IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Projecto Fim de Curso

Alien Arena

Curso de Engenharia Informática de Gestão

07-09-2009

Realizado por:
Pedro Andrade nº042205797
Rui Rodrigues nº042205770

Orientado por:
Professor Fausto Mourato
Professora Cláudia Viana

Índice

1. Introdução

*O projecto final de curso pretende ser o elo de ligação entre os conhecimentos adquiridos ao longo do percurso académico e a realidade empresarial, através do desenvolvimento de trabalho em parceria com uma empresa da especialidade. Neste caso concreto, o projecto proposto pela empresa Elemento Digital, trata-se de um jogo de computador intitulado "*Alien Arena*".*

Com a elaboração deste documento, pretendemos explicitar a informação relativa à análise de especificações para a aplicação acima enunciada. Ao longo das várias secções deste relatório pretendemos realçar a lógica de desenvolvimento desta aplicação, começando por efectuar um levantamento dos requisitos, e modelando o sistema usando as ferramentas que consideramos mais apropriadas para o mesmo.

Este projecto consiste na criação de um jogo de computador, utilizando técnicas de programação 3D baseado em Frameworks *e/ou motores de jogo de suporte aos processos de computação gráfica, para desta forma ser possível a introdução de publicidade dinamicamente. Pretende-se que as tecnologias a utilizar tenham uma boa base de suporte à criação de jogos em rede através da internet.*

1.1. Apresentação Sumária do Projecto

Ultimamente no ramo dos videojogos em 3D, temos assistido a um grande crescimento do mercado no que diz respeito ao componente multi-jogador online. A possibilidade de podermos jogar com um indivíduo que esteja noutra localização, em qualquer momento do dia, através da Internet, tem conquistado muitos adeptos ao longo dos últimos tempos.

A maior parte deste tipo de jogos é suportada através de licenças pagas pelo cliente final, daí surgiu a ideia de introduzir uma componente de advertisement*, que poderá futuramente suportar o custo de desenvolvimento da aplicação.*

*Posto isto, o desafio consiste na construção de um jogo multi-jogador do tipo "*First Person Shooter*" em 3 dimensões, no qual os jogadores encarnam a personagem de* aliens*, cujo objectivo é eliminar o adversário o maior número de vezes possível num determinado período de tempo. Deverá ser possível o carregamento de imagens contendo publicidade para o mapa onde irá ocorrer a acção.*

Para ir de encontro aos pontos acima descritos, utilizar-se-ão ferramentas de última geração no que toca à área da computação gráfica, técnicas de modelação 3D e mecanismos de computação em redes de computadores.

1.2. Recursos

1.2.1. Recursos Humanos

No desenvolvimento deste projecto irão participar cinco pessoas:

Pedro Andrade	*Desenvolvimento da aplicação*
Rui Rodrigues	*Desenvolvimento da aplicação*
Prof. Fausto Mourato	*Coordenação na área de Informática*
Prof. Cláudia Viana	*Coordenação na área de Gestão*
Eng. João Morais	*Coordenação externa e definição de requisitos*

1.2.2. Recursos Temporais

Está previsto que o Projecto Fim de Curso tenha a duração de um semestre escolar, isto é, aproximadamente quatro meses.

1.2.3. Recursos Tecnológicos

Em termos de equipamento tecnológico, teve-se o cuidado de seleccionar tecnologias e ferramentas que não acrescentassem custos à realização da aplicação.

Na parte de hardware *utilizamos dois computadores com placa gráfica de memória dedicada e com ligação à Internet por banda larga.*

De software usamos o sistema operativo Microsoft Windows XP. Outras aplicações serão o Microsoft Visual Studio Express Edition 2008, Microsoft XNA Game Studio 3.0, Microsoft Word 2007, Microsoft PowerPoint 2007 e Microsoft Visio 2007.

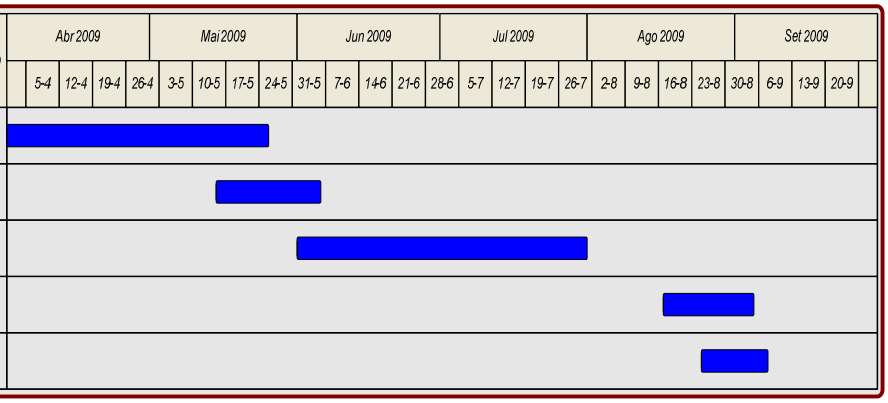
1.2.4. Recursos Monetários

Tratando-se de um trabalho de desígnio universitário, não existem quaisquer despesas financeiras a apontar.

1.3. Cronograma

De seguida expomos a informação relativa à distribuição de tempo pelas várias tarefas que nos foram propostas no âmbito da realização do Projecto Fim de Curso.

ID	Tarefa	Inicio	Termino	Duração
1	Requisitos	01-04-2009	25-05-2009	7,8s
2	Desenho	15-05-2009	05-06-2009	3,2s
3	Implementação	01-06-2009	31-07-2009	9s
4	Testes	17-08-2009	04-09-2009	3s
5	Documentação	25-08-2009	07-09-2009	2s



1.4. Modelo de desenvolvimento

Uma metodologia de desenvolvimento de software refere-se a uma estrutura para planear e controlar o processo de construção de um sistema de informação. Uma variedade de tais estruturas evoluiu ao longo dos anos, cada uma com as suas vantagens e fraquezas. Um sistema de modelo de desenvolvimento não é necessariamente adaptável para usar em todos os projectos. Cada modelo é mais apropriado para géneros específicos de projectos, baseando-se nas várias considerações técnicas, organizacionais, de projecto e equipa de trabalho.

São aceites vários métodos para combinar sistemas lineares e iterativos de metodologias de desenvolvimento, com o objectivo de reduzir o risco inerente ao projecto através da divisão do projecto em pequenos segmentos e disponibilizar facilidade a alterações durante o processo de desenvolvimento.

O modelo de desenvolvimento de software Incremental é um método em que o modelo é desenhado, implementado e testado incrementalmente até ao produto final. Envolvendo desenvolvimento e manutenção, o produto é definido como terminado quando cumpre todos os seus requerimentos. Este modelo combina elementos do modelo Cascata e a filosofia iterativa de Prototipagem.

O produto é decomposto num número de componentes, cada um dos quais são desenhados e construídos separadamente. Cada componente, ou build*, é entregue ao cliente quando completa, permitindo uma utilização parcial do produto e evita uma espera prolongada pelo desenvolvimento do projecto. Este modelo de software facilita o efeito de choque que existe ao entregar um produto completamente novo de uma só vez.*

Existem alguns problemas com este modelo. Um dos quais é o de cada build *ter de ser integrada com as* builds *prévias e em qualquer sistema que exista. A tarefa de divisão também não é trivial: se existirem poucas* builds *e estas degenerarem o modelo transforma-se no Construir e Corrigir (Pina, 2006). No entanto, se dividido em muitas* builds*, então muito pouco é adicionado a cada uma delas. E em casos mais complexos, existe mesmo a possibilidade em que a divisão em componentes não é possível.*

2. Enquadramento organizacional

2.1. A empresa

A Elemento Digital S.A. nasceu no ano de 2001, impulsionada por uma visão estratégica e de negócio partilhada pelos seus fundadores.

Instalados em Setúbal e em Lisboa em pleno Parque das Nações, a empresa conta actualmente com uma equipa de cerca de 21 colaboradores, desde Web Developers, Web-Designers, Designers e Flash Developers, Accounts, Marketeers e E-Marketeers. Independentemente da função que é desempenhada por cada um, incentiva-se que cada colaborador seja um verdadeiro elemento activo no desenvolvimento contínuo da empresa, co-habitando num ambiente familiar e descontraído (Elemento Digital S.A., 2009).

A gestão dos departamentos de produção, comercial e financeiro está tripartida pelos membros da Administração. Sendo a responsável do departamento de formação profissional a única que não representa a administração.

Organograma 1 – (Fonte: Elemento Digital)

2.2. Serviços

Actuando no habitat da Internet, envolve-se em todas as fases da comunicação, desde a fase da definição de estratégia, criatividade, produção e activação, até à análise e optimização de resultados, os quais acredita serem factores críticos, de sucesso de qualquer empresa.

2.3. Análise do Meio Envolvente

Sobrevivência de qualquer empresa depende, em primeiro lugar, da sua capacidade de interacção com o meio envolvente. A permanente evolução dos mercados e das indústrias gera múltiplas oportunidades e ameaças potenciais a que as empresas têm de saber dar resposta. O competidor que for menos rápido e eficaz a adequar-se às tendências do meio envolvente corre por isso o risco de perder clientes e, em última análise, de sair do negócio.

A análise do meio envolvente deve ser feita a dois níveis: o meio envolvente contextual (geral), comum a todas as organizações, e o meio envolvente transaccional, específico para cada indústria.

2.3.1. Contexto Tecnológico

Traduz o progresso técnico da sociedade. As principais variáveis são: inovações tecnológicas, inovações de processo, protecção de patentes, incentivos do governo e infra-estruturas.

Os sectores relacionados com as TI (Tecnologias de Informação) são áreas muito influenciadas pelas inovações tecnológicas (novas linguagens de programação; novos equipamentos electrónicos). Este sector está sempre preocupado com a tecnologia pois constantemente são confrontados com os avanços tecnológicos que cada vez mais simplificam e enriquecem os processos, fornecendo novas oportunidades e ideias que podem ser uma mais-valia para os negócios. Estes avanços além de serem

directamente vantajosos para as empresas que actuam neste sector nomeadamente ao nível de prazos de desenvolvimento e qualidade no trabalho realizado, tornam este sector mais atractivo para que possa satisfazer os clientes, que cada vez mais sobem o nível de exigência.

2.3.2. Contexto Económico

Numa perspectivo de contexto económico iremos analisar várias variáveis que condicionam a indústria das tecnologias de informação e que interferem de uma forma directa na análise à empresa em questão visto esta estar inserida nesta indústria, mais precisamente no habitat da internet. As principais variáveis são: produto interno bruto (PIB), taxa de inflação, taxa de câmbio, taxa de desemprego.

Produto Interno Bruto (PIB) aponta para uma diminuição de 3,7% em volume no 1º trimestre de 2009 face ao período homólogo (Instituto Nacional de Estatística, 2009). A forte contracção em termos homólogos do PIB no 1º trimestre esteve fundamentalmente associada à redução acentuada das Exportações de Bens e Serviços, do Investimento e, em menor grau, das Despesas de Consumo Final das Famílias, podemos concluir que irá haver tendências negativas no sector estudado visto originar uma diminuição da disponibilidade monetária para aquisição dos nossos serviços.

A taxa de desemprego estimada para o 1º trimestre de 2009 foi de 8,9%. Este valor é superior ao observado no período homólogo de 2008 em 1,3 pontos percentuais (pp.). A população desempregada foi estimada em 495 mil indivíduos (Instituto Nacional de Estatística, 2009). Os números elevados em termos absolutos e percentuais das taxas de desemprego são preocupantes para este sector.

A Inflação representa o crescimento contínuo e generalizado dos preços dos bens e é calculada como a taxa de variação do Índice de Preços no Consumidor (IPC), sendo que com a subida da taxa de inflação o poder de compra das famílias diminui o que origina uma diminuição da circulação de dinheiro.

A Taxa de câmbio é o preço de uma unidade monetária de uma moeda em unidades monetárias de outra moeda, como o dólar se encontra desvalorizado face ao euro, pode-se procurar adquirir licenças de produtos dos EUA com maior facilidade económico/financeira (Banco de Portugal, 2009).

2.3.3. Contexto Demográfico

Neste contexto, analisando as várias variáveis que aqui actuam, podemos dizer que são muito vastas. De facto, a Elemento Digital produz serviços para um grupo etário dos 5 aos 90 anos. Em termos de composição étnica é, também, generalista, uma vez que não coloca diferenças neste atributo. Como a base dos seus serviços é a Internet, a variável de alteração regional também não se aplica.

2.3.4. Contexto Sociocultural

No estilo de vida português, verifica-se muito o entusiasmo pelas novas tecnologias. Em Portugal verifica-se uma das maiores taxas de penetração do meio Internet na população. Assim sendo, era natural usar este meio como forma de comunicação (Instituto Nacional de Estatística, 2009).

2.3.5. Contexto Politíco-legal

Em termos de estabilidade política, a alternância de partidos no poder não se tem reflectido neste sector, pelo que o desenvolvimento desta área não tem sido impedido.

Na política económica existe uma recente liberalização da economia portuguesa que permite novas iniciativas privadas; a afluência de empresas à comercialização deste tipo de produtos tem sido crescente desde o final dos anos 90. (eMarketeer, 2007)

2.4. Meio Envolvente Transaccional

O meio envolvente transaccional é constituído por todos os agentes e factores que interagem directamente com a indústria em que a empresa actua. O seu impacto no desempenho económico dos vários concorrentes tende a ser por isso bastante acentuado e rápido.

Os principais elementos que integram o meio envolvente transaccional de qualquer empresa são: Clientes, concorrentes, fornecedores

2.4.1. Clientes

São os clientes que em conjunto constituem o mercado para que as empresas trabalham; isto é, são os consumidores e potenciais consumidores dos bens e serviços que a indústria oferece (Freire, 1997).

Os múltiplos clientes de uma indústria apresentam geralmente características distintas, consoante os seus objectivos, necessidades e padrões de consumo.

Esta indústria pode ter como seus clientes o mais variado tipo de indústrias, tendo estas apenas em comum o aproveitamento das novas tecnologias (internet) para publicitar o seu nome, negócios, produtos ou mesmo serviços. Entre os actuais clientes da empresa Elemento Digital podemos destacar os seguintes:

2.4.2. Concorrentes

Os concorrentes são adversários actuais e potenciais, bem como produtos substitutos, que satisfazem as mesmas necessidades do mercado. Em conjunto constituem a indústria ou a oferta (Freire, 1997).

Aproveitando a banalização das novas tecnologias, principalmente da internet, as organizações encontraram um novo formato que lhes possibilita a comunicação, demonstração de valor e administração do relacionamento com os clientes. Pensa-se que neste momento devido à aderência em massa do público ao mundo da internet, o mercado do marketing digital irá crescer muito rapidamente. As antigas empresas de marketing irão explorar esta oportunidade e certamente também as novas empresas na linha da Elemento Digital irão explorar a oportunidade.

2.4.3. Fornecedores

Os fornecedores são aqueles que prestam serviços e venda de matérias-primas e componentes intermédios necessários aos vários concorrentes da indústria, para que estes possam assim desenvolver a sua oferta, ou seja, são os agentes económicos que prestam serviços ou vendem produtos à indústria (Freire, 1997).

Podem ser classificados da seguinte forma:

Financeiros – empréstimos por obrigação, por títulos de participação, instituições de crédito, accionistas

Fornecedores de matérias-primas e outras existências: empresas do grupo e empresas de imobilizado

Sendo assim podemos concluir que os principais fornecedores do sector são os fornecedores de tecnologias de informação, no caso da empresa Elemento Digital, tecnologias ligadas à área da internet como é o caso do Adobe Flash.

2.5. Factores Críticos de Sucesso

Os factores críticos de sucesso são os pontos-chave que definem o sucesso ou o fracasso de um objectivo definido por um planeamento de determinada organização. Estes factores precisam de ser encontrados pelo estudo sobre os próprios objectivos, derivados deles, e tomados como condições fundamentais a serem cumpridos para que a instituição sobreviva e tenha sucesso na sua área. Quando bem definidos, os factores críticos de sucesso tornam-se um ponto de referência para toda a organização nas suas actividades voltadas para a sua missão.

Satisfação dos clientes: o quanto satisfeitos eles estão?

Qualidade: o quanto bom é o nosso produto ou serviço?

Desenvolvimento do produto ou serviço: o que traz de novo?

2.6. Análise SWOT

A Análise SWOT é uma ferramenta utilizada para fazer verificação estratégica da empresa no ambiente em que se insere, sendo usada como base para gestão e planeamento estratégico de uma empresa.

*O termo SWOT é acrónimo de Forças (*Strengths*), Fraquezas (*Weaknesses*), Oportunidades (*Opportunities*) e Ameaças (*Threats*).*

A Análise SWOT consiste em detectar pontos fortes e fracos no interior da empresa, em paralelo com as oportunidades e ameaças do exterior. Esta surge para apresentar de uma forma integrada o resultado do processo de análise estratégica, visando por um lado uma forma de atenuar ameaças, bem como uma redução dos pontos fracos da organização, e por outro lado aproveitar as oportunidades, assim como maximizar os pontos fortes.

Os pontos fortes e os pontos fracos são determinados pela situação da organização, tendo como base, um rigoroso diagnóstico interno (Recursos Humanos, Financeiros e Organizacionais).

As oportunidades e ameaças são resultado de uma profunda análise dos factores globais que são específicos à sua indústria (Clientes, Fornecedores, Concorrentes, Comunidade).

Matriz SWOT	Pontos Fortes	Pontos Fracos
	• Boas infra-estruturas; • Metodologia;	• Dimensão da empresa;
Oportunidades • Evolução das tecnologias; • Taxas de juro baixas; • Crescimento da procura;	• Incentivar o espírito inovador; • Produtos e serviços complementares;	• Falta de mão-de-obra qualificada; • Vigilância tecnológica;
Ameaças • Necessidade de Mão-de-obra qualificada; • Conjuntura económica desfavorável; • Maior intensidade competitiva;	• Diferenciação da oferta; • Colaboração com instituições de ensino Superior	• Apoios Financeiros; • Projectos desenvolvidos de acordo com as possibilidades e dimensão da empresa;

3. Fundamentos, objectivos e efeitos esperados do projecto

3.1. Fundamentos e Objectivos

3.1.1. Jogo em 3 Dimensões

A aplicação que se irá ser produzida consiste em utilizar técnicas de programação gráfica para construir um jogo com imagens que permitem a visualização da acção em tempo real a 3 dimensões, isto é, exposição de dados e informação em forma de representação do mundo real.

*Neste caso em particular, o género de jogo 3D solicitado foi "*First Person Shooter*" (*FPS*). Este género utiliza um ponto de visão na primeira pessoa onde o jogador visualiza a acção através dos olhos da personagem que controla e o principal intuito em títulos deste género é o combate utilizando armas de fogo. Este tipo de jogos de computador é considerado só por si um género de jogo de computador, não sendo aceite como um subgénero dos jogos de acção. Os títulos pioneiros neste campo foram considerados clássicos do mundo dos jogos de computador, obtendo enorme sucesso na sua época. Nomes como* Wolfenstein 3D *(1992) e* Doom *(1993) deram vida a este estilo de jogo que é o segundo mais solicitado pelos jogadores de computador, só sendo ultrapassado pelos jogos de estratégia em tempo real (*RTS*).*

Imagem 1 – Wolfenstein 3D

Imagem 2 – Doom

3.1.2. Fácil Distribuição

É também um objectivo essencial o facto de o jogo poder ser distribuído pela internet e instalado facilmente por o máximo de pessoas possível. Tem que se ter em conta que o tipo de jogador alvo desta aplicação será o jogador casual; este tipo de jogador prefere jogos pouco complexos, fáceis de jogar, sem qualquer tipo de história, possibilidade de potenciar um pouco de competitividade e principalmente dar uma sensação de objectivo cumprido num curto espaço de tempo (Lobão, Evangelista, & de Farias, 2008).

Não podemos deixar de observar que este tipo de utilizador irá apreciar o facto de que o processo de transferência e instalação seja praticamente imediato sem grandes tempos de espera, caso contrário corre-se o risco de o jogador desistir de jogar o jogo por se aborrecer com longos períodos de espera derivados da transferência ou instalação em disco.

Tendo em conta os factos acima descritos, pode observar-se facilmente que muitos dos aspectos do tipo de "jogador casual" são compatíveis com a distribuição do jogo pela internet, isto porque o facto da simplicidade de mecanismos pretendida pelo jogador em causa ser compatível com a necessidade de tamanho reduzido dos ficheiros de instalação, para transferência rápida através da internet.

Desta forma poderemos conseguir atingir o objectivo da distribuição/instalação simples da aplicação e ao mesmo tempo ir de encontro às expectativas dos jogadores alvo, quando procuram um jogo desta natureza.

