FERNANDO TSUDA
PAULA MORITA HOKAMA
THIAGO MOREIRA RODRIGUES

WE A.R. DANCIN': DESENVOLVIMENTO DE JOGOS UTILIZANDO REALIDADE AUMENTADA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia Elétrica com Ênfase em Computação

São Paulo
2006

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

FERNANDO TSUDA
PAULA MORITA HOKAMA
THIAGO MOREIRA RODRIGUES

WE A.R. DANCIN': DESENVOLVIMENTO DE JOGOS UTILIZANDO REALIDADE AUMENTADA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia Elétrica com Ênfase em Computação

Área de Concentração:
Engenharia de Computação

Orientador: Professor Associado Romero Tori
Co-orientador:
João Luiz Bernardes Jr.

São Paulo

2006

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

DEDICATÓRIA

Dedicamos este trabalho às pessoas que nos apoiaram nestes cinco árduos anos.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

AGRADECIMENTOS

Aos Orientadores Romero Tori e João Bernardes, pela orientação durante o desenvolvimento deste projeto.

Aos nossos pais, que nos apoiaram durante todo este trabalho e toda a nossa jornada até aqui.

Aos Professores Reginaldo Arakaki e Marcelo Carreño, pela ajuda desinteressada e espontânea.

Ao Gustavo Menezes, pelo auxilio dado para confecção dos objetos 3D utilizados pelo jogo.

Ao irmão da Paula Hokama, Lucas Hokama, pela criação das telas de apresentação do jogo.

À irmã do Thiago Moreira, Camila Moreira, por disponibilizar uma câmera para o desenvolvimento da solução de realidade aumentada.

Aos amigos que participaram nos testes da ferramenta de realidade aumentada e que foram os modelos para as fotos do pôster e dos relatórios.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

Aos amigos do curso de computação cooperativo pela ajuda e disponibilidade dos demos produzidos por eles.

Aos amigos da Promon Tecnologia e da Scopus Tecnologia, pelo apoio e auxilio apresentado durante o projeto, e pela compreensão.

Ao Afilhado do Thiago Moreira, Luiz Gustavo, apoiador e torcedor pelo sucesso deste trabalho.

Ao namorado da Paula Hokama, Leandro Seki, pela paciência e pelas caronas oferecidas nos momentos críticos.

Às amigas Luana Guimarães e Ana Gabriela Garcia pelas palavras de sabedoria.

Ao amigo Gustavo Samico e ao primo do Thiago, Luiz Filipe Carvalho, por auxiliar nos testes de usabilidade.

Aos professores, que nos possibilitaram conhecimento necessário para a devida execução deste trabalho.

Aos amigos Guilherme Fujii, Cíntia Yoshimura e à tia do Thiago, Meire Moreira, por nos prover a infra-estrutura necessária para conseguirmos terminar este trabalho.

Aos demais amigos e companheiros do curso de computação, turma 2006, por todos os momentos nestes anos de graduação.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

E nós devemos considerar todo dia perdido aquele no qual nós não dançamos pelo menos uma vez. E devemos chamar toda verdade de falsa se ela não for acompanhada por pelo menos uma risada.

(Friedrich Nietzsche)

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

RESUMO

O objetivo deste projeto é desenvolver um jogo que possua interações com o jogador utilizando a Realidade Aumentada, na qual o usuário, um elemento real, interage com um cenário virtual por meio da captura de seus movimentos, utilizando uma webcam, e convertendo-os em ações contextualizadas no jogo. Para o desenvolvimento deste jogo, o grupo realizou a integração da ferramenta jARToolKit e do enJine, um engine de jogos didático desenvolvido pelo Interlab (Laboratório de Tecnologias Interativas), além da elaboração de protótipos que serviram de prova de conceito.

Palavras-Chave: *Engines* de Jogos. JARToolKit. Jogos. Realidade Aumentada.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

ABSTRACT

This project's goal is to develop a game which interacts with the player using Augmented Reality, where the user, a real element, interacts with a virtual scene through the capture of his or her movements, using a webcam, and the conversion of these movements into game actions. For the development of this game, the development team integrated jARToolKit and enJine, a didactic game engine produced by Interlab (Laboratório de Tecnologias Interativas). Prototypes were also created and used as proofs of concept.

Keywords: Augmented Reality. Game Engines. Games. jARToolKit.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

LISTA DE ABREVIATURAS E SIGLAS

2D	2 Dimensional
3D	3 Dimensional
ABRAGAMES	Asociação Brasileira das Desenvolvedoras de Jogos Eletrônicos
AI	Artificial Intelligence – Inteligência Artificial
API	Application Programming Interface
AR	Augmented Reality – Realidade Aumentada
AV	Augmented Virtuality
CCE	Centro de Computação Eletrônica
CIF	Common Intermediate Format
FURPS	Functionality, Usability, Reliability, Performance, and Supportabiltiy
GPS	Global Positioning System
GPU	Graphics Processing Unit
GPU	Graphics Processing Unit
HMD	Headed-Mounted Display
INTERLAB	Laboratório de Tecnologias Interativas
J2SE	Java 2 Runtime Environment Standard Edition
LGPL	Lesser General Public License
MP3	MPEG-1/2 Audio Layer 3
MPEG	Moving Picture Experts Group
MR	Mixed Reality
MTBF	Mean Time Between Failure

PDA	Personal Digital Assistant
PMBoK	Project Managament Body of Knowledge
PPU	Physics Processing Unit
RAM	Random Access Memory
RPG	Role Playing Game
RUP	Rational Unified Process
UML	Unified Modeling Language
UP	Unified Process
USB	Universal Serial Bus
USP	Universidade de São Paulo
VGA	Video Graphics Array
VR	Virtual Reality
WBS	Works Breakdown Structure

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

LISTA DE SÍMBOLOS

Lux	Unidade de Medida de Luz
GHz	Gigaheartz
MB	Megabytes
GB	Gigabytes

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇAO E SISTEMAS DIGITAIS

SUMÁRIO

1 Introdução

1.1 Objetivo do Projeto de Formatura

O objetivo deste projeto é desenvolver um jogo que utilize os conceitos de Realidade Aumentada (Augmented Reality – AR) para fornecer ao usuário um novo mecanismo de interação humano-máquina. O desenvolvimento foi realizado com o uso do enJine, um Engine de jogos didático desenvolvida no Laboratório de Tecnologias Interativas (INTERLAB-USP) da Escola Politécnica da Universidade de São Paulo, e com uma biblioteca de Realidade Aumentada. Os conceitos aqui mencionados serão detalhados adiante.

1.2 Motivação

Os jogos possuem um papel de extrema importância nas áreas social, econômica e tecnológica.

Na área social, eles permitem a interação entre diversas pessoas de diferentes localidades. Esse fato é reforçado pelo crescimento da quantidade de jogos multi-jogadores on-line, onde milhares de pessoas interagem em um ambiente virtual comum a todos eles, transformando a experiência de jogo em um contato

interpessoal, em contraste com a jogabilidade isolada e unilateral, dominante até alguns anos atrás.

Na área econômica, o segmento de jogos eletrônicos mundial faturou 10 bilhões de dólares, ultrapassando assim o lucro obtido com a venda de ingressos pela indústria de cinema em Hollywood em 2001 (HAUSE, 2003), com previsão de crescimento de 20,1% ao ano até 2008. Apesar de o Brasil possuir 40 empresas desenvolvedoras de jogos (ABRAGAMES, 2004), o país ainda está longe de possuir tradição nessa área, principalmente devido à falta do regras e incentivos que são ocasionados principalmente pela falta de visão dos governantes que até 2004 não viam importância nesse setor. Porém, esse quadro tende a mudar, devido à criação da Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos (ABRAGAMES), responsável pela promoção da indústria brasileira de jogos eletrônicos, e à criação de concursos como o JogosBR – promovido pelo governo e demonstrando o seu interesse pela área.

Na área tecnológica, os jogos promovem um desenvolvimento tecnológico que motivam a criação de hardware, software e arquiteturas que sejam capazes de lidar com o aumento de sua complexidade, decorrente principalmente do uso de gráficos e sons em alta definição que trazem um aspecto realista ao jogo. Como exemplo, podemos citar o aumento da capacidade das placas gráficas (GPU – Graphics Processing Unit) e a criação de uma unidade responsável pelo processamento dos cálculos físicos dentro do jogo, chamado PPU (Physics Processing Unit), evoluções que foram fomentadas pela crescente complexidade dos jogos.

Um dos aspectos que dificultam a entrada de novos consumidores nesse mercado é a dificuldade crescente de se manusear os dispositivos de entrada. Cada

vez mais os jogos possuem mais comandos que são inseridos no jogo pelo teclado ou por controladores (joysticks) com muitos botões – atualmente eles possuem por volta de 8 botões de ação e 3 botões ou alavancas de direção –, forçando o usuário a decorar a maioria deles (INFOEXAME, 2006). Assim, novas formas de entrada têm sido utilizadas, como forma de se tornar a jogabilidade mais natural. Entre elas, podemos citar o uso de controladores com formas de instrumentos musicais, como guitarras, maracas e tambores, o uso de câmeras de vídeo como, por exemplo, o acessório eyeToy, utilizados pelos consoles Playstation da Sony, e o WiiMote, o controlador do console Wii, o qual é uma tentativa recente da Nintendo para atrair novos jogadores através de um novo modo inovador de jogabilidade, onde o movimento do controle é detectado e convertido em uma ação correspondente no jogo. Porém, os acessórios mais específicos, como os instrumentos musicais, ainda possuem um custo elevado e são muito específicos para o tipo de jogo que os utiliza.

Figura 1 - WiiMote, o controle do console Wii da Nintendo (GAMESPOT)

Figura 2 - eyeToy, a câmera para jogos de Realidade Aumentada da Sony (GAMESPOT)

A área de jogos é uma das mais interessantes para a exploração da Realidade Aumentada, pois além de não exigir a grande precisão requerida em áreas médicas ou em manutenções, o caráter lúdico dos jogos permite o desenvolvimento de jogos originais e com maior imersão do jogador no ambiente, melhorando assim a interface com o jogo, além de fomentar o desenvolvimento desta tecnologia, para que possa finalmente ser difundida entre as mais diversas áreas e tenha seu uso consolidado nestas outras aplicações que necessitam de confiabilidade muito maior.

Além disso, o projeto irá funcionar para testar e validar a arquitetura do enJine, principalmente em relação ao reconhecimento de comandos de entradas provenientes de dispositivos não convencionais, como a câmera de vídeo utilizada nesse projeto.

1.3 Organização

Inicialmente, o conceito sobre a Realidade Aumentada será apresentado para que o aspecto inovador de sua adoção em um jogo possa ser mais bem entendido.

A seguir será detalhada a metodologia de desenvolvimento do projeto, tanto para a solução de realidade aumentada quanto para a criação do jogo.

Depois, serão detalhadas as ferramentas que foram utilizadas nesse projeto, todo o processo de gestão do mesmo e de controle de qualidade, e, por fim, serão detalhados os aspectos referentes à especificação do projeto em si.

2 Aspectos Conceituais

2.1 Realidade Aumentada

A Realidade Aumentada (em inglês, Augmented Reality – AR) faz parte de um contexto mais amplo denominado Realidade Misturada (Mixed Reality – MR), o qual descreve de maneira genérica a união dos ambientes reais com os virtuais. Dentro desse contexto, Milgram (MILGRAM, 1994) definiu o conceito de Continuidade Virtual, representada pela figura 3:

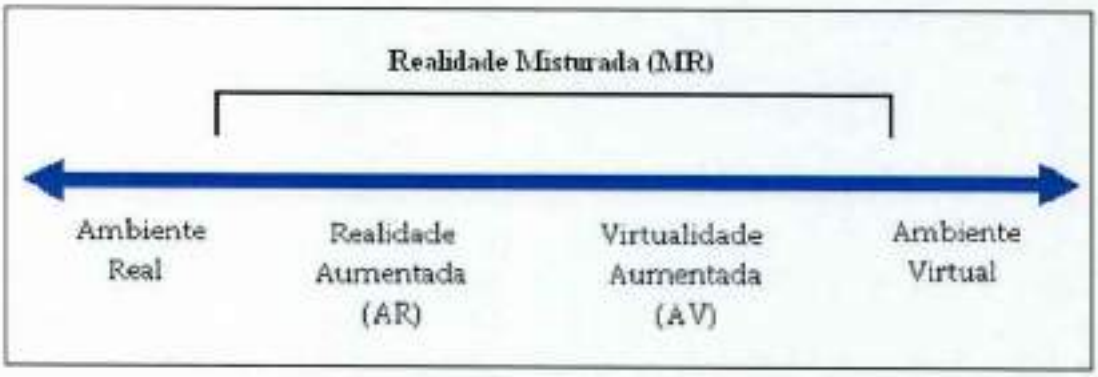


Figura 3 - Continuidade Virtual Definida por Milgram

Na representação da Continuidade Virtual, a Realidade Virtual (Virtual Reality – VR) representa uma simulação sintética em 3D do mundo real, a qual imerge completamente o usuário, não permitindo assim a visualização do mundo real ao seu redor. A Realidade Virtual é definida então como estando no extremo oposto aos Ambientes Reais.

A Realidade Aumentada (Augmented Reality – AR) é uma variação da VR e permite ao usuário a visualização do mundo real, com objetos virtuais dispostos dentro desse mundo real ou em composição com o mesmo. Assim, AR suplementa realidade, ao invés de completamente substituí-la. Essa definição contrasta com a Virtualidade Aumentada (Augmented Virtuality – AV), onde o usuário está imerso em um mundo completamente virtual com objetos 3D contendo texturas do mundo real, simulando assim esse ambiente. Para melhor entender o conceito de AR, idealmente, ao usuário seria como se os objetos virtuais e reais coexistissem no mesmo espaço, similar aos efeitos utilizados no filme "Uma Cilada Para Roger Rabbit". Alguns pesquisadores definem a AR como uma tecnologia que requer o uso de Head-Mounted Displays (HMDs). Para evitar limitações de AR a tecnologias especificas, será usada a definição usual de AR, como sistemas que possuem as três seguintes características:

1) Combina real e virtual;

2) Interage em Tempo Real;

3) Registra em 3D.

Esta definição engloba outras tecnologias além de HMDs enquanto preserva os componentes essenciais de AR. Por exemplo, não se inclui filmes ou overlays 2D. Filmes como "Jurassic Park" ou "Uma Cilada para Roger Habbit" não se integram nesta categoria por não possuírem esta interação em tempo real (e, no segundo caso, também não é feito este registro em 3D dos objetos virtuais) (AZUMA, 1997).

Figura 4 - Cena do filme "Uma Cilada para Roger Rabbit". Não pode ser considerado como AR, pois o ator não interage com o personagem em tempo real (ROGERRABBIT)

Figura 5 - "Livro Mágico", projeto que atende aos requisitos de AR. Interação em tempo real com o cenário virtual, combinação do real (personagem) e virtual (cenário) e registros 3D (ARTOOLKIT)

Para que os objetos 3D possam ser registrados corretamente no ambiente real, é necessária a utilização de elementos de posicionamento como, por exemplo, o GPS (Global Positioning System) e sensores magnéticos. Juntamente com os HMDs, esses dispositivos possuem um custo de aquisição elevado. Porém, dispositivos como webcams, monitores de vídeo convencionais, PDAs ou telefones celulares com câmera também podem ser utilizados, com a vantagem de esses possuírem um custo menor. Para o uso desses dispositivos, geralmente o registro é realizado através da visão computacional, onde alguma forma de padrão indica o posicionamento do ambiente ou usuário.

