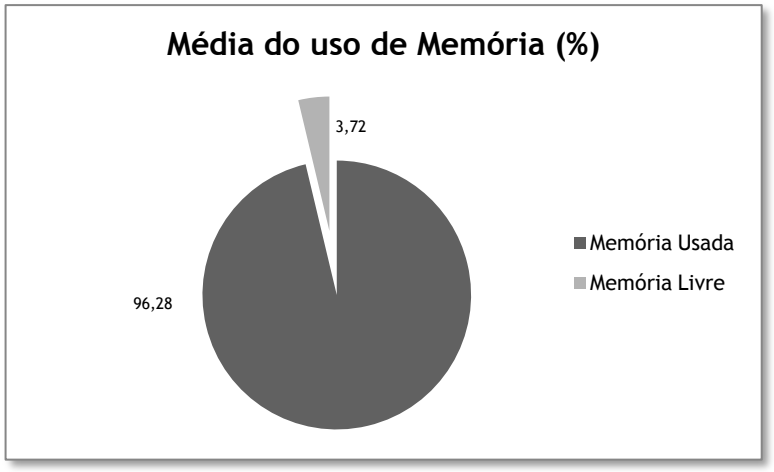

Figura 50 - Teste C: Gráfico da média do uso de memória

Como se tem observado nos testes anteriores, a percentagem de memória livre é bastante menor do que a de memória ocupada (Figura 50).

5.3.2.2 Processador

Tabela 18 - Teste C: Monitorização do processador

Tempo (s)	Utilizador (%)	Sistema (%)	Não Utilizado e Outros (%)
-5	4	1	95
0	35	14	51
5	68	30	2
10	69	27	4
15	69	28	3
20	65	32	3
25	70	27	3
30	70	26	4
35	76	18	6
40	69	27	4
45	76	21	3
50	7	1	92
55	0	0	100

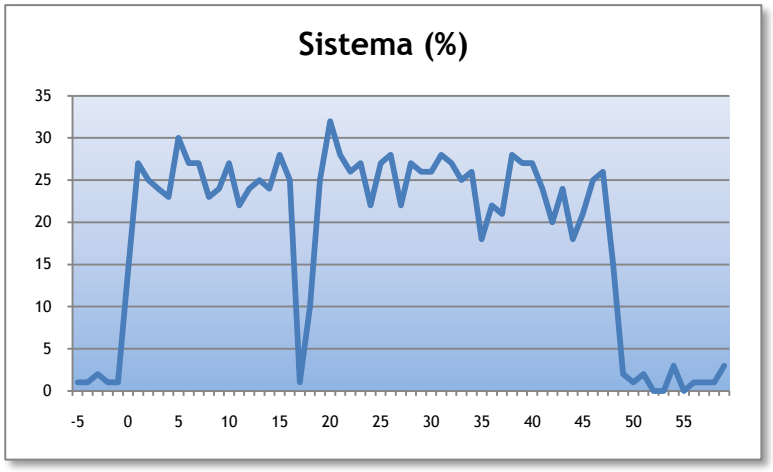

Figura 51 - Teste C: Gráfico do uso dos recursos do processador pelo sistema

Pela observação do gráfico da Figura 51, os resultados deste teste são bastante semelhantes aos obtidos no teste B, isto é, nota-se um uso bastante mais intenso do processador nestes dois testes comparativamente com o teste A. Mais uma vez, a utilização do processador por parte de processos ligados ao sistema situa-se entre os 20 e os 25%.

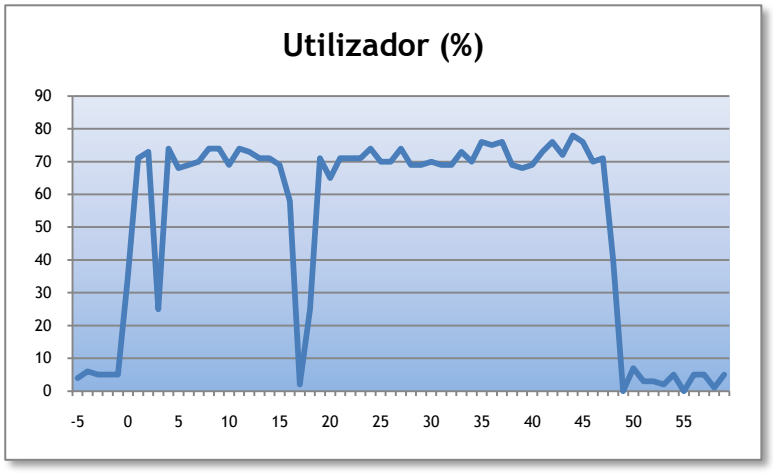

Figura 52 - Teste C: Gráfico do uso dos recursos do processador pelo utilizador

Os processos ligados ao utilizador (Figura 52) por seu lado gastam cerca de três vezes mais que os processos do sistema. Estando perto dos 70% o valor médio da utilização do processador por parte destes processos.

Após a leitura destes resultados, é de esperar que o processador esteja praticamente sempre a funcionar no limiar das suas capacidades como se pode observar pelo gráfico da Figura 53.

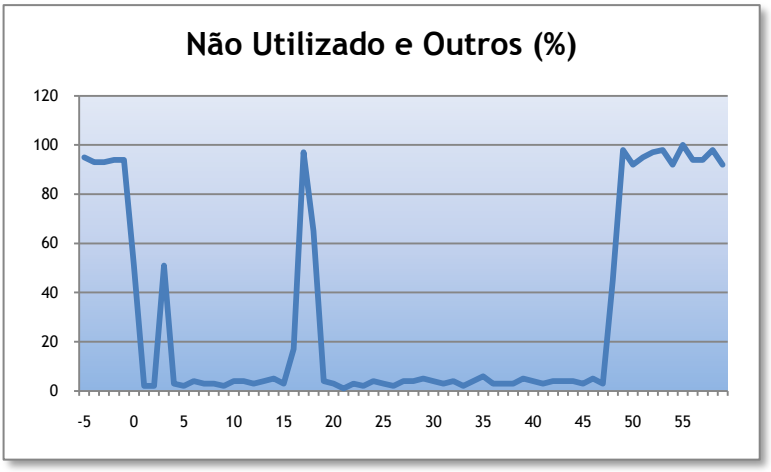

Figura 53 -Teste C: Gráfico da não utilização dos recursos do processador

O gráfico mostra duas zonas em que os recursos do processador são quase por completo utilizados.

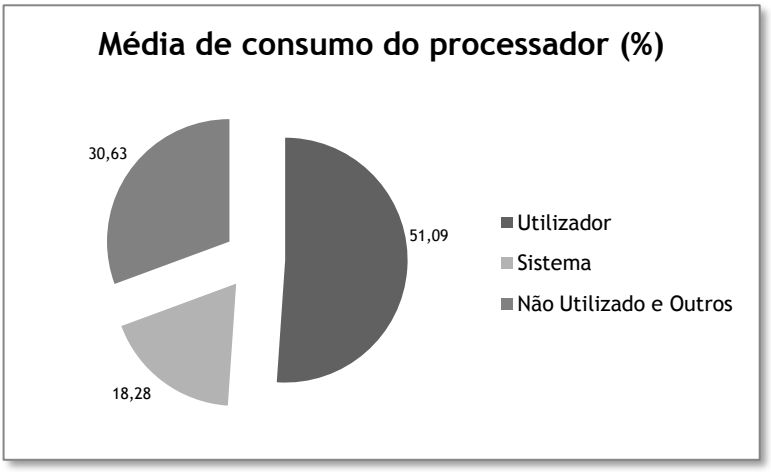

Figura 54 -Teste C: Gráfico da média de consumo dos recursos do processador

Pela observação da Figura 54 conclui-se que a quota de recursos usada pelos processos ligados ao utilizador continua a ser a maior – 51%- e que, em relação ao último teste, sofreu uma queda em detrimento dos recursos usados pelo sistema.

5.4 Teste D: Criar 100 Objectos Complexos

Este teste consiste na criação de cem objectos complexos. Cada objecto criado será do tipo autocarro, isto é, a referência passada será a de um autocarro: *bus*.

Cada objecto complexo com esta referência é constituído por cinco *prims*: um do tipo *box* (corpo do autocarro) e quatro cilindros (cada uma das rodas). Portanto, ao serem criados cem objectos complexos deste tipo estão a ser criados 500 *prims*.

5.4.1 Tempo de criação dos 100 objectos complexos

Na Tabela 19 podem ser observados os resultados dos tempos de execução deste teste assim como a média do tempo que o *2ndComing* demora a criar um objecto complexo deste tipo.

Tabela 19: Tempo de execução do teste D

Teste	Tempo	Tempo por objecto
D.1	10.7 segundos	107 milissegundos
D.2	10.7 segundos	107 milissegundos
D.3	10.7 segundos	107 milissegundos

Chega-se portanto à conclusão que cada objecto complexo do tipo autocarro demora, em média, 107milisegundos a ser criado.

5.4.2 Utilização da memória e processador

5.4.2.1 Memória

A Tabela 20 mostra os resultados obtidos na monitorização da memória do sistema durante o teste D.1. A tabela correspondente, com intervalos de 1segundo entre amostras, pode ser consultada no apêndice do documento.

Tabela 20 - Teste D: Monitorização do consumo de memória

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	47288	988076	95,43
0	44548	990816	95,70
5	43348	992016	95,81
10	39784	995580	96,16
15	39428	995936	96,19
20	39436	995928	96,19

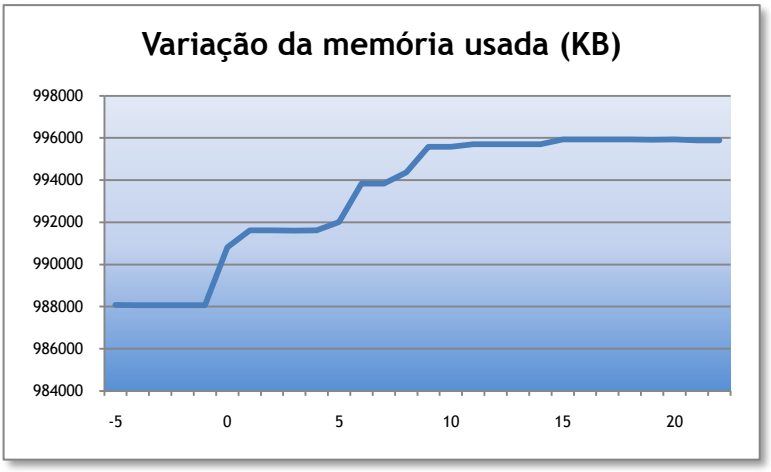

Figura 55 - Teste D: Gráfico da variação da memória usada

Mais uma vez, o consumo de memória ao longo do teste foi sempre aumentando sendo observável no gráfico da Figura 55 um efeito em escada.