3.1.3. Multi-jogador

*É um conceito que consiste em mais de uma pessoa poder jogar no mesmo ambiente em simultâneo. Ao contrário de muitos jogos que só oferecem componente "*Single Player*" em que o utilizador joga contra elementos oponentes controlados por inteligência artificial, a componente Multi-Jogador possibilita aos jogadores interagirem com outros jogadores humanos, organizando disputas, equipas e competições, providenciando interacção social entre eles.*

*Existem diferentes variantes de jogos Multi-Jogador comuns nos "*First Person Shooter*" são eles: "*DeathMatch*" consiste em eliminar outros jogadores, o maior número de vezes possível; "*TeamDeathMatch*" aqui uma equipa tenta eliminar a equipa adversaria o maior número de vezes possível; "*Capture The Flag*" consiste em capturar a bandeira da equipa adversaria o maior número de vezes possível; "*assault/defend*" uma equipa tenta tomar controlo de um ponto do mapa e a outra equipa defende esse mesmo ponto. Nesta aplicação o foco ira recair sobre a variante "*DeathMatch*", visto ser a que mais se relaciona com o estilo de jogador "casual" que pretendemos atingir. As outras variantes requerem componentes de estratégia, tácticas, treino e muita prática entre as equipas de jogadores, características que dizem respeito a outro tipo de jogadores. Alguns dos jogos mais jogados de todos os tempos são precisamente do género "*First Person Shooter*" especializados na componente Multi-Jogador. Exemplos disso são títulos como "*Unreal Tournament*" (1999) e "*Counter Strike*" (2000).*

Imagem 3 – Unreal Tournament

Imagem 4 – Counter Strike

3.1.4. Jogo com Publicidade

*Em inglês "*adverGame*" é a prática de usar um jogo de computador para publicitar um produto, uma organização ou um ponto de vista. Este conceito já é aplicado por grandes companhias que patrocinam vários jogos grátis online.*

*Com o crescimento da internet, os "*advergames*" têm-se multiplicado, dado a facilidade de distribuição que este canal proporciona, tornando-se por vezes parte dos planos de marketing das empresas, normalmente acompanhando o lançamento de um novo produto.*

*Existem 3 principais tipos de "*advergames*", são eles: "*above the line*" consistem normalmente em reeditar jogos clássicos e são construídos pelas próprias empresas que pretendem publicitar o seu produto com o jogo; "*Through the line*" são jogos de computador que possuem hiperligações para sites onde é depositada a publicidade; "*below the line*" os jogos são construídos de raiz por companhias especializadas e depois são incluídos patrocínios ao longo do jogo. Este último é o tipo que encaixa melhor com as nossas pretensões para esta aplicação.*

Pretende-se integrar publicidade nesta aplicação através de um repositório de imagens predefinido, ou seja, o jogo irá fazer o carregamento destas imagens, que possuem publicidade e os jogadores irão visualizar a publicidade enquanto jogam.

Imagem 5 – FIFA 94

Como podemos ver na Imagem 5, no jogo FIFA 94 pode visualizar-se publicidade à marca Adidas nos placards enquanto se joga.

3.2. Motivação

Um jogo a 3 dimensões com componente multi-Jogador para o tipo de jogadores "casual" com fácil distribuição pela internet poderá despertar a curiosidade de muitas pessoas, para fazer uma pequena competição com um amigo lá do bairro ou mesmo com alguém que está do outro lado do mundo. Se existir uma forma de controlar imagens publicitárias para que o jogador visualize enquanto de diverte, pensa-se que poderemos estar perante uma ideia de sucesso.

3.3. Articulação do projecto no seio da empresa

A empresa Elemento Digital que propôs este projecto trabalha essencialmente em produtos orientados para a Internet, logo a ideia de um jogo que tem a Internet como meio surge de forma relativamente natural e pacífica.

3.4. Descrição dos principais efeitos esperados a nível tecnológico, comercial e efeitos ao nível da produtividade

Espera-se que a partir do próximo ano seja possível começar a distribuir a aplicação pela Internet e que a empresa possa contar com um mecanismo para gerir de forma eficaz a publicidade que aparece nos terrenos do jogo, isto é, a possibilidade de carregar logótipos de parceiros e informação sobre os respectivos produtos.

4. Sistema de Informação

4.1. Análise de Requisitos

4.1.1. Requisitos Funcionais

ID	Descrição
1	A aplicação deverá permitir criar um jogo em rede
2	A aplicação deverá permitir procurar um jogo existente
3	A aplicação deverá permitir alterar opções de jogo
4	A aplicação deverá permitir visualizar estatísticas de jogo
5	A aplicação deverá fornecer ajuda sobre os modos de jogo
6	A aplicação deverá permitir terminar o jogo
7	A aplicação deverá permitir adicionar dados ao servidor
8	A aplicação deverá permitir modificar dados do servidor
9	A aplicação deverá permitir remover dados do servidor

Tendo em conta os requisitos funcionais do sistema construiu-se o seguinte diagrama UML de use cases:

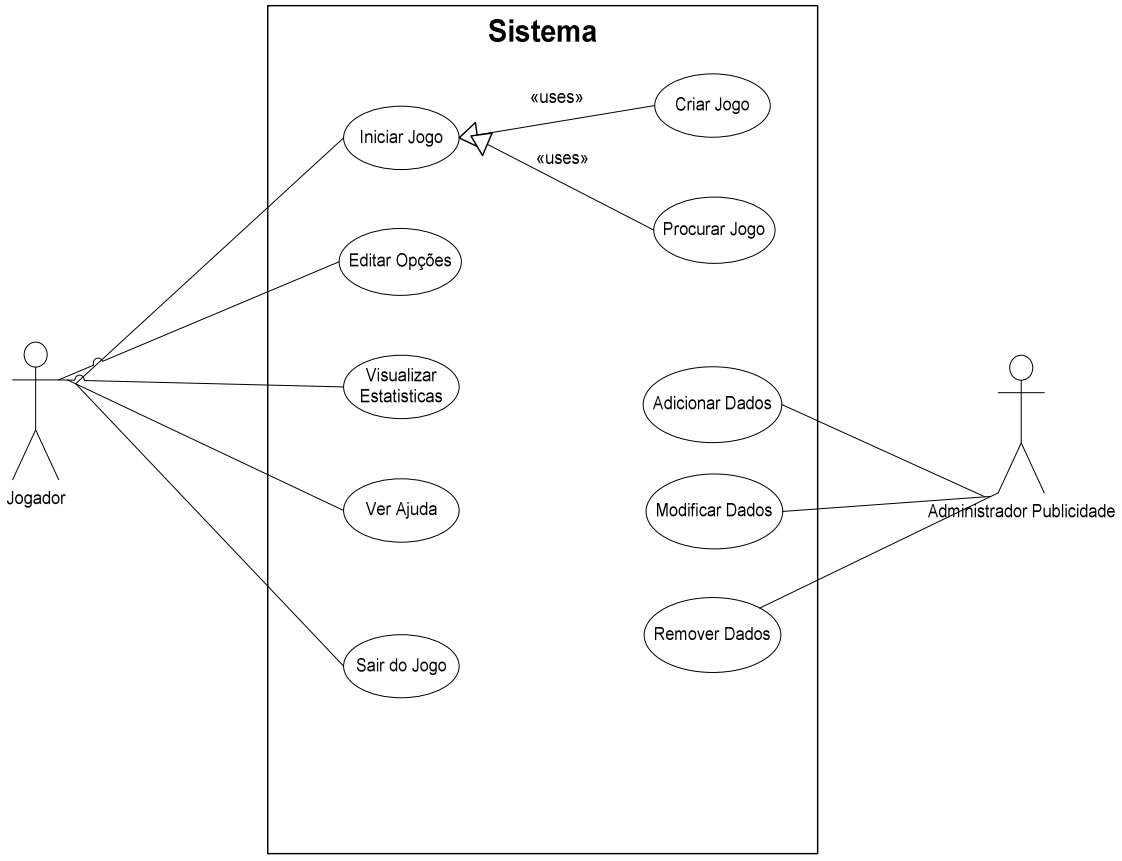


Diagrama de Classes 1 – Use Cases

4.1.2. Descrição dos Use Cases

ID	1
Actor	Jogador
Pré-condições	Maquina conectada à rede
Cenário Principal	Jogo em rede criado com sucesso
Cenários Alternativos	Erro na criação do jogo. Problema de configuração de rede. Conflito com firewall.
Pós-condições	O jogador é informado do resultado da operação

Tabela 1 – Criar Jogo

ID	2
Actor	Jogador
Pré-condições	Jogo criado por outro jogador
Cenário Principal	Associação ao jogo bem sucedida
Cenários Alternativos	Problema de configuração de rede. Conflito com firewall.
Pós-condições	O jogador é informado do resultado da operação e opta por associar-se ao jogo

Tabela 2 – Procurar Jogo

ID	3
Actor	Jogador
Pré-condições	Ficheiro de opções disponível
Cenário Principal	O utilizador edita as opções disponibilizadas
Cenários Alternativos	Erro ao gravar as alterações
Pós-condições	O jogador é informado do resultado da operação

Tabela 3 – Editar opções

ID	4
Actor	Jogador
Pré-condições	Ficheiro de estatísticas disponível
Cenário Principal	As estatísticas são escritas no ecrã
Cenários Alternativos	Erro ao ler o ficheiro de estatísticas
Pós-condições	O jogador é informado do resultado da operação e retorna ao menu inicial

Tabela 4 – Visualizar estatísticas

ID	5
Actor	Jogador
Pré-condições	Ficheiro de ajuda disponível
Cenário Principal	A ajuda é escrita no ecrã
Cenários Alternativos	Erro ao ler o ficheiro de ajuda
Pós-condições	O jogador é informado do resultado da operação e retorna ao menu inicial

Tabela 5 – Visualizar ajuda

ID	***6***
Actor	*Jogador*
Pré-condições	*Ter o jogo activo*
Cenário Principal	*Retornar ao sistema operativo*
Cenários Alternativos	*Erro na aplicação.* *Aplicação bloqueia.*
Pós-condições	

Tabela 6 – Sair Jogo

ID	***7***
Actor	*Administrador*
Pré-condições	*Directoria do repositório de publicidade criada*
Cenário Principal	*O ficheiro é adicionado*
Cenários Alternativos	*Erro de formato de ficheiro.* *Problema de configuração de rede.* *Conflito com firewall.*
Pós-condições	*O jogador é informado do resultado da operação*

Tabela 7 – Adicionar dados

ID	***8***
Actor	*Administrador*
Pré-condições	*Directoria do repositório de publicidade criada*
Cenário Principal	*O ficheiro é modificado*
Cenários Alternativos	*O ficheiro inicial não existe.* *Erro de formato de ficheiro.* *Problema de configuração de rede.* *Conflito com firewall.*
Pós-condições	*O jogador é informado do resultado da operação*

Tabela 8 – Modificar dados

ID	***9***
Actor	*Administrador*
Pré-condições	*Directoria do repositório de publicidade criada*
Cenário Principal	*O ficheiro é apagado*
Cenários Alternativos	*O ficheiro pretendido não existe.* *Problema de configuração de rede.* *Conflito com firewall.*
Pós-condições	*O jogador é informado do resultado da operação*

Tabela 9 – Apagar dados

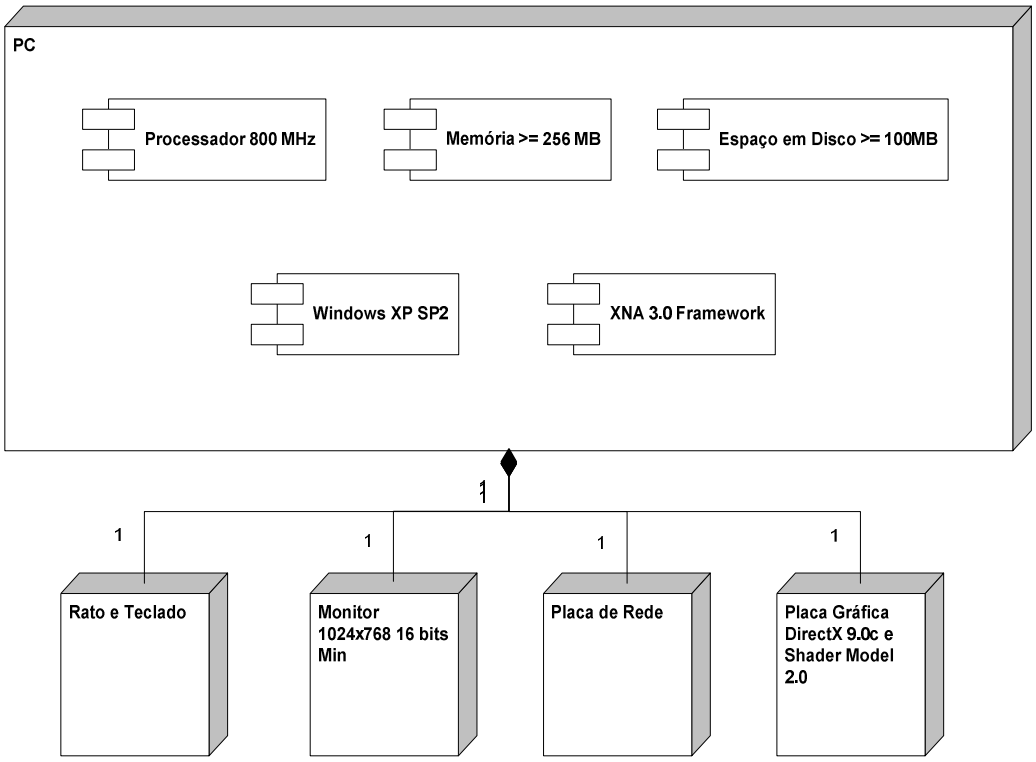
4.1.3. Requisitos Ambientais

Software*:*

- *Windows XP service pack 2 ou superior*
- *XNA Framework 3.0*

Hardware*:*

- *Processador Pentium III a 800 MHz ou superior*
- *Mínimo 256 MB de memória RAM*
- *Placa Gráfica com suporte DirectX 9.0c e* Shader Model 2.0
- *Ligação à Internet por banda larga*
- *Rato e teclado*
- *Monitor com resolução mínima 1024x768 a 16 bits*



4.1.4. Requisitos de Qualidade

No que toca ao atributo desempenho escolheu-se os seguintes testes ao sistema:

Identificação	*Capacidade de resposta*
Escala	*Milissegundos*
Teste	*Contabilizar o ping médio da rede durante um jogo*
Pior Caso	*100*
Planeado	*25*
Actual	*N/D*

Identificação	*Capacidade de processamento*
Escala	*Frames por segundo (FPS)*
Teste	*Contabilizar as FPS médio durante uma partida*
Pior Caso	*20*
Planeado	*60*
Actual	*N/D*

Na área do atributo disponibilidade escolheu-se o seguinte teste ao sistema:

Identificação	*Fiabilidade*
Escala	*Número de falhas*
Teste	*Contabilização de falhas ocorridas durante 120 minutos de jogo*
Pior Caso	*5*
Planeado	*0*
Actual	*N/D*

No campo do atributo adaptabilidade decidiu-se efectuar o seguinte teste:

Identificação	*Portabilidade*
Escala	*Dias*
Teste	*Tempo que levaria a efectuar uma conversão total para a plataforma XBOX 360*
Pior Caso	*30*
Planeado	*5*
Actual	*N/D*

No atributo usabilidade irá realizar-se os seguintes testes ao sistema:

Identificação	*Facilidade de aprendizagem*
Escala	*Minutos*
Teste	*O tempo que o utilizador necessita para aprender a jogar*
Pior Caso	*30*
Planeado	*5*
Actual	*N/D*

Identificação	*Eficiência na utilização*
Escala	*Minutos*
Teste	*Contabilizar o tempo desde a transferência do jogo até ao utilizador começar a jogar*
Pior Caso	*20*
Planeado	*5*
Actual	*N/D*

Identificação	*Satisfação*
Escala	*Percentagem*
Teste	*Percentagem de utilizadores satisfeitos depois de jogar*
Pior Caso	*50%*
Planeado	*90%*
Actual	*N/D*

4.2. Análise de Risco

Identificação de perigos de origem contratual, financeira, organizacional ou tecnológica.

ID	Descrição	Probabilidade	Efeito
1	Calendarização	Média	Marginal
2	Desistência de colaborador	Baixa	Dramático
3	Alteração de requisitos	Baixa	Crítico
4	Pouca experiencia em técnicas de computação gráfica	Alta	Crítico
5	Falta de comunicação entre partes envolvidas	Baixa	Crítico
6	Divergências entre colaboradores	Baixa	Marginal
7	Expectativas do cliente não satisfeitas	Média	Marginal
8	Abandono do projecto por parte do cliente	Baixa	Marginal

Da análise acima realizada resulta as seguintes possíveis soluções para cada risco apresentado.

ID	Solução
1	Seguir a calendarização, de modo a cada etapa ter tempo suficiente
2	Expor as dificuldades aos outros elementos e procurar soluções em grupo
3	Detalhar todos os objectivos do cliente para que não restem dúvidas
4	Consultar informação sobre as tecnologias a usar (manuais e tutoriais); recorrer aos orientadores sempre que necessário
5	Reuniões periódicas
6	Procurar chegar a acertos vantajosos para o projecto
7	Fazer uma boa análise de requisitos para ir ao encontro das expectativas
8	Tentar tornar o projecto novamente aliciante para o cliente

4.3. Diagrama de Classes

4.3.1. Classe Servidor

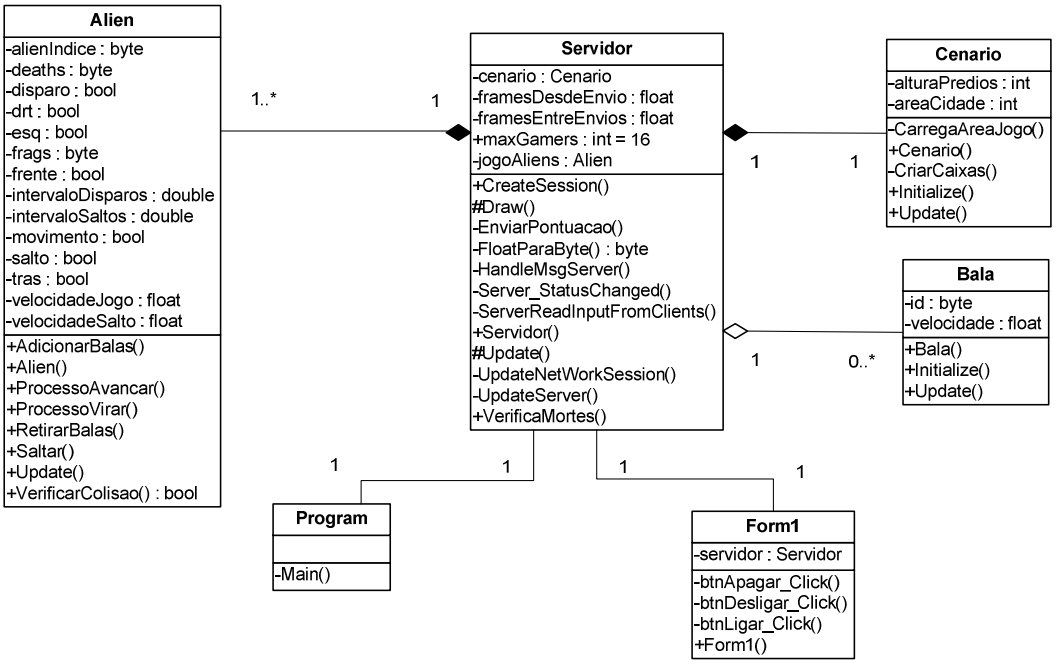


Diagrama de Classes 2

4.3.2. Classe Cliente

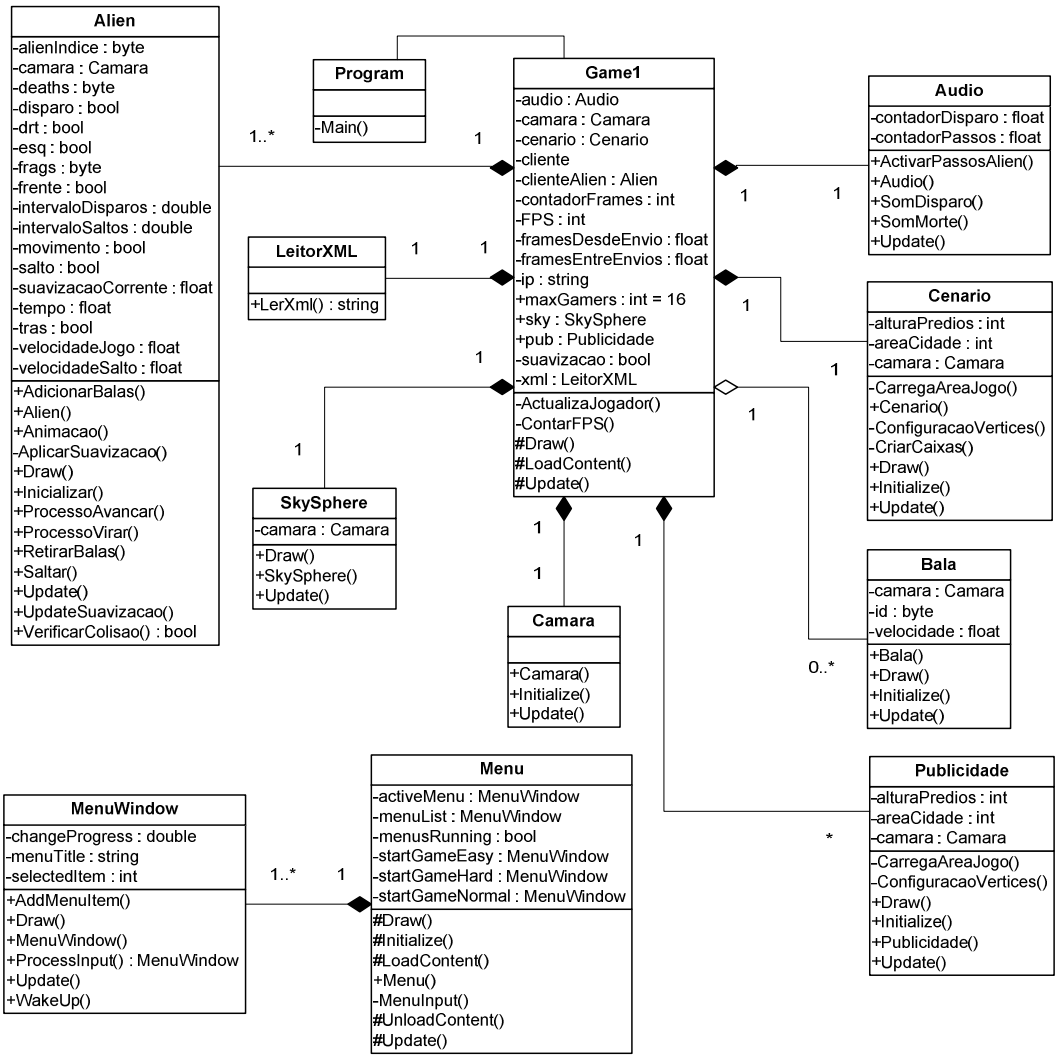


Diagrama de Classes 3

4.4. Descrição da tecnologia

4.4.1. Microsoft XNA

Microsoft XNA (XNA's Not Acronymed) é um conjunto de ferramentas disponibilizadas pela Microsoft que facilita o desenvolvimento de jogos de computador. O XNA foi anunciado pela Microsoft a 24 de Março de 2004 e a primeira versão data de 14 de Março de 2006. Trata-se de uma ferramenta recente no mercado dos jogos, pessoas em todo o mundo.