Algumas áreas de aplicação da Realidade Aumentada são citadas abaixo:

- Medicina: Em determinados tipos de exames médicos como, por exemplo, tomografias computadorizadas, ressonância magnética e ultra-som, o cirurgião pode ter a visão necessária da anatomia interna do paciente renderizada em tempo real, podendo assim planejar a cirurgia a ser realizada, como mostrado pela figura 6:

Figura 6 - Imagens renderizadas sobre os pacientes (AZUMA, 1997; BASTOS, N., 2005)

- Engenharia, Manutenção e Planejamento: Nessas áreas, a Realidade Aumentada pode ser bastante útil para a visualização de projetos dentro do ambiente de operação, desde a instalação de novos componentes até pequenos reparos, como pode ser visto pela figura 7:

Figura 7 - Exemplos de aplicações de AR nas áreas de Engenharia, Manutenção e Planejamento (AZUMA, 1997; BASTOS, N., 2005)

- Entretenimento e Jogos: Essa área concentra grande parte das aplicações de Realidade Aumentada até o momento que podem ser utilizadas desde as

transmissões de esportes ao vivo até a criação de jogos que usem a tecnologia, como mostrado pela figura 8:

Figura 8 - Aplicações na área de entretenimento e jogos (BASTOS, N., 2005)

Dentro da área de jogos, abordada pelo projeto, a utilização da Realidade Aumentada, permite a criação de jogos originais, com maior interação com o usuário, desde a sua imersão no jogo possibilitando a visualização de feições e maior contato com os outros jogadores até a projeção de diferentes jogos (xadrez, dama, gamão, ludo) num mesmo objeto real como um tabuleiro (realidade tangível), aumentando ainda mais o conceito de coletividade nos jogos. Além disto, o jogo torna-se ideal para o aperfeiçoamento da tecnologia, pois este não exige uma alta precisão como em aplicações médicas, por exemplo, permitindo a aquisição de equipamentos que possuem um baixo custo, entre eles câmeras de vídeo e o uso de processadores com mais de um núcleo, deixando que um deles seja exclusivamente

responsável pela captura e processamento das imagens provenientes do mundo real.

Existem diversas bibliotecas de Realidade Aumentada, e estas se utilizam de diversos tipos de linguagem para sua implementação, sendo que, após estudos e testes com estas bibliotecas, o grupo optou por utilizar a biblioteca jarToolkit, uma adaptação da biblioteca de Realidade Aumentada já consolidada ARToolkit (a qual foi desenvolvida utilizando C++) para Java. Maiores detalhes sobre esta parte do projeto serão exibidos em tópico posterior.

2.2 Engine de Jogos

Um engine de jogos é um componente de software fundamental para um jogo ou qualquer aplicação que possua interação gráfica em tempo-real. Ele provê a tecnologia para suporte, simplifica o desenvolvimento e, comumente possibilita a execução do jogo em múltiplas plataformas, como diferentes *consoles* ou então sistemas operacionais. Estas funcionalidades tipicamente providas por um engine de jogos incluem um aplicativo para renderização de gráficos tanto 2D quanto 3D, algoritmos que implementam modelos físicos e detectam colisão, ferramentas para som, scripts, animação, inteligência artificial, comunicação em rede e grafos de cenas (Game Engine).

Seu uso mostra-se fundamental principalmente para jogos que possuam interação 3D, facilitando enormemente o tratamento dos artefatos gráficos.

No decorrer do projeto, foi feito um estudo para a escolha do engine a ser utilizado, o qual será detalhado em posterior item deste relatório. Por fim, foi escolhido o enJine, um engine desenvolvido utilizando tecnologia Java, desenvolvido pelo próprio laboratório Interlab – Laboratório de Tecnologias Interativas – da Escola Politécnica, por se tratar de um engine didático, o também pelo fato deste projeto agregar valor ao engine, o que seria inviável em um engine comercial, já que a maioria destes não possuem seu código-fonte aberto.

3 Especificação do Projeto de Formatura

Nesta seção serão detalhados o projeto, sua especificação e toda a modelagem necessária para seu entendimento, tanto para sua posterior implementação, quanto para completo entendimento aos leitores deste trabalho, como também as ferramentas necessárias para esta especificação.

3.1 Ferramentas

3.1.1 Biblioteca de Realidade Aumentada

A escolha da biblioteca de realidade aumentada a ser utilizada teve inicio com pesquisas sobre bibliotecas existentes, para uma posterior análise e seleção da mais adequada, como mostrado pela tabela abaixo:

Tabela 1 - Pesquisa sobre Bibliotecas de Realidade Aumentada

Biblioteca	Descrição	Sistema	Pontos fortes	Pontos fracos
ARStudio (ARSTUDIO)	-		-	
DART (DART)	Extensão do Macromedia Director	Macromedia Director 8.5	-	Não é adaptado para nosso caso
ARToolkit (ARTOOLKIT)	Biblioteca mais consolidada	C++	-	-
JARToolkit (JARTOOLKIT)	Adaptação do ARToolkit para Java	Java	Utiliza a mesma linguagem do enJine	-
ARTag (ARTAG)	-	C++	Adaptado para nosso caso	-
ARToolkit Plus (APLUS)	Para handhelds	C++	-	Biblioteca para usuários avançados

ARToolKit for AMIRE (AMIRE)	Baseado no ARToolkit 2.65	AMIRE	-	Para uso no AMIRE
CperCV (OPENCV)	-	-	-	Não funcional para nossa aplicação
OpenIllusionist Project (OI PROJECT)	-	-	-	Não funcional para nossa aplicação
OperVIDIA (OPENVIDIA)	-	-	-	Requer hardware muito avançado, não adequado para pesquisa

O ARToolKit (ARTOOLKIT) é uma biblioteca que fornece funções que facilitam o desenvolvimento de aplicações de realidade aumentada. Basicamente, seu funcionamento baseia-se em técnicas de visão computacional para calcular a posição real da câmera e a orientação relativa aos marcadores, permitindo ao programador exibir objetos virtuais sobre eles.

A figura 9 mostra o funcionamento usual de um aplicativo que utiliza as funções do ARToolkit.

Figura 9 - Funcionamento de um aplicativo usual que se utiliza do ARToolkit (ARTOOLKIT)

Porém, o uso do ARToolkit está restrito aos usuários das linguagens C e C++, tornando-se este um pré-requisito para que se possam ser desenvolvidos aplicativos de realidade aumentada com o uso dessa biblioteca.

Para permitir o uso das funcionalidades do ARToolkit aos usuários da linguagem Java, foi desenvolvido o JARToolkit, que são classes que permitem o acesso a essas funcionalidades através da linguagem Java. Essa ligação é realizada facilmente através do JNI (Java Native Interface) e pela arquitetura do ARToolkit, que permite o acesso de suas funções através de classes (GEIGER et al., 2002).

Mas as funcionalidades 3D do ARToolkit não foram implementadas, tendo como objetivo o uso dos pacotes já existentes, como o Java3D (para a implementação em alto nível) e o GL4Java e o JOGL (os quais criam uma camada de acesso às funcionalidades da biblioteca OpenGL, de mais baixo nível, em Java), flexibilizando a renderização dos objetos.

A arquitetura do JARToolkit divide o ARToolkit em duas classes:

- JARToolkit - Classe que encapsula todas as funções do ARToolkit necessárias para o rastreamento. Por enquanto, somente as funções básicas estão implementadas, com o foco voltado para a versão Windows;
- JARFrameGrabber - Classe que encapsula todas as funções necessárias para o acesso à entrada de vídeo a partir de uma câmera. Para isso, são utilizadas funções do DirectShow.

Outro dado importante é que a integração entre as funcionalidades de Realidade Aumentada com as do Java3D é feita por um componente denominado *jARToolKit for Java3D*, com diversas classes que facilitam a inclusão de nós nos grafos de cena e o seu comportamento. A classe principal, denominada "*JARToolKit3D*" é responsável pela alocação e inicialização das principais instâncias utilizadas, entre elas o *jARToolKit* e *jARFrameGrabber*. Além desta, outras classes importantes são o "*ARPatternTransformGroup*", herdeira da classe "*TransformGroup*" do Java3D e responsável pelas informações de um marcador que

será utilizado e pela alocação do grupo do grafo de cena que será influenciado pela movimentação do marcador, e a classe "*ARBehavior*", herdeira indireta do "*Behavior*" (pois existem outras classes herdeiras entre as duas) do Java3D e responsável pela atualização das imagens obtidas da câmera e pela identificação dos padrões e renderização dos objetos virtuais sobre eles.

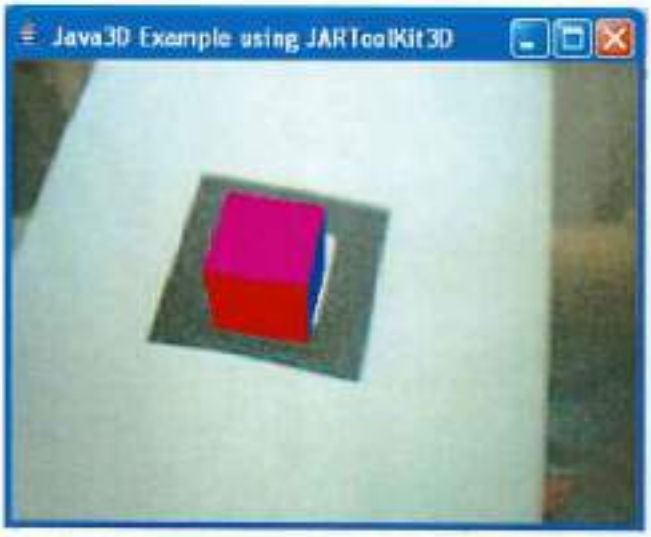


Figura 10 - Aplicativo de demonstração fornecido com o jARToolKit

Através do estudo realizado, descobriu-se que o stream de vídeo é na realidade um plano de fundo ("*background*") obtido pela classe "*JARToolKit3D*" a partir da imagem fornecida pelo "*ARBehavior*".

Além disso, verificou-se que através da imagem proveniente do vídeo é realizado um cálculo que retorna uma matriz de transformação geométrica de dimensão 4x4 que representa a translação, rotação e escala (a qual nesse caso não é utilizada) referente à posição do marcador. Essa matriz é utilizada então para realizar a mesma transformação no objeto virtual, oferecendo ao usuário a sensação do objeto posicionado sobre o marcador.

3.1.2 Java3D

O Java3D é uma interface criada para o desenvolvimento de aplicações gráficas tridimensionais em Java, executada sobre bibliotecas de mais baixo nível, tais como o OpenGL e DirectX (Direct3D). Possui várias tecnologias multimídia e gráficas que permitem o desenvolvimento de aplicações 3D que exploram o conjunto de facilidades e vantagens da plataforma Java, tais como a orientação a objetos, segurança e independência de plataforma (BICHO et al, 2002).

Com essa interface, os programadores não familiarizados com os detalhes pertinentes à programação 3D podem explorar esse novo universo mais facilmente, agilizando a criação e o desenvolvimento de aplicações.

Uma aplicação desenvolvida no Java3D é projetada a partir de um grafo de cena contendo objetos gráficos, luz, som, objetos de interação, entre outros, que possibilita ao programador criar mundos virtuais com personagens que interagem entre si e/ou com o usuário. Um grafo de cena é uma estrutura de dados que utiliza uma abordagem de alto nível para modelagem e gestão de cenas, ao contrário das APIs gráficas básicas (OpenGL e Direct3D). Os usuários desse tipo de estrutura de dados não necessariamente precisam conhecer os detalhes de implementação de baixo nível para utilizá-la. Um dos conceitos centrais a respeito de um grafo de cenas é que este implementa uma estrutura hierárquica (VALENTE, 2004).

A figura 11 mostra um exemplo de como os objetos são inseridos dentro de um grafo de cena:

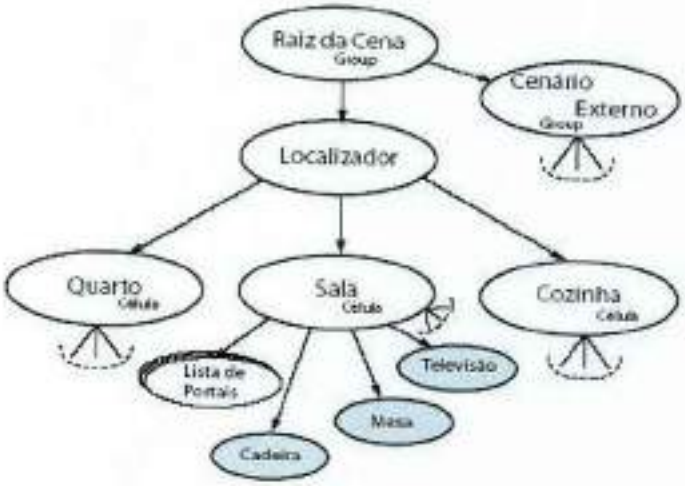


Figura 11 - Objetos em um grafo de cena (SILVA, 2003)

A figura 12 apresenta o grafo de cena típico do Java3D, criado utilizando o software Interlab3D:

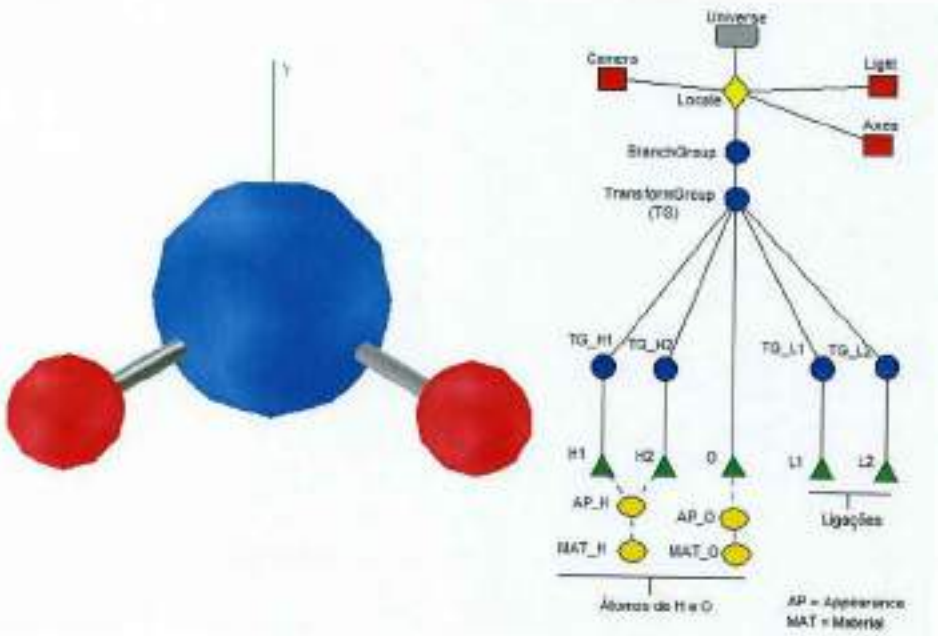


Figura 12 - Grafo de Cena - Interlab3D (INTERLAB3D)

3.1.3 enJine

O enJine é um conjunto de classes e funcionalidades que facilitam o desenvolvimento de um jogo implementado sobre a linguagem Java. Essas classes, agrupadas numa estrutura de pacotes reutilizáveis, fornecem vários serviços necessários ao funcionamento do jogo, como por exemplo, o tratamento dos dados provenientes da entrada, a manipulação e a exibição dos objetos do jogo e ambientes virtuais tridimensionais (utilizando-se das funções do Java3D), o comportamento desses objetos, a reprodução de efeitos sonoros, entre outros. Estes diversos pacotes, mostrados na figura 13, foram montados a partir das diversas funcionalidades do enJine, porém a estrutura apresentada foi criada na concepção do mesmo, logo alguns destes pacotes ainda não estão implementados – como o pacote de inteligência artificial (AI), o pacote Scripts e o pacote Haptics –, enquanto outros pacotes não estão indicados, como o pacote responsável pelas colisões.

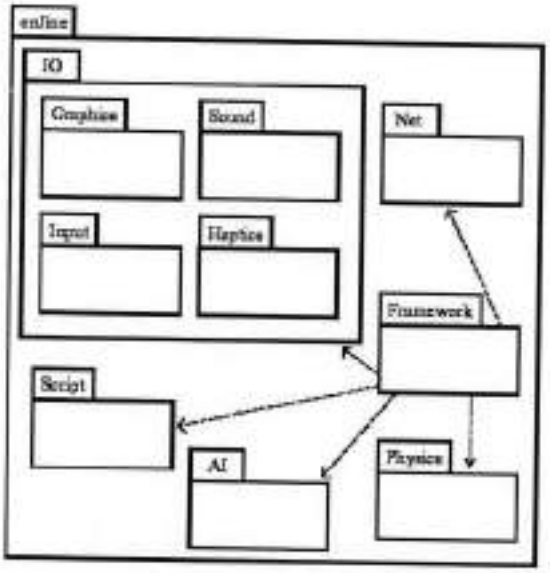


Figura 13 - Concepção dos pacotes do enJine (BERNARDES, J., 2005)

Analisando a figura anterior, pode-se notar que a interação do jogo com o ambiente externo citado anteriormente (atualização da imagem do jogo vista pelo usuário) é feita por meio dos pacotes IO (Input/Output) enquanto seu roteiro, reações a eventos externos é determinado nos pacotes restantes (AI, que envolve algoritmos de Inteligência Artificial, Physics, entre outros não listados que controlam as simulações físicas como colisão de objetos, por exemplo). Além disto, os pacotes foram criados genericamente, buscando sempre uma maior flexibilidade para a implementação dos jogos, sejam estes em rede ou não, multi-jogadores ou único, entre outros fatores.