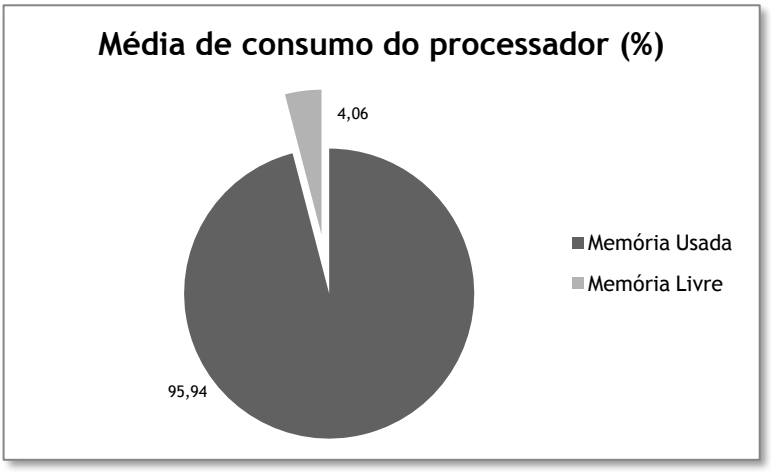

Figura 56 - Teste D: Gráfico da média de consumo dos recursos do processador

Durante este teste cerca de 4% da memória total do sistema estava livre, como aliás se pode observar na Figura 56.

5.4.2.2 Processador

Será de esperar neste teste um baixo consumo de recursos do processador, uma vez que por cada objecto complexo criado será preciso enviar uma mensagem com essa ordem.

Tabela 21: Teste D: Monitorização do Processador

Tempo (s)	Utilizador (%)	Sistema (%)	Não utilizado e Outros (%)
-5	7	1	92
0	19	3	78
5	11	3	86

10	12	7	81
15	4	0	96
20	8	2	90

Pelos valores presentes na Tabela 21 pode-se desde já confirmar a percentagem de consumo do processador será baixa.

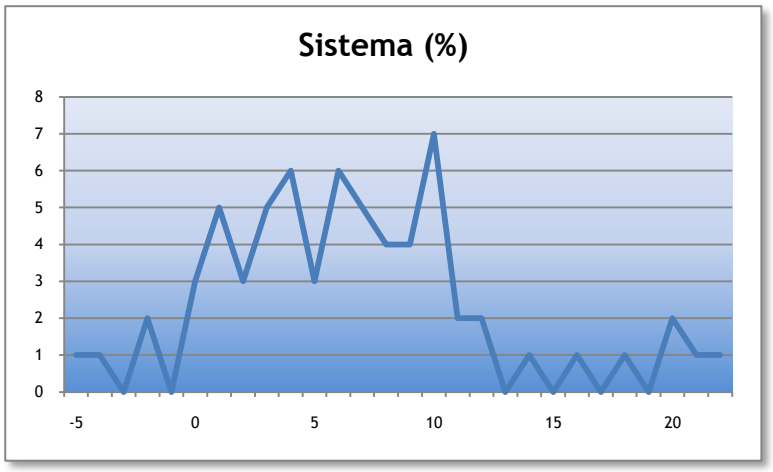

Figura 57 - Teste D: Gráfico da percentagem de consumo do processador pelo sistema

Como se pode observar pelo gráfico da Figura 57, a utilização dos recursos do processador por parte do sistema é bastante baixa e muito irregular, sendo o valor médio muito próximo dos 5%.

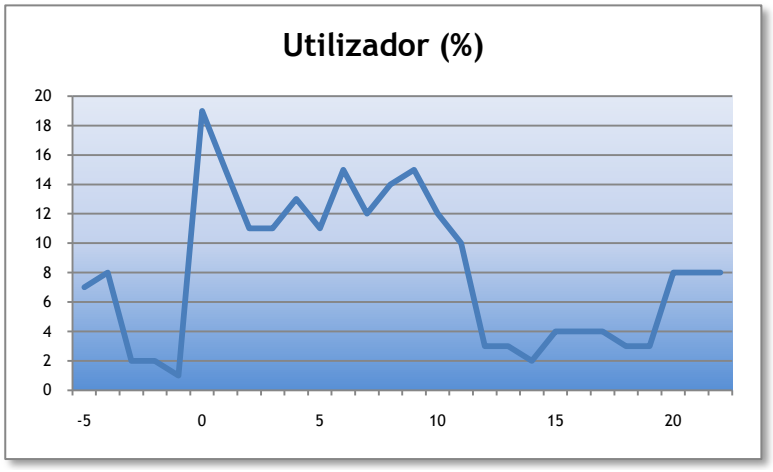

Figura 58 - Teste D: Gráfico da percentagem de consumo do processador pelo utilizador

Por parte dos processos ligados ao utilizador (Figura 58), o consumo dos recursos do processador são bastante maiores que os do sistema, como de resto se tem verificado em todos os testes. Porém, as percentagens são também bastante baixas, fazendo desde logo prever uma alta percentagem de não utilização do processador durante todo o teste.

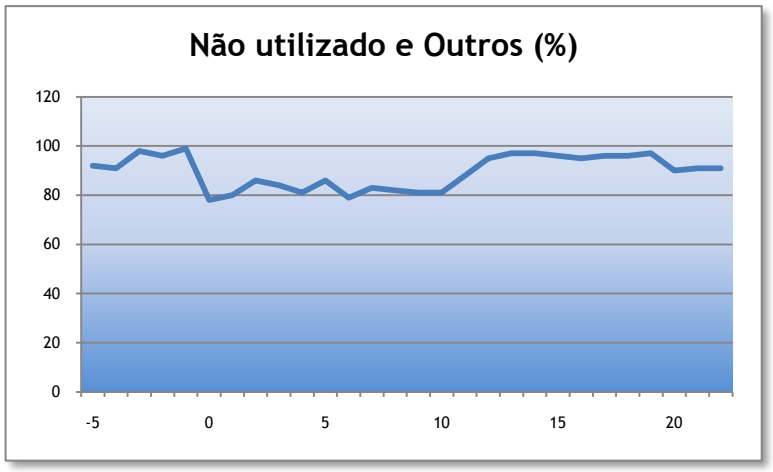

Figura 59 - Teste D: Gráfico da percentagem de não utilização do processador

Como era já esperado pelas análises nos gráficos anteriores, a percentagem de recursos do processador que não são utilizados é bastante elevada, ao longo de todo o teste nunca baixou dos 80% (Figura 59).

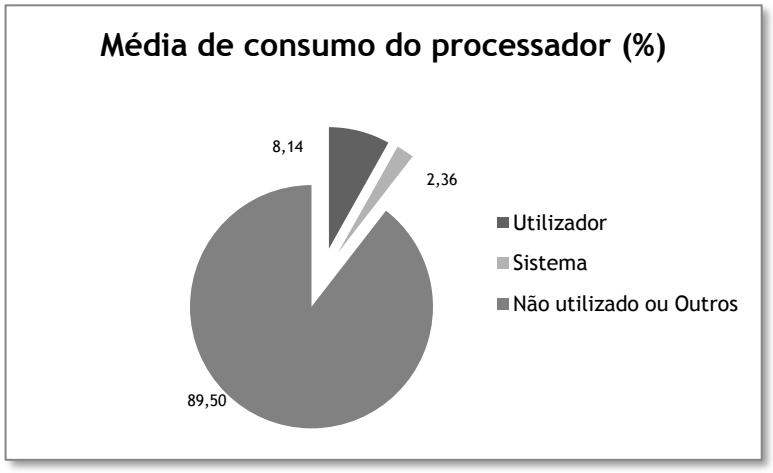

Figura 60 - Teste D: Média de consumo do processador

O gráfico da Figura 60 mostra, com bastante clareza, que durante todo o processo, os recursos do processador foram, em média, apenas usados cerca de 10%, isto é, apenas 10% dos recursos foram aproveitados durante todo o teste.

5.5 Teste E: Mover 100 objectos complexos

Este teste consiste, como o teste B, em fazer mover os objectos complexos criados no teste anterior, isto é, os 100 objectos criados no teste D.

Será apenas enviada uma mensagem a dar a ordem de movimento de todos os objectos presentes na cena e, durante alguns segundos, estes vão tremer um pouco à volta da sua actual posição.

5.5.1 Utilização da memória e processador

Visto que neste teste não é relevante o tempo de teste, serão apenas monitorizados o uso de memória e o consumo dos recursos do processador.

5.5.1.1 Memória

A Tabela 22 contém dados sobre a variação da memória do sistema com um intervalo temporal de cinco segundos entre amostras. A tabela correspondente com intervalos de apenas um segundo entre amostras pode ser encontrada no apêndice do documento.

Tabela 22: Teste E - Variação da memória do sistema de 5 em 5 segundos

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	43436	991928	95,80
0	40052	995312	96,13
5	39808	995556	96,16
10	41760	993604	95,97
15	41652	993712	95,98
20	41012	994352	96,04
25	40444	994920	96,09
30	40064	995300	96,13

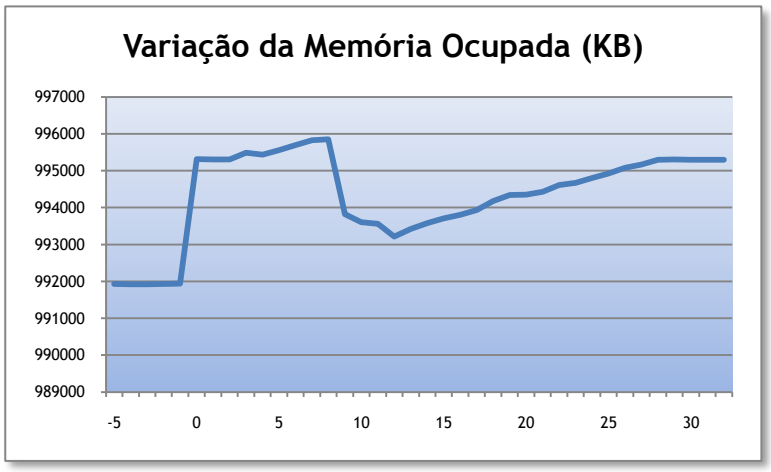

Figura 61 - Teste E: Variação da memória ocupada no sistema

Mais uma vez se assiste a um aumento do consumo de memória na utilização do *2ndComing* (Figura 61).