A Framework XNA é baseada na implementação nativa da .NET Compact Framework 2.0 da XBOX 360 e na .NET Framework 3.0 do Windows. Inclui uma extensão das bibliotecas, especificamente para o desenvolvimento de jogos, com o objectivo de providenciar o máximo de reutilização de código possível (Microsoft, 2009).

A XNA Framework encapsula os detalhes de baixo nível envolvidos na codificação de um jogo, garantindo que a Framework em si trata da diferença entre plataformas, quando são adaptados jogos compatíveis a partir de uma plataforma para outra, assim é permitido que os programadores de jogos se concentrem mais no conteúdo e experiência do jogo. A XNA Framework integra uma série de ferramentas, tais como XACT, para auxiliar na criação conteúdo, áudio neste caso. Fornece também suporte para jogos 2D e 3D, permite a utilização do comando da Xbox 360. Aplicações desktop podem ser distribuídas gratuitamente, nos termos actuais de licenciamento da Microsoft.

A versão actual desta ferramenta da Microsoft é o XNA Game Studio 3.0 (para Visual Studio 2008). A versão final foi lançada em 30 de Outubro de 2008. O XNA Game Studio 3.0 agora suporta C# 3.0, existem várias novas funcionalidades, que permite aos programadores adicionar facilmente conteúdos necessária para os todas as plataformas, criar jogos que funcionam em Windows, Xbox 360 e Zune, também a componente multi-jogador foi bastante trabalhada nesta nova versão, que será uma mais-valia.

Os jogos da Xbox 360 escritos em XNA Game Studio podem ser submetidos à comunidade de criadores. Os jogos submetidos são alvo de análises por outros criadores, se o jogo passar na análise é colocado na lista de jogos "Xbox Live Marketplace" (Microsoft, 2009). Os criadores recebem 70% do valor das vendas.

4.4.2. Microsoft C#

Microsoft C# (pronunciado C Sharp) é uma linguagem de programação multi-paradigma que engloba os tipos: funcional, imperativo, genérico, orientado a objectos e orientada a componentes. Foi desenvolvida pela Microsoft como parte da iniciativa .NET e mais tarde aprovada como standard pela ECMA (ECMA, 2006) e ISO (ISO, 2003). C# é uma linguagem de programação desenhada para a "Common Language Infrastructure".

C# foi idealizado para ser uma linguagem simples, moderna, de uso geral e orientada por objectos. O desenvolvimento da linguagem foi liderado por Anders Hejlsber, o designer do compilador da Borland e do Turbo Pascal. Tem uma sintaxe orientada a objectos baseada em C++. Foi inicialmente denominada "Cool", contudo, em Julho de 2000, quando a Microsoft apresentou o projecto ao público, o nome da linguagem de programação escolhido foi C#. A mais recente versão da linguagem é a 3.0, que foi lançada em conjunto com o .NET Framework 3.5, em 2007.

Microsoft Visual Studio é um "Integrated Development Environment"(IDE) da Microsoft. Este pode ser usado para desenvolver aplicações em consola, em modo gráfico, aplicações windows forms, sites web, serviços web, todos compatíveis com as diversas plataformas, Microsoft Windows, Windows Mobile, Windows CE, .NET Framework e a .NET Compact Framework (Microsoft, 2009).

O Visual Studio suporta linguagens (serviços das linguagens), que possibilita a cada linguagem de programação possuir suporte em diversos níveis. Através do editor de código e do debugger. O conjunto de linguagens suportadas inclui C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), e C# (via Visual C#). Também suporta XML/XSLT, HTML/XHTML, JavaScript e CSS. Também existe versões específicas do Visual Studio,

que possibilita ter o acesso limitador a um tipo preferido de linguagem por parte do utilizador.

4.4.3. XML

XML ou eXtensible Markup Language é uma maneira flexível de criar formatos comuns de informação a partilhar o formato e os dados na Internet, intranet ou qualquer que seja a conexão.

XML é uma recomendação formal da W3C (World Wide Web Consortium, 2009) e é similar à linguagem das páginas Web de hoje, o Hypertext Markup Language (HTML). O HTML e o XML partilham ambos os símbolos para descrever o conteúdo de uma página ou ficheiro. No entanto, o HTML descreve o conteúdo de uma página Web (principalmente gráficos e texto) apenas em termos de como posicionar e interagir. XML descreve o conteúdo em termos de quais os dados que vão ser descritos. Isto significa que um ficheiro XML pode ser processado simplesmente como dados por um programa ou pode ser armazenado com dados similares noutro computador ou, tal como com HTML, pode ser mostrado no ecrã (What is XML?, 2009).

XML é extensível porque, ao contrário do HTML, os símbolos são infinitos e auto-definidos.

5. Software Desenvolvido

5.1.1. Desenvolvimento de jogos em 3D

O rápido crescimento da capacidade de processamento dos processadores e placas gráficas permitiu enormes avanços na área do grafismo 3D. Ao carregar qualquer um dos últimos jogos de first-person shooter*, ficamos espantados com o nível de detalhe no jogo e no número de objectos a atravessar o nosso ecrã em qualquer momento. O grafismo 3D é o caminho para o futuro, e é aqui que iremos focar a maior parte do nosso tempo e atenção.*

Algumas das diferenças que notamos ao trabalhar em 3D ao invés de 2D é a adição de uma dimensão extra. Programar em 2D é semelhante a pintar uma imagem numa tela, desenha-se num espaço bidimensional. A coordenada (0, 0) aponta para o canto superior esquerdo do ecrã; X move positivamente para a direita, enquanto Y move positivamente para baixo.

Se desenhar em 2D é como pintar uma imagem, desenhar em 3D é como gravar um vídeo com uma câmara de vídeo. As coordenadas em 3D são baseadas num sistema tridimensional. Este sistema de coordenadas centra-se numa origem nas coordenadas (0, 0, 0). No entanto, ao contrário do que acontece em 2D, quando algo é desenhado na origem em 3D, não é possível garantir que o objecto seja visível no centro do ecrã, no canto do ecrã, ou em qualquer parte. Isto acontece porque, em 3D, existem dois componentes básicos para desenhar um cenário: colocar objectos no sistema tridimensional, e colocar uma câmara no sistema e especificar a direcção em que a câmara irá apontar. Apenas os objectos que a câmara vê serão visíveis no ecrã.

Dependendo de onde a câmara está posicionada e em que direcção aponta, um objecto que seja desenhado na origem num jogo 3D poderá estar no centro do ecrã, no fundo do ecrã, ou até completamente fora de vista.

Tipicamente, o eixo dos XX move-se positivamente para a direita e o eixo dos YY move-se positivamente para cima. No entanto, o eixo dos ZZ não está tão claramente definido. Existem dois tipos diferentes de sistemas de coordenadas, e em cada um deles o eixo dos ZZ move-se em direcções opostas. A direcção na qual o Z se move positivamente determina a orientação do sistema de coordenadas. As duas possíveis orientações são o sistema da mão direita e o sistema da mão esquerda.

Uma maneira de distinguir entre o sistema de coordenadas da mão direita do sistema de coordenadas da mão esquerda é colocar as mãos, palma virada para cima,

com os dedos a apontar na direcção positiva de X, e dobrados na direcção positiva de Y. A direcção para onde aponta o polegar indica a direcção do Z positivo para esse sistema de coordenadas.

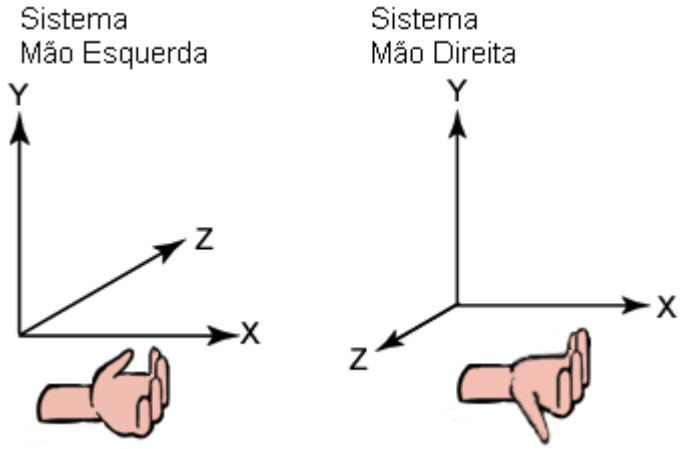


Imagem 6 – Sistema de Coordenadas

O XNA usa o sistema de coordenadas da mão direita, que significa que quando se olha para a origem de um ângulo tradicional em que X se move positivamente para a direita e Y se move para cima, Z move-se positivamente na nossa direcção.

8.1.1.1 Câmaras

Como indicado anteriormente, desenhar uma cena em 3D é como gravar um vídeo com uma câmara de vídeo. Será necessário definir onde a câmara está localizada, para onde está apontada e varias outras propriedades.

*Estas propriedades serão armazenadas num objecto **Matrix**.*

Existem dois objectos do tipo matriz que criam uma câmara em XNA: as matrizes de vista e projecção. A matriz vista armazena informação que determina onde a câmara se posiciona no mundo, em que direcção aponta, e qual a sua orientação. A matriz projecção reúne informação que determina as propriedades da câmara baseada no ângulo de visão, a quão longe a câmara consegue ver, e outras. Esta matriz representa a transformação do mundo 3D para o plano 2D do ecrã.

Apesar de podermos especificar uma posição e uma direcção para a câmara, isto ainda não determina como os objectos serão desenhados no ecrã. Com uma câmara de vídeo, é possível virar a câmara de lado, ou até ao contrário, e a direcção para onde está virada irá afectar a maneira que as imagens são gravadas. O mesmo

acontece com uma câmara em XNA. É possível especificar um vector para a câmara de maneira a que o XNA saiba, não apenas como posicionar a câmara no mundo, mas também qual a sua orientação.

A matriz projecção constrói o que é chamado de campo de visão para a câmara. Essencialmente, define uma área no espaço 3D que é visível pela câmara e será desenhada no ecrã. Os objectos que existam dentro desta área serão desenhados no ecrã. Pelo contrário, os objectos que estão fora do campo de visão da câmara, não serão desenhados.

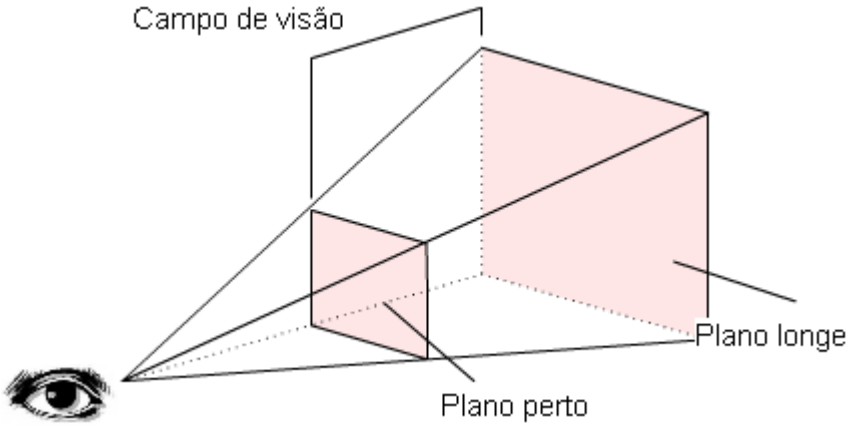


Imagem 7 – Campo de visão

É necessário algum cuidado na definição dos valores que definem o plano perto e o plano longe. É necessário ter em atenção que tudo o que está dentro desses planos, e dentro de outros limites do campo de visão, serão desenhados no ecrã. Se existirem muitos objectos no jogo, existe a possibilidade de existirem problemas de performance se o plano perto e o plano longe forem demasiado distantes um do outro.

Por exemplo, um jogo em que o jogador passeia por uma floresta, e de repente uma aparente montanha aparece no ecrã. Isto acontece porque a montanha estava por detrás do plano longe, e enquanto nos movemos em frente, esta entrou no nosso campo de visão. Se estes jogos não tivessem esta funcionalidade, a performance iria decair e a jogabilidade tornar-se-ia desagradável ao jogador. É fundamental incluir o suficiente do mundo para tornar o jogo agradável de jogar, e excluir o necessário para evitar problemas de performance.

8.1.1.2 Matrizes

Entender matrizes é parte fundamental de criar efectivamente jogos em XNA. As matrizes são usadas particularmente quando trabalhamos em 3D. No entanto, a matemática por detrás pode ser desafiadora. Apesar dos conceitos adquiridos nas aulas de Álgebra Linear e Geometria Analítica, o XNA fornece a maior parte das funcionalidades para matrizes para que não seja necessário preocupar com a matemática que esta usa. De qualquer modo, é necessário entender correctamente no que estas funcionalidades realmente realizam de forma a serem aplicadas correctamente no código.

Porque razões são tão importantes? Por um lado, é a única maneira de passar informações de transformação para um **Effect** *no código de renderização, ao invés de transformar cada vértice do modelo manualmente frame a frame. Outra razão é que matrizes permitem realizar transformações que podem incluir camadas de rotações, escalas, e translações, tudo de seguida.*

Uma matriz em XNA poderá ser visualizada como um array *4x4 de floats. Não existe necessariamente um* array *na estrutura* **Matrix***, mas é assim que poderá ser matematicamente visualizada. Em vez disso, os elementos da matriz são codificados como uma série de 16 atributos públicos. Cada campo tem a nomenclatura "M" + número de linha + número de coluna. Como exemplo, o elemento na linha 3 e coluna 2 seria M32.*

Como referido anteriormente, matrizes podem armazenar transformações. Transformações são operações matemáticas que realizamos no nosso jogo 3D, como translações, rotações e escalas.

É necessário, claro, uma matriz que represente a transformação de zero. Simplesmente colocamos a matriz a representar a origem (0, 0, 0) no espaço sem rotações e sem escala aplicada. Esta é chamada de Matriz Identidade e pode ser acedida através da propriedade **Matrix.Identity***. A matriz identidade é uma matriz que, no caso da multiplicação, actua como o número um. Isto é, qualquer matriz multiplicada pela matriz identidade devolve a própria matriz.*

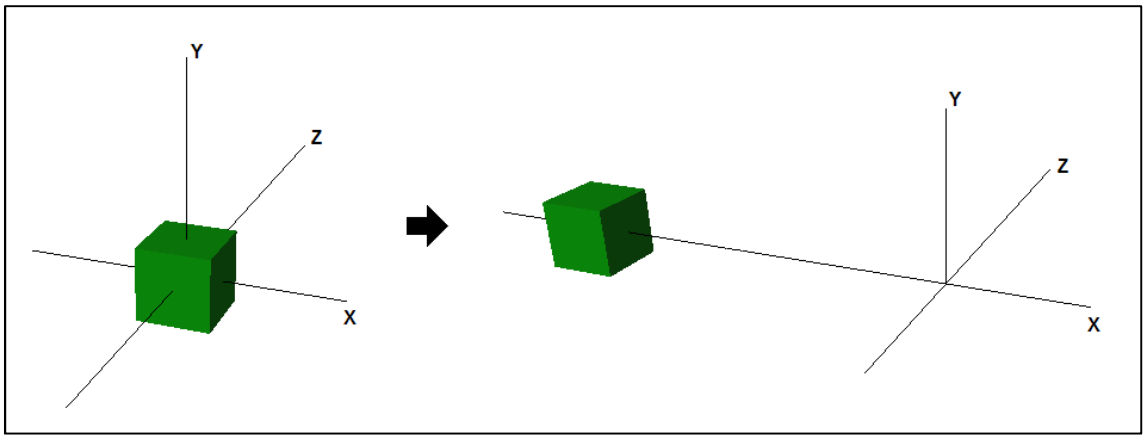


Imagem 8 - Translação

A translação é a transformação que move um objecto ou um ponto numa direcção especificada por um vector.

Existem vários métodos para rodar um objecto, e cada um deles depende do eixo pelo qual se quer rodar o objecto. Semelhante à rotação da Terra no seu eixo, toda a rotação em XNA revolve à volta de um eixo.

Existe várias maneiras de representar rotações: usando matrizes, quaterniões, ângulo de vectores, ou em graus e radianos. A maneira mais precisa e menos limitada de as armazenar são em matrizes.

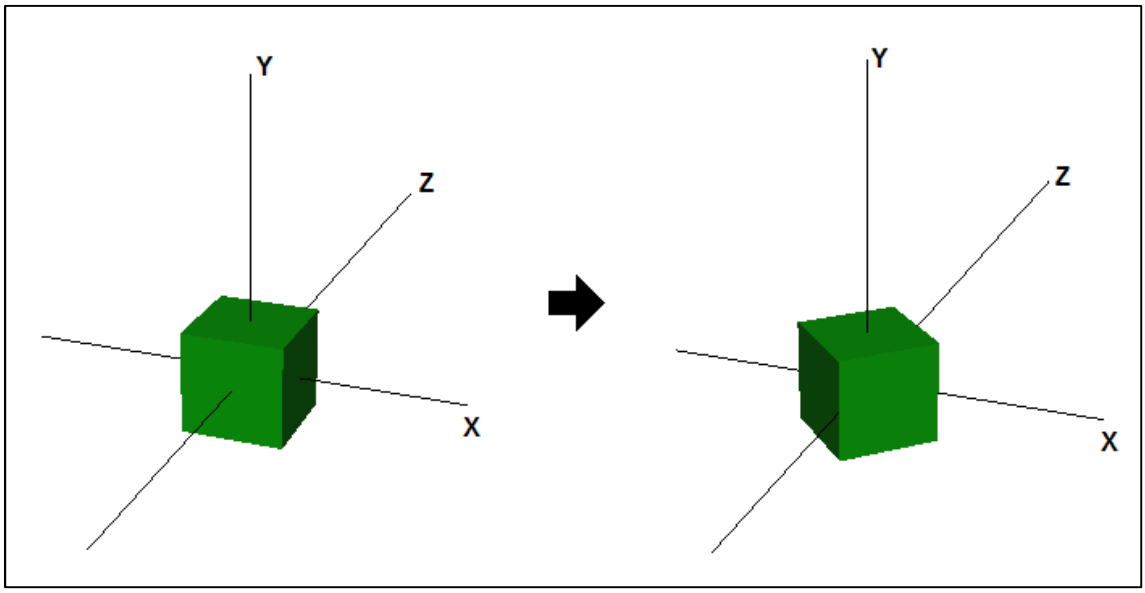


Imagem 9 - Rotação

Para rodar um objecto podemos usar os métodos **CreateRotationX**, **CreateRotationY** *e* **CreateRotationZ**, *que rodam em volta do eixo respectivo. Outro método usado para aplicar rotações é* **Matrix.CreateFromYawPitchRoll**. *Na sua essência, este método permite uma rotação que combina rotações em volta do eixo*

dos XX, YY e ZZ simultaneamente. Rotação Yaw indica uma rotação direita-esquerda, rotação Pitch é uma rotação em que o nariz empina e a cauda desce e rotação Roll em que uma asa desce para outra subir.

