Projeto e Implementação de Jogos Eletrônicos

Daniel Oliva Sales[1], **Wilian Soares Lacerda**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 37 – 37200-000 – Lavras (MG) – Brasil

dsales@comp.ufla.br, lacerda@ufla.br

Resumo. Jogos eletrônicos despertam cada vez mais interesse comercial e acadêmico. O desenvolvimento de jogos é um ramo multidisciplinar, já que um jogo é um software composto de vários módulos que devem funcionar perfeitamente entre si. Além disso, um jogo se caracteriza por ser uma aplicação em tempo real, o que exige a melhor interação possível entre hardware e software. O objetivo deste trabalho é descrever e aplicar os métodos utilizados na construção de um jogo para vídeo-game, desde sua concepção e arte até a fase final de programação e aprimoramentos, desenvolvendo para isso um jogo que exemplifica o processo. Este jogo foi desenvolvido em linguagem C++ e utiliza, entre outras técnicas e conceitos, módulos de Inteligência Artificial e de Computação Gráfica. Espera-se com esse estudo reunir as informações mais relevantes no processo de criação de jogos, servindo de referência inicial para trabalhos futuros.

Palavras-chave. desenvolvimento de jogos, vídeo-game, computação gráfica, inteligência artificial.

Electronic Games Design and Implementation

Abstract. *Electronic games are increasing each time more commercial and academic interest. The game development is a multidisciplinary area, because a game is a software composed of several modules which must work perfectly together. Moreover, a game is a real-time application, which demands interaction between the hardware and software of the best possible form. The objective of this work is to describe and execute the methods used in the development of a game, since its conception and art until the final phase of programming and improvements, developing a game which is an example for the process. This game was developed in C++ language and uses, beyond other techniques and concepts, Artificial Intelligence and Computer Graphics modules. The expect with this work is to gather the most important information in the game development process, acting as a initial reference for future works.*

Keywords: design, video-game, computer graphics, artificial intelligence

31/10/2008

1. Introdução

Um jogo eletrônico pode ser considerado um tipo de software especial, por conter os mais variados elementos como módulos de Computação Gráfica, Inteligência Artificial, Redes de Computadores, entre outros, que, além da necessidade de funcionar em perfeita harmonia, devem obedecer à característica fundamental de um jogo: devem responder em tempo real, exigindo que o hardware seja explorado da melhor maneira possível.

Como o principal objetivo de um jogo é entreter o usuário, diversos recursos artísticos e tecnológicos são combinados para criar a sensação de imersão. O desenvolvimento de jogos é, portanto, uma área extremamente interdisciplinar já que além de integrar várias áreas da computação, as aproxima de áreas como artes plásticas, design gráfico, música, etc. [1]

O objetivo deste trabalho é descrever e aplicar as diversas técnicas e conceitos utilizados no desenvolvi-

mento de jogos eletrônicos, desde a sua concepção até a programação, construindo um jogo que exemplifica o processo. São apresentados os passos utilizados e principais decisões de implementação como algoritmos escolhidos, modelos adotados, etc. Espera-se dessa forma tratar dos principais aspectos envolvidos no desenvolvimento de jogos, bem como servir de embasamento inicial para novos trabalhos, deixando claro para o leitor os passos a serem seguidos para o desenvolvimento de um jogo, seja ele de qualquer gênero.

O jogo produzido deverá ser executado em um sistema embarcado, no caso o console *Sony Playstation Portable* (PSP) [4], o qual foi escolhido por possuir um ótimo poder de processamento, boa qualidade e diversidade dos periféricos integrados, suporte nativo do firmware a *homebrews* – softwares desenvolvidos pelos próprios usuários através de ferramentas diversas –, além da simplicidade de programação e testes, já que é possível rodar o jogo criado diretamente da memória física do próprio dispositivo ou até mesmo de um computador, sem a necessidade de mídias auxiliares.

A motivação para uso de um console está justamente no fato de este ser um hardware dedicado a jogos, o que traz novos desafios na programação como a busca por otimização e adaptação do código a um sistema embarcado.

2. Desenvolvimento de Jogos

2.1. Processo de Elaboração de um Jogo

Como qualquer outro software, a produção de um jogo computadorizado, requer a adoção de um processo de desenvolvimento. No caso dos jogos eletrônicos, são tratados tanto os aspectos técnicos quanto os artísticos envolvidos no processo. Segundo ROLLINGS apud [1], o processo de desenvolvimento de um jogo envolve as seguintes etapas fundamentais:

1. Confecção do *Design Bible*;
2. Produção de áudio e imagens 2D;
3. Modelagem 3D;
4. Desenvolvimento dos artefatos computacionais.
5. Integração dos aspectos artísticos com os aspectos computacionais.