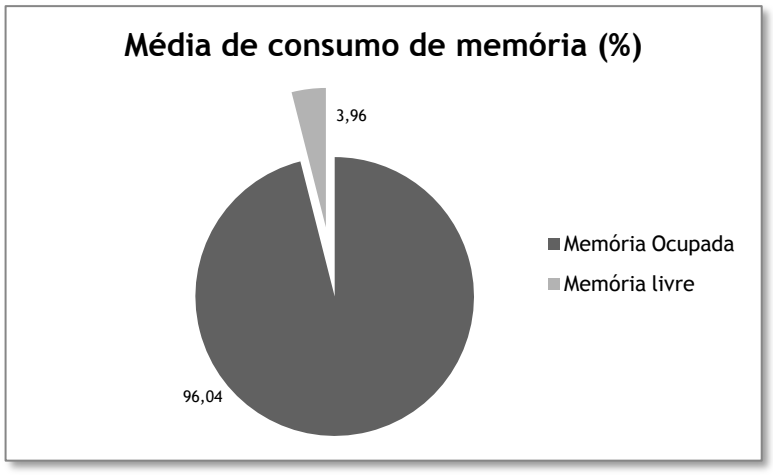

Figura 62 - Teste E: Média de consumo de memória

Como tem acontecido em todos os testes, em média, apenas cerca de 4% da memória total do sistema se encontra livre na altura da realização dos testes - **Figura 62**.

5.5.1.2 Processador

Espera-se com este teste o aumento do uso do processador, uma vez que apenas uma mensagem é trocada entre módulos e o restante do teste é baseado em processamento. Os resultados do teste podem ser consultados na Tabela 23.

Tabela 23: Teste E: Variação do consumo do processador

Tempo (s)	Utilizador (%)	Sistema (%)	Outros (%)
-5	3	0	97
0	19	5	76
5	39	24	37
10	52	16	32
15	44	22	34
20	46	21	33
25	49	21	30
30	15	2	83

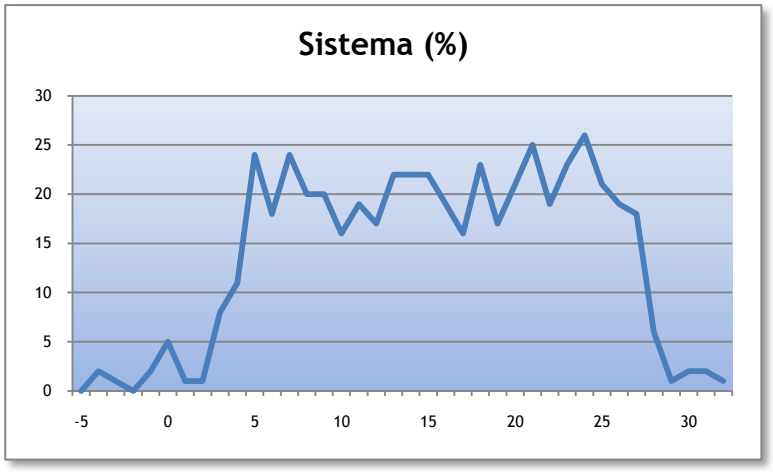

Figura 63 - Teste E: Gráfico da percentagem de consumo do processador pelo sistema

Como era de prever o uso de recursos do processador aumentou bastante, situando-se no valor médio de 20% o uso por parte dos processos relativos ao sistema (Figura 63).

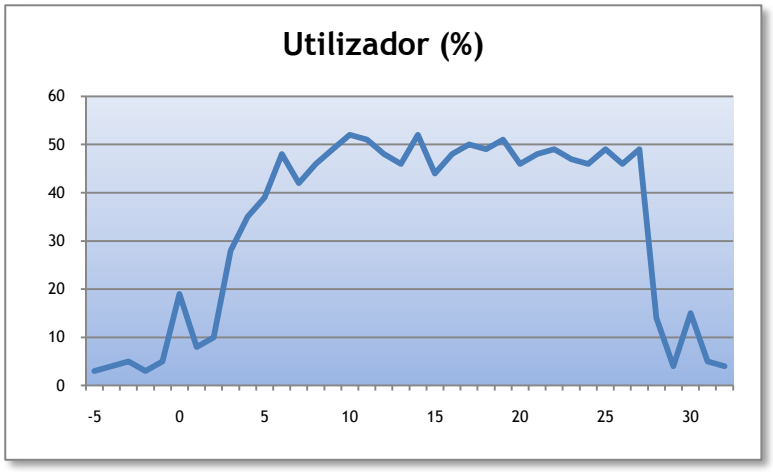

Figura 64 - Teste E: Gráfico da percentagem de consumo do processador pelo sistema

Pelos processos relativos ao utilizador, gráfico da Figura 64,o uso do processador também aumentou consideravelmente, situando-se o valor médio em cerca dos 40%. Apesar de bastante mais baixo que em alguns dos testes realizados anteriormente, este é também um valor bastante alto de uso dos recursos do processador.

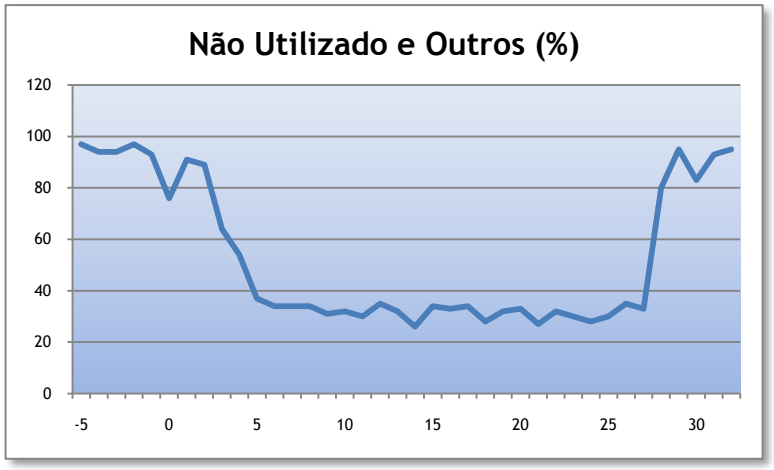

Figura 65 - Teste E: Gráfico da percentagem de não utilização do processador

Nota-se no gráfico da Figura 65 uma zona, pouco depois do início do teste, em que o uso dos recursos do processador foi bastante acentuado, porém, mesmo nessa zona, a quota de recursos não utilizados é alta – superior a 30%.

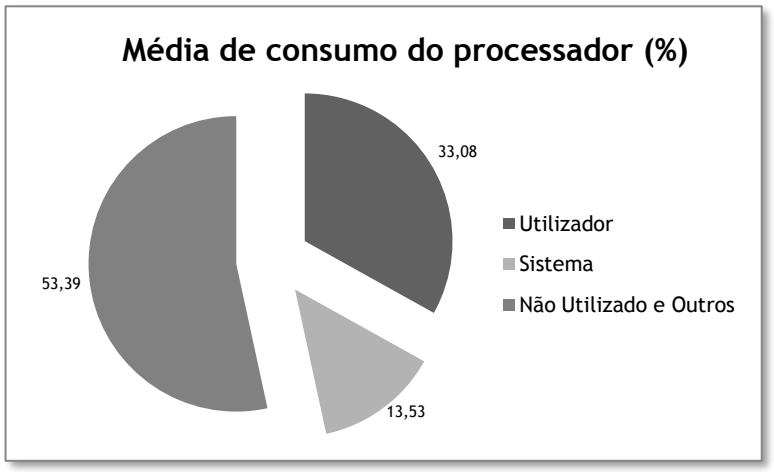

Figura 66 - Teste E: Gráfico da média de consumo do processador

Pela análise do gráfico da Figura 66 pode-se concluir que, apesar do aumento no uso dos recursos do processador, a maioria dos recursos não foram, em média, usados no decorrer deste teste: 53%. Contudo, os processos relativos ao utilizador continuam a necessitar de mais recursos que os processos relativos ao sistema.

5.6 Teste F: Remoção de 100 objectos complexos

Este teste consiste em eliminar da cena actual no mundo virtual, todos os objectos complexos criados anteriormente. Por outras palavras, por cada objecto criado, serão apagados cinco *prims*. Será importante comparar o resultado deste teste com o resultado do Teste C, relativo à remoção de *prims*, principalmente em termos de tempo de teste. Espera-

se, no entanto, que os resultados em termos de consumo de recursos de memória e de processador sejam idênticos em ambos os testes.

5.6.1 Tempo de remoção dos 100 objectos

Tabela 24: Tempo de execução do teste F

Teste	Tempo	Tempo por objecto
F.1	12.1 segundos	121 milissegundos
F.2	13.8 segundos	138 milissegundos
F.3	12.1 segundos	121 milissegundos

Assim sendo, para a remoção de cada objecto complexo foram necessários, em média, 127 milissegundos. Dado que cada objecto complexo é constituído por cinco *prims* pode-se estimar que cada *prim* demorou cerca de 25.4 milissegundos a ser eliminado.

5.6.2 Utilização da memória e processador

5.6.2.1 Memória

Na Tabela 25 podem ser consultados os resultados, com intervalos de 5 segundos entre amostras, relativos à monitorização da memória do sistema no teste F.1.