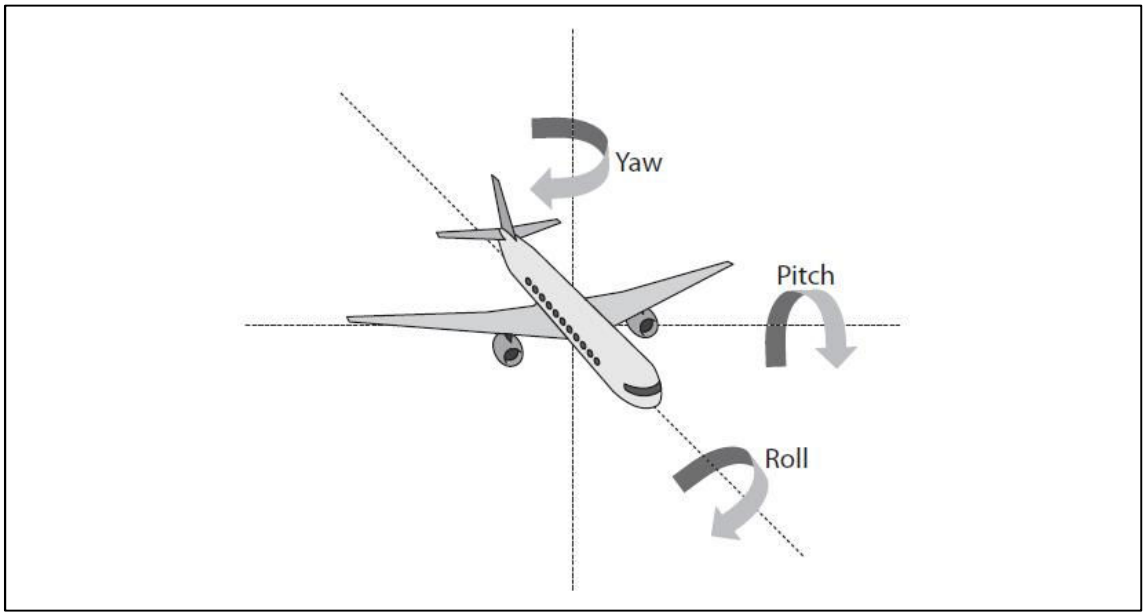


Imagem 10 – Rotações Yaw, Pitch e Roll

Quando usamos estes três ângulos para definir uma rotação, é muitas vezes referido como Representação de Ângulos de Euler. Gimbal Lock *é um problema que ocorre quando usamos os ângulos de* Euler *para realizar rotações. Ocorre quando dois eixos ficam alinhados durante a operação de rotação. Quando tal acontece, perde-se um grau de liberdade e não é possível voltar a ganha-lo usando ângulos de* Euler, *é necessária intervenção externa.*

A razão pela qual um Gimbal Lock *ocorre é porque quando são usados ângulos de* Euler *para efectuar uma rotação, cada eixo é processado independentemente, um a seguir ao outro, levando à possibilidade à possibilidade de um objecto poder ter o seu eixo rodado tal que, no momento em que o ultimo eixo é processado, outro dos eixos ficar alinhado com este. E, devido à maneira das funções de ângulos de* Euler, *tentar rodar enquanto dois eixos estão alinhados irá produzir resultados inesperados.*

Por exemplo, se efectuarmos uma rotação Yaw *(rotação no eixo dos YY) 45º e depois* Pitch *(rotação no eixo dos XX) -90º são o equivalente a um* Pitch *de -90º e depois um* Roll *(rotação no eixo dos ZZ) de 45º. Então, se a rotação* Pitch *for de ±90º este irá causar que os eixos de* Roll *e* Yaw *fiquem alinhados.*

Para evitar completamente o Gimbal Lock, *é necessário fazer algumas verificações cada vez que tal pode ocorrer, ou então usando um método diferente de rotação, especificamente Rotações Quaternião.*

Um quaternião é uma extensão ao número complexo. Em vez de apenas i, existem três números em que todos são raízes quadradas de -1,indicadas com i, j e k. Isto significa que $j * j = -1$ *e* $k * k = -1$. *Então um quaternião pode ser representado por:*

$$q = w + xi + yj + zk$$

Em que w é um escalar, e x, y e z são números complexos.

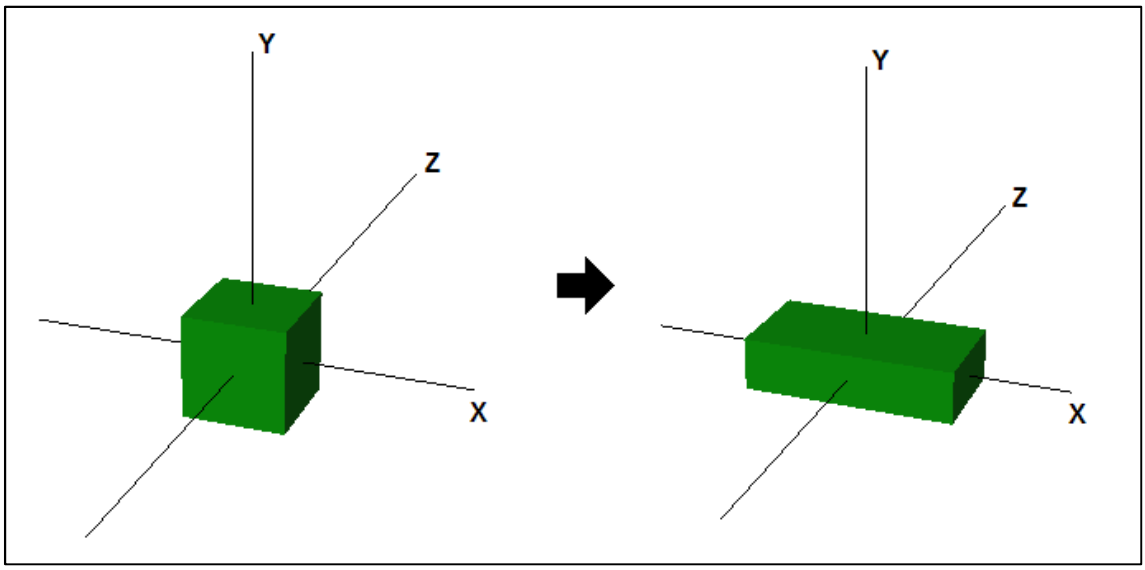


Imagem 11 – Escala

Alterar uma escala refere-se ao aumento ou diminuir o tamanho do componente vector ao longo da direcção do eixo.

8.1.1.3 Backface culling

Em computação gráfica, backface culling *determina se um determinado polígono de um objecto é visível. É um passo no* pipeline *gráfico que testa se os pontos do polígono aparecem no sentido dos ponteiros do relógio, ou ao contrario no ecrã. Se for especificado que os polígonos de face virada para a frente tem um sentido de ponteiro do relógio, e o polígono desenhado no ecrã tiver ordem do sentido contrário dos ponteiros do relógio, este não será desenhado.*

Esta funcionalidade torna o processo de rendering *mais rápido e mais eficiente ao reduzir o número de polígonos que será necessário desenhar. Uma bola de futebol, por exemplo, não há a necessidade de desenhar a parte de dentro da bola, esta parte irá estar oculta pelos lados que a serão realmente visíveis.*

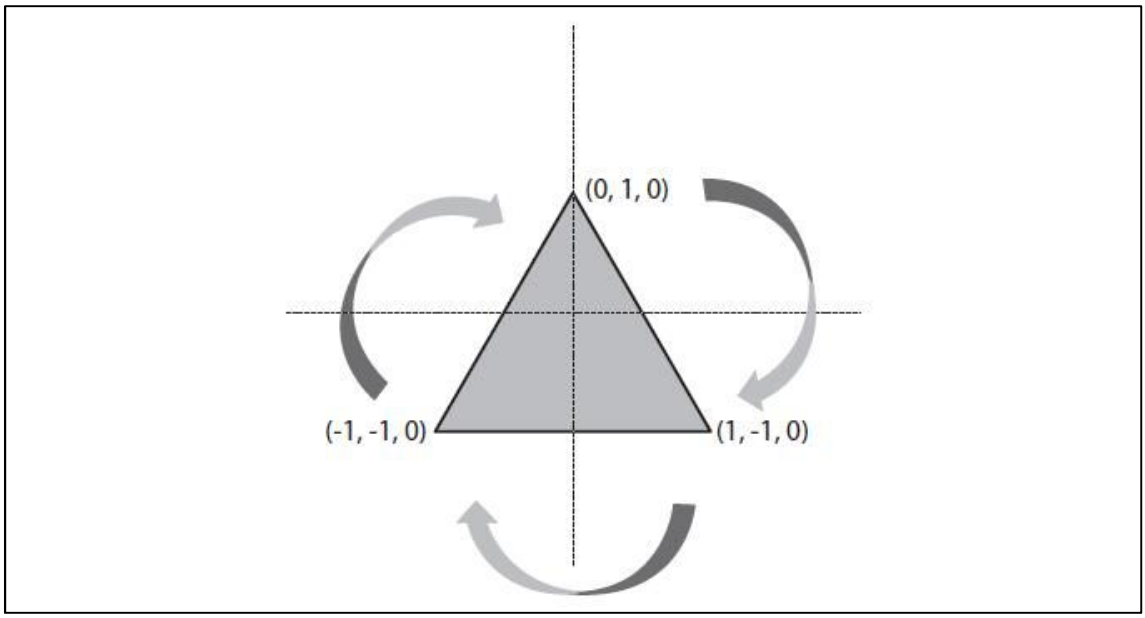


Imagem 12 – Desenhar vértices no sentido dos ponteiros do relógio

5.1.2. Modelos 3D

Enquanto é seguramente fácil desenhar alguns triângulos em 3D, como seria com uma nave espacial ou um dragão? Seria imensamente difícil definir uma forma como esta apenas definindo centenas ou milhares de vértices de triângulos.

Tipicamente para desenhar objectos mais complexos é usado modelos 3D. Essencialmente, um modelo 3D é uma colecção de pontos que formam vértices para triângulos. É possível aplicar texturas e cores no modelo. Estes modelos são criados fora do XNA em aplicações como 3D Studio Max, Maya, Blender, Lightwave ou Milkshape.

Os modelos criados nestas aplicações podem ser gravados em vários formatos para compatibilidade com as diferentes aplicações. XNA suporta o format .X e o .FBX para modelos 3D. Ao carregar e desenhar estes modelos num projecto XNA permite desenhar e manipular gráficos complexos sem a preocupação de especificar cada vértice e textura.

Para tal adicionamos o modelo ao conteúdo do projecto para podermos aceder a este elemento. Se o modelo necessitar de texturas adicionais, é necessário também importa-las para o conteúdo do projecto ou irá resultar num erro de compilação.

Neste projecto foi utilizado um modelo skysphere.fbx para simular a existência de um céu à volta do mundo.

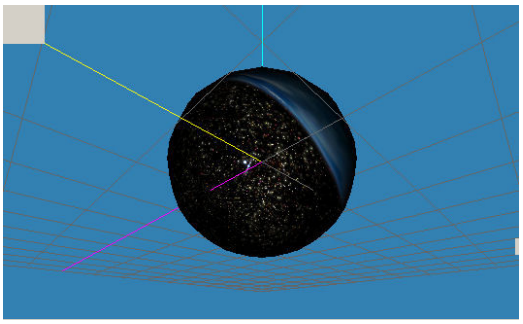


Imagem 13 – Modelo da esfera do céu

O modelo é carregado do conteúdo do projecto para um objecto do tipo **Model** *para depois ser enviado para o* shader*, juntamente com as matrizes da câmara, para poder ser visualizada. A matriz correspondente à escala da esfera foi aumenta 50 vezes de forma a poder englobar o mapa de jogo completamente e ainda dar o efeito de distância ao céu.*

5.1.3. Geração dinâmica de terreno

A raiz de todos os desenhos 3D é o triângulo. Praticamente tudo pode ser desenhado em 3D usando triângulos, até esferas. Para desenhar um triângulo, é necessário definir alguns pontos, ou vértices. Para criar vértices em XNA temos duas opções, **VertexPositionColor** *e* **VertexPositionTexture***. O primeiro cria um vértice com uma cor, e o segundo cria um vértice com as coordenadas de uma textura. Existe também* **VertexPositionNormalTexture** *que adiciona o vector normal à superfície.*

É necessário especificar algo como uma **VertexDeclaration***, que é essencialmente, algo que diz à placa gráfica que iremos enviar alguns dados, e especifica o tipo de dados que iremos enviar.*

Depois de termos a textura carregada no nosso projecto, definimos os três vértices que armazenaremos num array*. O formato dos vértices será* **VertexPositionTexture***. Definimos então os três vértices no espaço 3D, com o cuidado de estarem na ordem dos ponteiros do relógio, para serem desenhados.*

Depois de definirmos a sua posição, teremos de definir quais os pontos da textura a que vai corresponder o vértice. Estes pontos não são mais que as coordenadas X e Y da imagem.

Agora que podemos importar imagens simples para o projecto em XNA e coloca-las em triângulos, iremos criar um quantidade enorme de imagens. Com isto, iremos criar um método para o computador definir todos os vértices.

Primeiro, no jogo 3D será criada uma base de 20 por 20 quadrados. Isto significa 400 quadrados, 800 triângulos e 2400 vértices, apenas para a base. Como teremos de definir os vértices apenas uma vez, iremos armazena-los na RAM da placa gráfica ao enviar para um **VertexBuffer**.

A estrutura que irá definir o terreno será a seguinte:

```
private void CarregaAreaJogo()
{
    areaCidade = new int[,]
    {
        {0,0,0},
        {0,1,0},
        {0,0,0},
    };
}
```

Com esta estrutura, um 0 significa desenhar apenas a base, e 1 significa desenhar adicionalmente quatro paredes para formar uma parede (cubo).

Como exemplo, a geração da primeira base implicaria criar um triângulo com os vértices (0, 0, 0); (0, 0, -1); (1, 0, 0). O segundo triângulo irá coincidir com dois dos vértices do primeiro triângulo, mas não podemos esquecer a ordem. Por isso, o segundo será formado com os vértices (0, 0, -1); (1, 0, -1); (1, 0, 0).

Como podemos observar as únicas coordenadas que variam são o X e Z, uma vez que o Y que representa a altura vai estar fixo a zero pois trata-se da base.

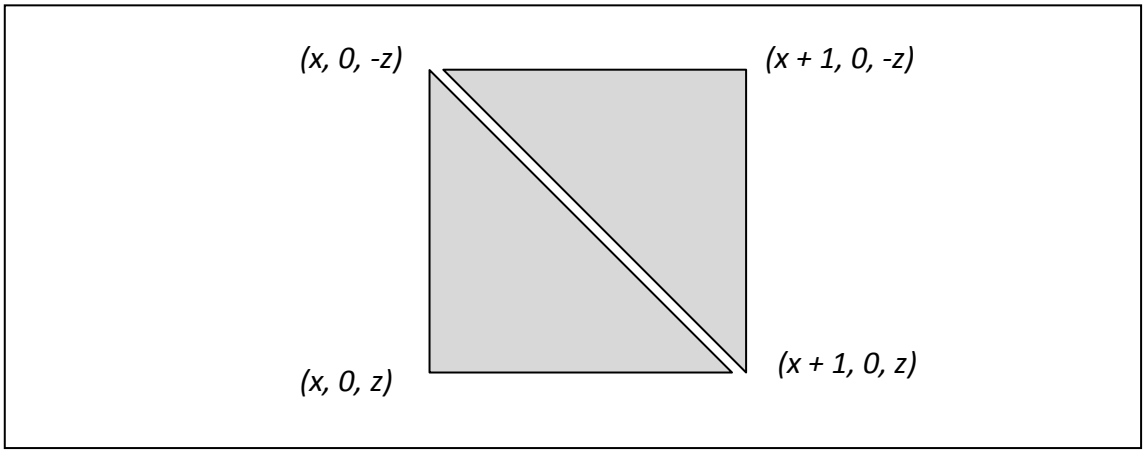


Imagem 14 – Exemplo de geração da base

Teremos de enviar também um vector a indicar a normal a este plano. Quando criamos um objecto que seja plano, posicionamos os vectores normais a apontar na perpendicular da superfície. No caso da base tem como vector (0, 1, 0). O XNA usa a normal de um plano para calcular o ângulo entre a fonte de luz e a superfície. Calcula os valores da intensidade da cor para os vértices e interpola-os para todos os pontos na

superfície. O XNA calcula a intensidade de luz usando o ângulo. Quanto maior o ângulo, menor luz brilha na superfície.

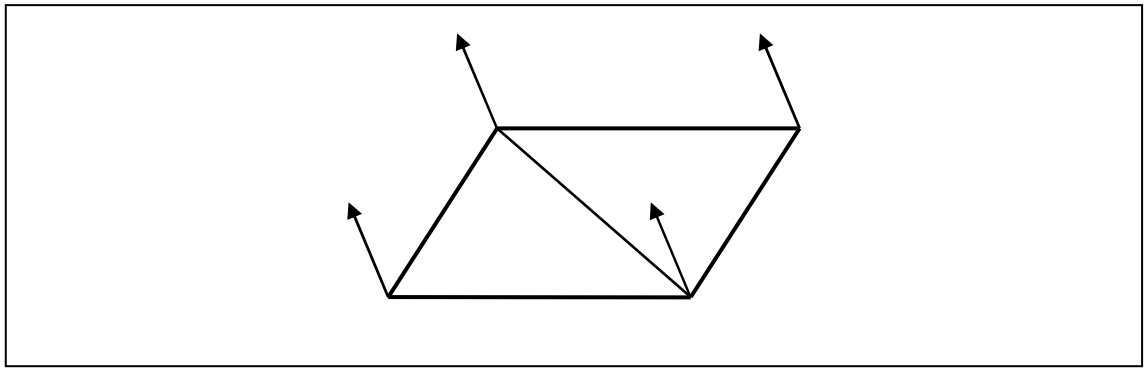


Imagem 15 – Vector normal a um plano

O algoritmo de geração dinâmica primeiro verifica a largura e altura do mapa. Depois cria uma **List** *capaz de armazenar* ***VertexPositionNormalTexture****. A maior vantagem em usar listas, é que não temos de definir à partida o tamanho do mapa, tornando-o mais dinâmico.*

Em seguida percorremos o conteúdo do array*, e sempre que encontrar um 0, iremos entrar num ciclo para criar seis vértices na lista, para formar dois triângulos. Transformamos a lista num* array *para que se possam copiar todos os vértices para a memória da placa gráfica.*

Ao percorrer do conteúdo do array, se encontrar um 1, iremos criar oito novos triângulos de forma a definir quatro novos quadrados para montar um cubo. Desenhar as paredes implica reutilizar o código de geração da base, mas com outras coordenadas, normais e coordenadas de texturas.

5.1.4. Publicidade

O algoritmo de geração de publicidade baseia-se na geração dinâmica de terreno. Partimos de uma estrutura array *que deve ter o mesmo tamanho que a estrutura do mapa. No entanto, este algoritmo não irá construir a base do mapa. O objectivo deste algoritmo é criar um género de placards que irão conter publicidade e que estarão por cima das paredes. Da mesma maneira que a geração dinâmica de terreno gera uma parede por cada 1 que encontrar na estrutura, este algoritmo quando encontra 1 desenha quatro placards nessa parede, com a publicidade desejada.*

Este algoritmo assenta na geração dinâmica de terreno, em que cada rectângulo irá estar afastado da parede de forma a ser visível. No caso deste algoritmo

esse valor ficou de 0,005 (tendo uma parede um comprimento de 1,000). Verificou-se que um valor abaixo deste provocaria conflitos acerca de qual a textura seria desenhada primeiro, se a da parede se a publicidade. Um valor acima poderia dar a ideia de a publicidade estar a flutuar em pleno ar.