Utilizando qualquer linguagem orientada a objetos, no caso o Java na implementação do enJine, além desta modularização em pacotes foi possível abstrair diversos elementos de um jogo e transformá-los em classes agregadas aos seus atributos, por exemplo, a criação de uma entidade "cenário" e suas características, a entidade personagem, entre outras (PRESSMAN, 2002). Desta

forma a manipulação do jogo se torna mais fácil e eficiente, o seu estado pode ser determinado apenas pelas mudanças de alguns atributos de objetos como o cenário (movimento de um objeto) ou do personagem (mudança de humor).

Uma estrutura de um jogo implementado com enJine pode ser descrito conforme a figura seguinte, onde os módulos "*Inicialização*" e "*Limpeza*" são executados apenas uma vez para alocar e liberar, respectivamente, os recursos necessários na execução do jogo. A real execução do jogo está localizada no "*Laço Principal*", onde inúmeras *threads* do Java3D são executadas paralelamente com o objetivo de representar o estado do jogo (renderização) ao usuário por meio de atualizações de cenários e personagens.

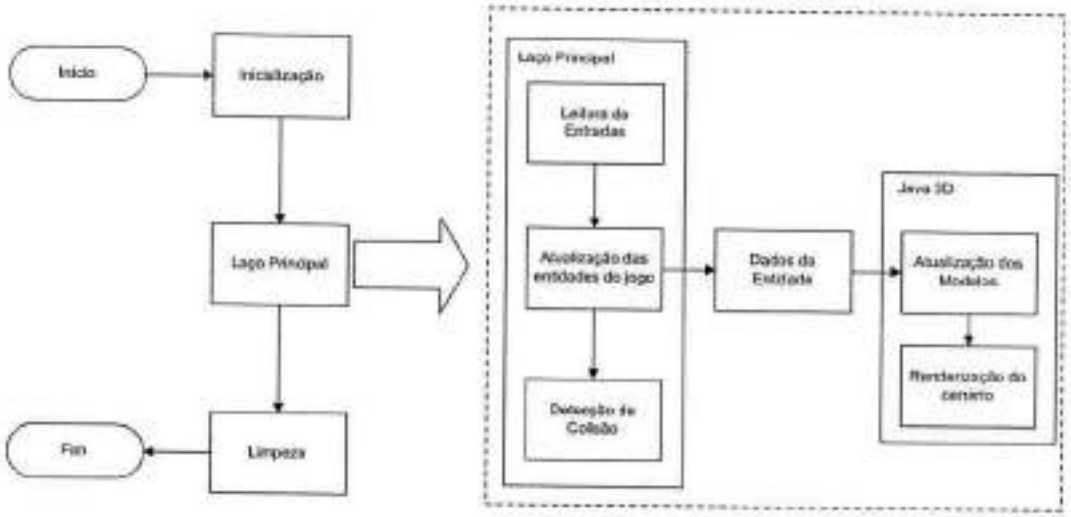


Figura 14 - Representação da estrutura de um jogo implementado no enJine

Como dito, o enJine ao criar um novo jogo (instância da classe *game*) gera abstrações de suas entidades como o cenário e outros objetos de jogo, e seus estágios (instâncias da classe *gameState*) e ainda consegue manipulá-los de maneira eficiente por meio do "Laço Principal" onde são implementadas as reações

aos eventos causadores de mudança de estágio, como pode ser notado na figura adiante.

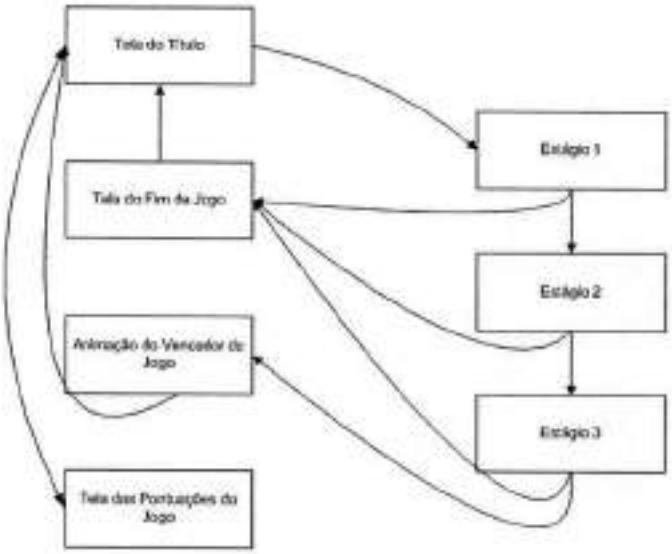


Figura 15 - Exemplo mostrando a seqüência de execução de um jogo com três estágios e outras telas "splash screen", as quais exibem mensagens ao jogador (enJine)

Lembrando, ainda, que o jogo desenvolvido recebe as ações do usuário por meio da Realidade Aumentada, foi necessário estudar métodos que capturassem o movimento do usuário e transformá-lo em uma ação reconhecida dentro de um ambiente de jogo (andar, pular, mover para frente, para o lado, para trás, atirar, bater, pegar, etc.). Por meio de pesquisas foram encontrados diversos plug-ins, já citados, que são capazes de realizar esta transformação e, dentre eles destacam-se o ARToolkit e o jARToolKit e, como visto, eles se diferem em algumas funcionalidades extras e a linguagem que foram implementados.

Desta forma, cabe ainda à solução transformar a captura de movimentos da câmera, feita pelo jARToolKit, em movimentos (valores de entrada) aceitos no jogo

(tratados no enJine). Na figura 16 pode-se observar um exemplo desta integração realizada.

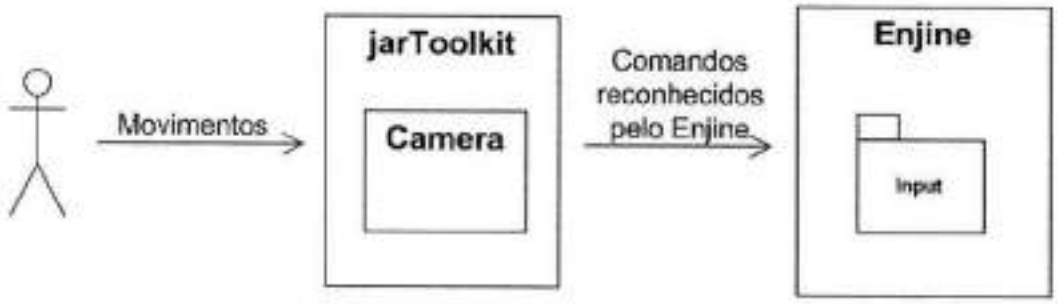


Figura 16 - Captura dos movimentos e a conversão em comandos no enJine

O pacote Input do enJine, descrito na figura 17, é baseado num modelo de abstração dos dispositivos de entrada formado por um conjunto desses dispositivos que fornecerão as ações compreendidas dentro de um jogo. Cada dispositivo (como um joystick ou teclado) é representado por uma subclasse "*InputDevice*" e este possui uma série de objetos "*InputSensor*" que representam elementos como botões do joystick ou teclas do teclado. E, a fim de vincular estes botões (*InputSensor*) a uma ação de um jogo (*InputAction*) como "pular", "mover para direita", por exemplo, existe uma última classe denominada "*InputManager*".

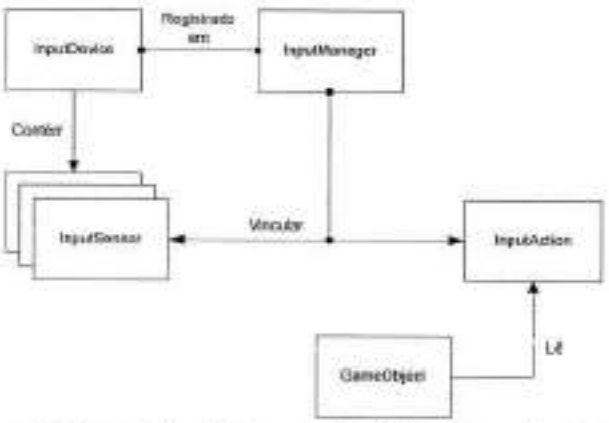


Figura 17 - Generalização das entradas do jogo no enJine (enJine)

Uma característica importante desse modelo de entrada é que tanto os *InputSensors* quanto as *InputActions* possuem um parâmetro de intensidade, permitindo o uso de dispositivos analógicos.

Enfim, determinadas todas as ações possíveis num determinado jogo (instanciações de *InputAction*), será necessário vinculá-las, por meio do "*InputManager*" a diferentes movimentos ("*InputSensor*") do usuário em frente à câmera ("InputDevice"). Estes movimentos, no entanto, serão gerados por uma nova camada a ser implementada que captura as diferentes posições do usuário – coordenadas (x, y, z), por exemplo – ao longo de um intervalo de tempo, fornecido pelo JarToolkit, e as relaciona com um movimento reconhecido pelo enJine (alguma instância do "InputSensor" criada).

Para que fosse possível a implementação de um jogo que interaja com o usuário por meio da Realidade Aumentada, foi necessário um estudo mais profundo sobre desenvolvimento de jogos, envolvendo sua arquitetura, modo de implementação, ferramentas auxiliares, por exemplo. A figura 18 descreve a arquitetura em camadas necessária para a execução de um jogo incluindo a plataforma escolhida para o projeto, desde recursos de hardware necessários para tal, relacionando todos os requisitos necessários, até a definição dos comandos de entrada para o jogo através da biblioteca de AR.

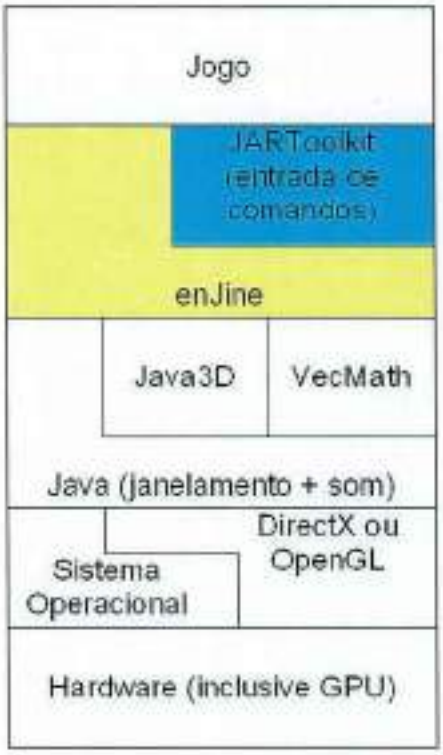


Figura 18 - Digrama em Camadas

Uma nova camada (middleware) foi desenvolvida de forma genérica a fim de aceitar inúmeras formas de movimentos e gerar, a partir destes, comandos flexíveis a qualquer jogo. No entanto, durante esta especificação e definição, duas possíveis alternativas de projeto para o grupo apareceram:

- Criação de um middleware genérico e o desenvolvimento de um "demo" para a demonstração do funcionamento do mesmo;
- Criação de um middleware específico a um determinado tipo de jogo sofisticado desenvolvido pelo grupo.

Esta decisão foi tomada em conjunto pelo grupo, após analisar e verificar os riscos que impactariam cada uma das soluções. A solução escolhida pelo grupo foi a criação de um middleware genérico, que pudesse ser adaptável para os mais diferentes jogos, como será descrito posteriormente.

3.1.4 Biblioteca de Decodificação de MP3

Esta funcionalidade não pertencia inicialmente ao escopo do projeto, porém o grupo avaliou sua necessidade e constatou que seu uso iria agregar muito valor, tanto à solução quanto ao enJine, no qual este seria incorporado.

Para tanto, foi feita uma pesquisa sobre bibliotecas que decodificam estes arquivos, e também sobre o funcionamento desta decodificação.

Dentre as bibliotecas pesquisadas, fui utilizada para implementação a biblioteca JavaZoom (JAVAZOOM), a qual consiste em outras duas bibliotecas:

- jl1.0.jar: Biblioteca responsável pela decodificação dos arquivos de música;
- MP3.jar: Biblioteca responsável pela obtenção de informações do arquivo de música, tais como *bitrate e freqüência.*

Esta biblioteca basicamente lê o arquivo MP3, converte seu *bitrate* e freqüência para o padrão sem compactação de áudio e executa os métodos já existentes na máquina virtual. Um outro fator importante é que esta biblioteca é livre, e está disponível sob a licença LGPL - Lesser General Public License –, ou seja, é livre para o uso com o enJine.

3.2 Especificação da Interface de Realidade Aumentada

O objetivo do desenvolvimento de uma interface entre o Engine e a biblioteca de Realidade Aumentada é acrescentar ao Engine a capacidade de

desenvolvimento de jogos que utilizem este conceito, sendo que esta solução deve ser completa, e precisa suportar todos os requisitos necessários, tanto funcionais quanto não-funcionais.

Sua concepção abrange etapas de aprendizado e familiarização com este tipo de ferramenta, e também etapas de refinamento da ferramenta, adaptando-a ao seu uso no projeto, as quais deverão prover entradas para as fases seguinte, detalhando e implementando os conceitos aqui apresentados para o desenvolvimento do jogo.

A interface de realidade aumentada foi tratada pelo grupo como um pacote adicional ao enJine, já que este se baseia em componentes e a inserção de um novo componente pode ser feita de maneira simples.

Porém, durante a especificação do jogo, o grupo percebeu que algumas das funcionalidades do enJine não estavam desenvolvidas. Um destes problemas foi a falta de suporte do enJine a arquivos de música no formato MP3. Assim, o escopo do projeto foi um pouco expandido, para que esta funcionalidade – que o grupo também avaliou como importante, pois o jogo escolhido para ser desenvolvido foi um jogo de dança – fizesse também parte do escopo do projeto.

A seguir serão mostrados os requisitos, casos de uso e outros diagramas que especificaram esta solução do projeto:

3.2.1 Requisitos Funcionais

- EnJine deve fornecer como entrada, além das duas já existentes – teclado e mouse –, a entrada como realidade aumentada;

- EnJine deve fornecer músicas e sons, além dos dois formatos já suportados – wave e midi –, músicas no formato MP3 (formato que utiliza compressão de áudio);
- EnJine deve calibrar automaticamente a posição dos marcadores a serem utilizados pelos jogos que utilizem Realidade Aumentada (posições x, y e z do marcador), de forma que não seja necessária uma calibração pelo usuário cada vez que ele for jogar.

3.2.2 Requisitos Não-Funcionais

Os requisitos não-funcionais foram levantados segundo a regra criada pela Hewlett-Packard – um conjunto de fatores de qualidade de software – ao qual foi designado a sigla FURPS, derivada das palavras inglesas para funcionalidade, usabilidade, confiabilidade, desempenho e suportabilidade (JUNIOR, 2003):

- A funcionalidade ("functionality" em inglês) é aferida avaliando-se o conjunto de características e as capacidades do programa, a generalidade das funções que são entregues e a segurança do sistema global. Para tanto, o enJine deve fornecer formas diferentes de interação usuário-jogo utilizando realidade aumentada, para que esta solução seja adaptável para os diferentes tipos de jogos existentes (esta funcionalidade será mais bem descrita no tópico "Metodologia");
- A usabilidade ("usability" em inglês) é avaliada considerando-se os fatores humanos, a estética global, a consistência e a documentação. A solução deve

ser contida no enJine, e seu uso deve ser tão simples quanto o uso do mesmo para o desenvolvimento de jogos que não se utilizem de Realidade Aumentada;

- A confiabilidade ("reliability" em inglês) é avaliada medindo-se a freqüência e a gravidade de falhas, a acurácia dos resultados de saída, o tempo médio entre as falhas (MTBF), a capacidade de recuperar falhas e a previsibilidade do programa. O enJine deve reconhecer os padrões utilizados para captura de movimento, com uma porcentagem mínima de 80% dos movimentos executados (movimentos estes executados a uma velocidade máxima de 1 m/s), em ambientes com iluminação intermediária definida em, no mínimo, 1000 lux (lembrando que lux é a unidade de medida que indica a incidência de lux em uma área de um metro quadrado). Outro fator importante é a distância do jogador à câmera, a qual deve ser de, no máximo, dois metros e meio;
- O Desempenho ("performance" em inglês) é medido avaliando a velocidade de processamento, o tempo de resposta, o consumo de recursos, o throughput e a eficiência. O processamento de um jogo desenvolvido utilizando Realidade Aumentada deve ter o mesmo desempenho que um jogo normal. O tempo de resposta deve aceitar atrasos de, no máximo, 1 segundo – levando-se em conta a rapidez de processamento das câmeras utilizadas atualmente;
- A suportabilidade ("supportability" em inglês) combina a capacidade de ampliar o programa (extensibilidade), adaptabilidade, capacidade de serviço (esses três atributos representam um termo mais comum - manutenibilidade), além de capacidade de teste, compatibilidade, configurabilidade, a facilidade

com que um sistema pode ser instalado e a facilidade com que problemas podem ser detectados. Como já mencionado, a solução deve ser tratada como um componente adicional do enJine, não interferindo nos outros componentes já existentes.