Tabela 25: Teste F - Variação da memória no sistema de 5 em 5 segundos

Tempo (s)	Memória Livre (KB)	Memória Ocupada (KB)	Memória Ocupada (%)
-5	45672	989692	95,59
0	43028	992336	95,84
5	45904	989460	95,57
10	46316	989048	95,53
15	46332	989032	95,53

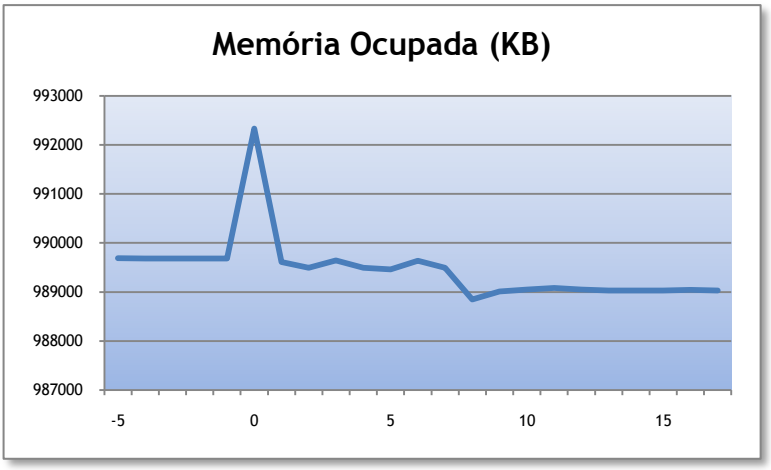

Figura 67 - Teste F: Variação da memória ocupada

Apesar de um pico no valor da memória ocupada logo no inicio do teste, este valor foi diminuindo, embora muito ligeiramente, durante todo o processo do teste (Figura 67).

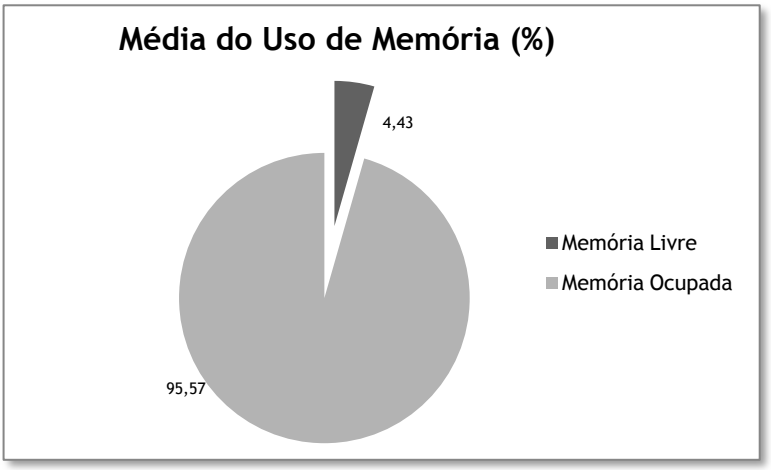

Figura 68 - Teste F: Média do uso de memória

A percentagem média da memória livre é ligeiramente, menos que meio ponto percentual, superior à percentagem obtida nos testes anteriores - Figura 68.

5.6.2.2 Processador

Tabela 26: Teste F - Variação do uso dos recursos do processador de 5 em 5 segundos

Tempo (s)	Utilizador (%)	Sistema (%)	Não Utilizado ou Outros (%)
-5	8	0	92
0	67	7	26
5	89	11	0
10	89	11	0

15	0	1	99

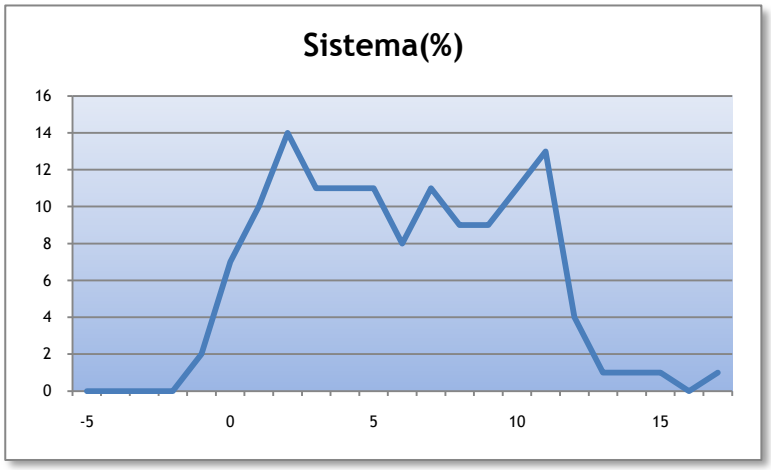

Figura 69 . Teste F: Gráfico da percentagem de consumo do processador pelo sistema

No gráfico da Figura 69 pode-se verificar um uso médio de cerca de 10% dos recursos do processador. É necessário observar como se comportam os processos relativos ao utilizador para se saber qual a percentagem total de recursos de utilizadores não usados.

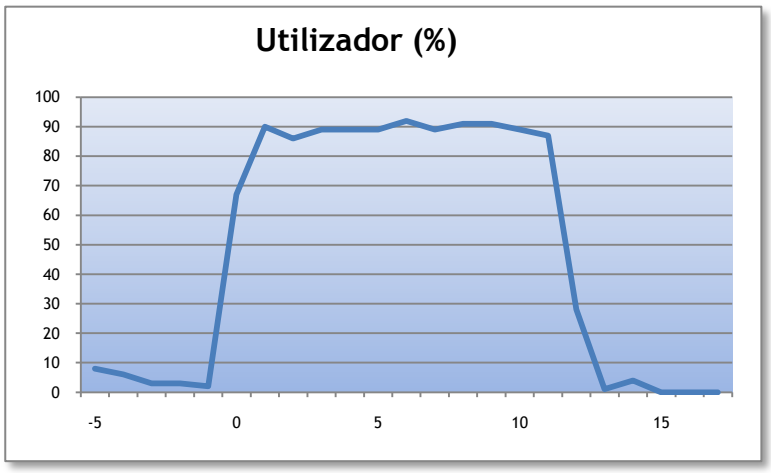

Figura 70 - Teste F: Gráfico da percentagem de consumo do processador pelo utilizador

Pela observação do gráfico da Figura 70, prevê-se que durante o tempo em que decorreu o teste, a percentagem de recursos do processador livres seja muito próxima de zero.

Este é aliás o teste em que existe um maior uso dos recursos por parte dos processos associados ao utilizador. Durante o tempo de teste, a percentagem de utilização foi quase sempre superior a 90%.

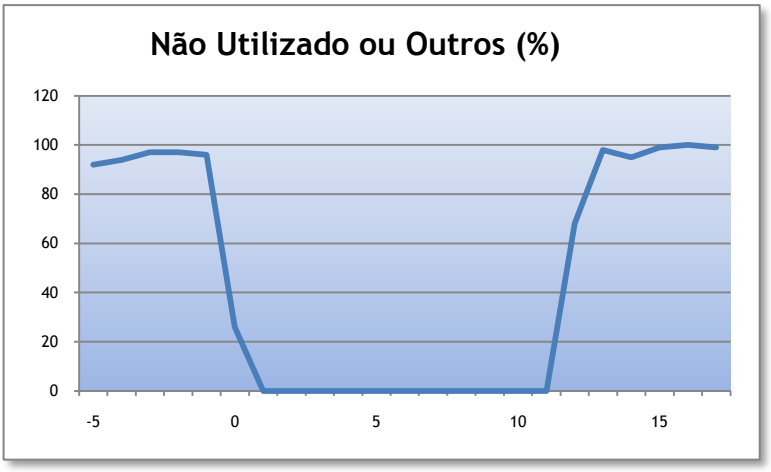

Figura 71 - Teste F: Gráfico da percentagem de não utilização do processador

Como era espectável, face aos resultados obtidos anteriormente, durante o decorrer do teste não existiram recursos livres - Figura 71. Este foi o primeiro teste em que o *2ndComing* necessitou de todos os recursos disponíveis para realizar as suas acções.

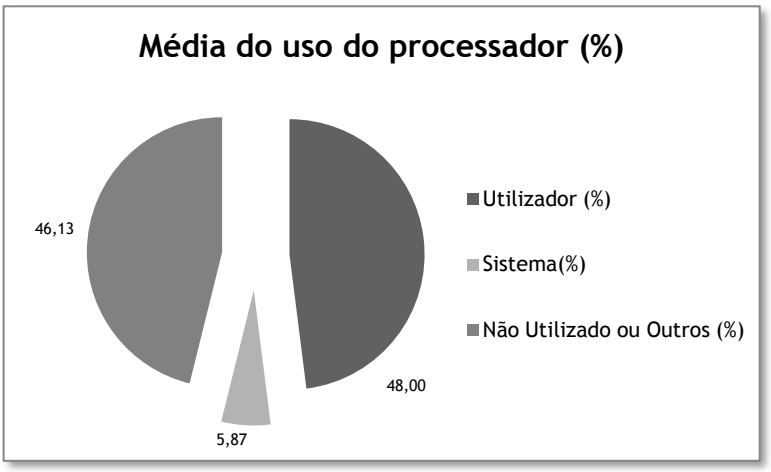

Figura 72 - Teste F: Gráfico da média do uso do processador

Pela observação da Figura 72, pode-se concluir que os processos associados ao utilizador foram os que mais usaram, em média, os recursos em todo o teste. Este valor seria bastante mais elevado no caso de os dados apresentados fossem referentes apenas ao tempo de teste uma vez que durante este tempo não havia recursos livres.

Porém, como os resultados são mais abrangentes, existe também uma grande percentagem de recursos não utilizados: 46%.

Mais uma vez, a menor fatia diz respeito aos processos relacionados com o sistema.

5.7 Teste G: Tempo de criação de Ruas de Cidades

Este teste encontra-se dividido em duas partes. Na primeira parte é medido o tempo necessário para que o 2ndComing RoadGenerator envie toda a informação necessária para criar as estradas de uma zona. Na segunda parte é determinado o tempo necessário para transformar toda a informação recebida em conteúdo no mundo virtual.