2.1.1. Design Bible

Trata-se de um documento que contém todas as instruções de desenvolvimento e características desejadas do jogo a ser desenvolvido, já que é impossível desenvolver uma aplicação sem antes ter todas as suas especificações. Segundo [1], a produção deste documento é uma etapa fundamental, tanto que o processo de desenvolvimento não começa sem que esse esteja pronto. O *Design Bible* deve conter os seguintes elementos descritos abaixo:

Roteiro

É um item fundamental para o processo de criação, já que é crucial para convencer os investidores da potencialidade do produto, mostrando seu diferencial em relação aos outros. Os roteiros de jogos são chamados de roteiros interativos, pois diferentemente dos roteiros de filmes, devem ter espaço para interferência do usuário no desencadeamento da história . [1]

Game Design

Segundo [1], entende-se por *game design* a conceituação artística do jogo. Nesta etapa são definidas as principais características dos cenários, desenhados esboços de personagens (Figura 1), descritas as texturas fundamentais, mapas e fases (também chamado de *level design*).

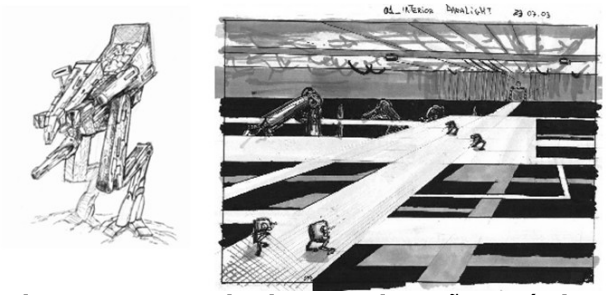

Figura 1: Exemplo de conceituação artística de um personagem e de um cenário

FONTE: CLUA, 2005

Gameplay

Nesta parte do documento deve descrever-se como serão as regras do jogo, e o balanceamento das mesmas. O conteúdo desta seção é fundamental para os programadores na etapa de implementação, já que descreve as características que parte do software deve apresentar.

Interface

Pode-se dividir a interface em *ingame* e *outgame*. A primeira é responsável pela entrada de dados do jogador para a aplicação, ou seja, botões pressionados e ações realizadas. A interface *outgame* é a forma de apresentar todos os dados do jogo, entre outras operações de suporte. Uma boa interface deve passar despercebida para o jogador, permitindo que o mesmo possa focar-se no desenrolar da história e das ações.

Ainda segundo [1], terminada a etapa de conceituação e elaboração, o desenvolvimento de um jogo divide-se em dois caminhos distintos: o de criação artística e o de programação, com grande interseção entre ambas. A criação artística corresponde à criação de todos vários elementos que deverão ser incluídos no jogo como modelos 3D, texturas, terrenos, sons, músicas e arquivos de configuração, enquanto a programação trata das regras do jogo e demais módulos, integrando os elementos desenvolvidos com os recursos criados .

2.1.2. Produção de Áudio e Imagens 2D

No aspecto de áudio, para incrementar a imersividade é fundamental adicionar a percepção sonora no jogo. Geralmente são usadas ferramentas específicas de edição e produção de músicas para este fim.

Quanto às imagens, é importante ressaltar que os jogos 3D não são construídos somente com modelos tridimensionais, na produção de um jogo também é necessário compor imagens bidimensionais. Em geral, tais imagens serão usadas como texturas, mas também serão usadas para compor a interface gráfica *ingame* e *outgame*, tais como, botões, janelas, barras de energia, menus e outros componentes gráficos. Para produzir estas imagens existe uma série de ferramentas para editoração gráfica, sendo o *Adobe Photoshop* um dos softwares mais tradicionais para desempenhar tal atividade

2.1.3. Modelagem 3D

A equipe de modelagem 3D será responsável por criar os objetos geométricos das fases. A geometria de um jogo pode ser dividida em dois tipos: modelagem estrutural e modelagem de elementos dinâmicos. [1] A modelagem estrutural consiste na criação dos elementos estáticos como o cenário, e a modelagem de elementos dinâmicos consiste na criação de objetos que possuem movimento, como personagens.