3.2.3 Especificação Funcional

Para a especificação da interface de realidade aumentada, como para o jogo, foi utilizado UML – Unified Modeling Language – como linguagem para a modelagem, para que fosse possível visualizar os produtos e serviços de seu trabalho em diagramas padronizados.

Para a especificação das funcionalidades que fazem parte do escopo desta parte do projeto, não faz sentido a descrição das mesmas utilizando casos de uso usuais, pois as mesmas não são funcionalidades apresentadas diretamente para atores, e sim para jogos que utilizem a plataforma como Engine. Assim, foi criado um ator abstrato, que é o jogo a ser desenvolvido utilizando o enJine. A figura 19 mostra este diagrama de casos de uso:

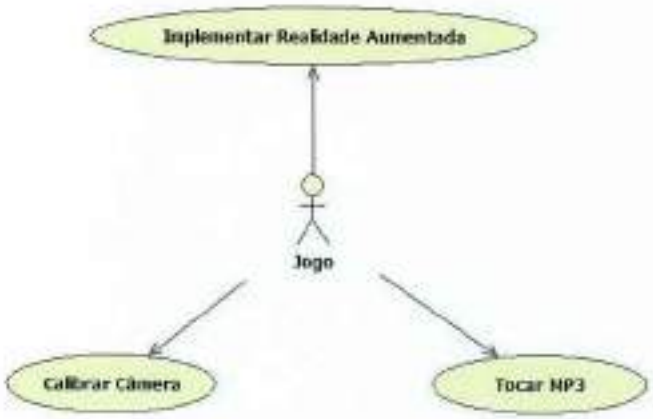


Figura 19 - Casos de Uso da Solução para enJine

A seguir, estas funcionalidades são descritas, através de diagramas tanto estáticos – como diagramas de classes – quanto dinâmicos – como de seqüência e de atividades, para que se tenha uma visão completa da solução, abrangendo tanto a interação destas funcionalidades com os atores quanto provendo o necessário para a estruturação da arquitetura e desenvolvimento da solução.

Para implementar realidade aumentada, foi descrita a atuação da solução como integração entre a biblioteca de realidade aumentada e o enJine através do diagrama de atividades descrito pela figura 20:

Figura 20 - Implementar Realidade Aumentada - Diagrama de Atividades

Para modelar a calibração da câmera, o seguinte diagrama de atividades foi criado primeiramente:

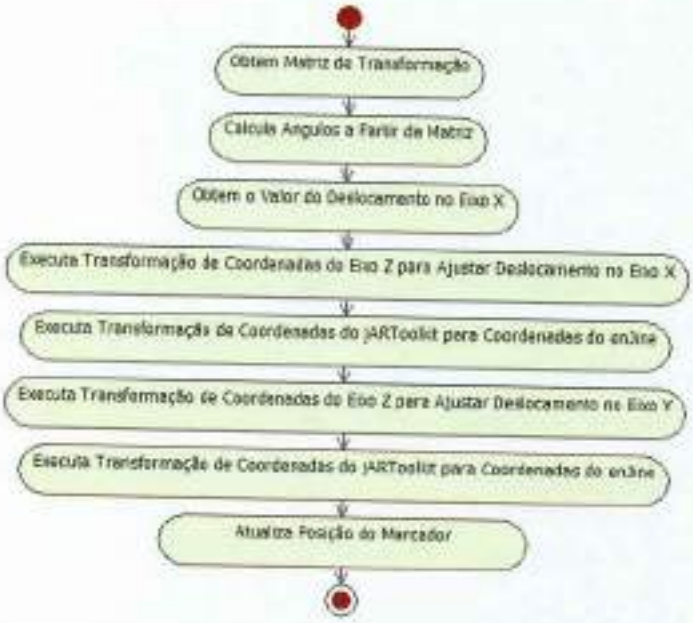


Figura 21 - Calibrar Câmera - Diagrama de Atividades

Porém, como esta solução se mostrou incompleta, já que a calibração para posições no eixo z não é levada em conta, foi implementada uma nova solução utilizando a calibração já existente no jarToolkit, a qual leva em consideração ambos os olhos do usuário para cálculo da matriz de transformação. Esta solução será mais bem explicada em seção posterior.

O suporte a arquivos de música no formato MP3, funcionalidade que originalmente não fazia parte do escopo, mas que se mostrou ser de extrema importância para o jogo a ser desenvolvido, pode ser detalhado pelo diagrama de seqüência a seguir:

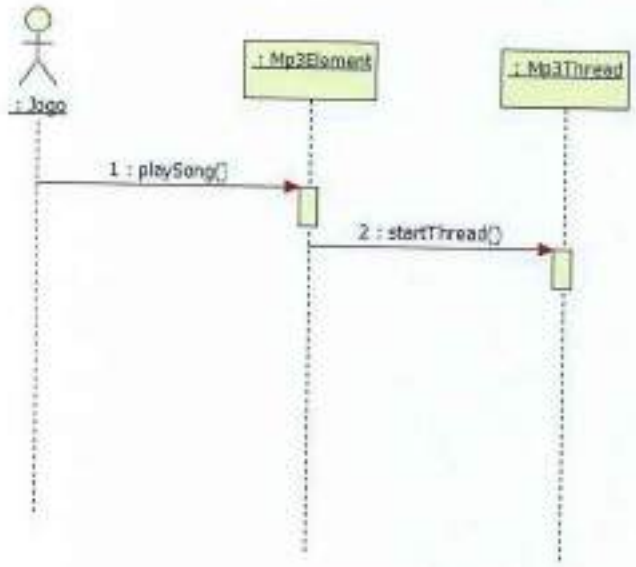


Figura 22 - Tocar MP3 - Diagrama de Seqüência

Foram também gerados os diagramas de classes para o posterior desenvolvimento destas soluções, e estes serão descritos no item "Projeto e Implementação".

3.3 Especificação do Jogo Utilizando Realidade Aumentada

O desenvolvimento de um jogo não tem seus processos alterados pelo fato da entrada de comandos para o personagem ser a usual – através de um controle, por exemplo -, ou então se utilizando realidade aumentada – capturando os movimentos do jogador através de uma câmera. A estruturação do engine escolhido para desenvolvimento, como mostrado na figura 13, permite que diversos tipos interfaces

com o jogador possam ser utilizados, sem influenciar o jogo em si, graças à orientação a objetos e à arquitetura do enJine.

Para tal, foram definidos o escopo e enredo do jogo a ser desenvolvido, utilizando técnicas de gestão de projetos, para um desenvolvimento otimizado.

O projeto foi desenvolvido tendo como base uma customização do Unified Process, processo criado com base no RUP – Rational Unified Process –, e que é distribuído gratuitamente para sua aplicação e customização (em oposição ao RUP, que é proprietário), através de um processo iterativo, onde ao final de cada iteração, métricas devem ser feitas para medida de todos os pontos de qualidade do software e também para verificação do valor agregado ao software (JACOBSON, 1999).

O ciclo de desenvolvimento foi dividido, utilizando os conceitos do UP (FERREIRA, 2006), em:

- Fase de Elaboração, a qual consiste principalmente na definição do escopo, definição do tipo de jogo a ser desenvolvido;
- Fase de Concepção, a qual define a arquitetura do sistema como um todo, definindo as ferramentas de AR a serem utilizadas, e também a arquitetura do jogo;
- Fase de Desenvolvimento: na qual é criada a primeira versão da aplicação, como prova de conceito unificando o uso de Realidade Aumentada com o jogo desenvolvido;
- Fase de Transição, que compreende o desenvolvimento e entrega da ultima versão da ferramenta de AR desenvolvida e do jogo.

Durante as fases de concepção e desenvolvimento, a organização do desenvolvimento do protótipo com o intuito de facilitar a implementação do mesmo

mostrou-se muito importante, assim como para otimizar o tempo gasto. O detalhamento do processo pode ser exemplificado pela figura 23.

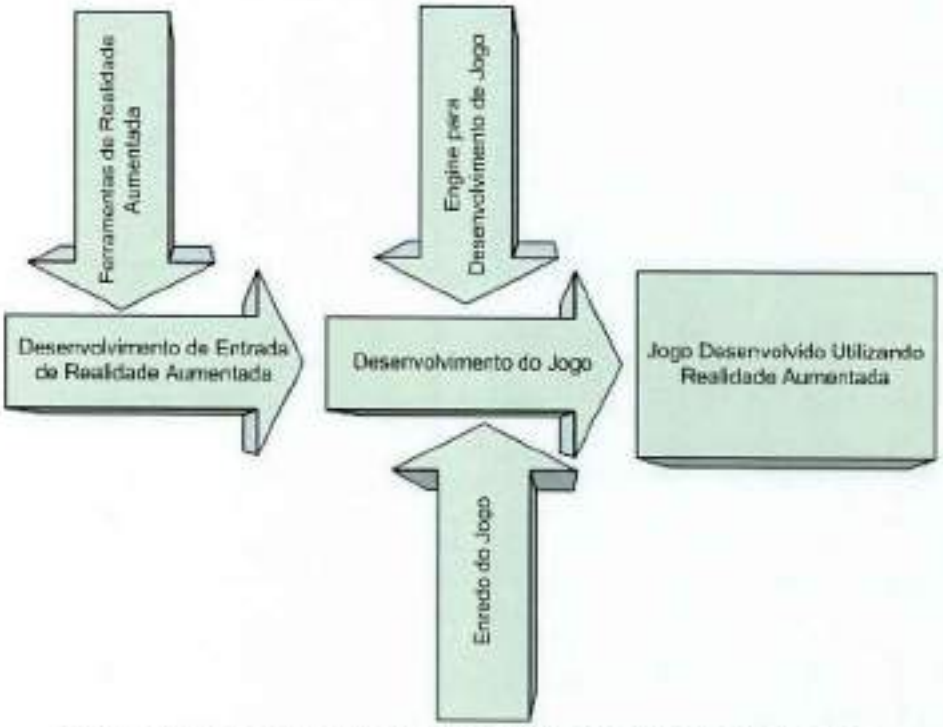


Figura 23 - Processos a serem realizados para o desenvolvimento do projeto

Para definição do tipo de jogo a ser desenvolvido, inicialmente foi realizado um "brainstorm" entre os integrantes do grupo, visando à listagem de algumas modalidades a serem implementadas. Após a listagem dos estilos, as vantagens e desvantagens de se adotar cada um deles foram discutidas considerando as limitações da ferramenta utilizada e os equipamentos utilizados.

Tabela 2 - Escolha do Jogo a ser Desenvolvido

Tipo de Jogo	*Contras*	*Prós*
Adventure (3D)	Pouca interação; Muito tempo para enredo; Complicado para inserir jogador no cenário (R.A.);	Utilização de I.A. e algoritmos
Side-scrooling	Pouca interação; Complicado para inserir jogador no cenário (R.A.); Muito complexo.	Utilização de I.A. e algoritmos
RPG	Pouca interação; Complicado para inserir jogador no cenário (R.A.); Muito tempo para enredo; Muito complexo.	Utilização de I.A. e algoritmos
Corrida	Complicado para inserir jogador no cenário (R.A.); Podemos usar um demo de jogo de corrida já existente.	Bastante funcional (uso de R.A. para simular volante)
Puzzle (Letras embaralhadas)	Pouca interação; Simples demais.	
"Bate-jacaré"	Complicado para inserir jogador no cenário (R.A.); Simples demais.	Alta customização do jogo, novo tipo de interação do jogador. Possibilidade de vários jogadores simultâneos.
Dança	Não utiliza conceitos de física.	Alta customização do jogo, novo tipo de interação do jogador. Possibilidade de vários jogadores simultâneos.
Luta	Complicado para inserir jogador no cenário (R.A.); Muito complexo.	Alta customização do jogo, novo tipo de interação do jogador. Possibilidade de vários jogadores simultâneos.
Pinball	Pouca interação; Complicado para inserir jogador no cenário (R.A.).	
Boliche	Pouca interação; Simples demais.	Alta customização do jogo, novo tipo de interação do jogador.

A partir da análise dos prós e contras, o grupo optou pela implementação de um jogo de dança. Como outros fatores favoráveis, podemos citar que esse é um estilo que oferece ao jogador a sensação de imersão em um ambiente contendo os objetos virtuais sem que estes dificultem sua noção de posicionamento na janela do jogo e a pouca necessidade de elaboração de enredos e modelos gráficos complexos, permitindo focar o trabalho na jogabilidade.

Em relação aos requisitos do jogo, foram definidos os seguintes pontos:

- O jogador deve utilizar um ou mais marcadores para "tocar" no objeto apresentado na tela;
- Possibilidade de dois jogadores dividirem a tela, permitindo assim partidas multi jogadores;
- Utilização das mãos e dos joelhos (representando os pés) para a realização dos movimentos. Assim, cada jogador deve utilizar quatro marcadores, um em cada membro, como exibido pela figura 24;
- Uso de cores para orientar o jogador qual membro utilizar para "tocar" cada objeto.

Figura 24 - Jogador com Marcadores

Nessa fase, também foi definido o nome do jogo, intitulado "we A.R. Dancin'", onde o "A.R." é um trocadilho entre o verbo "estamos" em inglês ("are") e a sigla

utilizada para representar a palavra "Realidade Aumentada". A figura 25 abaixo mostra a primeira versão do logo do jogo:

Figura 25 - Logo do Jogo

A partir destes requisitos e do "brainstorm" feito para que o grupo pudesse discutir e criar idéias para o jogo, as funcionalidades do jogo e os casos de uso que deveriam ser criados para atender as mesmas foram definidos, como exibido na figura 26:

Figura 26 - Casos de Uso do Jogo de Dança

3.3.1 Descrição dos Atores

Os atores definidos para o jogo são e seus papéis no escopo do projeto são:

Figura 27 - Atores do Jogo

- Usuário, pessoa jogadora do jogo, a qual interagirá com o jogo através dos marcadores os quais ela terá nas mãos e nas pernas;

- MakerTimer, timer que será responsável pela manipulação das marcas para acompanhamento na tela;
- Mp3Thread, responsável pela execução de arquivos de música MP3 em paralelo com o jogo, através de um "Thread" de Java (uma tarefa que é tratada pelo enJine e executada em paralelo com o jogo).

3.3.2 Descrição dos Casos de Uso

Nesta seção serão descritos os casos de uso que compõem o jogo desenvolvido, através de sua modelagem em UML, segundo sua prioridade para o projeto.

3.3.2.1 Mostrar Marcas para Acompanhar Música na Tela

Este caso de uso é responsável pela exibição de marcas na tela para que o jogador possa acompanhá-las utilizando-se dos marcadores os quais ele segura. Estas marcas estão definidas como uma lista de eventos a ser verificada e, caso o instante de verificação seja igual ao instante em um dos eventos, deverá ser exibida a marca na tela.