5.7.1 Criação de ruas do Porto

Para este primeiro teste foi escolhida uma pequena zona da cidade do Porto correspondente à figura apresentada abaixo:

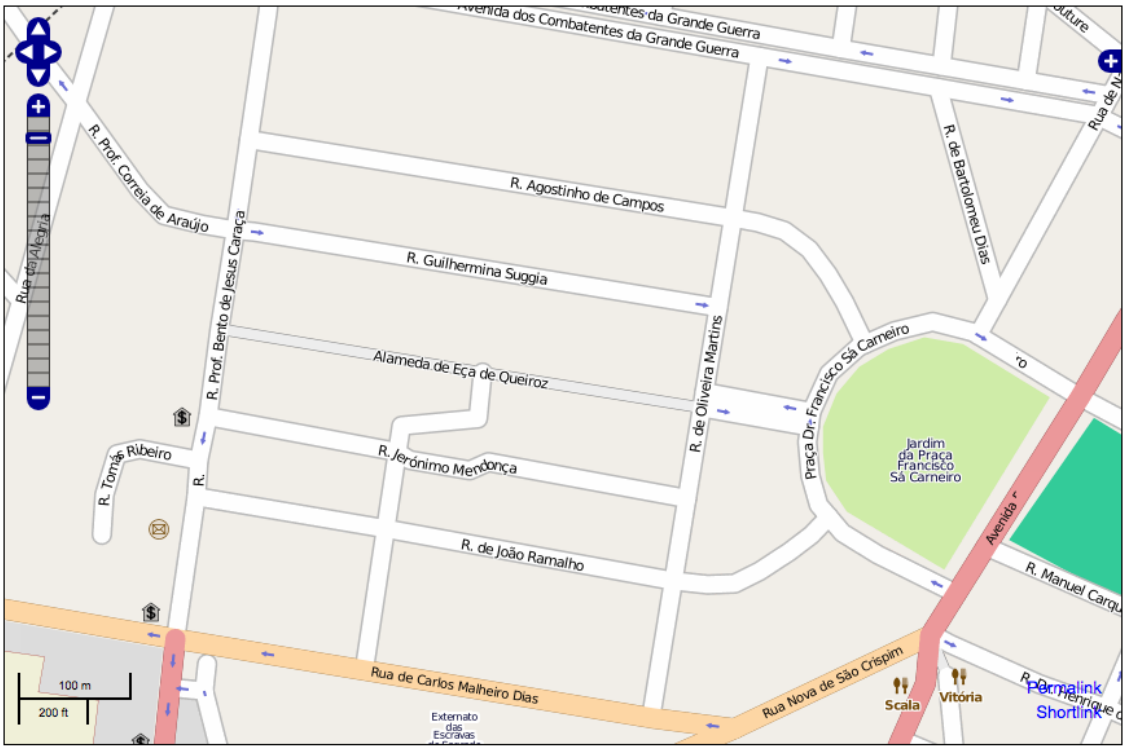

Figura 73 - Teste G: Imagem do OSM da zona do porto gerada no 2ndComing

Como se pode observar na Figura 73, o número de estradas é bastante reduzido e as que existem são praticamente todas rectas pelo que se espera que o tempo de criação desta zona seja bastante reduzido, tanto para o envio da informação como para a geração da mesma no mundo virtual.

5.7.1.1 Envio da informação

```
[2ndComing] Starting time: 6/19/2009 11:19:39 PM
[2ndComing] Stop time: 6/19/2009 11:20:20 PM
[2ndComing] 00:00:40.1926790
```

Conclui-se então que a informação demorou 40segundos a ser completamente enviada para o 2ndComing. O teste seguinte mostra quanto tempo é necessário para transformar esta informação em conteúdo.

5.7.1.2 Transformação da informação em conteúdo

```
[2ndComing] Startingtime: 6/19/2009 11:27:46 PM
[2ndComing] Stop time: 6/19/2009 11:28:05 PM
[2ndComing] 00:00:19.0705640
```

O tempo da transformação da informação obtida em conteúdo no mundo virtual foi cerca de metade do envio da informação. Isto porque nem toda a informação recebida pelo 2ndComing será depois usada para gerar conteúdo.

Por fim, pode-se realizar uma estimativa de quantos *prims* foram gerados pelo tempo total do teste. Visto que esta funcionalidade usa o comando *createPrim* do teste A, pode-se usar os resultados obtidos nesse teste, nomeadamente o tempo médio de criação de um *prim* para fazer esta estimativa. Assim sendo, chega-se à conclusão que foram criados cerca de 182 *prims* para gerar a zona do Porto deste teste.

5.7.2 Criação de ruas de Vila Nova de Gaia

Este teste será bastante mais intenso que o anterior no sentido em que a zona escolhida é bastante mais exigente a nível de densidade de estradas - Figura 74.

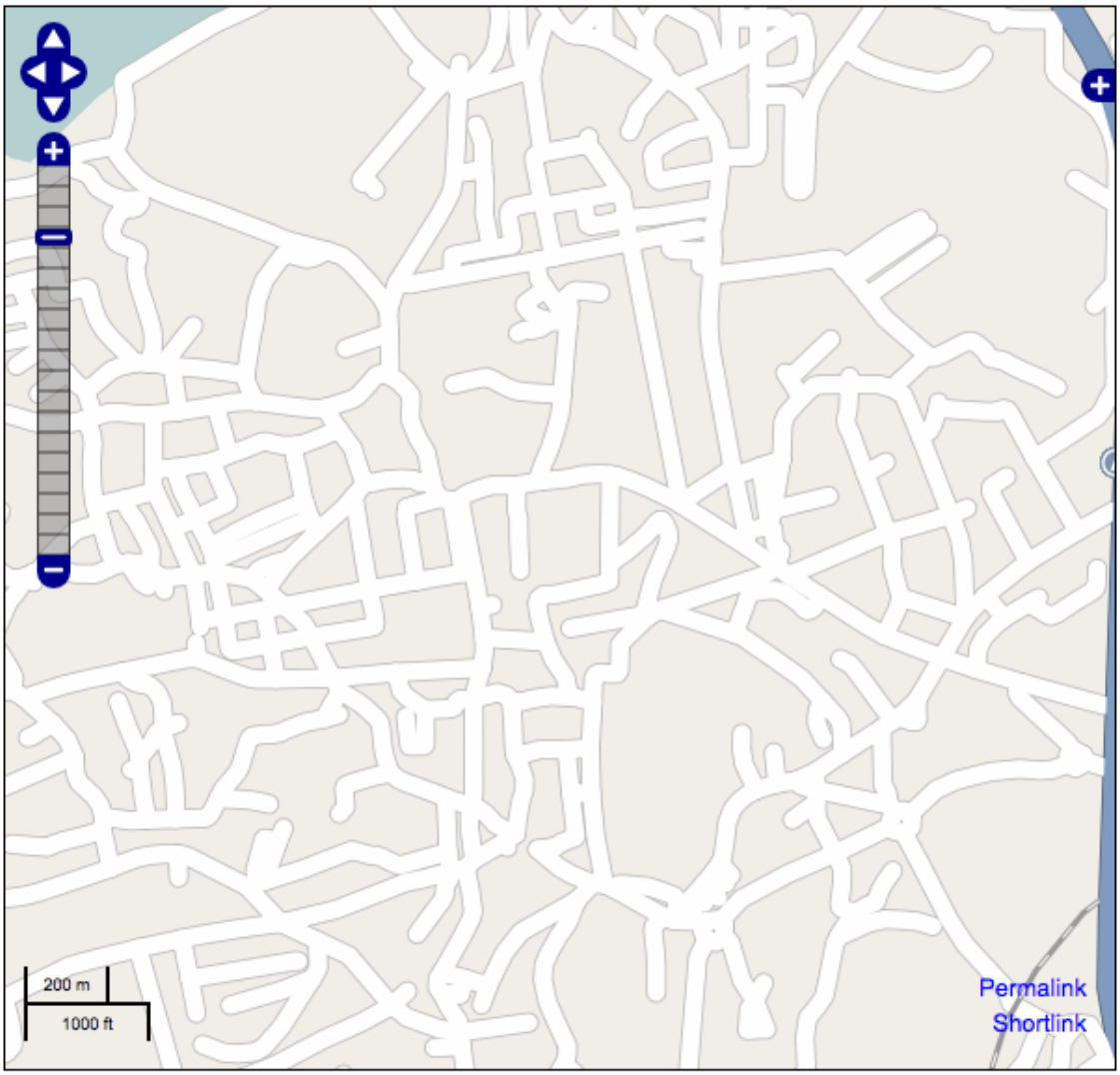

Figura 74 – Teste G – Imagem OSM da zona de V. N. De Gaia gerada no *2ndComing*

Como se pode observar na Figura 74, não existe comparação possível em termos de densidade de estradas com a zona do teste anterior. Espera-se, portanto, um tempo de criação bastante superior também, principalmente durante o processo de transmissão de toda a informação do *RoadGenerator* para o *2ndComing*.

5.7.2.1 Envio da informação

```
[2ndComing] Startingtime: 6/20/2009 12:39:01 AM
[2ndComing] Stop time: 6/20/2009 12:45:56 AM
[2ndComing] 00:06:54.7813080
```

Como era esperado, o tempo de envio da informação foi bastante superior ao do teste anterior. Sete minutos é bastante tempo para o envio da informação mesmo tendo em conta a alta concentração de estradas da zona.

5.7.2.2 Transformação da informação em conteúdo

```
[2ndComing] Startingtime: 6/20/2009 12:55:26 AM
[2ndComing] Stop time: 6/20/2009 12:57:22 AM
[2ndComing] 00:01:56.5194630
```

Ao contrário do resultado obtido no teste anterior, quase dois minutos para a transformação da informação em conteúdo é um tempo bastante aceitável, apesar de haver sempre lugar para aumentar a eficácia do 2ndComing. O tempo da transformação da informação em conteúdo é menor que um terço do tempo de envio da informação.

Usando os resultados do teste A para determinar a estimativa de *prims* criados conclui-se que foram criados cerca de 1110 *prims* para gerar a zona de teste da cidade de Vila Nova de Gaia. Apesar de um elevado tempo de espera, a quantidade de *prims* gerados neste teste foi quase seis vezes maior que a do teste anterior.

5.8 Análise de resultados

Os resultados obtidos nos testes anteriores dizem muito acerca do *2ndComing*.

A primeira conclusão a tirar é a de que o que demora mais no processo de geração de conteúdo são as trocas de mensagens. Pelos resultados do teste A, em que cada *prim* era gerado com o envio de uma mensagem, notou-se um fraco uso dos recursos do processador quando comparado, por exemplo, com o teste B em que somente existiu uma troca de mensagens e o resto do teste baseou-se em processamento.

Este facto é particularmente visível quando comparando o teste A com o teste D. No teste A, por cada *prim* criado é enviada uma mensagem enquanto no teste D é enviada uma mensagem por cada objecto complexo. No fundo, isto quer dizer que no teste D, por cada mensagem são criados cinco *prims*.