O algoritmo foi também alterado no sentido em que o cartaz não irá ser começar a ser desenhado a partir do chão. Nem terá a largura da parede completa. A textura que compõe o cartaz irá começar com uma altura de 0,4 e termina nos 0,6, numa parede que mede 1,0 por 1,0. Este valor é variável e poderá ser utilizado para utilizar proporções de imagem que não o habitual quadrado.

A textura a ser usada como publicidade não está fixada no projecto. Isto é, não foi importada para o projecto. O objectivo é que a publicidade seja facilmente alterada sem que seja necessário alterar o jogo em si.

Para isso foi criada uma estrutura em XML com a seguinte forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <!-- Publicidade -->
    <pub value = "3">
        <ficheiro>easports.jpg</ficheiro>
        <altura>120</altura>
        <largura>121</largura>
    </pub>
</root>
```

Em que é definido uma publicidade na tag **pub** *indicando o seu caminho absoluto na* tag **ficheiro** *e a sua altura e largura nas* tags *respectivas.*

Com esta estrutura criada, criamos um objecto do tipo **XmlDocument** *para carregar todos os nós* **publicidade**, *armazenando todos os nós* **ficheiro**, **altura** *e* **largura** *em listas de nós do tipo* **XmlNodeList**. *Como nesta fase apenas iremos usar o caminho absoluto do ficheiro, recolhemos todos os conteúdos das* tags **ficheiro** *para uma lista auxiliar de* **strings**.

Iremos depois criar a textura para usar no algoritmo de geração de publicidade usando o método **Texture2D.FromFile.**

Com este algoritmo é possível visualizar vários cartazes publicitários em várias partes do mapa, alterando apenas a estrutura array do mapa publicitário.

5.1.5. Rede

*Apesar de o estilo do jogo ter-se modificado ligeiramente, devido a alterações de requisitos por parte da empresa, a parte Multi-Jogador manteve-se praticamente inalterada. No lugar do ponto de visão do jogador estar dentro do modelo (*First-Person-Shooter*), este irá encontrar-se atrás do modelo (*Third-Person Shooter*). Em termos de rede o mecanismo de comunicação de informação entre máquinas não é minimamente alterado.*

Os videojogos em rede contêm dificuldades extra em relação aos jogos programados para serem jogados localmente. Isto porque têm que existir preocupação com as mensagens que são recebidas do servidor, ou de outros jogadores (dependendo da arquitectura), enviar mensagens de volta para estes, o input do jogador local e o processamento e cálculos de física do jogo. Tudo isto sem deixar que o ecrã congele de uma frame para outra.

A Framework XNA possui bibliotecas destinadas a alguns destes tipos de problemas e ajudam-nos a abstrair de grande parte dos detalhes de baixo nível, mas infelizmente, a Microsoft exige que os jogadores sejam registados e que paguem licença para poder jogar através da internet aos jogos que são construídos com auxílio da sua biblioteca. Esta restrição é grave, visto que como foi declarado anteriormente esta aplicação tem que ser grátis e tem que ser possível jogar em rede, caso contrário o jogo não faria sentido.

Posto este problema, procuram-se alternativas à biblioteca de rede da Framework XNA, alternativas estas que se revelassem tão robustas e potentes em termos de funcionalidades como a biblioteca de rede do XNA, e ao mesmo tempo de baixo custo em termos comerciais. Também não podemos esquecer que a nova biblioteca teria de ser compatível com o código já desenvolvido, ou seja, as tecnologias teriam de continuar as mesmas.

A ferramenta escolhida foi a "Lidgren", uma biblioteca de rede para a .NET Framework da Microsoft que utiliza socket com o protocolo UDP para efectuar conexões, leitura e envio de mensagens. Trata-se de um projecto de código aberto totalmente gratuito e compatível com as ferramentas já usadas e, apesar de não ser tão refinada para videojogos como a biblioteca do XNA, desempenha muito bem a função. Tem como principais características a capacidade de confirmação ou não confirmação da chegada das pacotes de rede ao destino, a entrega de pacotes de rede segundo a ordem de envio ou descartando a ordem de envio, estatísticas de conexão, capacidade de implementação de arquitectura Peer-to-Peer *e simulação de latência para testes, são algumas das características mais importantes desta ferramenta.*

Depois de se ter encontrado a ferramenta perfeita para trabalhar este módulo da aplicação, o próximo passo é a escolha da arquitectura a usar na comunicação entre as máquinas envolvidas no processo de jogo. Existe dois estilos de arquitecturas de rede que encaixam bem no que toca à construção de redes em videojogos, são elas a arquitectura Peer-to-Peer *e Cliente / Servidor. Como a ferramenta escolhida não nos restringe a escolha, podemos organizar o nosso código de rede segundo a arquitectura que se venha a revelar mais vantajosa.*

Nas conexões Peer-to-Peer *os jogadores estão informados de todos os outros através de mensagens enviadas e recebidas de todos os participantes, ou seja a nossa máquina encontra-se em contacto com todas as outras e todas as outras estão em contacto com a nossa máquina.*

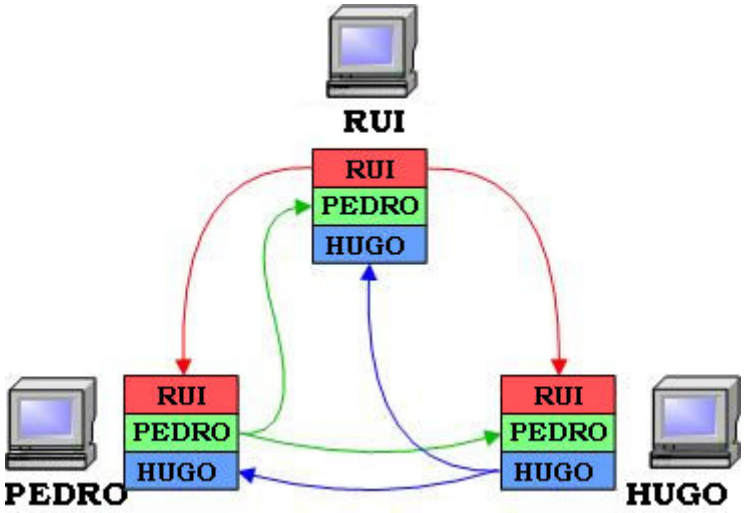


Imagem 16 – Peer to Peer

*A maior vantagem de usar esta organização na nossa rede é a de que não ser necessário um servidor dedicado e de grande capacidade de resposta para jogar, bem como o facto de se um elemento da rede cair, os outros continuam a jogar como se nada tivesse acontecido. O principal problema acontece quando o número de jogadores aumenta, uma vez que o número de mensagens na rede irá incrementar muito rapidamente. Por exemplo quando a máquina do Hugo precisa de actualizar o seu estado na rede, esta envia 2 mensagens para a rede (para o Rui e para o Pedro), isto porque a rede tem 3 jogadores. Hipoteticamente descrevendo, se durante o jogo precisarmos de mudar de estado 10 vezes serão 10 * 2 mensagens, mas, se tivermos 9 jogadores seriam 9 * 9 mensagens e assim sucessivamente. Partindo do princípio que cada mensagem possui dezenas de bytes e a rede é um recurso escasso, iremos consumir grandes percentagem de largura de banda cada vez que é adicionado um jogador à sessão. Normalmente este tipo de organização de rede não pode exceder os*

10 jogadores. As consequências de ultrapassar este número são o iniciar de congelamentos de ecrã.

Outra tipologia de rede muito usada para jogos é o Cliente / Servidor. Neste tipo de rede, todos os jogadores estão conectados a um servidor (podendo ou não ser jogador) que recebe mensagens, processa informação e envia de volta o estado de jogo a todos os clientes, de uma forma correcta e sincronizada, como podemos observar na figura seguinte.

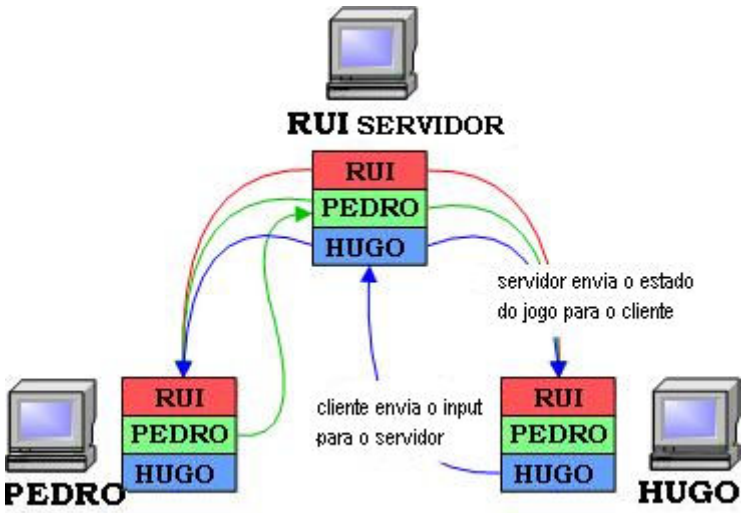


Imagem 17 – Cliente / Servidor

A metodologia Cliente / Servidor consome muito menos largura de banda por jogador, o que permite enviar um pouco mais de informação nos pacotes e tornar o jogo mais rico em termos de funcionalidades durante o desenvolvimento. Por outro lado se o servidor tiver algum problema e deixar o jogo, todos os jogadores vão imediatamente a baixo e é impossível de continuar o jogo.

Do ponto de vista do programador torna-se mais flexível programar um jogo segundo a metodologia Cliente / Servidor, uma vez que se tem o poder de decisão sobre que acções têm lugar na máquina cliente e que acções têm lugar na máquina servidor. Isto é, seria melhor colocar todo o cálculo físico na máquina do cliente e o servidor fazer unicamente distribuição de informação, ou por outro lado programar-se o servidor de forma a calcular grande parte das acções cruciais de jogo e o cliente apenas desenhar no ecrã e receber inputs… cada jogo é um jogo e depende muitas variáveis (quantos jogadores em media irá receber cada servidor, o quanto irá custar ao processador a calcular uma determinada tarefa do jogo, o custo de processamento da tarefa na máquina do cliente, a largura de banda que essa informação ocupa, etc.). De acordo com a maior parte dos autores analisados, existe necessidade de aproximação gradual a um ponto de equilíbrio entre o que é processado no cliente, o

que é processado no servidor, o que pode ser enviado pela rede para o servidor e o que o servidor pode enviar pela rede para o cliente.

Posto isto, foi requisitada a opinião de todos os intervenientes no projecto, sobre os factos analisados das diferentes arquitecturas de rede. Todas as partes envolvidas no projecto concordaram que as vantagens de uma arquitectura cliente/servidor iriam ser benéficas para o tipo de projecto que está a ser desenvolvido. Importa também realçar que de acordo coma empresa, não se pretende que o elemento de rede que efectua o trabalho de servidor possa participar no jogo com os outros clientes, Pretende-se apenas que o servidor se dedique exclusivamente a cálculos e não efectue geração de imagem de jogo.

Decidida a arquitectura a utilizar, investigou-se seguidamente os tipos de jogo em rede que existem na actualidade, Turn-based Games *e os* Real-time Games*, a diferença entre eles é tão profunda que ao escolher um estilo de jogo como* Third-Person-Shooter*, implicitamente escolhe-se o tipo de jogo em rede, seguidamente ir-se-á analisar o porquê deste facto.*

No tipo de jogo em rede Turn-Based*, cada jogador decide a sua acção, seguidamente o controlo do jogo passa para o adversário e é a vez de este decidir a sua acção no jogo. Ao descrever este tipo de jogo em rede, pensasse imediatamente em jogos como monopólio, o xadrez e todos os jogos de cartas. Este tipo de jogo em rede é substancialmente mais fácil de programar de acordo com os autores analisados, não precisando de grandes preocupações com a latência da rede inclusive. Neste tipo de jogos não é aconselhável ter muitos jogadores, uma vez que o tempo de espera de um jogador até jogar novamente pode levar a um grande desinteresse.*

A elaboração do tipo Real-Time Network Game *é bastante mais desafiante, isto porque necessitamos de transferir enormes quantidades de dados num curto espaço de tempo, sabendo-se que a rede é um recurso escasso que se consome rapidamente, tem de se efectuar um trabalho de optimização sério de maneira a prevenir um eventual atraso que prejudique a experiencia de jogo. Ao mesmo tempo temos de garantir que toda a informação está sincronizada; adquire uma especial importância nos jogos em que os jogadores lutam entre si.*

Como é perceptível pelas descrições acima enunciadas, a aplicação a ser desenvolvida enquadra-se perfeitamente no tipo de rede Real-Time*. Pretende-se uma optimização rigorosa do tráfego de pacotes na rede de forma a evitar atrasos, assim como uma proposta algorítmica de forma a prevenir eventuais atrasos inesperados, para que a experiencia de jogo não seja afectada minimamente.*

Durante o desenvolvimento da rede do videojogos multi-jogador, tiveram-se em conta práticas de trabalho para prevenir os problemas mais comuns neste tipo de projecto.

Para projectar uma boa rede é preciso detalhar ao máximo as ideias que pretendemos implementar, isto porque, vamos ter diferentes indivíduos (máquinas) a interagirem e comunicarem um com o outro por um corredor (rede). Torna-se imperativo descrever cada pacote que chega a cada indivíduo e o que este tem que fazer com a informação que lhe chega. Deve-se definir o "onde" e "quando" cada processo vai ocorrer, para garantir um perfeita sincronização. Por exemplo, se no nosso caso a sincronização não for garantida pode acontecer que numa das máquinas um jogador visualize um seu tiro a acertar no adversário, mas na outra máquina o adversário conseguir esquivar e ver o tiro a passar ao seu lado. Temos que prevenir este tipo de situações tendo em conta que não podemos utilizar muita largura de banda, uma vez que é um recurso escasso na internet.

No videojogo multi-jogador, torna-se essencial codificar desde o primeiro minuto a pensar no desempenho da rede. Em todas as decisões sobre o jogo tem que se ter em conta as consequências que estas podem trazer para a comunicação em rede. Mesmo que a ideia seja criar uma pequena versão de testes em Single Player *primeiro, convém que algumas partes do código sejam criadas já a pensar na futura rede, por exemplo, devemos isolar procedimentos que trabalham com os dados de entrada do utilizador do resto do jogo, para que se possa alterar estes mesmos procedimentos para receber dados de entrada remotamente.*

Como já foi mencionado anteriormente, a largura de banda é um recurso extremamente escasso e tem que aproveitado da melhor maneira. Isto leva a um trabalho bastante rigoroso que por vezes pode parecer exagerado para profissionais que não estejam nesta área, mas que é de extrema importância para atingir boas performances na troca de informação entre as máquinas conectadas. Depois de se discutir e analisar todas as mensagens que necessitamos de enviar pela rede (o mínimo possível), passamos à parte de analisar o conteúdo de cada mensagem (pacote de dados), com o intuito de confirmar se estamos a usar o mínimo de espaço possível. Para efectuar este trabalho com eficácia necessitamos de conhecer bem o tamanho dos tipos de dados da linguagem de programação que estamos a utilizar e escolher sempre o tipo que satisfaz as nossas pretensões e que simultaneamente ocupe menos espaço. Por exemplo no lugar de utilizar **string** *(20 bytes) para comunicar inputs, utiliza-se* **bool** *(1 byte) que ocupa 20 vezes menos espaço.*

Como foi referido anteriormente, a aplicação servidor irá estar separada da aplicação cliente, logo não existe necessidade de desenhar qual queres imagens do

jogo na aplicação servidor. Pretende-se que exista alguns parâmetros variáveis que o utilizador pode personalizar, tais como, número máximo de jogadores que se poderão conectar, nome do jogo na rede e a porta a que é feita a conexão.

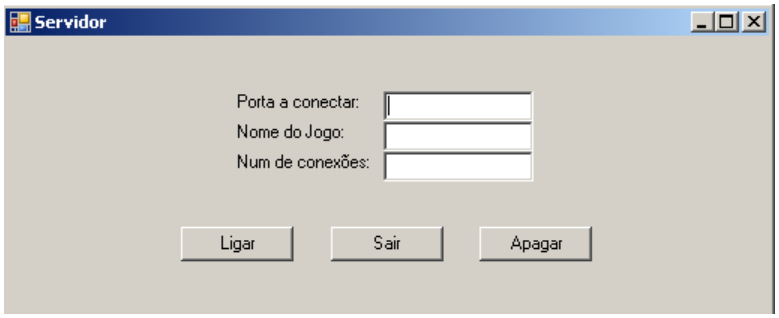


Imagem 18 – Menu Servidor

Para mais informações consultar Manual Técnico (em anexo) que contem em maior detalhe toda a programação e questões técnicas da plataforma desenvolvida, e o Manual de Utilizador (em anexo) que contém uma visita guiada pela aplicação, que ajudará a navegar pela aplicação.

5.1.6. Billboarding

Para compreender a prática utilizada no desenvolvimento das balas no projecto necessitamos primeiro de compreender o que é a técnica de billboarding *em 3D.*

Para um mundo a três dimensões num jogo parecer mais impressionante, este necessita de conter vários objectos desenhados no ecrã. Estes objectos transmitem a ideia ao utilizador de um cenário mais realista, especialmente se o jogo tem como lugar um espaço exterior. Por exemplo, partindo do princípio que o nosso jogo 3D decorre no interior de uma floresta, iríamos precisar de centenas de árvores e arbustos para o jogo parecer minimamente realista aos jogadores.

Certamente que está fora de questão desenhar no ecrã centenas de modelos 3D de arvores e arbusto, pois isto iria estrangular completamente a nossa máquina devido á necessidade brutal de poder de processamento gráfico, tornando a pratica do jogo impossível.

É possível resolver este problema trocando o objecto três dimensões por uma imagem em 2 dimensões. No entanto, quando a câmara estiver posicionada ao lado da imagem 2D, o jogador irá constatar que é simplesmente uma imagem a 2 dimensões.

Para resolver este problema, cada imagem terá definido 2 triângulos no espaço 3D que vão desenhar a imagem, estes triângulos serão rodados para que a imagem esteja sempre de frente para a câmara.