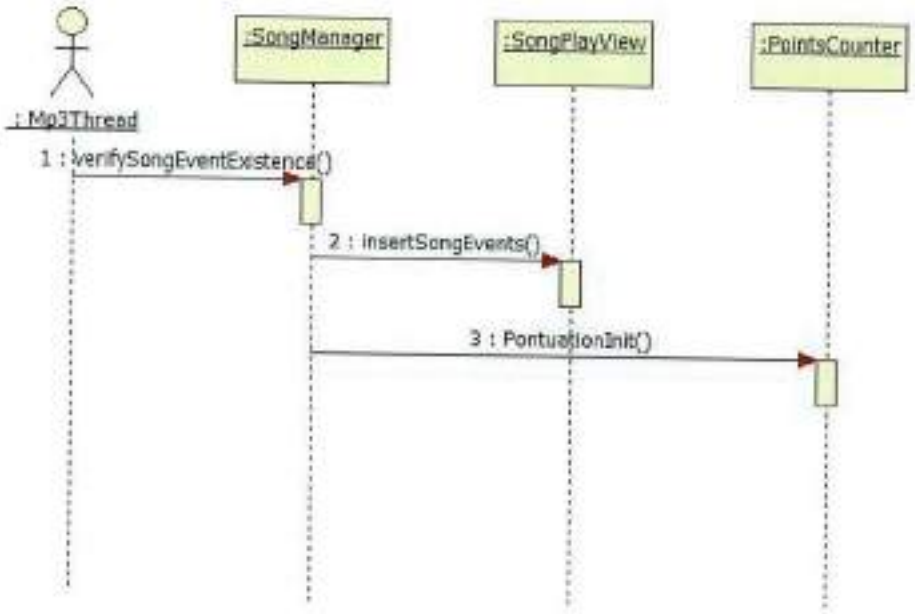


Figura 28 - Mostrar Marcas para Acompanhar Música na Tela - Diagrama de Seqüência

3.3.2.2 Listar Músicas do Jogo

Este caso de uso é responsável pela obtenção da lista de músicas existentes para o jogo. Esta lista é obtida dinamicamente, verificando os arquivos com extensão "mp3" e "arm" existentes na pasta de músicas (ou seja, o jogador pode inserir as próprias músicas). Em seguida é criada uma lista de músicas para que o jogador possa escolher a música para a qual ele deseja jogar.

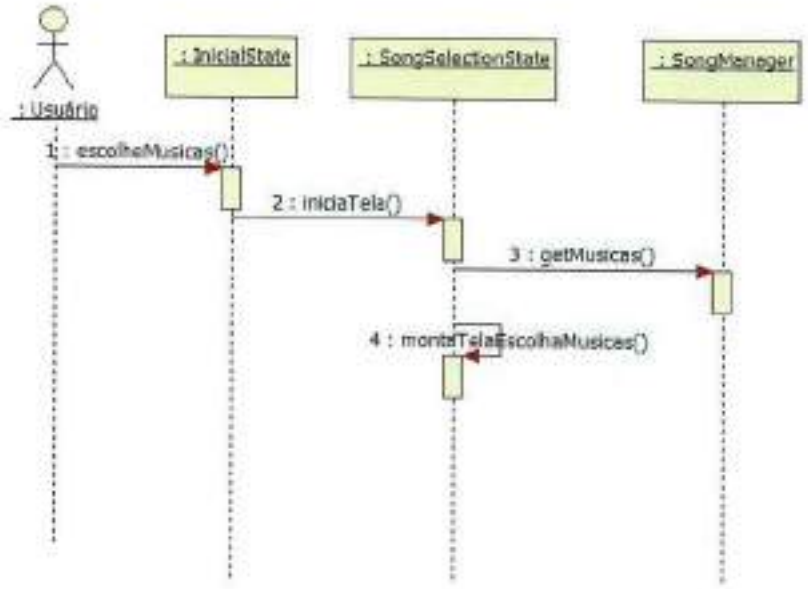


Figura 29 - Listar Músicas do Jogo - Diagrama de Seqüência

Este caso de uso também é responsável pela criação e manipulação da lista de músicas na tela do jogo. Esta deve funcionar do seguinte modo: deverá aparecer o nome de uma música, e setas para o jogador navegar entre as músicas existentes, para que ele possa escolher uma delas.

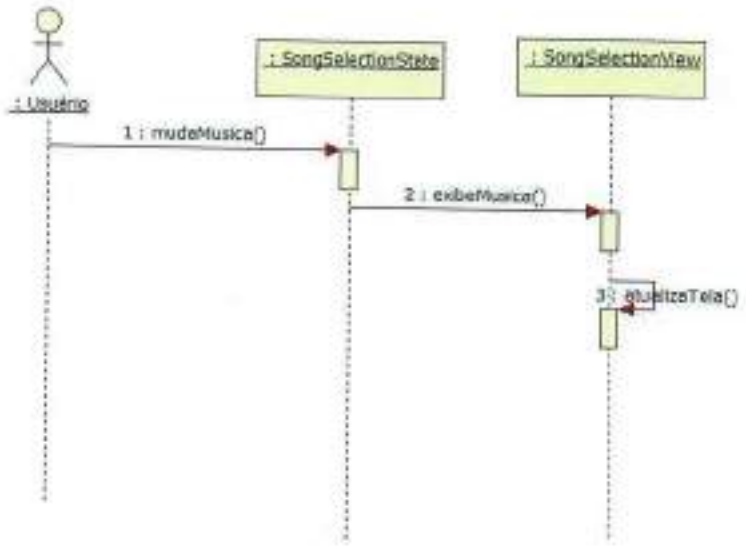


Figura 30 - Navegar pela Lista de Músicas - Diagrama de Seqüência

3.3.2.3 Carregar Música

Este caso de uso se inicia quando o jogador seleciona uma música através da lista de músicas. O jogo deverá ler o arquivo "arm" (que significa A. R. Música), criar um objeto que contenha todos os dados necessários pelo jogo sobre a música, e encaminhar para a tela de jogo.

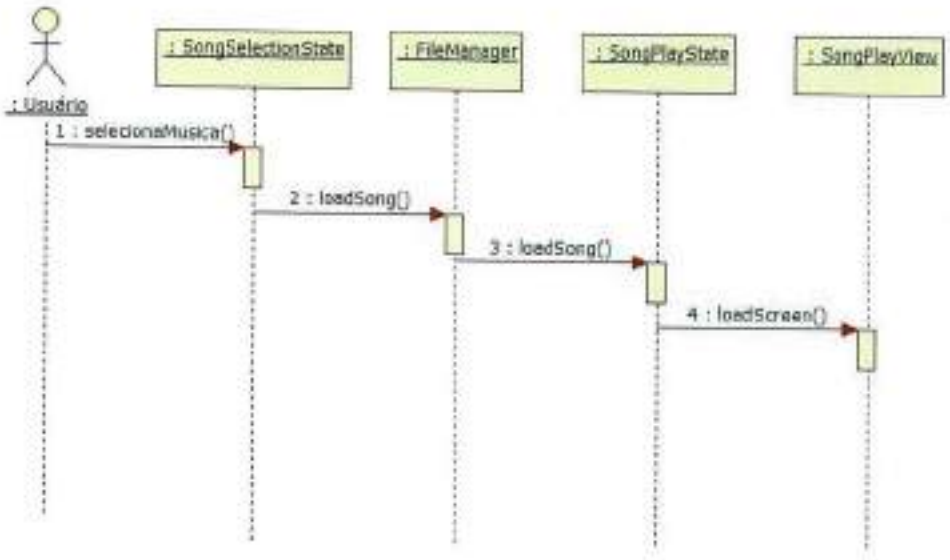


Figura 31 - Carregar Música - Diagrama de Seqüência

3.3.2.4 Escolher Nível de Dificuldade

Este caso de uso possibilita ao jogador a escolha entre três níveis de dificuldade: fácil ("Easy"), normal ou difícil ("Hard"). Esta dificuldade se retrata na quantidade de marcas que é apresentada na tela (quão maior é esta dificuldade, maior o número de marcas que aparecerão).

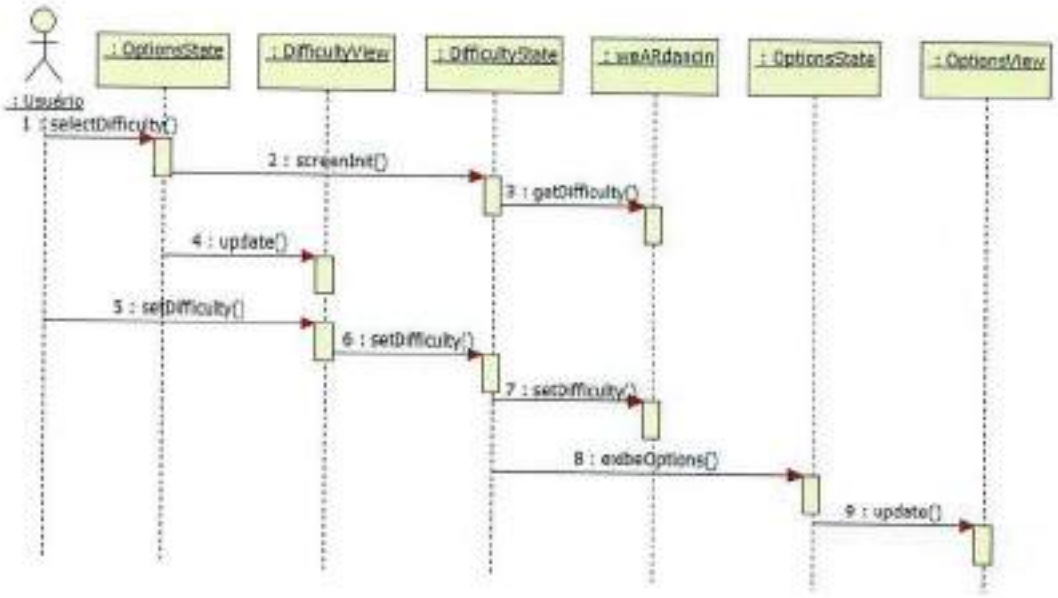


Figura 32 - Escolher Nível de Dificuldade - Diagrama de Seqüência

3.3.2.5 Escolher Número de Jogadores

Este caso de uso possibilita ao jogador a escolha entre jogos com um jogador ou com dois jogadores. Para dois jogadores, a tela de jogo será dividida verticalmente no meio, e os marcadores aparecerão iguais para ambos os jogadores, para que haja uma competição entre os mesmos.

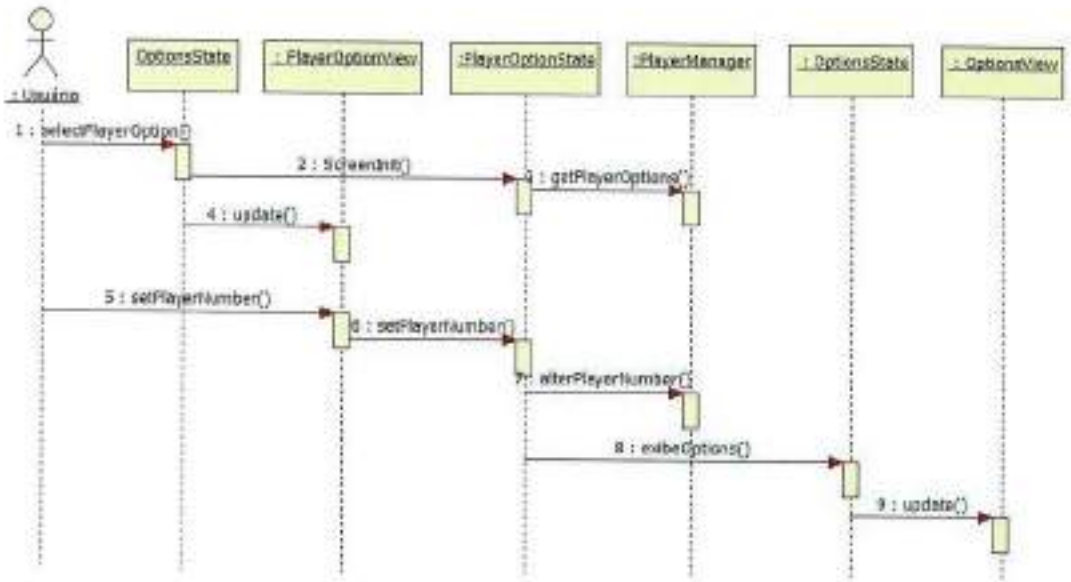


Figura 33 - Escolher Número de Jogadores - Diagrama de Seqüência

3.3.2.6 Pontuar Movimento

Este caso de uso é responsável pela pontuação durante o jogo. Ele controla a duração da exibição de cada marca na tela, e os pontos atribuídos a cada período de exibição. Também é responsável pela remoção de marcas que já estão sendo exibidas a um intervalo de tempo pré-definido como tempo máximo de jogada ("timeout"). Após execução do caso de uso "Mostrar Marcas para Acompanhar Música na Tela" este caso de uso é iniciado.

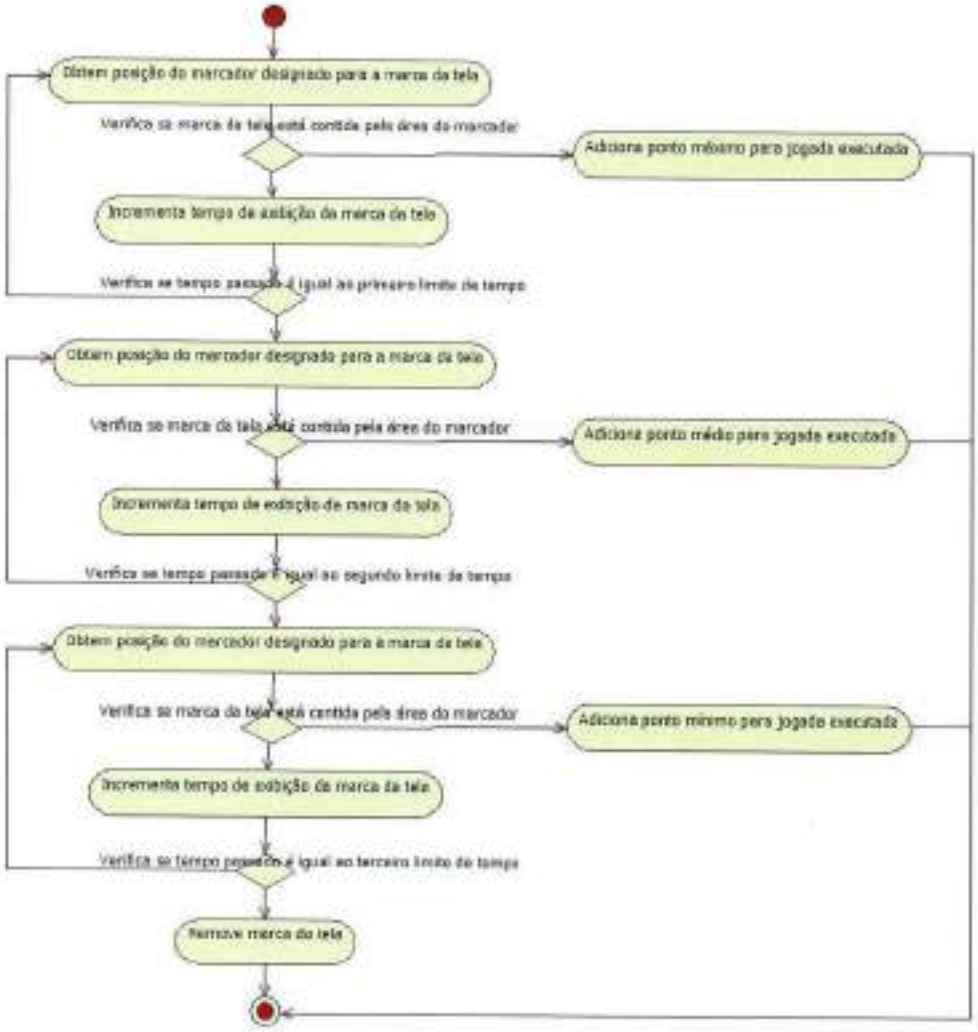


Figura 34 - Pontuar Movimento - Diagrama de Atividades

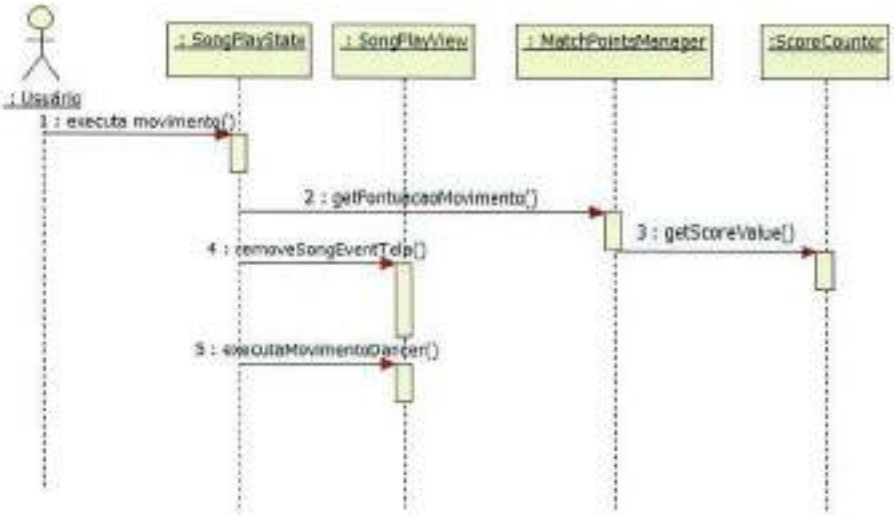


Figura 35 - Pontuar Movimento - Diagrama de Seqüência

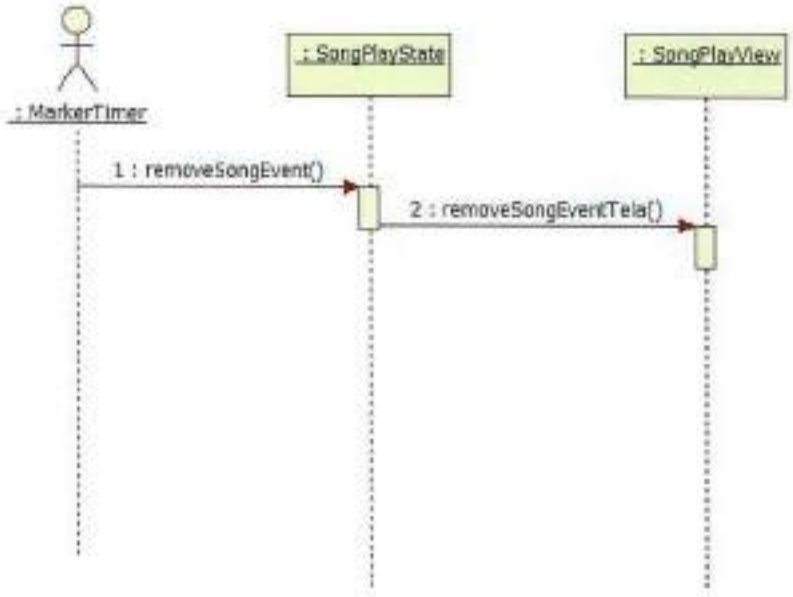


Figura 36 - Não Pontuar Movimento ("timeout") - Diagrama de Seqüência

3.3.2.7 Manter Pontos da Partida

Este caso de uso é responsável pelo armazenamento de pontos para uma partida, a fim de que o jogador seja avaliado quanto ao seu desempenho.