Tabela 27: Comparação dos tempos dos testes A e D

Teste	Tempo Total	Tempo por prim
Teste A - criar 1 prim	105ms	105ms
Teste D - criar 5 prims	107ms	21.4ms

Analisando os dados da Tabela **27** rapidamente se conclui que enquanto no teste A se obteve uma média de 105milissegundos por cada *prim* gerado, no teste B o resultado foi de 107milissegundos por cada objecto criado. Isto quer dizer que por cada *prim* criado, o teste D apenas demorou 21.4milissegundo contra os 105milissegundos do teste A. Está aqui bem demonstrado o *overhead* temporal que as mensagens trazem à criação de conteúdo.

Em termos de consumo de memória, em praticamente todos os testes notou-se um ligeiro aumento do consumo com a realização dos testes. Porém, a maior parte, mais de 95% do total da memória, estava já ocupada na altura do teste.

Os testes relativos ao movimento de *prims*, testes B e E, são testes com alto teor de processamento e servem apenas para mostrar como se comporta o *OpenSimulator* e o processador da máquina onde este está a ser executado. Pelos resultados obtidos, apenas se notou alguma perda de fluidez no mundo virtual nos testes de remoção de conteúdo, ou seja, testes C e F. Estes testes requerem muito processamento pois além de ser necessário remover os objectos da cena virtual é também necessário apagá-los da base de dados do *OpenSimulator* que acumula alguns objectos e depois os apaga simultaneamente de forma a aumentar a eficiência. Porém, quando o *OpenSimulator* executa este processo, em acumulação com os testes que estão a ser realizados, o consumo dos recursos do processador é praticamente total.

Por último, o teste de controlo de tempo da geração de estradas, teste G, mostra que o processo pode, em alguns casos, ser muito demorado. Quanto mais densa for a área que se deseja criar, mais tempo esta levará a ser criada. O resultado dos testes permitiu observar a grande disparidade de tempos necessários para os dois processos que envolvem a criação de estradas: a angariação de informação necessária e o uso da informação para criar conteúdo. O primeiro processo demora bastante mais tempo que o segundo visto toda a informação da zona ser enviada por mensagens HTTP para o *2ndComing*. O segundo processo apenas usa a informação necessária de toda aquela que foi compilada no primeiro para gerar conteúdo. O tempo excessivo gasto no envio de todas as informações disponíveis no primeiro processo acabará posteriormente por ser compensado quando outros geradores de conteúdo, como o *BusGenerator*, usarem a informação angariada no processo de criação de ruas para criarem os seus conteúdos. A adição futura de novos geradores de conteúdo poderá também usar esta informação.

Capítulo 6

Conclusão

O tema deste projecto é, sem qualquer dúvida, bastante vasto e vago. As possibilidades de escolha para o conteúdo que se deveria começar por gerar são praticamente infinitas e num projecto limitado no tempo como este, é necessário escolher apenas uma pequena parte dessas. A escolha recaiu sobre a geração de estradas, simulação de movimento de autocarros e objectos com algum grau de complexidade. Com estes três conteúdos tipos de conteúdo distintos consegue-se gerar conteúdo estático e dinâmico.

Para facilitar a futura adição de outros geradores de conteúdo o *software* foi desenvolvido por módulos, deixando também bastante espaço para a interligação da informação que diferentes módulos transmitem através da centralidade conferida ao *2ndComing*.

Os resultados dos testes realizados permitem concluir que o *2ndComing* não é muito exigente em termos de consumo de recursos da máquina onde está instalado. Como se viu anteriormente, apenas os testes que exigiam muita capacidade de processamento é que levaram o processador aos limites das suas capacidades. Os únicos testes que conseguiram este feito foram os de remoção de objectos da cena. Esta funcionalidade do *2ndComing* será utilizada em ocasiões raras e especiais, pelo que, a momentânea perda de fluidez no mundo virtual não será muito sentida.

As trocas de mensagens entre os módulos são o principal factor no consumo de tempo. Tempo esse que é cada vez mais indispensável para os utilizadores deste tipo de *software*. O processo de troca de mensagens realizado pelo *2ndComing RoadGenerator* é muito intenso e demorado. Porém, as estradas são uma parte fundamental para uma simulação num mundo virtual e a quantidade de conteúdo que se poderá gerar para elas é muito variada. Neste momento apenas o *2ndComing BusGenerator* aproveita a informação anterior, diminuindo assim de forma drástica a quantidade de informação que precisa de enviar. No futuro, outros

módulos poderão utilizar essa mesma informação e assim compensar o tempo gasto na primeira fase pelo *RoadGenerator*.

A instabilidade do código do *OpenSimulator* tornou-se, durante o tempo do projecto, uma contrariedade. As constantes evoluções mexeram em algumas secções importantes que estavam a ser utilizadas neste projecto. Por outro lado, a falta de documentação das funções suportadas pelo *OpenSim*, bem como as funcionalidades por implementar que de certa forma não ajudaram à evolução do *2ndComing* foram outras das adversidades encontradas. Um exemplo dessas funcionalidades não implementadas é a função *moveToTarget(targetCoords, timeToReachTarget)*. Esta função seria óptima para ser usada nas simulações de autocarros, porém não foi possível colocá-la operacional.

6.1 Trabalho Futuro

É complicado colocar um limite à quantidade de conteúdo que se poderá criar para o *2ndComing*. A abrangência da informação disponível para todos hoje em dia é demasiado grande, permitindo possibilidades ilimitadas para gerar conteúdo. Assim, um dos pontos futuros deste projecto não poderá deixar de passar pela adição de novos geradores.

Actualmente, o *2ndComing* apenas é capaz de gerar ruas a duas dimensões não tendo informações de relevo do terreno. À medida que novos serviços, abrangendo estas informações sejam disponibilizados e o acesso a esta informação seja livre, outro ponto a melhorar seria a adição de relevo ao terreno no mundo virtual.

A geração de objectos complexos por parte do *2ndComing* não é a melhor, pelo menos, não a que se desejaria. No futuro, não se pode deixar de pensar em introduzir uma maior resolução nos objectos, isto é, quantidade de *prims* que constituem cada objecto. Implementar uma forma mais fácil e eficaz de gerar os objectos complexos é também um dos possíveis trabalhos futuros. Este ponto pode passar pelo uso da interface de construção do *OpenSimulator* / *Second Life* como ferramenta para a construção e edição dos objectos. Posteriormente, a adição de uma base de dados para guardar a informação dos objectos completos criados no *2ndComing* de forma a que possam ser carregados a partir da base de dados e não de memória, como actualmente.

O uso do protocolo HTTP para as trocas de mensagens entre módulos *2ndComing*, um dos protocolos mais usados por *softwares* semelhantes neste meio, permite que se pense na interactividade do *2ndComing* com outros simuladores de mundos virtuais. Um dos possíveis e mais forte candidato a ser suportado pelo *2ndComing* seria o *Google Earth* que, como já se viu, cada vez é mais apoiado quer por utilizadores quer por novas funcionalidades.

Outro aspecto do *2ndComing* a ser revisto no futuro é a sua interface. As interfaces por linha de comandos são cada vez menos usadas e justificar-se-ia investir numa interface mais *standard* e apelativa esteticamente.

Referências

[1] AbuSalehMd. MahfujurRahman, MohamadEid, AbdulmotalebElSaddik. "KissMe: Bringing Virtual Events to the Real World". *VECIMS 2008 – IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*. Istambul, Turkey, 14-16 July 2008.

[2] Mike Macedonia. "Generation 3D: Living in Virtual Worlds". *Computer. Page(s): 99-101. October 2007. IEEE.*

[3] Chris Edwards. Another World. *Engineering&Technology. Page(s): 28-32.*

[4] Sujoyini Mandal, Ee-Peng Lim. "Second Life: Limits Of Creativity Or Cyber Threat?" *2008 IEEE Conference on Technologies for Homeland Security. Page(s): 498-503.*

[5] Internet Game Exchange. [online] http://igxe.com. Acedido em Junho 2009.

[6] Qing Zhu, Tao Wang, Yufu Jia. "Second Life: A New Platform for Education". *Information Technologies and Applications in Education. First IEEE International Symposium on 23-25 November 2007. Page(s): 201-204.*

[7] Donna Russell, Molly Davies, Iris Totten. "GEOWORLDS: Utilizing Second Life to Develop Advanced Geosciences Knowledge". *Second IEEE International Conference on Digital Games and Intelligent Toys Based*. 2008.

[8] Hyungsung Park, Bokjin Shin, Xiangzhe Cui, Jihyun Hwang. "What will happen to field trips? Beyond the classroom". *Second IEEE International Conference on Digital Games and Intelligent Toys Based Education.*

[9] Linden Lab. Second Life. [online] http://secondlife.com/. Acedido em Junho 2009.

[10] Google. [online] http://www.google.com/corporate. Acedido em Junho 2009.

[11] Fortune Magazine, artigo publicado pela CNN. [online] http://money.cnn.com/magazines/fortune/bestcompanies/2007/full_list/. Acedido em Junho 2009.

[12] Millward Brown Group. [Online] http://www.millwardbrown.com/Sites/optimor/Media/Pdfs/en/BrandZ/BrandZ-2008-Report.pdf. Acedido em Junho 2009.

[13] Lively. [online] http://www.lively.com. Acedido em Fevereiro 2009.

[14] Google Earth [online] http://earth.google.com/. Acedido em Junho 2009.

[15] Google Mars [online] http://www.google.com/mars/. Acedido em Junho 2009.

[16] Google Sky [online] http://www.google.com/sky/. Acedido em Junho 2009.

[17] Google Ocean [online] http://www.justmagic.com/GM-GE.html. Acedido em junho 2009.

[18] Google Moon. [online] http://www.google.com/moon/. Acedido em Junho 2009.

[19] Google Earth API [online] http://code.google.com/apis/earth/. Acedido em Junho 2009.

[20] Google Maps [online] http://maps.google.com. Acedido em Junho 2009.

[21] Microsoft [online] http://www.microsoft.com. Acedido em Junho 2009.

[22] Microsoft. *Xbox Live* [online] http://www.xbox.com. Acedido em Junho 2009.

[23] Microsoft. *Project Natal*. [online] http://www.xbox.com/en-US/live/projectnatal/. Acedido em Junho 2009.

[24] OpenStreetMap. [online] http://www.openstreetmap.org. Acedido em Junho 2009.