Imagem 19 – Colin McRae Rally

Podemos verificar o acontecimento de rotação de árvores billboarding *no jogo Colin McRae Rally na imagem acima. O tempo e o espaço de captação é o mesmo como podemos observar pelo contador. Para facilitar a compreensão utilizou-se duas câmaras a apontar para a mesma* sprite*, neste caso a árvore, de posições distintas. No ecrã de cima, da imagem exemplo, podemos observar que a árvore rodou para a nossa câmara ao ponto de cortar o automóvel azul ao meio; enquanto no ecrã de baixo da imagem a* sprite *ao rodar na direcção da câmara do jogador inferior, transmite o aspecto de o carro nem sequer estar cortado.*

No que toca ao projecto a desenvolver, foi considerada a opção de um modelo 3D para representar da bala disparada pelo alien*, tal opção chegou a ser testada e apresentada ao orientador. Todavia decidiu-se que a representação por* billboarding *seria o caminho mais adequado, com o objectivo de prevenir decréscimos acentuados de performance por parte da máquina e respectiva placa gráfica. Por exemplo,*

imagine-se que se tinham juntado cerca de 10 jogadores ao jogo e cada um destes jogadores disparou 10 balas (modelo 3D), logo a placa gráfica iria ter que desenhar 110 modelos, certamente assistiríamos a um decréscimo da capacidade de geração de frames por segundo.

Para implementação da técnica, especifica-se um ponto central no espaço 3D, nesse ponto irá ser desenhada a textura em questão sempre virada para a câmara dos jogadores e reflectindo o tamanho em concordância com a distancia.

Uma imagem 2D é também conhecida no XNA como sprite*, esta apenas necessita de um ponto central para ser desenhada num ambiente 3D. Uma vez que só é necessário um ponto, vai-se consumir muito menos poder de processamento da nossa placa gráfica*

Quando é disparado um projéctil, pretende-se que este se mova para a frente continuamente, tendo em conta a direcção e posição do jogador que o disparou.

Imagem 20 – Alien Arena

5.1.7. Optimização da rede

Montada a estrutura Cliente / Servidor com as características mais acima referidas, iniciou-se uma fase de testes e refinamentos deste importante módulo do projecto.

Propusemo-nos a guiar os nossos testes pelas práticas de boa programação da Microsoft em termos de rede para jogos desenvolvidos em XNA, apesar de estarmos a usar uma biblioteca externa. Definimos então a nossa meta em termos de ocupação de largura de banda abaixo dos 64 kilobits por segundo.

Nos primeiros testes á comunicação em rede entre duas máquinas verificou-se que os números que obtivemos não nos satisfaziam de maneira nenhuma, visto que estava-se a utilizar praticamente o dobro da largura de banda a que nos propusemos utilizar para efectuar as comunicações necessárias. Logo observou-se a necessidade de elaboração de um plano que nos permitisse uma optimização da troca de pacotes.

Seguidamente efectuou-se um estudo sobre procedimentos usados por especialistas de computação em redes para delinear estratégias que nos pudessem ser úteis, na tarefa de redução da utilização de largura de banda. Decidiu-se explorar quatro interessantes tópicos, eliminação de string *dos pacotes de dados, restrição do número de pacotes enviados, mecanismo de suavização do movimento em cliente e compressão de pacotes.*

As cadeias de caracteres ou string*. Pelo facto de as cadeias de caracteres serem tão importantes em qualquer tarefa de programação, os engenheiros que criaram a linguagem resolveram dar-lhe um tratamento especial que transcende o simples objecto. Em C# um objecto do tipo* string *ocupa sempre no mínimo 20 bytes contendo um conjunto imutável de caracteres. É aqui que nasce o principal problema de comunicar em rede utilizando cadeias de caracteres, estas ocupam muito espaço.*

Estudou-se todos os tipos de dados da linguagem de programação, a fim de seleccionar os que fossem simultaneamente úteis para comunicar e pequenos no que diz respeito a espaço. A máquina cliente passou a comunicar o "input" do utilizador em variáveis booleanas (boolean = 1byte) e a enviar o estado destas pela rede. A máquina passou a converter os float *informativos do estado de jogo directamente para bytes, evitando as* strings*.*

Exemplo do envio por parte do cliente:

	Pacote de dados enviado pelo cliente para o servidor
Antes de optimização	*1 String → ocupação de 20 bytes por envio*
Depois de optimização	*6 boolean → ocupação de 6 bytes por envio*

Antes de nos iniciarmos a optimização de jogo, o método que envia o estado do jogo para o cliente enviava á velocidade do ciclo de jogo de jogo, ou seja, se existisse 60 iterações do ciclo de jogo por segundo, este método tentava introduzir 60 pacotes na rede com o estado de jogo. Esta prática é extremamente ineficaz, o que nos levou a restringir os envios de pacotes. A restrição funciona da seguinte forma: a cada 6 iterações da máquina servidor enviamos um pacote com estado de jogo para as máquinas cliente, feitas as contas os pacotes lançados na rede por segundo decrescem de 60 para 10.

Admitindo que cada pacote tem 60 bits:

	Servidor para o Cliente
Antes de optimização	*60 Bits (tamanho) * 60 Envios / Seg. → 3600 bits por segundo*
Depois de optimização	*60 Bits (tamanho) * 10 Envios / Seg. → 600 bits por segundo*

O mecanismo descrito acima de restrição de envio é muito proveitoso em termos de poupança de ocupação de largura de banda, mas, provoca um acontecimento não desejado na jogabilidade do cliente. Estamos a falar de uma experiencia de jogo em que é transmitida a sensação de que o modelo que controlamos está a mover-se de uma forma pouco suave á vista humana. Isto acontece devido ao facto de não serem enviadas todas as coordenadas de posição que estão a ser processadas no servidor, uma vez que só é recebida a posição de 6 em 6 iterações do servidor.

Com a finalidade de contrariar este fenómeno, recorreu-se a um algoritmo que tem como objectivo suavizar o movimento do modelo 3D controlado pelo cliente. Este algoritmo recorre a uma técnica matemática conhecida por interpolação linear.

A interpolação linear pode ser definida como: Uma interpolação que permite fazer a reconstituição (aproximada) de uma função, apenas conhecendo algumas das suas abcissas e respectivas ordenadas, ou seja, trate-se de uma prática que permite construir um novo conjunto de dados, a partir de um conjunto discreto de dados pontuais conhecidos.

Em termos de XNA precisa-se de 3 valores para calcular uma interpolação linear, ponto inicial, ponto final e quantia (entre 0 e 1).

O valor retornado pelo método de interpolação é gerado segundo a seguinte formula: " ***ponto final + [(ponto inicial – ponto final) * Quantia]****". A quantia sendo um valor entre 0 e 1, definida inicialmente com valor 1, é calculada segundo a seguinte formula: "****Quantia = Quantia – (1 / nº valores)*** *". Como podemos observar, se a quantia for 0, é retornado o ponto final; se a quantia for 1 é retornado o ponto inicial.*

Exemplo: Pretende-se 6 valores (nº valores) no intervalo entre 10 (Pto. final) e 20 (Pto. inicial):

Cálculo do 1º valor:

Quantia = 1 - 1/6 = 5/6;

*Int. Linear = 20 + [(10 – 20) * 5/6] = 11,6667*

Cálculo do 2º valor:

Quantia = 5/6 – 1/6 = 4/6;

*Int. Linear = 20 + [(10 – 20) * 4/6] = 13,3333*

Cálculo do 3º valor:

Quantia = 4/6 - 1/6 = 3/6;

*Int. Linear = 20 + [(10 – 20) * 3/6] = 15*

Cálculo do 4º valor:

Quantia = 3/6 - 1/6 = 2/6;

*Int. Linear = 20 + [(10 – 20) * 2/6] = 16,6666*

Cálculo do 5º valor:

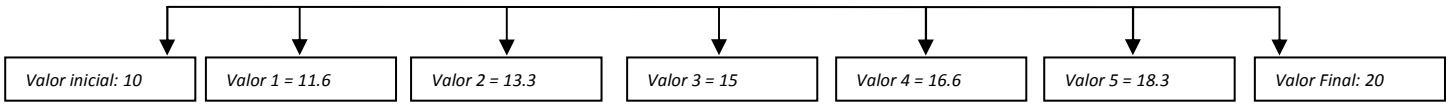
Quantia = 2/6 - 1/6 = 1/6;

*Int. Linear = 20 + [(10 – 20) * 1/6] = 18.3333*

Cálculo do 6º valor:

Quantia = 1/6 - 1/6 = 0;

*Int. Linear = 20 + [(10 – 20) * 0] = 20*



Compreendendo o exemplo anterior, fica muito mais claro entender o que foi realizado no algoritmo de suavização incluído no projecto. O Ponto iniciar e final poderia ser uma coordenada x ou y ou z de um vector posição recebido do servidor, com esta técnica facilmente pode calcular pontos intermédios e desenha-los no ecrã. Desta forma o nosso modelo passará por 6 posições em vez de duas, o que nos dá uma experiencia de jogo muito mais interessante.

Em termos de código para conseguir introduzir esta técnica, o nosso modelo Alien tem de possuir 3 Atributos de posição e 3 variáveis de rotação. Estas têm a função de guardar o estado antigo, estado recente, estado actual.

Quando é recebido por rede um novo pacote de dados com o estado de jogo, ele é colocado no estado recente e o valor que antes estava no estado recente, passa para o estado antigo. O estado actual é aquele que é desenhado no ecrã, logo é interpolado localmente usando a técnica anteriormente descrita. Fazendo a analogia com o

exemplo, ponto inicial = estado antigo, ponto final igual a estado recente e o resultado da interpolação dos 2 é o estado actual que vai ser desenhado no ecrã.

Foi adicionado também um novo atributo para controlar a "quantia" este atributo foi designado de suavidade corrente, visto que este é que controla em que parte do intervalo está num determinado instante.

A técnica de compressão de pacotes de dados antes de envio pela rede não foi implementada, visto que se concluiu que o ganho em termos de desempenho de rede comparado com a perda em termos de performance de processador (compressão/descompressão) não se justificava.

Implementadas as técnicas anteriormente referidas, foi-nos possível optimizar a ocupação de largura de forma a tornar a experiencia de jogo mais agradável, fiável e a cumprir os requisitos de ocupação que nos propusemos.

Nos últimos testes realizados o nível de ocupação de rede foi de 30 kilobits, isto representa uma redução de cerca de 70% em relação aos valores inicialmente obtidos.

5.1.8. Animação Tridimensional

O cenário de jogo é principalmente constituído por objectos estáticos, por isso para aumentar o realismo das acções pode-se acrescentar alguns modelos animados á mecânica do jogo. Pode-se criar animações de diferentes formas. Por exemplo, nos jogos de corridas o carro deveria ser um modelo animado porque convêm as rodas rodar quando o veículo se desloca. Logo pode-se rodar facilmente as rodas em torno do eixo dos XX.

Mas se for necessário animar uma personagem com movimentos de corrida, saltos, marcha, etc. Nestes casos há necessidade de alteração da malha do modelo, o que torna o processo muito mais complicado.

Existe duas principais técnicas de animação, a animação por frames " e a animação põe esqueleto. Cada tipo de animação é usada em diferentes situações e cada uma delas tem as suas vantagens e desvantagens.

Na animação por frames, guarda-se uma malha estática para cada frame da animação. Este tipo utiliza diversas frames intermédias para fazer passar a sensação de suavidade.

Uma das vantagens da animação por frames *é a sua rapidez, porque nada necessita de ser calculado durante a animação. Todos os* frames *estão guardados em memória e durante a animação apenas necessitamos de trocar a frame que é*

desenhada no ecrã. Uma das desvantagens desta prática é ter de se guardar todas as frames em memória, se o modelo tiver centenas de animações, estas vão ocupar muita memória.

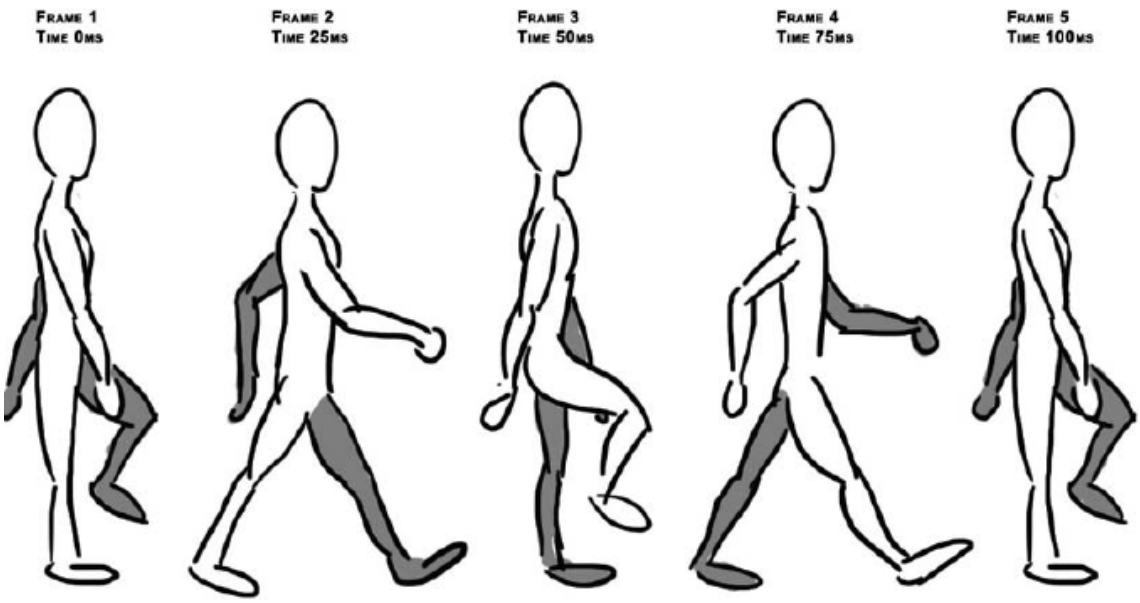


***Imagem 21 – Animação por* frames**

Uma outra maneira de animar um modelo é pelo seu esqueleto. Neste processo, precisa-se de construir um modelo para o esqueleto em questão, composto por ossos, e depois conectar todas as malhas de vértices nos ossos do esqueleto. Desta forma quando o osso é animado, a malha associada também o segue.

Para construção de um modelo com esqueleto associado às malhas, pode-se usar programas como o Blender, 3d Max, Maya, entre outros. Depois de criar o modelo, é necessária a exportação para formato que suporte a animação do esqueleto. A Framework XNA suporta de um modo nativo os formatos DirectX e FBX.

A animação por esqueleto tem mais vantagens que a animação por frames*. Esta permite que as animações sejam facilmente combinadas, permitindo a aplicação de diferentes animações sobre o modelo, ao mesmo tempo. Por exemplo, pode-se aplicar duas animações diferentes para o modelo, onde uma animação fará o modelo andar e outra animação faria o modelo olhar em volta (rodando o pescoço). A animação por esqueleto permite ainda a um osso de um objecto para ser conectado a um osso de outro objecto. Por exemplo, se tiver uma personagem a empunhar uma espada, será possível conectar o osso da espada ao osso da mão, o que fará a espada mover-se à medida do movimento da mão.*

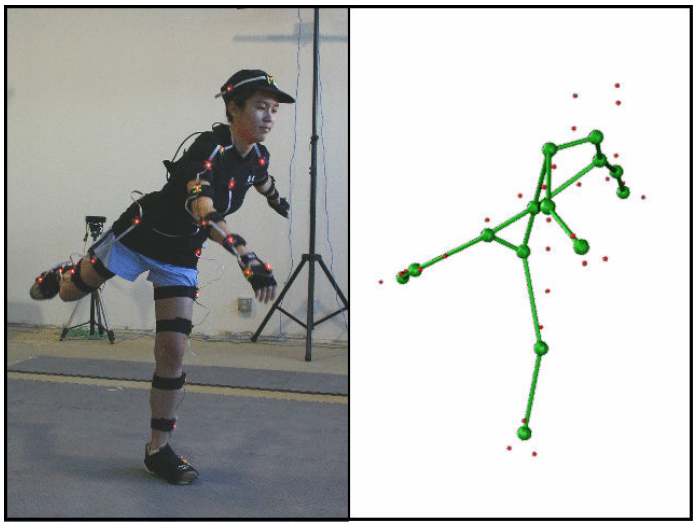


Imagem 22 – Motion Capture

A Framework XNA fornece um suporte parcial em termos de animação. Esta define um objecto intermédio para guardar os dados de animação dentro do contentor, estes dados podem ser importados de formatos FBX ou DirectX.
Infelizmente a Framework XNA ainda não possui classes que ajudem a trabalhar com animações em tempo real.

Foi proposto ao orientador do projecto pelos alunos a utilização de uma biblioteca de apoio fornecida pela Microsoft, esta biblioteca possui classes que nos facilitam o controlo das animações. Desta forma é reproduzida a animação da personagem principal do jogo, a animação de movimento do Alien, já incluída no modelo utilizado.

5.1.9. Áudio

Nesta secção vai-se compreender as técnicas utilizadas para adição de som ao jogo Alien Arena. Para isso explorar-se-á alguns dos conceitos básicos da XNA Framework. O funcionamento do áudio na XNA Framework é semelhante ao funcionamento dos gráficos, o som é apenas outro tipo de conteúdo.

Mas existe uma diferença: podemos adicionar gráficos directamente ao contentor enquanto no som é preciso ajustar o tipo de formato, gerado pela Microsoft Cross-Platform Audio Creation Tool, conhecida por XACT.

Para reproduzir áudio em XNA é necessário criar um novo tipo de projecto XACT, e adicionar os ficheiros de áudio pretendidos a este projecto. Seguidamente adiciona-se o ficheiro do projecto XACT ao contentor.

Para aceder à reprodução de sons "wave", é necessário criar três objectos fundamentais:

AudioEngine - *é usado para ajustar parâmetros de som*

WaveBank - *é a colecção de sons "*wave*" que estão no projecto XACT*

SoundBank - *detalhes de reprodução dos ficheiros.*

Em termos de som, decidiu-se incluir a reprodução de passos durante a animação do modelo do Alien*, som de disparo de bala e som de morte por parte de um jogador atingido por um projéctil.*

Adicionou-se ao algoritmo de reprodução de som uma restrição temporal para a reprodução dos passos, para efeitos de sincronização com a animação que o Alien possui no seu andamento.

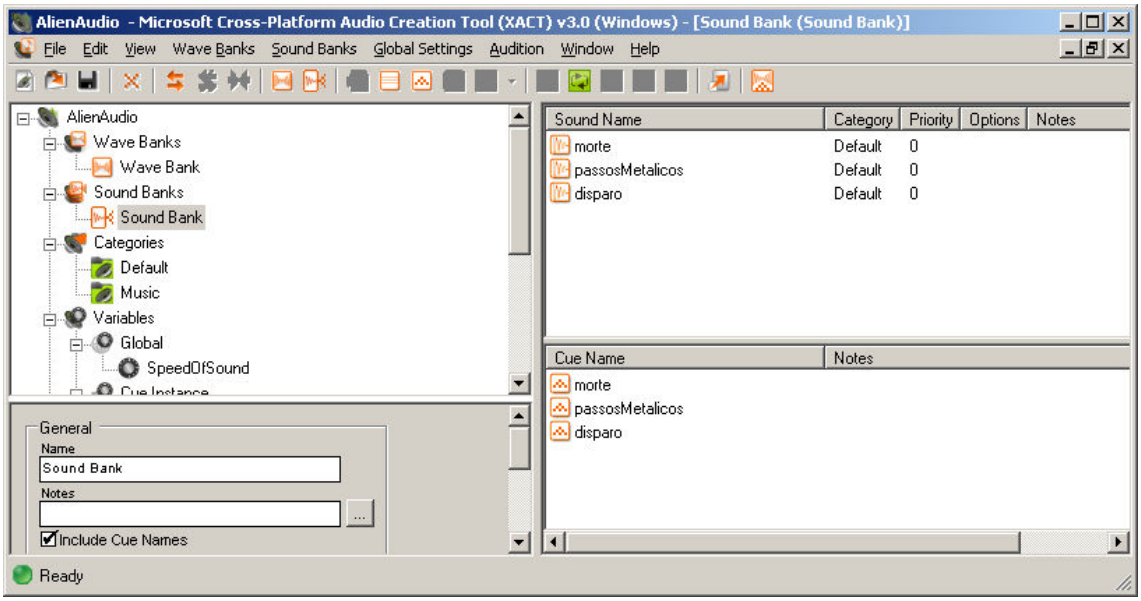


Imagem 23 – XACT 3

6. Avaliação do Projecto

Para uma avaliação qualitativa adequada deste projecto é necessário verificar se os objectivos definidos nos requisitos de qualidade são alcançados.

Os requisitos de qualidade irão definir os atributos necessários para a descrição do software. Existem quatro categorias, ***Desempenho****,* ***Disponibilidade****,* ***Adaptabilidade*** *e* ***Usabilidade****. Em cada uma das categorias acima mencionadas encontramos vários requisitos de qualidade.*

Este projecto engloba requisitos de todas as categorias.

Na categoria Desempenho encontra-se a capacidade de resposta e capacidade de processamento.

Na categoria Disponibilidade existe o requisito fiabilidade.

A portabilidade é o requisito de qualidade contido na categoria Adaptabilidade.