Para esta etapa os principais softwares utilizados são o *3DStudio MAX*, *MAYA*, *Avid Softimage* e *Ligthwave*, pois fornecem recursos avançados tornando-os ótimos para este processo. Manipulam fácil e intuitivamente os polígonos criados, aplicam com facilidade as texturas nos polígonos, além de otimizar a modelagem dos objetos, utilizando uma quantidade reduzida de polígonos.

As ferramentas utilizadas na modelagem 3D devem possuir uma boa interface de visualização, pois para o artista, é importante que à medida que um objeto seja construído esse processo seja acompanhado em tempo real, sabendo a *priori* como o mesmo será visto no jogo.

Neste processo, é importante que os artistas tenham capacidade de modelar objetos com quantidade reduzida de polígonos. Otimizando-se a modelagem, será possível que o cenário possa ser mais extenso e que mais objetos possam ser inseridos no mesmo. É possível observar na Figura 2 que o resultado final é bastante semelhante, embora o avião da direita possua 15 vezes mais polígonos que o da esquerda. Deve-se estar atento aos limites do hardware, e quantidade de elementos que farão parte da cena para que seja utilizada uma quantidade adequada de polígonos.

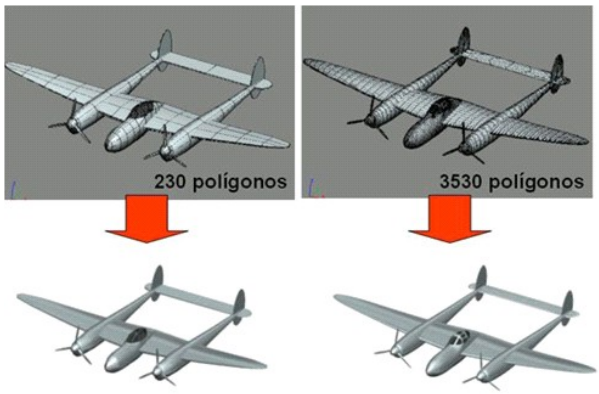

Figura 2: Representação de um modelo em resoluções de polígonos diferentes

FONTE: CLUA, 2005

2.1.4. Artefatos Computacionais

Na etapa de desenvolvimento dos artefatos computacionais, são escolhidas ou criadas ferramentas para o desenvolvimento dos modelos, interação com o hardware gráfico, controle da IA, etc. Elas devem permitir que o desenvolvedor possa criar diversos jogos diferentes, reaproveitar com facilidade o código desenvolvido em projetos anteriores, abstrair a manipulação de APIs, além de possibilitar uma fácil integração entre código e modelagem 3D. [1]

2.1.5. Integração

A última etapa consiste na integração dos componentes artísticos com os computacionais, ou seja, atribuir aos modelos as propriedades idealizadas, aplicar as texturas, etc. para por fim incluí-los no projeto, importando os modelos criados para o programa principal, montando assim as cenas desejadas.

2.2. Principais Componentes

Alguns dos principais aspectos a serem tratados, possuindo inclusive *engines* exclusivas para seu tratamento são: renderização, física, som, e inteligência artificial por exemplo.

A renderização é basicamente o tratamento do *pipeline* gráfico, que consiste nas transformações 3D, projeção, seleção e corte dos polígonos que serão visualizados, iluminação e texturização [1].

A física é o tratamento de eventos do mundo virtual que devem se assemelhar com os do mundo real como colisões, resultantes de forças, etc.

O controle do som é feito definindo-se quais os tipos e intensidades de sons que os objetos e ambiente devem apresentar durante o jogo através da manipulação dos arquivos de áudio.

A inteligência artificial descreve o comportamento de entidades não controladas pelo jogador [1]. Para isso são usadas máquinas de estado, busca em árvores de jogo, ou outras técnicas como, por exemplo, Redes Neurais Artificiais e Algoritmos Genéticos.

Para jogos *online* e/ou *multiplayer,* os conceitos de Redes de Computadores são utilizados para definir como o jogo irá acessar o hardware de comunicação, e a forma como os dados serão compartilhados para que seja mantida a interação em tempo real, e a sensação de imersividade desejada em jogos.

3. Material e Métodos

3.1. Ferramentas

A principal ferramenta utilizada para o desenvolvimento é o pacote PSPDev [5] que deve funcionar em um ambiente de desenvolvimento C++, ou com outro editor de código em ambiente Windows. O projeto será compilado com a versão 4.0.2 do compilador Gcc contido no pacote PSPDev, que possui além deste recurso, algumas das classes e funções necessárias para a programação do dispositivo.