[25] 3D CAD Browser. [online] http://www.3dcadbrowser.com/.Acedido em Fevereiro 2009.

[26] Graphics and Gaming Group. Institute of Technology Blanchardstown. Automatic Building Generation. 2006-2008. [online] http://www.gamesitb.com/automaticbuildings.html. Acedido em Fevereiro 2009.

[27] W. J. Tam, Liang Zhang. "3D-TV Content Generation: 2D-to-3D Conversion". *IEEE International Conference on Multimedia and Expo*. 2006.

[28] Greuter Stefan, Parker Jeremy, Stewart Nigel, Leach Geoff. "Real-time Procedural generation of 'pseudo infinite' cities". *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. 2003.

[29] OpenSimulator. [online] http://opensimulator.org/. Acedido em Junho 2009.

[30] OSGrid. [online] http://osgrid.org. Acedido em Junho 2009.

[31] Gupta, S. "Linear quaternion equations with application to spacecraft attitude propagation". *Aerospace Conference. Proceedings., IEEE. Volume 1, 21-28 March 1998 Page(s): 69 - 76.*

Apêndice A

Tabelas de Resultados

Tabela 28: Apêndice A - Monitorização de Memória do teste A

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	35504	999860	96,57
-4	35504	999860	96,57
-3	35504	999860	96,57
-2	35504	999860	96,57
-1	35496	999868	96,57
0	32760	1002604	96,84
1	31844	1003520	96,92
2	31828	1003536	96,93
3	31808	1003556	96,93
4	31828	1003536	96,93
5	31672	1003692	96,94
6	31412	1003952	96,97
7	30544	1004820	97,05
8	29524	1005840	97,15
9	28988	1006376	97,20
10	28952	1006412	97,20
11	28584	1006780	97,24
12	27576	1007788	97,34
13	26608	1008756	97,43
14	25840	1009524	97,50
15	24928	1010436	97,59
16	24064	1011300	97,68
17	23180	1012184	97,76
18	23164	1012200	97,76
19	23124	1012240	97,77
20	23148	1012216	97,76

21	23124	1012240	97,77
22	23108	1012256	97,77
23	23108	1012256	97,77
24	22844	1012520	97,79
25	21960	1013404	97,88
26	21684	1013680	97,91
27	21660	1013704	97,91
28	21644	1013720	97,91
29	21636	1013728	97,91
30	21504	1013860	97,92
31	21084	1014280	97,96
32	19984	1015380	98,07
33	19720	1015644	98,10
34	19816	1015548	98,09
35	19772	1015592	98,09
36	19764	1015600	98,09
37	19776	1015588	98,09
38	19752	1015612	98,09
39	19744	1015620	98,09
40	19744	1015620	98,09
41	19700	1015664	98,10
42	19816	1015548	98,09
43	19792	1015572	98,09
44	19776	1015588	98,09
45	19776	1015588	98,09
46	19732	1015632	98,09
47	19752	1015612	98,09
48	19584	1015780	98,11
49	19800	1015564	98,09
50	19664	1015700	98,10
51	19668	1015696	98,10
52	19624	1015740	98,10
53	19644	1015720	98,10
54	19584	1015780	98,11
55	19604	1015760	98,11
56	19684	1015680	98,10
57	19644	1015720	98,10
58	19668	1015696	98,10
59	19620	1015744	98,11
60	19484	1015880	98,12
61	19504	1015860	98,12

62	19536	1015828	98,11
63	19532	1015832	98,11
64	19552	1015812	98,11
65	19504	1015860	98,12
66	19512	1015852	98,12
67	19512	1015852	98,12
68	19468	1015896	98,12
69	19568	1015796	98,11
70	19552	1015812	98,11
71	19508	1015856	98,12
72	19500	1015864	98,12
73	19520	1015844	98,11
74	19472	1015892	98,12
75	19568	1015796	98,11
76	18912	1016452	98,17
77	18764	1016600	98,19
78	25248	1010116	97,56
79	25140	1010224	97,57
80	24636	1010728	97,62
81	24644	1010720	97,62
82	24600	1010764	97,62
83	24500	1010864	97,63
84	24116	1011248	97,67
85	23252	1012112	97,75
86	22384	1012980	97,84
87	21380	1013984	97,94
88	20480	1014884	98,02
89	20544	1014820	98,02
90	20528	1014836	98,02
91	20132	1015232	98,06
92	19256	1016108	98,14
93	18356	1017008	98,23
94	17696	1017668	98,29
95	17084	1018280	98,35
96	16820	1018544	98,38
97	22272	1013092	97,85
98	21992	1013372	97,88
99	21656	1013708	97,91
100	21656	1013708	97,91
101	21640	1013724	97,91
102	21252	1014112	97,95

103	20404	1014960	98,03
104	19484	1015880	98,12
105	18628	1016736	98,20
106	18688	1016676	98,20
107	18688	1016676	98,20
108	18688	1016676	98,20
109	18688	1016676	98,20
110	18688	1016676	98,20

Tabela 29: Apêndice A - Monitorização do processador no teste A

Tempo (s)	Utilizador (%)	Sistema (%)	Não Usado ou Outros
-5	3	0	97,00
-4	4	1	95,00
-3	2	0	98,00
-2	6	1	93,00
-1	0	1	99,00
0	16	5	79,00
1	13	5	82,00
2	9	4	87,00
3	12	3	85,00
4	11	4	85,00
5	7	5	88,00
6	9	4	87,00
7	10	4	86,00
8	9	3	88,00
9	12	5	83,00
10	8	5	87,00
11	10	3	87,00
12	10	5	85,00
13	8	4	88,00
14	8	6	86,00
15	8	4	88,00
16	8	7	85,00
17	10	3	87,00
18	11	4	85,00
19	9	4	87,00
20	12	2	86,00
21	17	3	80,00
22	9	5	86,00

23	14	2	84,00
24	9	4	87,00
25	11	3	86,00
26	9	7	84,00
27	13	2	85,00
28	13	3	84,00
29	11	3	86,00
30	12	5	83,00
31	12	2	86,00
32	10	6	84,00
33	12	4	84,00
34	14	5	81,00
35	8	4	88,00
36	13	3	84,00
37	9	5	86,00
38	10	4	86,00
39	9	4	87,00
40	13	4	83,00
41	11	5	84,00
42	13	3	84,00
43	8	4	88,00
44	11	5	84,00
45	8	2	90,00
46	9	4	87,00
47	12	3	85,00
48	10	6	84,00
49	12	3	85,00
50	12	5	83,00
51	14	3	83,00
52	12	4	84,00
53	12	3	85,00
54	12	4	84,00
55	11	3	86,00
56	14	5	81,00
57	13	4	83,00
58	13	3	84,00
59	12	3	85,00
60	13	4	83,00
61	12	4	84,00
62	14	3	83,00
63	10	4	86,00

64	11	4	85,00
65	9	4	87,00
66	11	5	84,00
67	13	3	84,00
68	11	4	85,00
69	15	2	83,00
70	10	8	82,00
71	12	3	85,00
72	9	4	87,00
73	10	4	86,00
74	13	3	84,00
75	10	4	86,00
76	38	15	47,00
77	55	23	22,00
78	64	29	7,00
79	70	24	6,00
80	65	27	8,00
81	30	9	61,00
82	10	6	84,00
83	12	4	84,00
84	14	4	82,00
85	12	4	84,00
86	12	5	83,00
87	13	4	83,00
88	10	5	85,00
89	15	4	81,00
90	11	6	83,00
91	13	2	85,00
92	12	5	83,00
93	11	3	86,00
94	17	6	77,00
95	32	11	57,00
96	63	30	7,00
97	70	24	6,00
98	70	27	3,00
99	66	19	15,00
100	15	5	80,00
101	13	3	84,00
102	12	5	83,00
103	16	4	80,00
104	12	4	84,00

105	12	2	86,00
106	5	1	94,00
107	4	0	96,00
108	4	1	95,00
109	4	0	96,00
110	4	0	96,00

Tabela 30: Apêndice A - Monitorização da memória do teste B

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	33828	1001536	96,73
-4	33828	1001536	96,73
-3	33828	1001536	96,73
-2	33820	1001544	96,73
-1	33828	1001536	96,73
0	33608	1001756	96,75
1	33472	1001892	96,77
2	33344	1002020	96,78
3	33856	1001508	96,73
4	33892	1001472	96,73
5	36264	999100	96,50
6	36836	998528	96,44
7	36452	998912	96,48
8	36172	999192	96,51
9	35616	999748	96,56
10	35104	1000260	96,61
11	34792	1000572	96,64
12	34448	1000916	96,67
13	34324	1001040	96,68
14	33996	1001368	96,72
15	33480	1001884	96,77
16	33220	1002144	96,79
17	32764	1002600	96,84
18	32316	1003048	96,88
19	31632	1003732	96,94
20	31532	1003832	96,95
21	31012	1004352	97,00
22	30764	1004600	97,03
23	30296	1005068	97,07
24	29800	1005564	97,12

25	29392	1005972	97,16
26	29180	1006184	97,18
27	28648	1006716	97,23
28	28292	1007072	97,27
29	27932	1007432	97,30
30	27532	1007832	97,34
31	27140	1008224	97,38
32	26808	1008556	97,41
33	26428	1008936	97,45
34	25912	1009452	97,50
35	25576	1009788	97,53
36	25196	1010168	97,57
37	24464	1010900	97,64
38	24196	1011168	97,66
39	23732	1011632	97,71
40	22844	1012520	97,79
41	22564	1012800	97,82
42	22072	1013292	97,87
43	21804	1013560	97,89
44	21316	1014048	97,94
45	20824	1014540	97,99
46	20484	1014880	98,02
47	20120	1015244	98,06
48	19724	1015640	98,09
49	19348	1016016	98,13
50	18712	1016652	98,19
51	18208	1017156	98,24
52	17888	1017476	98,27
53	17392	1017972	98,32
54	16860	1018504	98,37
55	16376	1018988	98,42
56	15904	1019460	98,46
57	15664	1019700	98,49
58	15168	1020196	98,54
59	14732	1020632	98,58
60	14648	1020716	98,59
61	14008	1021356	98,65
62	14404	1020960	98,61
63	14080	1021284	98,64