Finalmente, na categoria Usabilidade temos os requisitos facilidade de aprendizagem, eficiência de utilização e satisfação do utilizador.

Para verificar o cumprimento destes requisitos foram planificados vários testes (mencionados na primeira parte deste documento, secção 4.1.4).

Avaliando qualitativamente a aplicação desenvolvida, concluiu-se que:

- *Na categoria Usabilidade, o projecto cumpriu os objectivos inicialmente propostos, enquadrando-se dentro do intervalo de valores a alcançar nos testes planeados.*
- *Respeitante à Adaptabilidade, não se verificou o ponto proposto devido ao aparente falta de valor acrescentado que poderia advir de uma eventual portabilidade para uma consola XBOX 360.*
- *Na categoria Disponibilidade, a aplicação cumpriu com eficácia os requisitos estipulados.*
- *Quanto ao Desempenho, dependendo da ligação à internet utilizada, a capacidade de resposta é variável.*

Para uma avaliação do projecto, simulámos uma análise de custo:

Número de Elementos	2	Pessoas

Nº de dias úteis/mês	22	Dias
Nº meses trabalho/ano	4	Meses
Nº horas trabalho/dia	8	Horas
Ordenado individual		800,00 €
Nº de meses pagos/ano	5	Meses

Custo / Pessoa / Ano	4.000,00 €
Custo Horas / Pessoa	5,68 €

Fases de desenvolvimento	Meses	Dias / semana	Semanas / mês	Média de horas	Custo Total
Análise e Levantamento de Requisitos	2	5	4	2	909,09 €
Desenho	0,75	5	4	2	340,91 €
Desenvolvimento	1,25	5	4	5	1.420,45 €
Testes	1	5	4	4	909,09 €

Custos Estimados	Valor	Horas
Análise e Levantamento de Requisitos	909,09 €	80 h
Desenho	340,91 €	30 h
Desenvolvimento	1.420,45 €	125 h
Testes	909,09 €	80 h
Manutenção	0,00 €	0 h
Total:	3.579,55 €	315 h

Os dados acima descritos foram simulados, uma vez que não foi possível obter dados reais da evolução custo / benefício da aplicação. Assumimos que o número de placards vendidos no momento da análise seria de 30 unidades com o preço de 0,01€ por placard por jogador que entre em sessão de jogo.

A média de utilizadores por dia é de 52, num mês que contém 30 dias. Valores puramente supostos, uma vez que não existem estatísticas anteriores que nos permitam avaliar tal realidade.

Verifica-se uma rentabilidade mensal de 468€ por mês; concluímos que, com estes valores, o investimento estará completamente recuperado no oitavo mês depois do seu lançamento.

Simulação de utilização da aplicação	
Número de placares vendidos no mapa	30
Valor por participação de jogador	0,01 €
Número de dias	30

Média de Utilizadores	52	Pessoas/dia
	Total:	468,00 €

	Valor inicial	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6	Mês 7	Mês 8
Custo da aplicação	3579,55								
Valor recuperado		468,00	468,00	468,00	468,00	468,00	468,00	468,00	468,00
Total		-3111,55	-2643,55	-2175,55	-1707,55	-1239,55	-771,55	-303,55	164,45

7. Conclusão e trabalho futuro

A elaboração deste projecto proporcionou-nos uma aprendizagem na área da programação tridimensional.

Para o conseguir foi necessário um vasto trabalho de pesquisa sobre computação gráfica 3D, redes de dados de computador e a linguagem XNA para aplicação de conhecimentos. Com o estudo desenvolvido foram criados pequenos exemplos passíveis de estudo, sendo posteriormente utilizados na criação da aplicação final.

Em suma, cremos ter atingido os objectivos propostos, nomeadamente o desenvolvimento de um jogo a 3 dimensões, jogo este com capacidade para multi-jogador e possuidor de publicidade no cenário de jogo.

Ao cumprir estes objectivos certificamos que o projecto cumpre os requisitos especificados previamente, ou seja, a possibilidade de criar um jogo em rede, juntar a um jogo já existente e jogar em multi-jogador.

Consideramos a aplicação com uma utilização simples e intuitiva, graças à simplicidade dos comandos existentes.

Como trabalho futuro, é do nosso interesse melhorar o efeito visual do jogo aplicando mais efeitos de luz, anti-aliasing*, introdução de sombras, mais e diferentes modelos de Alien e mais e melhores texturas.*

A inclusão de mais efeitos sonoros também é desejável, assim como uma banda sonora constante e aplicada ao tema do jogo. A inclusão de um botão para desligar o som permite um melhor controlo da qualidade de jogo.

No futuro também seria favorável um novo algoritmo de geração de terreno, baseado em heightmaps*. A inclusão de mais e diferentes mapas iria com certeza incrementar a continuação do jogo. A existência de diferentes níveis de dificuldade seria também aprazível.*

Utilizar um chat *para comunicar entre jogadores seria também um ponto a favor no aumento de qualidade de jogo.*

Um sistema de pausa de jogo, em que o jogador estaria temporariamente fora de jogo, mas com a possibilidade de voltar com a pontuação que teria antes. Tal pausa teria de ter um tempo limite, findo este o jogador seria retirado da sessão de jogo. Isto

permite ao jogador parar temporariamente o seu jogo, sem desistir completamente de o jogar.

O sistema de leitura de publicidade está implementado, apenas fica a faltar para desenvolvimento futuro, uma gestão backoffice *das imagens para publicidade.*

8. Anexos

8.1. Manual Técnico

A aplicação foi desenvolvida na linguagem de programação C# totalmente compatível com todas as tecnologias utilizadas no decorrer do projecto.

De seguida passaremos a descrever as rotinas que desempenham as funções recepção, processamento e distribuição síncrona de pacotes de dados por todos os clientes que se encontram a participar no jogo.

Classe servidor

Atributos:

`GraphicsDeviceManager graphics;` → *Para dar acesso às definições gráficas*

`NetServer servidor;` → *guarda informação para o cliente se poder conectar*

`NetLog Log;` → *guarda informação de ocorrência de eventos no servidor*

`NetAppConfiguration nac;` → *informação de configuração, nome, porta*

`int maxGamers;` → *número máximo de jogadores que o servidor pode receber*

`List<Alien> jogoAliens;` → *lista dinâmica de Jogadores que estão ligados ao servidor*

`Cenario cenario;` → *para efeitos de colisão unicamente (saber onde estão as paredes)*

`float framesEntreEnvios;` → *intervalo de frames entre envio de pacotes pela rede*

No método Criar Sessão, é onde se localiza a inicialização dos mais importantes atributos de rede. O atributo "nac" irá receber directamente dois parâmetros introduzidos pelo utilizador, o nome a que o jogo passará a ser identificável na rede e a porta onde os clientes se ligam á máquina servidor (tem que estar desbloqueada na firewall). O atributo "log" serve registar a informação de ocorrência de eventos no servidor, muito útil para a tarefa de identificação de eventuais anomalias que possam ocorrer durante a utilização da aplicação servidor. O objecto do tipo servidor necessita da informação destes 2 últimos atributos acima descritos, de realçar a importância extrema deste atributo para a rede, este possibilita o envio e recepção de pacotes de dados relativos a todos os jogadores presentes na rede, assim como, a identificação de eventuais eventos que podem ocorrer numa comunicação de máquinas.

ServerStatusChanged é um método de evento que é disparado quando um possível cliente tenta conectar-se, é muito útil visto que foi aproveitado para distribuição de identificação de jogadores no jogo. Por outras palavras, o servidor quando verifica que outra maquina esta a conectar-se, efectua uma contagem de

quantos jogadores estão conectados para atribuir Id. (nº conexões + 1), constrói um pacote de dados com a identificação e envia-o para o cliente.

ActualizarServidor actualiza o estado do jogo, tendo em conta a informação recebida nos pacotes com origem nos jogadores clientes. Entenda-se por estado de jogo as seguintes funcionalidades, processo de translação no espaço do modelo controlado pelo cliente, processo de rotação no espaço do modelo controlado pelo cliente, mecanismo de jogabilidade do cliente. Seguidamente construímos um pacote de dados que contem o estado actualizado de todos os envolvidos no jogo, este é posteriormente enviado de volta para cada cliente. Devido ao esforço de optimização da rede, que vai ser aprofundado mais á frente neste relatório, o servidor não envia informação á mesma velocidade que actualiza o estado, isto é, por seis vezes que actualizamos o estado de jogo, envia-se apenas um pacote para os clientes com o estado do jogo.

Com vista a construir pacotes de dados robustos para enviar aos jogadores clientes da rede, surgiu a necessidade de trabalhar o máximo possível com a unidade mínima de processamento, o "byte". Como todos os cálculos de rotações e translações são feitos no tipo de dados "float" (4 bytes), antes de os enviar pela rede, surgiu a necessidade de conversão entre eles. O método FloatParaByte recebe um jogador Alien por argumento e retorna uma lista de bytes com toda a informação sobre a posição e rotação actual. Para efectuar esta conversão utilizou-se o método "GetBytes()" da classe auxiliar "BitConverter" incluída na .NET Framework, este método recebe uma coordenada float e retorna um conjunto de 4 bytes que a representam. Depois de convertida a coordenada é efectuado um ciclo onde adicionamos os bytes a uma lista, isto para todas as coordenadas de posição e rotação.

*Já foi explicado a forma como construímos os pacotes de dados e os enviamos para o cliente, para que este se mantenha actualizado, mas, para que o próprio servidor obtenha conhecimento do que se passa do lado do cliente, este tem que saber interpretar as mensagens que o cliente lhe envia. Surgiu então necessidade de desenvolver a funcionalidade **HandleMsgServer**, esta função recebe por argumento um objecto do tipo **NetMessage**. Seguidamente percorre-se todos os jogadores no jogo, efectuando-se comparação entre o índice do jogador iterado e o índice do pacote. Encontrado o jogador, Actualiza-se os atributos conforme os dados recebidos na mensagem, por exemplo: se o quinto byte do pacote de dados é igual a "true", significa que o cliente nos está a comunicar um salto.*

Descrição dos pacotes de rede enviados pelo servidor durante o jogo:

Pacote de dados / Bytes	Identificação cliente	Estado de jogo	Pacote Pontuação
0	Byte que contem o índice de registo do cliente no jogo		(255)
1		Bytes de Informação relativa a coordenada X (float) da posição no espaço do modelo controlado pelo cliente	Byte que contém o índice de registo do cliente no jogo.
2			Byte com informação do número de pontos (FRAGS) do jogador.
3			Byte com informação do número de "DEATHS" do jogador.
4			
5		Bytes de Informação relativa a coordenada Y (float) da posição no espaço do modelo controlado pelo cliente	
6			
7			
8			
9		Bytes de Informação relativa a coordenada Z (float) da posição no espaço do modelo controlado pelo cliente	
10			
11			
12			
13		Bytes de Informação relativa a coordenada X (float) necessária para criar um quaternião, objecto que nos permite trabalhar a do modelo controlado pelo cliente	
14			
15			
16			
17		Bytes de Informação relativa a coordenada Y (float) necessária para criar um quaternião, objecto que nos permite trabalhar a do modelo controlado pelo cliente	
18			
19			
20			
21		Bytes de Informação relativa a coordenada Z (float) necessária para criar um quaternião, objecto que nos permite trabalhar a do modelo controlado pelo cliente	
22			
23			
24			
25		Bytes de Informação relativa a coordenada W (float) necessária para criar um quaternião, objecto que nos permite trabalhar a do modelo controlado pelo cliente	
26			
27			
28			
29		Ultimo byte deste player ira conter info se houve disparo deste tanque	

CLASSE CLIENTE

Quando um utilizador se quer juntar a um jogo, tem que seguir determinados procedimentos, para isto foi criado o método JuntarSessao(). Este procedimento necessita de informação do endereço de rede da máquina (IP) onde se encontra a correr o servidor de jogo e a porta de acesso na máquina. Necessitamos também de um objecto do tipo "log", que guarda os eventos que vão ocorrendo durante a execução do cliente e de um objecto do tipo "nac" que contem o nome da rede. Depois disso podemos inicializar o cliente, invocar o método connect() deste e se a informação estiver correcta o servidor irá responder com a nossa identificação no jogo.

No caso de se encarnar o papel de jogador cliente, tem-se de ter em conta que a máquina do servidor necessita de obter informação sobre o que os seus clientes pretendem fazer no decorrer do desafio, para isto acontecer foi criada a rotina ActualizarJogadorLocal(). Aqui pretende-se tratar duas tarefas importantes, em primeiro lugar efectuar uma leitura do teclado do utilizador, em segundo lugar empacotar essa informação e enviar para o servidor. No que diz respeito ao input do cliente a Framework do XNA possuiu métodos auxiliares que nos permitem verificar se uma determinada tecla do teclado está a ser pressionada num determinado momento, as teclas que estamos a analisar são as necessárias em termos de jogabilidade, "space","alt", "ctrl" e setas, a informação guarda-se para posterior comunicação.

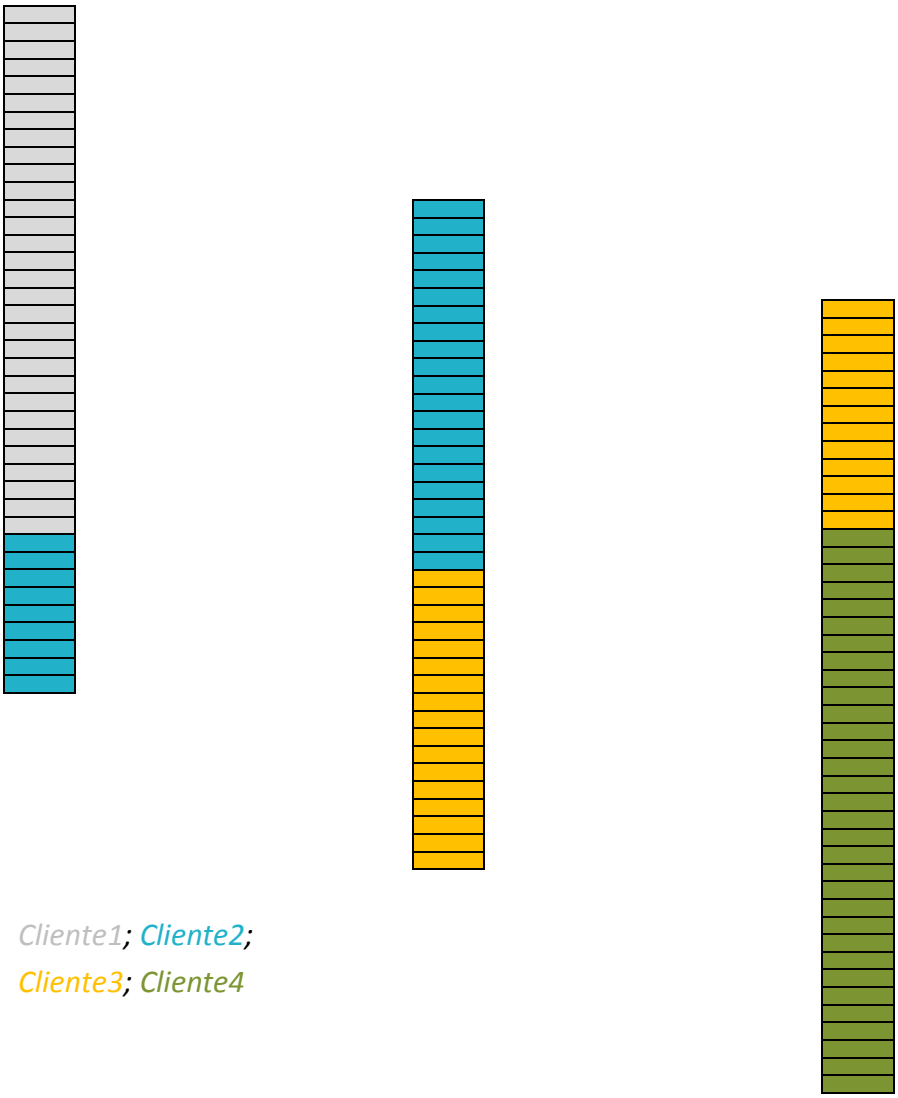
Em termos de comunicar ao servidor o estado das teclas do seu cliente, cria-se um pacote de rede com os bytes necessários e envia-se a mensagem para a rede. A estrutura deste pacote de dados é a seguinte:

Byte	*Descrição*
0	*Identificação do pacote com o índice a que o cliente está associado*
1	*Informação sobre o cursor "UP"*
2	*Informação sobre o cursor "DOWN"*
3	*Informação sobre o cursor "LEFT"*
4	*Informação sobre o cursor "RIGHT"*
5	*Informação sobre o cursor "SPACE"*
6	*Informação sobre o cursor "CTRL"*

Já se explicou a forma como o cliente envia informação para o servidor, naturalmente vamos seguir o raciocínio lógico e comentar a forma como o cliente recebe informação do servidor. Para desenvolver esta função criou-se o método "HandleMsgClient", quando recebemos uma mensagem do servidor a primeira coisa a fazer é saber o tipo de pacote de dados, existe três hipóteses: pacote identificação, pacote de estado de jogo ou pacote informativo da pontuação (já foram explicados na classe servidor). Se o pacote for de identificação, faz-se a actualização directa do atributo que guarda o índice no objecto "Alien". Se o pacote recebido é do tipo "estado

de jogo", itera-se o mesmo, durante a iteração vai-se guardando os bytes numa estrutura auxiliar. Sempre que a estrutura auxiliar atinge o tamanho de 30 bytes (informação de 1 jogador), esta mesma estrutura auxiliar é enviada para processamento. Este processamento está relacionado com a actualização do jogador com a nova informação. Este processo desenrola-se até se chegar ao final do pacote de dados, desta forma actualizamos todos os jogadores que estão no nosso jogo.

Pacote de dados "estado de jogo" (rectângulo = byte):



Cliente1; *Cliente2*;
Cliente3; *Cliente4*

Blocos cinza representam os bytes informativos do Cliente Alien com o índice 1, dentro do pacote de dados enviado pela rede com o estado de jogo. A ilustração anterior simplifica a percepção do mecanismo descrito para actualizar todo o jogo na máquina do cliente com o pacote "estado jogo" enviado pelo servidor. Blocos castanhos representam os bytes informativos do Cliente Alien com o índice 2 dentro do pacote de dados enviado pela rede com o estado de jogo, e assim sucessivamente.

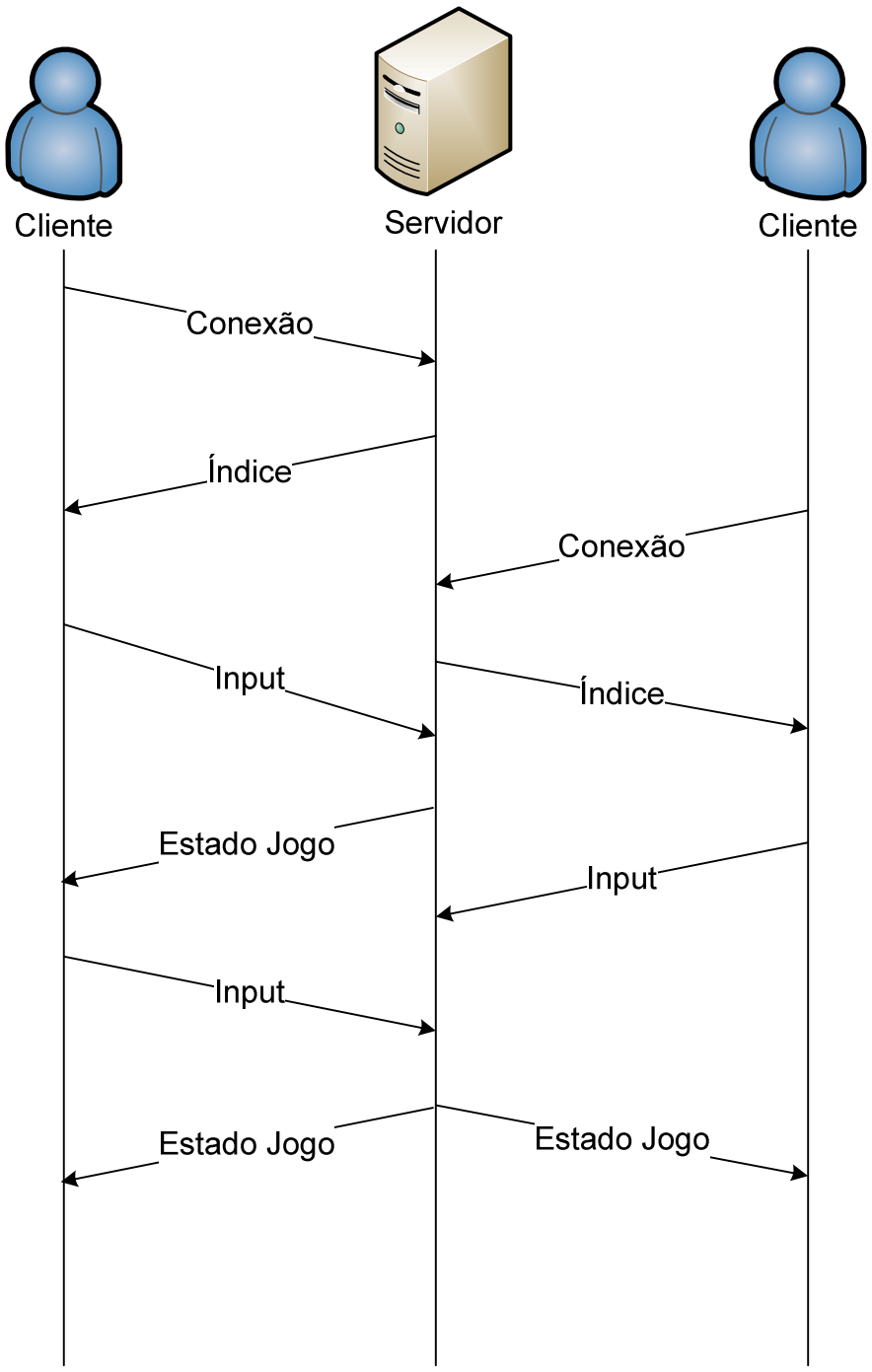


Imagem 24 – Diagrama Temporal

8.2 Manual de Utilizador

8.2.1 Aplicação Servidor

8.2.1.1 Introdução