As outras bibliotecas necessárias como PSPGL, libmad, etc. estão no pacote PSPDevLibInstall [5]. Trata-se de um executável que já instala e configura todas as bibliotecas no mesmo diretório onde foi instalado o PSPDev.

Os testes foram realizados no próprio dispositivo, já que o arquivo executável gerado pode ser transferido através de uma conexão USB para a memória física do console, de onde já é diretamente executado.

3.2. Softwares Utilizados

Os seguintes softwares foram utilizados:

- Adobe Photoshop CS3 – Produção e edição de imagens 2D
- Audacity – Conversão dos arquivos de áudio produzidos
- Bloodshed Dev C++ – Ambiente de desenvolvimento C++
- Guitar Pro 4 – Criação das músicas do jogo

3.3. Hardware Final

O console adotado é o Sony PSP Slim (Figura 3) [4]. Trata-se de um ótimo sistema de entretenimento portátil, lançado em 2005 na sua versão inicial, e aperfeiçoado em 2007 quando foi lançado então na versão Slim.

Figura 3: Console Sony PSP Slim

3.3.1. Firmware

O PSP possui um firmware atualizável que o permite além de rodar jogos, reproduzir filmes em MPEG4, aplicações em Java e Flash, músicas em MP3 e WMA, e fotos em JPEG de um disco UMD ou diretamente do Memory Stick. Além disso, possui um browser para navegação na Internet através de sua conexão Wi-fi. O firmware também já está preparado para utilizar os acessórios opcionais como câmera digital, sintonizador de TV Digital e até mesmo GPS. Sua interface apresenta de forma simples e intuitiva os recursos disponíveis ao usuário.

Até a versão 1.5, o firmware oficial da Sony suportava *homebrew* (softwares desenvolvidos pelos usuários). As versões mais novas não possuem mais esse suporte nativo, porém foram lançados *Custom Firmwares* (firmwares não-oficiais), que acompanham todas as funcionalidades dos firmwares oficiais recém-lançados, com suporte a *homebrew*.

Para execução do jogo desenvolvido, é necessário portanto que o console esteja com a versão 1.5 oficial ou anterior, ou com uma Custom Firmware. Mais informações sobre custom firmware, bem como tutoriais para atualização de firmware podem ser facilmente encontrados na internet.

4. Desenvolvimento

4.1. Concepção

4.1.1. Design Bible

Como citado anteriormente, o primeiro passo no desenvolvimento de um jogo é a confecção de um *Design Bible*, documento que possui todas as informações necessárias para a execução dos passos seguintes.

O *Design Bible* do jogo desenvolvido conta com os seguintes elementos:

- Roteiro: por ser um jogo exemplo, não foi elaborado um roteiro completo, mas sim um contexto para o jogo.
- *Game Design*: foi definido o cenário principal, criados os esboços das texturas posteriormente produzidas, esboços das peças (personagens).
- *Gameplay*: o balanceamento das regras foi feito tendo como bases outros jogos já famosos. Procurou-se criar regras simples, fáceis de ser aprendidas, mas ao mesmo tempo interessantes. Além disso, foram incluídos elementos que dão ao jogo características não-determinísticas, diminuindo as possibilidades de se prever qual o próximo movimento do adversário.
- Interface *ingame*: os controles do jogador foram definidos usando como base jogos de vídeo-game conhecidos, e procurou-se dar liberdade ao usuário para controlar além dos seus próprios personagens, a visualização do cenário.
- Interface *outgame*: trata-se de um tabuleiro 8x8, visualizado tridimensionalmente, há um menu lateral com informações de ajuda e sobre o jogo. As principais regras e comandos estão listados em uma tela de ajuda que o usuário pode acessar quando desejar.

Além destas informações, foram citados alguns aspectos técnicos como forma de execução do jogo no console, e decisões de implementação que complementam a descrição do *gameplay* e servem de base para a fase de programação.

4.2. Produção de Áudio e Imagens 2D

4.2.1. Áudio

A principal ferramenta para esta fase do desenvolvimento é o software Guitar Pro 4. Apesar da facilidade de uso do programa, é importante que o compositor tenha bons conhecimentos em teoria e percepção musical, sabendo ler e escrever partituras.