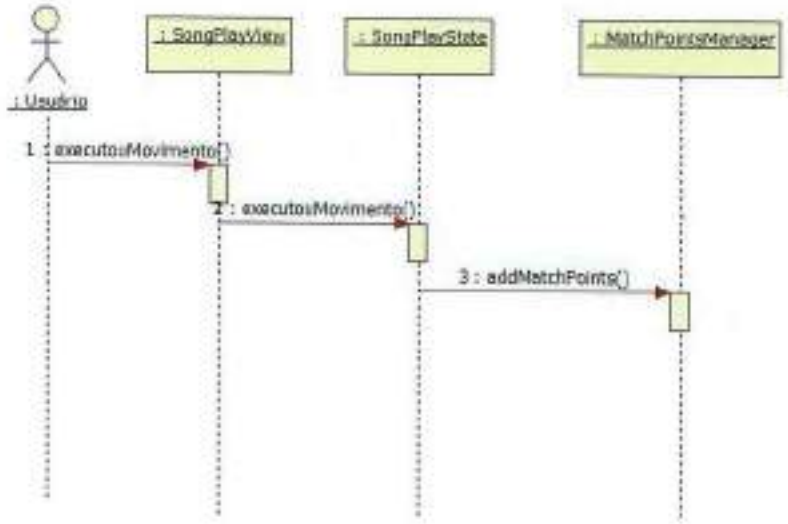


Figura 37 - Manter Pontos da Partida - Diagrama de Seqüência

3.3.2.8 Manter Ranking

Este caso de uso é responsável pelo armazenamento de informações de melhores pontuações para cada jogo, atualização destas informações caso uma destas melhores pontuações for superada, e exibição destas informações ao jogador.

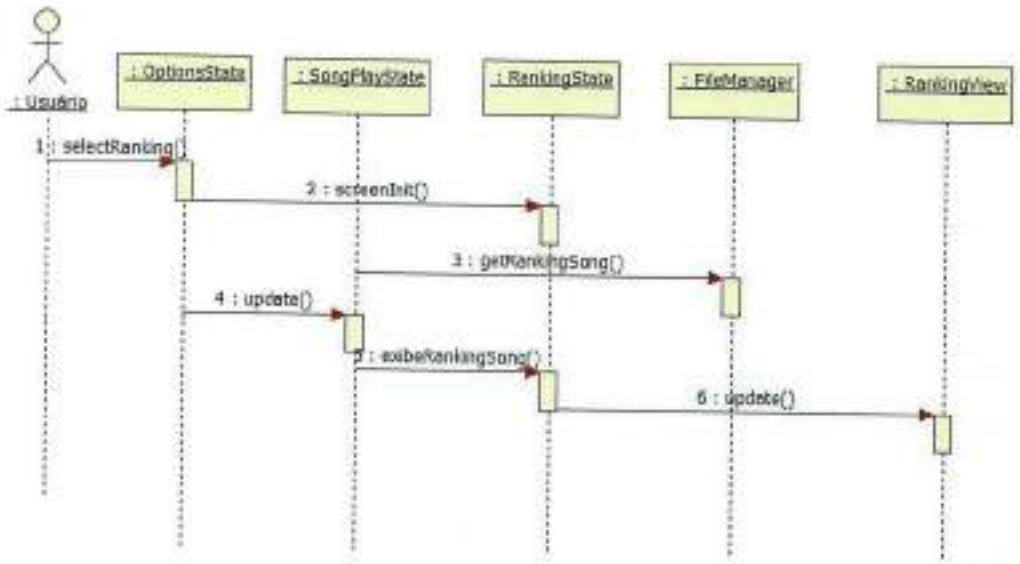


Figura 38 - Listar Ranking - Diagrama de Seqüência

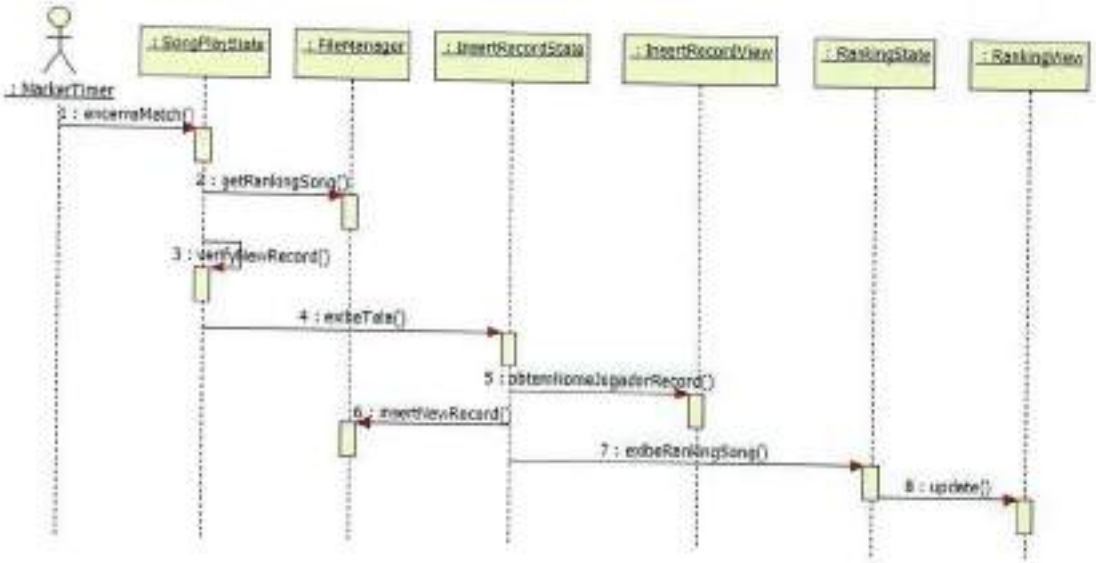


Figura 39 - Inserir Pontuação no Ranking - Diagrama de Seqüência

3.3.2.9 Acompanhar Movimento do Jogador

Este caso de uso é responsável pelo acompanhamento dos movimentos do jogador por uma espécie de "robô" na tela, para aumentar a interação entre o jogador e o jogo. Este caso de uso foi definido como o de menor prioridade por não agregar valor à solução. Assim, face ao prazo do projeto, ele foi tratado a partir da etapa de construção do UP como um trabalho futuro.

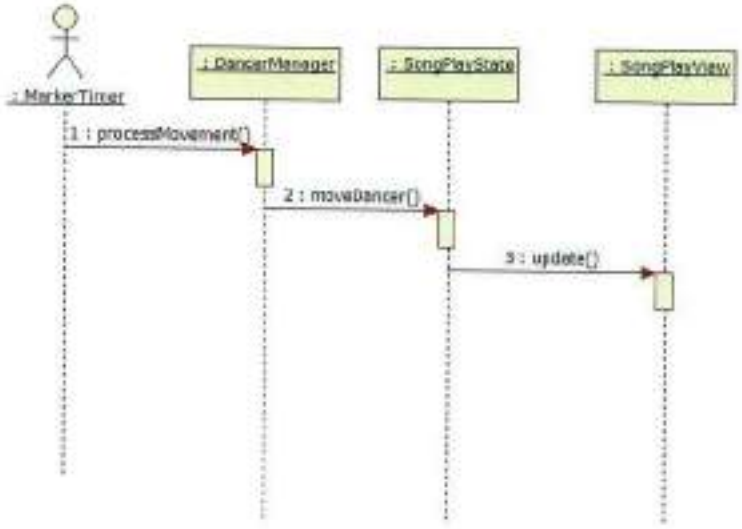


Figura 40 - Acompanhar Movimento do Jogador - Diagrama de Seqüência

4 Metodologia

Com o intuito de monitorar e controlar o projeto para que seu escopo se mantenha conforme proposto anteriormente e seja implementado no tempo determinado utilizando os recursos disponíveis, o grupo adotou algumas técnicas do PMBoK – Project Management Body of Knowlegde – (PMBOK) e que são detalhadas a seguir.

4.1 Escopo

Analisando o objetivo proposto neste projeto, o grupo identificou inúmeras atividades – Análise, Projeto, Desenvolvimento, Documentação, Alinhamento, Testes - que devem ser realizadas para que o resultado esperado seja atingido. Essas atividades foram divididas em atividades menores e mais especializadas, permitindo, assim, seu melhor detalhamento e podem ser vistas no diagrama hierárquico (WBS - Work Breakdowm Structure) abaixo.

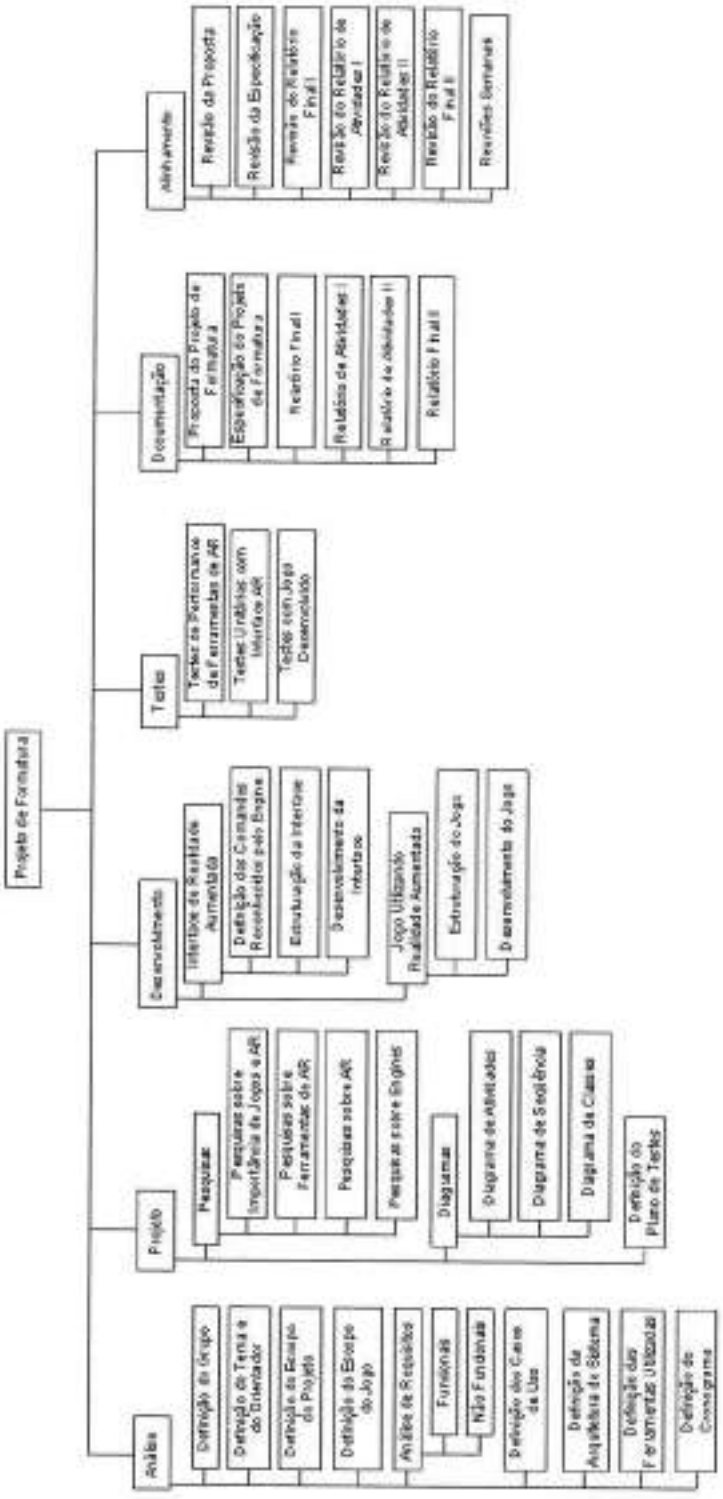


Figura 41 - WBS do Projeto

A seguir serão detalhadas cada uma das atividades e suas respectivas tarefas. Note que ainda não existe uma ordem cronológica destas tarefas até o momento e, que se trata apenas de uma separação por semelhança de conceito entre essas tarefas.

4.1.1 Análise

Nesta atividade estão envolvidas todas as tarefas necessárias para delinear o escopo do projeto já que definem: o grupo que implementará o projeto de formatura ("Definição do grupo"), a área de computação em que ele estará situado e o seu orientador dentro deste departamento ("Definição do Tema e do Orientador"), o objetivo do projeto ("Definição do Escopo do Projeto" e "Definição do Escopo do Jogo"), o que será implementado ("Análise de Requisitos", "Definição dos Casos de Uso"), como será implementado ("Definição da Arquitetura do Sistema", "Definição das Ferramentas Utilizadas") e quando será realizado ("Definição do Cronograma").

4.1.2 Projeto

As tarefas incluídas nesta atividade estão relacionadas ao que deve ser projetado para que o escopo definido seja atingido como resultado de todo o desenvolvimento. E que, portanto, envolve inúmeras pesquisas para o entendimento

dos conceitos necessários para o desenvolvimento do projeto ("Pesquisas sobre a Importância de Jogos e AR", "Pesquisas sobre AR", "Pesquisas sobre Engines", "Pesquisas sobre Ferramentas de AR"), assim como, o planejamento e a descrição de todo o escopo do jogo por meio de diagramas ("Diagrama de Atividades", "Diagrama de Seqüência", "Diagrama de Classes") e plano de testes ("Definição do Plano de Testes"), que permitem um melhor desempenho da implementação do mesmo.

4.1.3 Desenvolvimento

Nesta atividade foram descritos todos os pacotes de desenvolvimento ("Interface de Realidade Aumentada" e "Jogo Utilizando Realidade Aumentada") que foram identificados e devem ser implementados. Notando que dentro de cada pacote, encontram-se as sub-tarefas a serem realizadas e que foram determinadas após a intensa análise do escopo e das ferramentas que seriam utilizadas no projeto que serão detalhadas mais adiante.

4.1.4 Testes

As tarefas descritas nestas atividades permitem a validação das decisões tomadas e do desenvolvimento já realizado até o momento de sua respectiva

execução ("Testes de Desempenho das Ferramentas de AR", "Testes Unitários com Interface AR", "Testes com Jogo Desenvolvido") e seus resultados são posteriormente detalhados.