Este manual tem como objectivo ajudar o utilizador a compreender e utilizar a aplicação. Para tal, iremos guiá-lo nas várias etapas, desde a instalação dos componentes necessários à aplicação até a um exemplo prático da mesma.

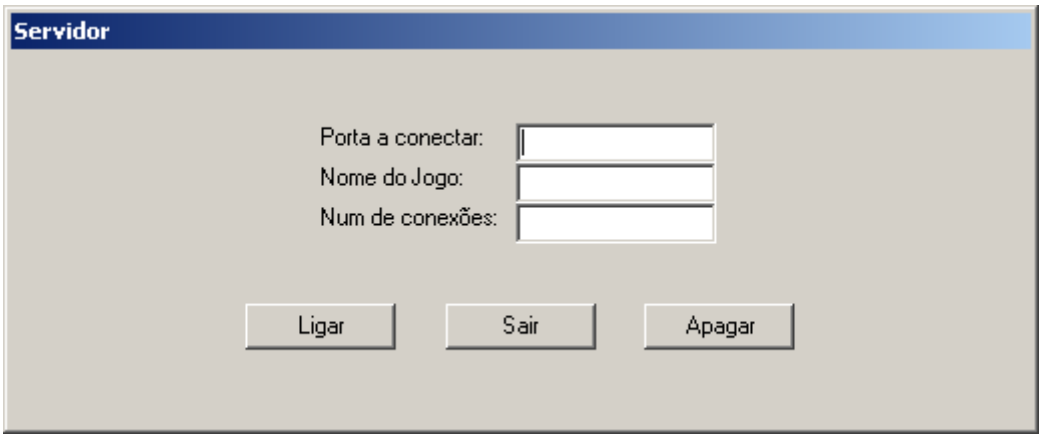


Imagem 25 – Configuração do Servidor

Para configurar o servidor, é necessário indicar uma porta válida a conectar, um nome do Jogo e o número de conexões.

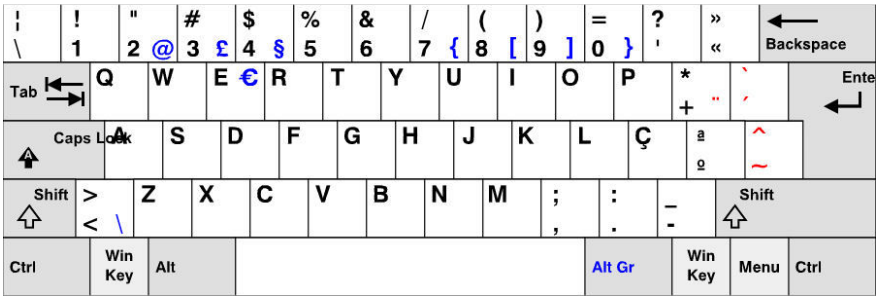
*Depois de configurado, clique em Ligar para iniciar uma sessão Servidor de "*Alien Arena*".*

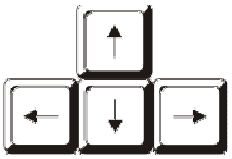
Poderá ser necessário colocar este endereço de rede fora da zona de protecção da firewall *que esteja a usar.*

8.2.2 Aplicação Cliente

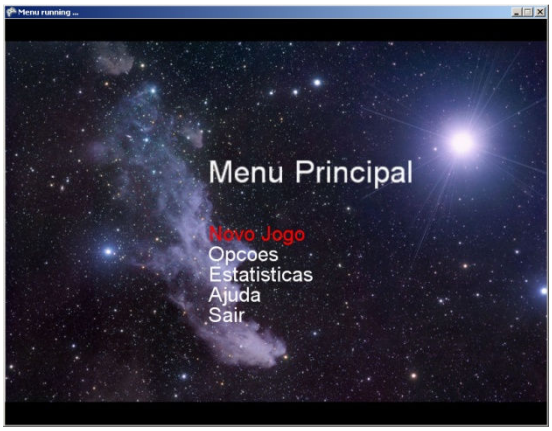
8.2.2.1 Introdução

Este manual tem como objectivo ajudar o utilizador a compreender e utilizar o jogo.



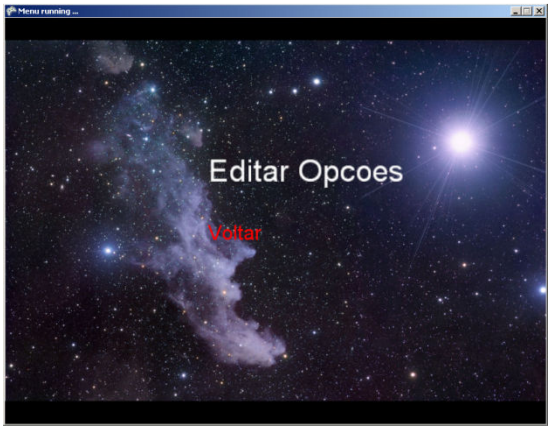


8.2.2.2 Começar



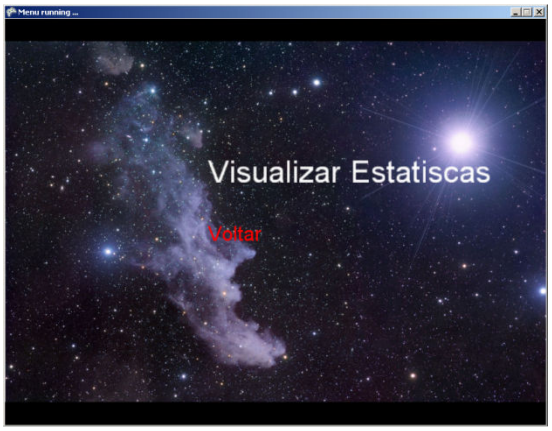
Usa as setas para mover o cursor pelas opções do menu principal. Carrega ENTER *para activar uma selecção. Inicia um novo jogo ao seleccionar* ***Novo Jogo****.*

8.2.2.3 Menu Opções



*Neste menu, poderemos alterar algumas opções, como por exemplo, o nome (*nick*) do jogador.*

8.2.2.4 Menu Estatísticas



Neste menu, é possível visualizar as estatísticas de jogo.

8.2.2.5 Menu Ajuda

Neste menu, serão dadas explicações breves sobre os modos de jogo e de como jogar.

8.2.2.6 Alien Arena

Apenas tu e o resto do mundo. Apenas estratégia e técnica apurada. A tua arma? Está incorporada em ti. Não importa seres ganancioso, as munições não acabam.

Se falhares, és automaticamente recolocado no mapa para voltares à acção.

Carrega no CTRL para disparar e ESPAÇO para saltar.

8.2.3 Aspectos Comuns

Ambas as aplicações trabalham usando .NET Framework 3.5 e XNA Framework. Se o utilizador não tiver uma ou nenhuma destas aplicações instaladas, o instalador irá requerer ao utilizador que as instale.

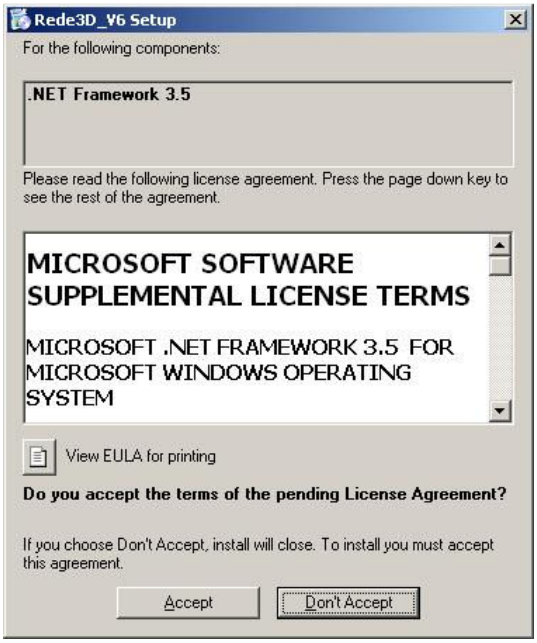


Imagem 26 – Instalação de .NET Framework 3.5

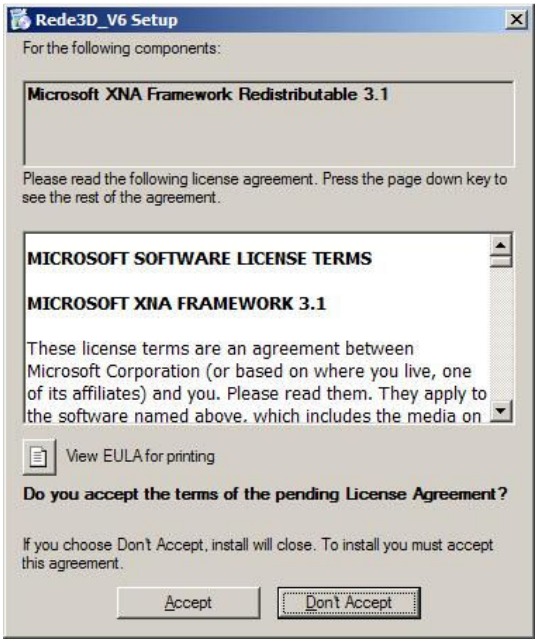


Imagem 27 – Instalação de XNA Framework

Para conseguir jogar ao "Alien Arena" é necessária a instalação de ambos os suplementos. Será, provavelmente, necessário reiniciar o PC depois da instalação de algum dos suplementos.

8.3 Resumos de Actas

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #1

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Eng. João Morais, Pedro Andrade, Rui Rodrigues*

***Data**: 2 de Fevereiro de 2009*

***Resumo dos assuntos discutidos**: Nesta reunião foi apresentada a proposta de trabalho por parte do Eng. João Morais da empresa Elemento Digital, a qual consiste no desenvolvimento de um videojogo multi-jogador com inclusão de uma componente de publicidade. Também foi passada a informação dos principais requisitos pretendidos para este projecto, onde há a destacar a mecânica simples da jogabilidade em ambiente tridimensional e fácil distribuição e instalação. Algumas sugestões importantes sobre tecnologias a utilizar foram discutidas em conjunto com o Proj. Fausto Mourato, tendo em conta os requisitos anteriormente apontados.*

***Principais conclusões**: Nesta reunião ficou claro o tipo de projecto pretendido pelo responsável da empresa assim como as principais características pretendidas.*

***Próximos passos**: Estudar as alternativas sugeridas em termos de tecnologias a utilizar.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #2

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 13 de Março de 2009*

***Resumo dos assuntos discutidos**: Foi nesta reunião analisada a estrutura do trabalho final, feito um ponto de situação e organização de ideias. Da parte do professor Fausto Mourato, que guiou a reunião, foi realizada uma demonstração de alguns exemplos e técnicas de Computação Gráfica. Também foram debatidas algumas vantagens e desvantagens de algumas tecnologias possíveis de utilizar.*

***Principais conclusões**: Desta reunião apuraram-se requisitos, assim como algumas técnicas de programação gráfica importantes.*

***Próximos passos**: Iniciar fase de requisitos.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #3

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 03 de Abril de 2009*

***Resumo dos assuntos discutidos**: Foram nesta reunião debatidos os requisitos levantados pelos alunos e tiradas algumas dúvidas sobre a apresentação intermédia em Inglês. Da parte do professor Fausto Mourato, que guiou a reunião, foi realizada uma explicação sobre as metas pretendidas tanto para a apresentação como para o relatório intermédio.*

***Principais conclusões**: Desta reunião analisou-se que a linguagem mais apropriada para o desenvolvimento da aplicação será C#, através da Framework XNA da Microsoft, bem como os tópicos a desenvolver na apreciação intermédia do projecto.*

***Próximos passos**: Estudo da Framework XNA e conceitos 3D essenciais.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #4

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 04 de Maio de 2009*

***Resumo dos assuntos discutidos**: Nesta reunião debateram-se os casos de utilização criados pelos alunos e esclarecidas algumas dúvidas referentes à apresentação intermédia em inglês. Ficou também decidido a não implementação de efeitos (shaders) em HLSL devido à sua complexidade. Em sua alternativa, irá ser usado o shader de origem do XNA.*

***Principais conclusões**: Desta reunião em que foram analisados os "use cases", resultou a sua correcção e posterior aprovação, bem como da apresentação em inglês.*

***Próximos passos**: Continuação do estudo da Framework XNA e conceitos 3D essenciais. Início da elaboração do documento para a primeira fase do projecto.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #5

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues, Eng.º João Morais*

***Data**: 12 de Maio de 2009*

***Resumo dos assuntos discutidos**: Nesta reunião, por parte dos alunos, foram apresentados alguns exemplos práticos, resultantes do estudo decorrente das ferramentas escolhidas para a elaboração da aplicação. Após apreciação dos mesmos, tanto por parte do Prof. Fausto Mourato, como por parte do Eng.º João Morais, foram efectuadas sugestões sobre como melhorar os exemplos elaborados, para determinar o melhor caminho a seguir. Foram ainda obtidos dados sobre a Elemento Digital, necessários para a primeira fase do documento a entregar.*

***Principais conclusões**: Determinou-se nesta reunião a forma mais favorável de evolução dos exemplos desenvolvidos pelos alunos, bem como foram sugeridas novas aplicações-exemplo fundamentais para o desenvolvimento deste projecto.*

***Próximos passos**: Continuação do estudo da Framework XNA e conceitos 3D essenciais. Desenvolvimento de pequenas aplicações-exemplo.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #6

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 08 de Junho de 2009*

***Resumo dos assuntos discutidos**: Foram abordados os últimos detalhes da fase de requisitos e clarificaram-se passos a seguir. Da parte do Prof. Fausto Mourato esclareceu questões pertinentes a secções do documento a entregar.*

***Principais conclusões**: Nesta reunião foram analisadas as várias secções do manual, esclarecendo pontos a melhorar.*

***Próximos passos**: Conclusão da elaboração do documento a entregar na primeira fase do projecto.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #7

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 15 de Junho*

***Resumo dos assuntos discutidos**: Foram nesta reunião tratados aspectos referentes às melhores formas de execução dos passos a seguir.*
O Prof. Fausto Mourato esclareceu questões quanto aos melhores métodos de criação da programação, para atingir os requisitos propostos. Devido à não integração de uma biblioteca nativa de animação do XNA, foi sugerido pelos alunos a utilização de uma biblioteca freeware e open-source da Microsoft elaborada para XNA, com a aprovação do orientador.
Quanto aos alunos receberam conselhos para o desenvolvimento da aplicação, e consolidação de conhecimentos.

***Principais conclusões**: Nesta reunião foram analisados os passos a seguir para o desenvolvimento do projecto.*

***Próximos passos**: Continuação do desenvolvimento da aplicação com as novas bibliotecas.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #8

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 15 de Julho*

***Resumo dos assuntos discutidos**: Foram nesta reunião debatido os passos a seguir para a continuação de um bom desenvolvimento desta aplicação.*
Da parte do Prof. Fausto Mourato, orientador da reunião, analisou os progressos que os alunos demonstraram na aprendizagem autónoma, esclarecendo questões surgidas após o desenvolvimento da mesma.
Partiu dos alunos, com a aprovação do Prof. Fausto Mourato, a sugestão de desistir do uso da API de rede do XNA, para uma API open-source e freeware que melhor preenche os requisitos da empresa.
Quanto aos alunos, viram esclarecidas as suas dúvidas, tendo obtido aprovação do trabalho já desenvolvido até então.

***Principais conclusões**: Foram, nesta reunião, debatidos os passos a seguir para a continuidade do desenvolvimento da aplicação, para a conclusão do mesmo.*

***Próximos passos**: Continuação do desenvolvimento da aplicação. Alteração da API XNA para Lidgren.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #9

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues, Eng. João Morais*

***Data**: 30 de Julho de 2009*

***Resumo dos assuntos discutidos**: Foram nesta reunião visualizados os resultados obtidos da concepção da aplicação de software.*
Da parte dos alunos, mostraram o trabalho desenvolvido e a abordagem seguida para o executar, esclarecendo dúvidas pontuais existentes.
Da parte do Prof. Fausto Mourato, analisou os resultados que os alunos elaboraram, esclarecendo questões surgidas para a conclusão da mesma.
Da parte do Eng. João Morais, houve alteração dos requisitos iniciais, como a alteração da visão em primeira pessoa para terceira pessoa.

***Principais conclusões**: Nesta reunião foram aprovados os pontos em falta para a conclusão do trabalho, e os novos requisitos.*

***Próximos passos**: Conclusão da aplicação e dos manuais necessários à aplicação.*

ACTA de REUNIÃO de PROJECTO de FIM de CURSO #10

***Projecto**: Alien Arena*

***Presentes**: Prof. Fausto Mourato, Pedro Andrade, Rui Rodrigues*

***Data**: 03 de Setembro de 2009*

***Resumo dos assuntos discutidos**: Da parte dos alunos, mostrou-se o trabalho desenvolvido e corrigiram-se os detalhes em falta.*
Da parte do Prof. Fausto Mourato, analisou os progressos que os alunos mostraram, esclarecendo questões despontadas para a conclusão da mesma. Foram também analisadas as dúvidas dos alunos em relação à documentação final a entregar.

***Principais conclusões**: Nesta reunião foram aprovados os pontos em falta para a conclusão do trabalho.*

***Próximos passos**: Conclusão dos manuais necessários à aplicação.*

9. Bibliografia

Banco de Portugal. (12 de Junho de 2009). Banco de Portugal*. Obtido em 14 de Junho de 2009, de Taxa de Câmbio de Referência diárias: http://www.bportugal.pt/*

ECMA. (2006). Standard ECMA-334*. Obtido de ECMA International: http://www.ecma-international.org/publications/standards/Ecma-334.htm*

Elemento Digital S.A. (2009). Obtido de Elemento Digital: http://www.elementodigital.pt/

eMarketeer. (28 de Fevereiro de 2007). Digital Inteligense*. Obtido em 14 de Junho de 2009, de eMarketeer: http://www.emarketer.com/Article.aspx?R=1004635*

Freire, A. (1997). Estratégia. Sucesso em Portugal. *Editorial Verbo.*

Instituto Nacional de Estatística. (15 de Maio de 2009). INE*. Obtido em 14 de Junho de 2009, de Principais Indicadores: http://www.ine.pt/*

ISO. (2003). ISO/IEC 23270*. Obtido de International Organization for Standardization: http://www.iso.org/iso/catalogue_detail.htm?csnumber=36768*

Lobão, A., Evangelista, B., & de Farias, J. A. (2008). Beginning XNA 2.0 Game Programming. *Apress.*

Microsoft. (2009). Xbox LIVE Marketplace*. Obtido de http://marketplace.xbox.com/pt-PT/*

Microsoft. (2009). XNA creators club online*. Obtido de XNA creators club online: http://creators.xna.com/*

Microsoft. (2009). XNA Developer Center*. Obtido de http://msdn.microsoft.com/en-us/xna/default.aspx*

Pina, N. (2006). Modelos de Desenvolvimento de Software. *Setúbal.*

What is XML? *(2009). Obtido em 2009 de Março de 30, de TechTarget: http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci213404,00.html*

World Wide Web Consortium*. (2009). Obtido de Web Standards: http://www.w3.org/*

10. Glossário

Para facilitar a compreensão de determinados conceitos neste manual elaborou-se este glossário. Os termos encontram-se dispostos por ordem alfabética.

- Advergame*: nome dado à estratégia de mercado (a nível de marketing) que usa jogos, principalmente electrónicos, como ferramentas para divulgar e promover marcas, produtos, organizações e/ou pontos de vista.*
- Advertisement*: actividade profissional dedicada à difusão pública de ideias associadas a empresas, produtos ou serviços.*
- *Computação gráfica: área da computação destinada à criação de imagens em geral, em forma de representação de informação e dados, ou em forma de simulação do mundo real, por intermédio de fórmulas matemáticas e algoritmos complexos. Possui uma infinidade de aplicações para diversas áreas, tais como a produção de interfaces gráficas para software, sistemas operativos e sites na internet, bem como na produção de animações e jogos.*
- *First Person Shooter (FPS): estilo de jogo de computador e jogos de vídeo, no qual se vê apenas o ponto de vista do protagonista, como se o jogador e a personagem do jogo fossem o mesmo observador.*
- *Framework: abstracção que une códigos comuns entre vários projectos de software promovendo funcionalidade genérica. "Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação" (FAYAD e SCHMIDT).*
- *Microsoft XNA: Framework que serve para desenvolvimento de jogos para computador com Windows, bem como para a consola Xbox 360. Sendo uma plataforma de desenvolvimento, é constituída por vários componentes, nomeadamente:*
  - *XNA Game Studio Express: versão baseada no Visual C# Express;*
  - *XNA Framework: conjunto de classes necessárias para executar um jogo XNA. Funciona sobre o .Net Framework para jogos no Windows ou do .Net Compact Framework para jogos na Xbox 360.*

- *Modelação 3D: área da computação gráfica que tem como objectivo a criação de entidades em três dimensões, geração de cenas estáticas, imagens em movimento (designadas de animação), com ou sem interactividade. Basicamente, consiste na criação de formas, objectos, cenários e personagens.*
- *Multi-jogador: conceito que permite a mais de uma pessoa poder jogar no mesmo ambiente em simultâneo. Possibilita aos jogadores da aplicação interagirem com outros jogadores humanos, facilitando a interacção social entre eles.*
- *Quaterniões: são uma extensão* $\mathbb{H}$ *do conjunto dos números complexos* $\mathbb{C}$*. Mais precisamente, o conjunto* $\mathbb{H}$ *é uma álgebra associativa formada pelos números na forma* $u + xi + yj + zk$*, onde* $u, x, y, z \in \mathbb{R}$ *e* i*,* j *e* k *são unidades imaginárias. Os quaterniões são usados para a rotação de vectores em 3D.*
- *Threads: linha de execução; forma de um processo se dividir a si mesmo em duas ou mais tarefas que podem ser executadas em simultâneo.*