Cada trilha do projeto corresponde a um instrumento a ser utilizado, e é editada individualmente. Inúmeras combinações de instrumentos podem ser utilizadas. O software permite ainda importar músicas já prontas em formato MIDI e editá-las.

Uma vez criada a música e salvo o projeto, é possível exportar o áudio em formato MIDI. Torna-se então necessária a utilização de um software que converta a música criada para um formato compatível com o PSP. Para este fim foi utilizado o software Audacity. Ele também permite a edição do áudio, porém não nota por nota como o Guitar Pro, mas sim, edição da forma de onda final, já com todos os instrumentos. Nele é possível remover ruídos indesejáveis, aumentar a amplitude, incluir efeitos e cortar/incluir trechos.

Após a edição ser concluída, o projeto é salvo, e o áudio deve ser exportado como MP3, formato a ser utilizado pelo PSP.

4.2.2. Imagens 2D

Esta etapa engloba não apenas as texturas propriamente ditas, mas também as outras imagens do jogo como menus e demais telas. Está muito próxima da etapa de texturização, e portanto é necessário conhecer antecipadamente qual o tamanho de imagem em pixels utilizado pela biblioteca gráfica e qual o formato do arquivo.

As imagens usadas como menus, apresentação e informações também são carregadas como texturas em retângulos do tamanho da janela de visualização, passando portanto pelo mesmo processo que as demais texturas.

O software utilizado nesta etapa é o Adobe Photoshop CS3.

No caso, o método escolhido para carregar texturas utiliza imagens em formato Targa (.TGA), de 32 bits, e as dimensões da figura (altura e largura) devem ser potências de dois.

O display do PSP possui 480x272 pixels. Para que as proporções se mantenham durante a execução, a imagem deve ser editada utilizando estas medidas em pixels, e só depois de pronta ser redimensionada para as dimensões finais, que são as potências de dois mais próximas (no caso 512x256). Isso vale para todas as outras imagens, sempre deve-se editá-las nas proporções reais desejadas e só por fim redimensioná-las.

4.3. Modelagem 3D

Assim como a produção de imagens 2D, a modelagem 3D está intimamente ligada com a etapa de implementação gráfica. Os modelos 3D produzidos são incluídos no jogo posteriormente. Geralmente são utilizados softwares específicos para este fim como o 3DStudio e Maya, porém como o jogo desenvolvido possui modelos muito simples, estes foram criados diretamente no código fonte do programa.

Um modelo consiste basicamente de um vetor de vértices, que possui além da informação de posição dos vértices, a cor desejada e as coordenadas para o mapeamento de texturas.

Para o PSP especificamente, há diversas formas de descrever os modelos através dos vértices, visto que há várias formas de desenho dos modelos. Mais uma vez é necessário que a forma de implementação seja bem estabelecida, já que o projeto do modelo depende completamente da forma de desenho a ser utilizada pelo hardware gráfico.

Para os modelos criados, foi escolhido o tipo TRIANGLE_STRIP, onde para desenhar um triângulo basta especificar os três pontos, e para polígonos de mais vértices usa-se uma composição de triângulos, como quadrados por exemplo, em que se especifica-dois triângulos que possuem dois vértices coincidentes.

Na especificação dos vértices no código fonte, cada linha especifica um único vértice.

Os dois primeiros parâmetros são as coordenadas de mapeamento da textura. No exemplo mostrado esta é aplicada somente na frente da peça. O valor “-0” é utilizado para determinar que neste vértice nenhuma textura será aplicada. As coordenadas para mapeamento 2D são especificadas conforme a figura abaixo.

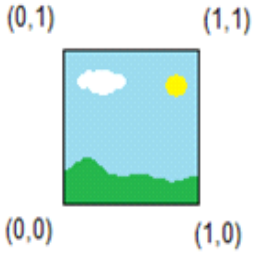

Figura 4: Coordenadas de mapeamento de texturas 2D

O terceiro parâmetro é a cor do fragmento desenhado em RGBA. É necessário saber qual o modo de texturização utilizado para aplicar a cor correta. No jogo foi usado o modo MODULATE, onde a cor final do fragmento corresponde à multiplicação entre a cor original do fragmento e a cor da textura aplicada. Com isso, caso a textura possua partes brancas, estas funcionam como transparente, e caso o fragmento original seja branco, as cores predominantes são as da textura, uma propriedade bastante útil.

Os últimos três parâmetros correspondem às coordenadas do vértice no SRU.