Tabela 31: Apêndice A - Monitorização do processador no teste B

Tempo (s)	Utilizador (%)	Sistema (%)	Não utilizado e Outros (%)
-5	5	0	95
-4	5	0	95
-3	4	0	96
-2	3	1	96
-1	3	0	97
0	46	15	39
1	66	26	8
2	75	23	2
3	69	20	11
4	72	26	2
5	70	24	6
6	70	24	6
7	68	27	5
8	72	22	6
9	72	21	7
10	71	23	6
11	68	30	2
12	72	26	2
13	73	24	3
14	71	22	7
15	72	24	4
16	71	24	5
17	68	24	8
18	76	22	2
19	72	22	6
20	71	21	8
21	70	25	5
22	77	17	6
23	71	21	8
24	73	20	7
25	80	18	2
26	72	17	11
27	78	20	2
28	69	19	12
29	71	24	5
30	75	24	1
31	76	23	1

32	71	19	10
33	79	19	2
34	72	24	4
35	67	22	11
36	67	25	8
37	68	25	7
38	65	22	13
39	72	21	7
40	72	21	7
41	75	21	4
42	79	20	1
43	77	21	2
44	78	15	7
45	73	23	4
46	74	23	3
47	71	26	3
48	75	15	10
49	72	23	5
50	84	15	1
51	68	22	10
52	72	23	5
53	70	24	6
54	74	21	5
55	75	21	4
56	72	22	6
57	71	21	8
58	73	22	5
59	78	18	4
60	76	16	8
61	72	23	5
62	67	25	8
63	5	3	92

Tabela 32: Apêndice A - Monitorização da memória no teste C

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	39252	996112	96,21
-4	39252	996112	96,21
-3	39252	996112	96,21

-2	39252	996112	96,21
-1	39260	996104	96,21
0	39008	996356	96,23
1	38964	996400	96,24
2	38972	996392	96,24
3	38980	996384	96,24
4	38980	996384	96,24
5	38980	996384	96,24
6	38944	996420	96,24
7	38980	996384	96,24
8	38840	996524	96,25
9	38856	996508	96,25
10	38820	996544	96,25
11	38840	996524	96,25
12	38872	996492	96,25
13	38856	996508	96,25
14	38872	996492	96,25
15	38836	996528	96,25
16	38732	996632	96,26
17	38748	996616	96,26
18	38748	996616	96,26
19	38748	996616	96,26
20	38756	996608	96,26
21	38580	996784	96,27
22	38616	996748	96,27
23	38632	996732	96,27
24	38596	996768	96,27
25	38624	996740	96,27
26	38584	996780	96,27
27	38608	996756	96,27
28	38624	996740	96,27
29	38624	996740	96,27
30	38484	996880	96,28
31	38476	996888	96,28
32	38500	996864	96,28
33	38464	996900	96,28
34	38360	997004	96,30
35	38128	997236	96,32
36	38128	997236	96,32
37	38344	997020	96,30
38	38376	996988	96,29

39	38344	997020	96,30
40	38196	997168	96,31
41	38220	997144	96,31
42	38252	997112	96,31
43	38128	997236	96,32
44	37864	997500	96,34
45	37880	997484	96,34
46	37864	997500	96,34
47	37836	997528	96,35
48	37880	997484	96,34
49	37880	997484	96,34
50	37872	997492	96,34
51	37872	997492	96,34
52	37872	997492	96,34
53	37872	997492	96,34
54	37872	997492	96,34
55	37864	997500	96,34
56	38864	996500	96,25
57	38856	996508	96,25
58	38856	996508	96,25
59	38856	996508	96,25

Tabela 33: Apêndice A - Monitorização do processador no teste C

Tempo (s)	Utilizador (%)	Sistema (%)	Não Utilizado e Outros (%)
-5	4	1	95
-4	6	1	93
-3	5	2	93
-2	5	1	94
-1	5	1	94
0	35	14	51
1	71	27	2
2	73	25	2
3	25	24	51
4	74	23	3
5	68	30	2
6	69	27	4
7	70	27	3
8	74	23	3
9	74	24	2

10	69	27	4
11	74	22	4
12	73	24	3
13	71	25	4
14	71	24	5
15	69	28	3
16	58	25	17
17	2	1	97
18	25	10	65
19	71	25	4
20	65	32	3
21	71	28	1
22	71	26	3
23	71	27	2
24	74	22	4
25	70	27	3
26	70	28	2
27	74	22	4
28	69	27	4
29	69	26	5
30	70	26	4
31	69	28	3
32	69	27	4
33	73	25	2
34	70	26	4
35	76	18	6
36	75	22	3
37	76	21	3
38	69	28	3
39	68	27	5
40	69	27	4
41	73	24	3
42	76	20	4
43	72	24	4
44	78	18	4
45	76	21	3
46	70	25	5
47	71	26	3
48	40	15	45
49	0	2	98
50	7	1	92

51	3	2	95
52	3	0	97
53	2	0	98
54	5	3	92
55	0	0	100
56	5	1	94
57	5	1	94
58	1	1	98
59	5	3	92

Tabela 34: Apêndice A - Monitorização da ocupação de memória no teste D

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	47288	988076	95,43
-4	47296	988068	95,43
-3	47296	988068	95,43
-2	47296	988068	95,43
-1	47296	988068	95,43
0	44548	990816	95,70
1	43752	991612	95,77
2	43752	991612	95,77
3	43768	991596	95,77
4	43744	991620	95,78
5	43348	992016	95,81
6	41536	993828	95,99
7	41528	993836	95,99
8	41004	994360	96,04
9	39784	995580	96,16
10	39784	995580	96,16
11	39668	995696	96,17
12	39668	995696	96,17
13	39668	995696	96,17
14	39668	995696	96,17
15	39428	995936	96,19
16	39436	995928	96,19
17	39436	995928	96,19
18	39436	995928	96,19
19	39444	995920	96,19
20	39436	995928	96,19
21	39476	995888	96,19

22	39476	995888	96,19

Tabela 35: Apêndice A - Monitorização do processador no teste D

Tempo (s)	Utilizador (%)	Sistema (%)	Não utilizado e Outros (%)
-5	7	1	92
-4	8	1	91
-3	2	0	98
-2	2	2	96
-1	1	0	99
0	19	3	78
1	15	5	80
2	11	3	86
3	11	5	84
4	13	6	81
5	11	3	86
6	15	6	79
7	12	5	83
8	14	4	82
9	15	4	81
10	12	7	81
11	10	2	88
12	3	2	95
13	3	0	97
14	2	1	97
15	4	0	96
16	4	1	95
17	4	0	96
18	3	1	96
19	3	0	97
20	8	2	90
21	8	1	91
22	8	1	91

Tabela 36: Apêndice A - Monitorização da memória no teste E

Tempo (s)	Memória Livre (KB)	Memória Usada (KB)	Memória Usada (%)
-5	43436	991928	95,80

-4	43444	991920	95,80
-3	43444	991920	95,80
-2	43436	991928	95,80
-1	43428	991936	95,81
0	40052	995312	96,13
1	40056	995308	96,13
2	40056	995308	96,13
3	39872	995492	96,15
4	39924	995440	96,14
5	39808	995556	96,16
6	39668	995696	96,17
7	39536	995828	96,18
8	39508	995856	96,18
9	41540	993824	95,99
10	41760	993604	95,97
11	41800	993564	95,96
12	42148	993216	95,93
13	41940	993424	95,95
14	41784	993580	95,96
15	41652	993712	95,98
16	41560	993804	95,99
17	41428	993936	96,00
18	41188	994176	96,02
19	41020	994344	96,04
20	41012	994352	96,04
21	40932	994432	96,05
22	40756	994608	96,06
23	40692	994672	96,07
24	40560	994804	96,08
25	40444	994920	96,09
26	40280	995084	96,11
27	40196	995168	96,12
28	40064	995300	96,13
29	40056	995308	96,13
30	40064	995300	96,13
31	40064	995300	96,13
32	40064	995300	96,13

Tabela 37: Apêndice A - Monitorização do processador no teste E

Tempo (s)	Utilizador (%)	Sistema (%)	Não Utilizado e Outros (%)
-5	3	0	97
-4	4	2	94
-3	5	1	94
-2	3	0	97
-1	5	2	93
0	19	5	76
1	8	1	91
2	10	1	89
3	28	8	64
4	35	11	54
5	39	24	37
6	48	18	34
7	42	24	34
8	46	20	34
9	49	20	31
10	52	16	32
11	51	19	30
12	48	17	35
13	46	22	32
14	52	22	26
15	44	22	34
16	48	19	33
17	50	16	34
18	49	23	28
19	51	17	32
20	46	21	33
21	48	25	27
22	49	19	32
23	47	23	30
24	46	26	28
25	49	21	30
26	46	19	35
27	49	18	33
28	14	6	80
29	4	1	95
30	15	2	83
31	5	2	93
32	4	1	95

Tabela 38: Apêndice A - Monitorização da memória no teste F

Tempo (s)	Memória Livre (KB)	Memória Ocupada (KB)	Memória Ocupada (%)
-5	45672	989692	95,59
-4	45680	989684	95,59
-3	45680	989684	95,59
-2	45680	989684	95,59
-1	45680	989684	95,59
0	43028	992336	95,84
1	45752	989612	95,58
2	45868	989496	95,57
3	45720	989644	95,58
4	45872	989492	95,57
5	45904	989460	95,57
6	45724	989640	95,58
7	45868	989496	95,57
8	46516	988848	95,51
9	46352	989012	95,52
10	46316	989048	95,53
11	46280	989084	95,53
12	46316	989048	95,53
13	46332	989032	95,53
14	46332	989032	95,53
15	46332	989032	95,53
16	46324	989040	95,53
17	46332	989032	95,53

Tabela 39: Apêndice A - Monitorização do processador no teste F

Tempo (s)	Utilizador (%)	Sistema(%)	Não Utilizado ou Outros (%)
-5	8	0	92
-4	6	0	94
-3	3	0	97
-2	3	0	97
-1	2	2	96
0	67	7	26
1	90	10	0

2	86	14	0
3	89	11	0
4	89	11	0
5	89	11	0
6	92	8	0
7	89	11	0
8	91	9	0
9	91	9	0
10	89	11	0
11	87	13	0
12	28	4	68
13	1	1	98
14	4	1	95
15	0	1	99
16	0	0	100
17	0	1	99