4.1.5 Documentação

Nesta atividade são exibidos todos os documentos que devem ser gerados ao longo do projeto de formatura ("Proposta do Projeto de Formatura", "Especificação do Projeto de Formatura", "Relatório Final I", "Relatório de Atividades I", "Relatório de Atividades II", "Relatório Final II") e que representa o registro e a análise das atividades já realizadas até o momento de sua entrega.

4.1.6 Alinhamento

O alinhamento envolve todas as tarefas (revisão dos documentos e reuniões semanais) que permitem a integração do grupo e a correção de todas as outras tarefas executadas.

4.2 Prazo

Para assegurar a finalização do projeto na data estipulada, foi necessária uma análise de todas as tarefas descritas anteriormente, estimar o tempo necessário para executá-las utilizando os recursos disponíveis, além de determinar a interdependência entre elas, ou seja, quais tarefas precisam ser finalizadas para que outra se inicie. A partir de então, foi construído um diagrama de rede, e, a partir do mesmo, o gráfico de Gantt, apresentado no apêndice 1, os quais possibilitaram a detecção do tempo mínimo necessário (caminho crítico) para executar o projeto e quais atividades devem ser monitoradas com maior atenção. E, por meio deste diagrama foi possível obter o cronograma que se encontra como apêndice 2, e que também será descrito posteriormente.

4.3 Qualidade

Note que a qualidade do projeto é mantida por meio das atividades de Alinhamento e Testes que revisam e avaliam, respectivamente, todas as tarefas executadas em função dos requisitos levantados nas atividades de Análise e Projeto. E, caso estes não sejam atendidos, tarefas são re-executadas até que a qualidade requerida seja atendida.

Além disto, o conceito de qualidade foi disseminado por toda a equipe, e cada componente do grupo foi responsável por um aspecto do desenvolvimento do

projeto. A metodologia foi assim dividida para que cada parte crítica do projeto ficasse sobre responsabilidade de um dos componentes do grupo, o qual deveria durante todo o projeto verificar e administrar os riscos existentes.

Esta metodologia de desenvolvimento pode ser exemplificada pela figura 42:

Figura 42 - Metodologia de Desenvolvimento

Também devido ao grande porte do projeto, esta divisão de tarefas e responsabilidades se mostrou fundamental. Cada qual teve sua responsabilidade no contexto do mesmo:

- Fernando Tsuda será responsável pela pesquisa, detalhamento e desenvolvimento das ferramentas de AR; Também será responsável pela definição e implementação de uma metodologia de desenvolvimento otimizada para o projeto;
- Paula Hokama será responsável pelo estudo e detalhamento das ferramentas de desenvolvimento de jogos – os engines –; Também será responsável pela definição da arquitetura do jogo a ser criado e seu desenvolvimento;

- Thiago Moreira será responsável pela criação dos artefatos de gerência do projeto e seu controle; e também pelo controle de qualidade das partes criadas e pelo produto final do projeto, asseguradas através do plano de testes.

4.4 Comunicação e Integração

Deve-se observar também que a comunicação e integração do grupo é relevante, já que reuniões periódicas são realizadas e permitem que o conceito absorvido por todos seja homogêneo e problemas e soluções sejam expostos e discutidos por todos. Isso facilita o desenvolvimento do projeto e melhora a qualidade do mesmo, já que todos os indivíduos apresentam uma mesma visão sistêmica do projeto e estão em constante avaliação pelos outros devido às entregas estabelecidas para cada reunião.

Para que esta integração fosse possível e simples durante o ano, o grupo também criou um grupo de discussão eletrônico utilizando o Yahoo, no qual estavam os integrantes do grupo, juntamente com o orientador e o co-orientador. Além disso, por falta de um repositório que pudesse ser utilizado com facilidade e com baixos custos, o grupo criou um e-mail específico para o grupo, no qual todos os componentes puderam armazenar os arquivos referentes ao projeto, e ter controle de versões. Este e-mail foi criado no serviço de e-mail gratuito do Google – gmail – e possui uma estrutura para armazenamento em "pastas" separadas dos relatórios, código-fonte, artigos, apresentações e e-mails.

5 Projeto e Implementação

5.1 Solução de Realidade Aumentada

A primeira tarefa importante desta etapa do projeto foi um estudo sobre o funcionamento de uma possível integração entre jARToolKit e enJine, fase durante a qual o grupo apenas se preocupou com esta atividade, para garantir uma solução rápida e robusta.

Inicialmente, através do aplicativo de demonstração fornecido, exibido na figura 10, e do código fonte fornecido com a ferramenta, foram identificadas as classes responsáveis pela detecção dos marcadores na cena e pela exibição do stream de vídeo proveniente da câmera.

Para o enJine, foi dado um enfoque sobre o pacote dos dispositivos de entrada e no de saída gráfica.

Para a criação do middleware de integração com o enJine, os pontos observados foram levados em consideração e as seguintes classes foram criadas:

- ARInstance: Classe que inicializa e faz o acesso às funções principais do "*JARToolKit3D*". Entre as suas principais funções, estão: o registro dos marcadores que serão utilizados nos jogos e criação do grafo de cena responsável pela captura e leitura das imagens obtidas da câmera. Somente uma instância dessa classe é permitida;

- ARInput: Classe que representa um dispositivo de entrada a partir das imagens da câmera e herdeira do "*InputDevice*". Na implementação, cada marcador utilizado é considerado um dispositivo e deve conter um "*ARPatternTransformGroup*" que será adicionado ao grafo de cena do jogo para que possa ser identificado em um imagem. Também é responsável pela conversão dos valores obtidos da matriz de transformação nos comandos que serão utilizados nos jogos;
- ARTimerTask: Herdeira da classe "*TimerTask*" do Java, é responsável pela obtenção da matriz de transformação do ramo do grafo de cena que identifica o marcador na imagem e a transferência dessa matriz para o dispositivo ARInput;
- ARDisplay:Classe herdeira do "*GameDisplay*" do enJine, foi criada para permitir a adição do plano de fundo proveniente da câmera e pela adição, através dos métodos do ARInstance, do grafo de cena que representam os dispositivos de AR.

Quanto à matriz de transformação obtida do jARToolKit, esta possui dimensão 4, onde as três primeiras colunas representam a rotação do objeto e a última coluna representa a translação. Ela também pode conter informações referentes à escala, porém essa informação não é utilizada. A última linha é utilizada para manter a simetria da matriz. A sua forma final representa a multiplicação de várias matrizes que representam transformações atômicas, como por exemplo, a translação, a rotação sobre um determinado eixo e a escala. A matriz apresentada pela equação 1 é obtida multiplicando-se, respectivamente, a matriz de

transformação, a rotação sobre o eixo x com ângulo α, a rotação sobre o eixo y com ângulo β, a rotação sobre o eixo z com ângulo γ e a escala.

Equação 1 - Matriz de Transformação

$$\begin{bmatrix} s_x \cos\gamma\cos\beta & -s_y \sin\gamma\cos\beta & s_z \sin\beta & x \\ s_x \cos\gamma\sin\beta\sin\alpha + s_x \sin\gamma\cos\alpha & s_y \cos\gamma\cos\alpha - s_y \sin\gamma\sin\beta\sin\alpha & -s_z \cos\beta\sin\alpha & y \\ s_x \sin\gamma\sin\alpha - s_x \cos\gamma\sin\beta\cos\alpha & s_y \sin\gamma\sin\beta\cos\alpha + s_y \sin\alpha\cos\gamma & s_z \cos\beta\cos\alpha & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A figura abaixo detalha o funcionamento interno do jarToolkit, diferenciando as classes conforme os pacotes nos quais elas pertencem (jarToolkit, jarToolkit for Java3D ou Java3D).

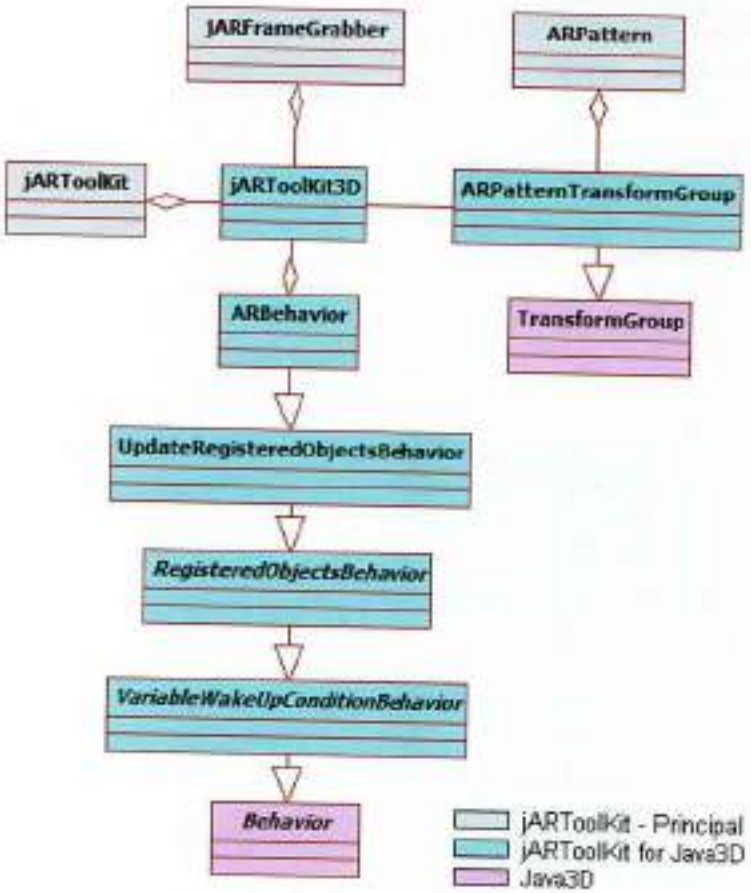


Figura 43 - Classes do jARToolKit

Abaixo é detalhada a solução de integração do jarToolkit com o enJine, para integração das funcionalidades dos mesmos:

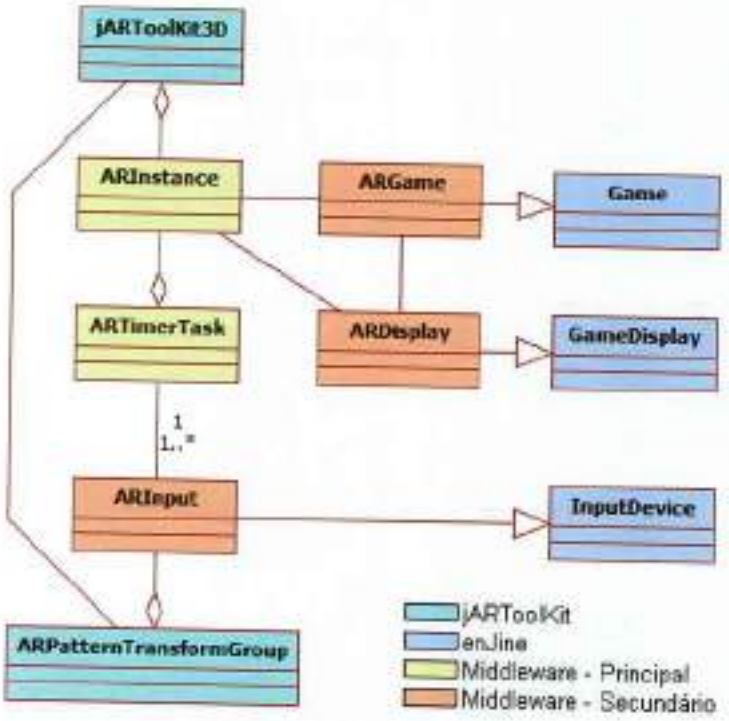


Figura 44 - Classes do Middleware desenvolvido e interação com o enJine e jARToolKit

Também, completando o projeto da solução, é apresentado o diagrama de seqüência de obtenção dos dados relativos à realidade aumentada para seu tratamento abaixo:

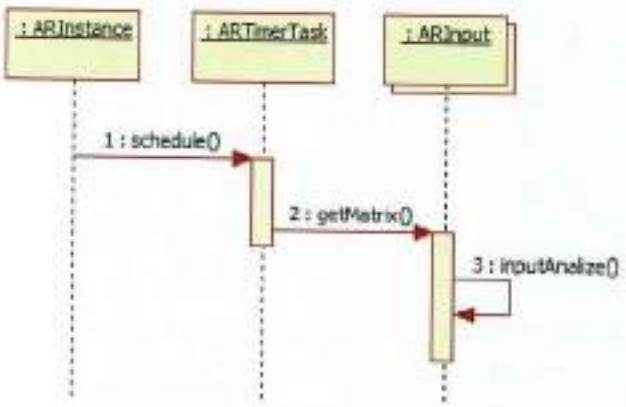


Figura 45 - Seqüência de obtenção da matriz de transformação e análise

Para facilitar a criação de um jogo que utilize a Realidade Aumentada, foi criada uma classe dentro do pacote "Framework" denominada "ARGame", herdeira da classe "Game". Ela foi criada com o intuito de incluir os atributos e métodos necessários para que a interface entre o enJine e o jARToolKit seja realizada corretamente.

Com a criação destes novos módulos no enJine, foi possível realizar a comunicação do mesmo com o jARToolKit e realizar alguns testes para avaliar o desempenho obtido. Entretanto, com o resultados destes testes, foi possível detectar que o rastreamento dos objetos não era realizado com grande precisão sob qualquer distância da câmera, ou seja, o rastreamento do marcador realizado pelo jARToolKit e a conseqüente deposição do objeto, pelo enJine, sob a posição recebida como entrada desta comunicação entre as ferramentas não foi realizada de maneira adequada. Observe na figura 46 o fluxo desta comunicação.

Figura 46 - Fluxo de comunicação entre o jARToolKir e o enJine

Os movimentos que foram pré-configurados para interação foram as translações nos três eixos (x, y e z), e também as rotações em torno do eixo z. O grupo fez o refinamento do tratamento desta matriz de transformação pelo enJine para obtenção do valor correto da rotação, que ainda contém alguns erros. Em paralelo, o grupo fez a modelagem da solução integrada com o enJine, para que na próxima versão deste engine a funcionalidade "Jogo utilizando Realidade Aumentada" esteja completamente suportada. Para tal foram definidas maneiras diferentes de obtenção do movimento do marcador pelo jogo:

- Movimento do marcador relativo à tela – o jogo desenvolvido para este projeto de formatura utiliza este tipo de movimento;
- Movimento do marcador absoluto – os casos nos quais o marcador fica em uma posição e o movimento é obtido através do movimento relativo à esta posição inicial (um exemplo seria um jogo de corrida, no qual o marcador seria o volante);
- Movimento do marcador relativo ao tempo – para os jogos os quais necessitem de mais interação do jogador, fazendo que este permaneça em constante movimento para que a ação seja continua no jogo (um exemplo seria em um jogo de aventura, no qual o jogador deve mover os braços para indicar que o personagem está andando, por exemplo).

Porém, a comunicação entre jarToolkit e enJine apresentou uma falha. Buscando detectar a mesma, o grupo analisou sob diferentes situações a principal informação que navega neste fluxo: a matriz de posição. E encontrou os seguintes aspectos antes não considerados:

- A matriz de posição fornecido pelo jARToolKit não é simétrica, isto é, a posição 0 (x = 0, y = 0) não está localizada no centro da tela;
- A visão da câmera é perspectiva (observe a figura 48), assim, a faixa de valores que representa o eixo x e y varia conforme o valor de z (distância em relação à câmera). Por exemplo, se à distância de 1 metro o valor de x na matriz de posição varia de -100 a +100 para cada extremo da tela, a 3 metros esses valores se alteram proporcionalmente;
- Ao longo de toda a extensão que o jARToolKit é capaz de reconhecer o marcador, não existe nenhuma distância em que a câmera esteja suficientemente calibrada para todo ponto no plano xy. Isto porque para pontos próximos do centro da tela, o objeto fica sobre o marcador, entretanto, à medida que o marcador se afasta do centro, o objeto vai se deslocando numa velocidade diferente do mesmo.

Com o intuito de solucionar os problemas citados acima, o grupo criou funções denominadas "calibração" que recebem como entrada os valores das posições da matriz obtida pelo jARToolKit e promovem transformações nestes valores retirando os impactos observados anteriormente.