4.4. Implementação

Na etapa de implementação, todos os módulos são integrados conforme as especificações do *Design Bible*.

4.4.1. Projeto Inicial

Deve-se levar em conta todos os detalhes especificados previamente. Nesta etapa, são criados os “esqueletos” das classes com as chamadas das principais funções e são definidos os tipos das principais variáveis envolvidas.

Para o jogo criado, foram criadas duas classes: “ia.h”, que contém as funções da inteligência artificial e define as jogadas da CPU, e “tabuleiro.h”, que contém as regras de movimentação e ataque das peças, e cria os

parâmetros iniciais do jogo. A interface gráfica foi implementada no arquivo "main.cpp", que contém ainda a implementação do jogo propriamente dito, com as chamadas para as funções das outras classes criadas, e leitura dos botões de controle.

Como a biblioteca gráfica do PSP é baseada em OpenGL, o funcionamento do programa deve seguir o padrão de um programa OpenGL comum. A PSPGU não conta com a GLUT, logo os *callbacks* de teclado e redesenho por exemplo foram implementados manualmente com as funções específicas do hardware.

Com o início da função main inicia-se um *loop* infinito onde a função "teclado" é chamada, seguida pela função "desenha". Como nesta etapa a interface gráfica final ainda não está pronta, esta função "desenha" possui os comandos para exibir as informações do jogo em modo texto, possibilitando os testes dos outros módulos e simplificando posteriormente a inclusão da interface gráfica, sendo necessário apenas substituir os comandos de texto pelos comandos da PSPGU, e incluir as funções adicionais criadas.

A inclusão das músicas no jogo pode ser implementada a qualquer momento da etapa do desenvolvimento, porém quanto mais cedo isto acontecer mais fácil se torna a detecção e correção de possíveis erros. No modelo criado, foi usada a biblioteca libmad contida no pacote PSPDevLibInstall para este fim.

4.4.2. Regras do Jogo

Com base no Design Bible, as regras foram implementadas no arquivo "tabuleiro.h" em uma única função que tem a finalidade de retornar ao programa se a jogada testada é válida ou não. Com isso o jogador não movimenta as peças do adversário, não movimenta suas peças além do permitido, e nem ataca além do alcance da peça selecionada. Neste mesmo arquivo é implementada a função que avalia se as condições de fim de jogo foram atendidas ou não, e a função que realiza a jogada desejada, alterando a matriz do tabuleiro quando necessário.

Além destas funções foram criadas funções auxiliares como a função que lança os dados para testar se um ataque é realizado ou não, e a função que busca se uma determinada peça ainda está no tabuleiro, por exemplo.

Em resumo, esta classe possui implementações relacionadas ao tabuleiro e à jogada, independente do jogador.

4.4.3. Inteligência Artificial

É importante que a classe com a implementação das regras já esteja pronta nesta fase do desenvolvimento, pois não há como realizar os testes com a IA sem ela. Nesta etapa do desenvolvimento são aplicados algoritmos e técnicas para que o jogador possa interagir com a CPU.

Uma das estratégias mais utilizadas para jogos de tabuleiro é o algoritmo Minimax. Este foi utilizado com o algoritmo de poda alfa-beta para obtenção de um melhor desempenho. A classe "ia.h" recebe do módulo principal o estado atual do tabuleiro e após o processamento é escolhida a jogada da CPU. É preenchido então um vetor com os dados necessários (linha e coluna inicial e final da jogada da CPU), e estes valores são passados para a classe tabuleiro para que a jogada seja realizada.

Como o uso do Minimax convencional em jogos mais complexos é impraticável, há uma forma de se obter um resultado aproximado em tempo satisfatório através de heurísticas [3]. Essa forma de implementação foi adotada, substituindo a função de utilidade por uma função que avalia se um estado é o melhor, e ainda definindo quando utilizá-la. Em termos práticos, isso foi feito criando uma função que atribui um peso para cada estado e um limite para a profundidade da árvore de jogo. Os estados terminais passam a ser os estados com profundidade máxima. Para cada um deles é atribuído um peso, e no final da execução é escolhida a jogada que levou ao tabuleiro de maior peso.

A heurística que atribui pesos funciona da seguinte forma: é criada uma variável com valor zero. Em seguida, cada posição do tabuleiro é lida. Para cada peça do jogador que ainda está no tabuleiro, é somada a essa variável um determinado valor; para cada peça do adversário que ainda se encontra em jogo, é subtraído um determinado valor, forçando assim que a CPU ataque quando possível. Além disso, é somado um valor adicional quando as peças estão mais próximas do fundo do tabuleiro do adversário, o que força a CPU a movimentar suas peças para frente, e não apenas esperar que o adversário se aproxime.

Durante a execução do minimax, o algoritmo não joga dados para simulação de ataques, ou seja, a CPU

joga sem avaliar a probabilidade de um ataque ser perdido.

Para diminuir a previsibilidade das jogadas da CPU, foi incluído um teste antes da jogada, no qual há 10% de chance de que ela realize uma jogada completamente randômica.

4.4.4. Computação Gráfica

Deve-se iniciar esta etapa com um estudo profundo das funções do hardware gráfico através de tutoriais, exemplos e documentação.

A interface do jogo criado está incluída no arquivo "main.cpp" e se baseia em duas funções principais: "desenha" e "desenhaMenu". A primeira é responsável por desenhar o tabuleiro e as outras telas do jogo, enquanto a segunda é responsável pelo desenho da tela inicial e menu principal do jogo.

Os principais passos usados pela função "desenha" são:

1-Limpa a janela de visualização

2-Define-se a projeção a ser utilizada e seus principais parâmetros

3-Posiciona o observador virtual

4-Desenha o tabuleiro e aplica sua textura

5-Lê a matriz do tabuleiro e para cada peça encontrada desenha uma peça da cor correspondente e aplica a textura correspondente à sua função. Em seguida essa peça é transladada da origem (de onde é desenhada inicialmente) até a sua posição real no jogo.

6-É desenhado o cursor com base na sua posição atual. A posição inicial do cursor e posição final recebem texturas diferentes durante as jogadas.

7-É carregada a matriz identidade para que as transformações anteriores não tenham efeito a partir deste ponto, e então é desenhado um retângulo que recebe a textura de menu lateral com as informações necessárias. Este menu permanece fixo em sua posição, bem como todos os elementos que aparecem neste menu.

Já a função "desenhaMenu" executa os seguintes passos:

1-Limpa a janela de visualização.

2-Define-se a projeção a ser utilizada e seus principais parâmetros

3-Posiciona o observador virtual

4-Desenha um retângulo do tamanho da janela de visualização.

5-Se esta função está sendo chamada pela primeira vez, aplica seqüencialmente as texturas de apresentação com nome do autor, universidade, etc.

6-Aplica a textura com o menu principal (a opção atualmente selecionada fica em destaque)

Outra função importante para a interface é a função InitGu. Esta função é chamada no início do programa e nela são aplicados vários parâmetros que definem como será feita a visualização dos elementos posteriormente. São definidos, por exemplo, o modo de texturização adotado, parâmetros da janela de visualização, cores utilizadas, entre outros.

Outras funções auxiliares foram criadas para atender chamadas de eventos especiais, como *Game Over*, *Pause*, etc. e possuem funcionamento semelhante à função "desenhaMenu", ou seja, é desenhado um retângulo do tamanho da janela de visualização e aplicada a textura com a imagem adequada.

4.4.5. Função Principal

Incluída no arquivo "main.cpp", implementa os modos de jogo possíveis e inicializa todas as variáveis. A execução consiste nos seguintes passos:

1 – É definida a freqüência de operação do processador, são iniciados os *callbacks* principais e é definido o modo de operação do controle.

2 – São inicializados os dados do jogo, e o áudio

3 – São carregados todos os arquivos de imagens, e rearranjados os dados nas matrizes através da função *swizzle*. Cada arquivo de imagem é carregado em um novo objeto da classe TGALoader.

4 – Inicializa-se a GU com seus parâmetros definidos anteriormente

5 – Inicia o *loop* do jogo

O *loop* do jogo é, a princípio, o menu principal. A partir dele as opções são acessadas e os diferentes modos de jogo iniciados.

As funções tecladoMenu e tecladoJogo fazem as alterações necessárias nas variáveis e executam várias funções quando solicitado. Sempre as funções teclado

são seguidas de uma função desenha correspondente que mostra na tela eventuais alterações realizadas.

4.4.6. Testes e Aprimoramentos

Os testes foram realizados exaustivamente, e em todas as etapas do desenvolvimento. A cada compilação do código, um novo teste era realizado no próprio dispositivo. Após o jogo funcionando corretamente, alguns recursos foram adicionados como por exemplo um modo *multiplayer* onde o jogador interage com um segundo jogador no mesmo dispositivo, foram incluídas novas informações para o usuário no menu lateral do jogo e vários trechos do código foram otimizados em busca de melhor performance.

5. Resultados

A execução dos passos apresentados neste trabalho permitiu a implementação de um jogo para vídeogame atendendo todos os requisitos especificados no seu projeto.

As técnicas de Inteligência Artificial utilizadas se mostraram eficientes na resolução do problema proposto, consumindo uma quantidade aceitável de recursos do hardware e em um baixo tempo de execução. A heurística implementada faz com que a CPU avance no tabuleiro quando possível e ataque se alguma peça adversária estiver ao alcance. Como o minimax percorre grande parte da árvore de jogo, e o tamanho desta é proporcional à quantidade de jogadas possíveis, a IA gasta mais tempo para realizar as jogadas quando há mais peças no jogo, e quando há possibilidade de ataque. 10% das jogadas são completamente randômicas.

As técnicas de Computação Gráfica foram usadas na produção de uma interface 3D (Figuras 5, 6 e 7), que além de exibir as informações do jogo ainda apresenta as principais regras e objetivos do jogo quando solicitado, ajudando o usuário progredir no jogo.

Figura 5: Menu Principal

Figura 6: Interface do Jogo

Figura 7: Jogo em Andamento

A produção artística (áudio, imagens 2D, etc.) foi incluída no jogo da forma desejada, aumentando a imersividade esperada.

O código foi todo comentado, facilitando assim o seu entendimento e possíveis alterações.

Obteve-se como produto final tanto o jogo desenvolvido como o próprio trabalho, o qual pode ser usado como referência inicial para trabalhos futuros nesta área.

Sendo assim, a principal contribuição desse estudo é a sistematização do processo de criação de jogos eletrônicos através do exemplo do software desenvolvido, proporcionando ao leitor uma visão abrangente e ao mesmo tempo detalhada do processo de desenvolvimento de um jogo.

6. Conclusão

Vários aspectos interessantes foram observados durante o desenvolvimento:

· É extremamente importante que o projetista conheça a fundo todas as áreas do desenvolvimento, pois deve definir todas as ferramentas e parâmetros necessários para execução do trabalho, inclusive aspectos técnicos envolvidos em cada módulo.

· As interfaces das classes devem ser muito bem definidas no projeto, pois, alterações após o início da implementação comprometem todo o desenvolvimento.

· A demanda por tempo e quantidade de pessoas na equipe de desenvolvimento aumenta proporcionalmente à complexidade do projeto.

· É importante que todos os envolvidos no projeto conheçam as limitações do hardware final e tenham como meta um bom aproveitamento dos seus recursos, mesmo para os módulos que não utilizam as funções do hardware diretamente.

Conclui-se portanto que a adoção de uma metodologia adequada é fundamental para o sucesso de um projeto.

7. Trabalhos Futuros

Pretende-se aprofundar o estudo do desenvolvimento de jogos futuramente abordando outras áreas do desenvolvimento como a implementação de um modo *multiplayer* através da interface de rede *wireless* que o PSP possui, bem como estender a fase de aprimoramentos com a inclusão de mais cenários, modos de jogo e recursos.

Pretende-se também aprofundar o estudo da etapa de Inteligência Artificial para jogos com a implementação de outras técnicas, no caso o uso de Redes Neurais Artificiais em conjunto com o algoritmo Minimax já implementado. Os dados para treinamento da rede serão gerados manualmente, e pela execução do minimax já implementado. O desempenho das duas implementações será comparado e analisado.

Outra possibilidade de aprofundamento está na criação de modelos 3D mais complexos, adotando um mapeamento de texturas alternativo, criado para este fim.

8. Referências Bibliográficas

[1] CLUA, ESTEBAN W.G.; BITTENCOURT, JOÃO R. Desenvolvimento de Jogos 3D: Concepção, Design e Programação In: **XXV Congresso da Sociedade Brasileira de Computação**. São Leopoldo/RS: SBC, Julho/2005, pag 1313-1358.

[2] RUSSELL, S; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro, Ed. Campus, 2004, 1040p.

[3] SONY COMPUTER ENTERTAINMENT. **Playstation Portable.** Disponível em http://www.us.playstation.com/psp/. Consultado em 20 Jun. 2008.

[4] XORLOSER. **PSPDev for Win32.** Disponível em http://xorloser.com/. Consultado em 05. Mai. 2008