Primeiramente o grupo optou em solucionar o problema do acompanhamento do objeto pelo marcador com a mesma velocidade numa certa distância "z" fixa e, analisando essa defasagem por meio de testes, pôde-se notar que esta variação é

linear tanto no eixo x como no eixo y. Promovendo inúmeros testes, o grupo obteve por métodos empíricos, os valores das constantes, coeficiente angular (k_1) e de deslocamento da reta (k_2) segundo a equação abaixo, necessárias para as funções de calibração.

Equação 2 - Deslocamento de reta

$$y = k_1 \cdot x + k_2$$

Entretanto, estas funções são capazes de solucionar apenas a calibração do plano xy para a distância utilizada nos testes (z=1600 mm). Assim, a solução que permite adequar estas funções de calibração para qualquer distância é projetar (calibração do eixo z) todos os pontos obtidos pela câmera para o plano xy à distância testada (z=1600 mm) antes de promover a calibração dos eixos x e y, tornando-se este plano, portanto, o plano de referência. Observe nas figuras abaixo a seqüência necessária para a obtenção da calibração de cada ponto e um exemplo deste processo, respectivamente.

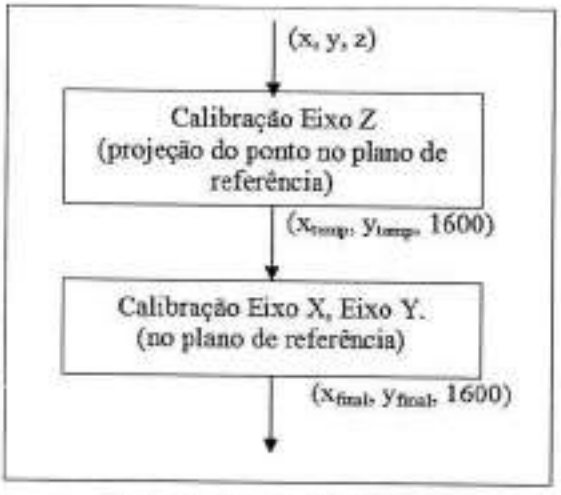


Figura 47 - Processo de Calibração

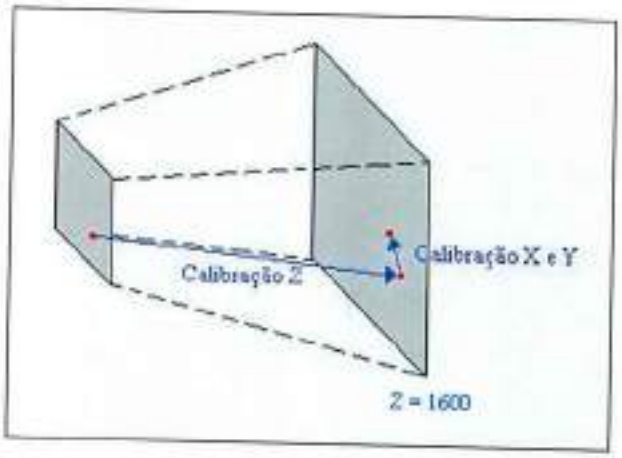


Figura 48 - Exemplo de Calibração

Esta calibração, porém, se mostrou falha, uma vez que não suporta calibração para variações de posição no eixo z. Assim, o grupo pesquisou a biblioteca jarToolkit e conseguiu adaptar a calibração nativa da mesma para o enJine. Assim toda a calibração necessária foi substituída pela calibração já existente, tornando mais precisa. O uso desta calibração em jogos de realidade aumentada pode ser verificado no apêndice 3. Esta solução leva em consideração os dois olhos do usuário, e aplica a matriz de transformação com base nestes dados (para dois referenciais distintos), fornecendo assim um resultado com menos falhas e desvios.

5.2 *Jogo utilizando Realidade Aumentada*

O desenvolvimento de um jogo mais completo visa aplicação de alguns conceitos de Game Design e deverá utilizar a solução integrada desenvolvida anteriormente e considerando-se as observações feitas durante os testes deste.

Durante o desenvolvimento, para simplificar o fluxo de dados passado pela aplicação, o grupo optou por fixar as posições onde o objeto virtual pode aparecer (vide caso de uso "Mostrar Marcas para Acompanhar Música na Tela"), permitindo o tratamento destes através de índices. Para cada mão, foram definidos 9 posições diferentes e para cada pé 3 posições, as quais podem ser verificadas na figura seguinte:

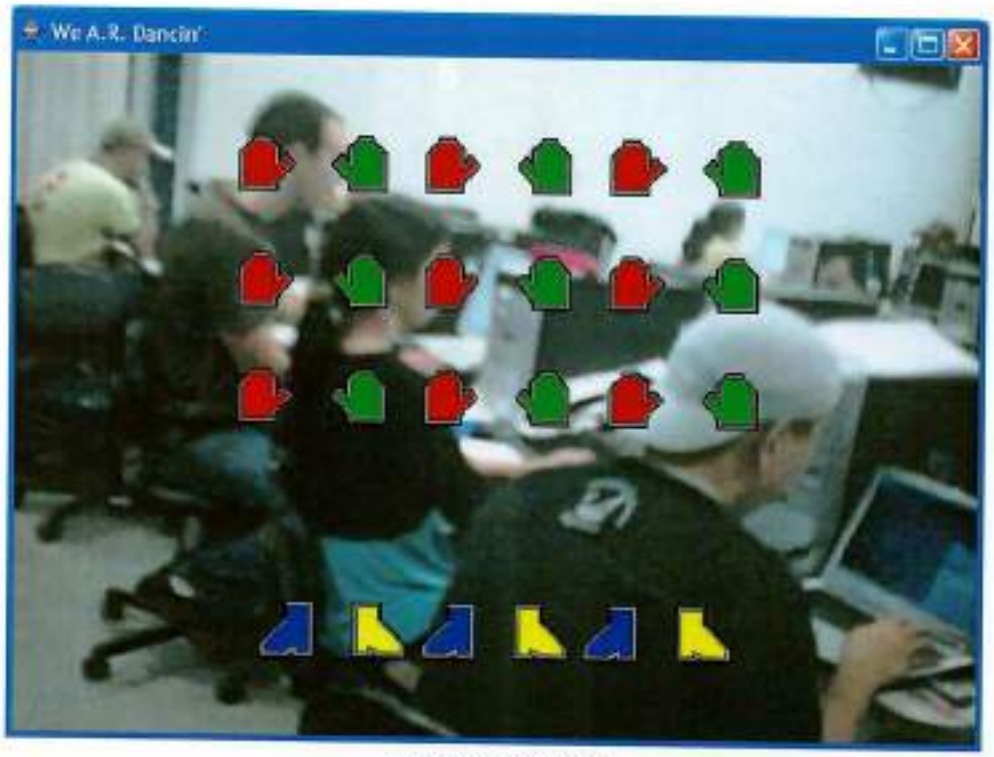


Figura 49 - Posições

Outra decisão foi o controle do jogo ser feito totalmente com o uso dos marcadores, dispensando assim o uso do teclado e permitindo com que o jogador escolhesse as opções da distância necessária entre ele e a câmera exigida pelo jogo (entre 1,5 e 2,0 metros).

Também foram definidos todos os estados existentes no jogo, ilustrada pela figura 50. Cada estado foi implementado a partir da herança da classe "*GameState*" do enJine. A partir da tela de abertura (*Opening*), a qual é a inicial, o jogador deve ir para outra tela onde é possível escolher se ele irá jogar sozinho (*1 Player*) ou contra outro jogador (*2 Players*). Também é possível o acesso à tela de Opções (*Options*), onde ele pode ver os recordes de pontuação (*Ranking*), definir a dificuldade do jogo (*Difficulty*) em fácil (*Easy*), normal (*Normal*) e difícil (*Hard*) ou voltar para a tela anterior (*Return*).

Após a seleção do modo de um ou dois jogadores, o jogador deve escolher a música (*Song Select*). Feito isso, ele entrará no jogo (*InGame*), o qual representa o jogo propriamente dito e que será detalhado adiante. Após o fim do jogo, será mostrada uma tela ao jogador contendo sua pontuação e, caso ela seja um recorde, ele deverá entrar três letras para que ele seja identificado. Após isso, o jogo volta para a tela inicial. A figura abaixo contém estes estados e suas transições:

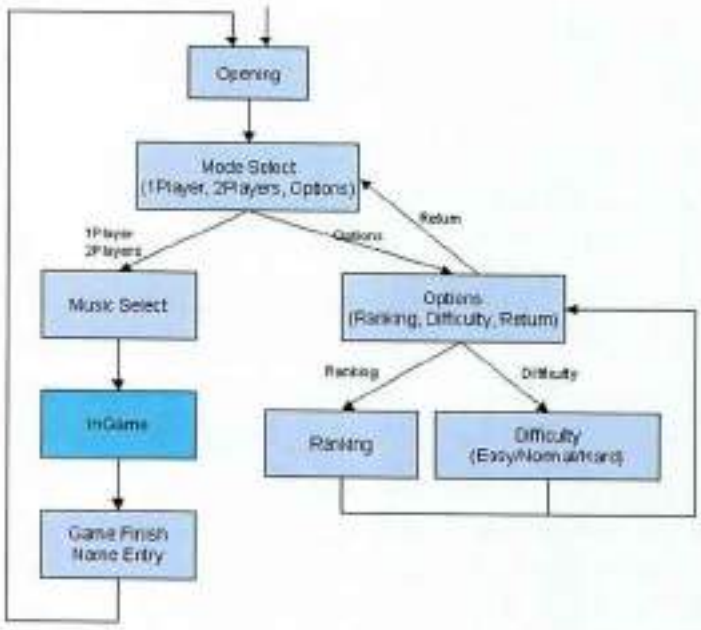


Figura 50 - Estados e Transições do Jogo

No estado *InGame*, existem outros três estados principais: o de preparação (*READY STATE*), onde um contador regressivo é exibido para permitir ao jogador se posicionar de maneira adequada. Depois que a contagem encerra, o jogo inicia, entrando no estado de jogada (*PLAY STATE*). Nesse estado, o jogador deve utilizar o marcador para "tocar" no objeto posicionado na tela, as quais representam as mãos ou os "pés" (joelhos) que o jogador deve utilizar. Além do formato representando eles, foram utilizadas cores, as quais permitem uma identificação mais rápida por parte do jogador, e o tempo restante para que o objeto seja "tocado". Após o final da música, o jogo entra no estado de conclusão (*CLEAR STATE*), onde é apresentada a pontuação ao jogador e a transição para a tela de encerramento é preparada. Deve ser explicado que o toque entre o membro e o objeto é reconhecido através da colisão deles, os quais representam uma extensão da classe "*GameObject*" do enJine.

Para o cálculo da pontuação, foram definidos três níveis possíveis: no primeiro, se o jogador "tocar" o marcador antes de um primeiro tempo limite, ele recebe totalmente os pontos. Caso esse limite seja ultrapassado, mas esteja antes do segundo limite, ele recebe uma pontuação média. Se o segundo limite for ultrapassado, a pontuação é a menor. Se o tempo de exibição do marcador se esgotar, os pontos não são computados. Os limites são definidos de acordo com a dificuldade do jogo. As coreografias a serem realizadas pelo jogador são definidas dentro de um arquivo texto com extensão ".arm" (A.R. Música), o qual possui a seguinte regra gramatical (representada na figura abaixo, usando a notação de Wirth (JOSÉ NETO, J., 1988)):

```
nome_música
pontuação_máxima
tempo indice_membro indice_posicao prioridade
{tempo indice_membro indice_posicao prioridade}

Exemplo:
Festa no Apê
5000
0 1 5 1
0 2 8 1
5 3 2 2
5 4 0 1
...
```

Figura 51 - Exemplo de Arquivo .arm

A leitura desses arquivos é feita de forma dinâmica pelo programa, ou seja, basta colocar o arquivo juntamente com a música na pasta reservada para elas que, ao entrar na tela de escolha da música, elas são identificadas e inseridas no jogo.

A navegação entre as telas de opções foi implementada com sucesso. Para evitar que o posicionamento do marcador sobre o ícone da tela resultasse em uma transição indesejada, foi implementado um contador que indica em quanto tempo a

transição irá ocorrer. Nessas telas também se optou por esconder o stream de vídeo que, se por um lado dificulta a movimentação do marcador nas telas de opções, por outro provoca um maior impacto no jogador quando estado de jogada é iniciado.

No jogo, notou-se uma pequena lentidão na detecção dos marcadores, fazendo com que movimentos muito rápidos não sejam detectados corretamente. Para contornar isso, foi definido um reconhecimento com menor precisão entre a colisão do membro com o objeto virtual.

6 Cronograma

Conforme citado durante a descrição da metodologia de gerenciamento adotada pelo grupo, a partir das atividades descritas na WBS, foi realizada uma profunda análise de suas interdependências e estimado o tempo de execução de cada uma utilizando os recursos disponíveis. Desta forma, foi possível obter um cronograma do projeto contendo todas estas atividades e as suas datas de início e término detalhados no apêndice 2.

Além disto, buscando facilitar a compreensão do andamento do projeto, o grupo criou uma versão simplificada do mesmo, figura 52, onde o projeto está dividido não pelas atividades definidas, mas pelas fases explicitadas na metodologia UP e que foram detalhadas anteriormente.

Vale lembrar, que segundo o UP, a qualidade do projeto é obtida por meio de inúmeras interações, onde em cada uma delas são executadas as fases de desenvolvimento e transição. E da mesma forma, como pode ser visto na figura 52, visando à garantia de integração dos módulos do projeto, houve uma interação durante o primeiro semestre, na qual a arquitetura do módulo de interface do jARToolKit com o enJine foi escolhida (fase de elaboração e concepção), desenvolvida e validada (fase de construção e transição) e, uma outra interação no decorrer do segundo semestre, que foi responsável pela escolha do jogo e sua arquitetura (fase de elaboração e concepção) e o desenvolvimento e validação do mesmo (fase de construção e transição).

Figura 52 - Cronograma Simplificado para o Desenvolvimento do Projeto

Note ainda que este cronograma não foi estático desde o início do projeto, porém serviu para que, caso houvesse algum atraso, este fosse identificado previamente e tivesse seu risco mitigado, administrando, assim, a duração das próximas atividades. Isto foi importante principalmente na fase de concepção, na qual o grupo enfrentou diferentes problemas para a definição da arquitetura e da solução a ser utilizada.

7 Recursos e Infra-estrutura Requeridos

Para execução deste projeto, as ferramentas utilizadas e também elaboradas são exibidas na tabela 3. Porém, deve-se mencionar que a maioria dos recursos necessários já estava disponível para o uso antes do início do mesmo, assim o valor encontrado apenas ilustra o caso onde é necessária a compra dos equipamentos.

Tabela 3 - Recursos a serem utilizados no projeto, com os respectivos valores de aquisição

Recurso	Utilização	Valor	Observação
Câmera de vídeo - Modelo CMS-V11	Câmera utilizada para a captura dos movimentos	R$ 123,00	Modelo a venda no Japão por 6.279 ienes. O valor em reais foi obtido utilizando-se a cotação do dia 20/06/2006
Microcomputador PC	Computador no qual serão realizadas as atividades referentes ao projeto	R$ 2.300,00	Considerando o modelo Dell Dimension 5150n com um monitor de 17"
Sistema operacional Microsoft Windows XP	Sistema operacional onde o jogo será desenvolvido e executado	R$ 689,00	Versão Profissional com SP2, em português e completo. Preço obtido na loja Brasoftware
Microsoft Office 2003	Aplicativos para a elaboração dos relatórios e apresentações	R$ 1.099,00	Versão Standard, em português e completo. Preço obtido na loja Brasoftware
Microsoft Project 2003	Software utilizado para o gerenciamento do projeto	R$ 1.965,00	Versão em português e completo. Preço obtido na loja Brasoftware
Ambiente de desenvolvimento e execução Java (J2SE)	Inclui as bibliotecas necessárias para o desenvolvimento e execução do jogo e o funcionamento das demais bibliotecas adicionais	-	-
Biblicteca Java3D	Biblioteca necessária para o tratamento e exibição na tela dos gráficos tridimensionais	-	-
API DirectX	API necessária para o funcionamento do Java3D		Este recurso é obtido juntamente com o Microsoft Windows XP
Biblicteca JARToolkit	Biblioteca necessária para a criação das ferramentas de Realidade Aumentada a serem utilizadas no jogo	-	-

Engine de jogos enJine	Biblioteca com as funções a serem utilizadas no desenvolvimento do jogo	-	-

Vale ressaltar que este tipo de captura de movimentos que esta sendo utilizado corresponde a sensores passivos, portanto os sensores são os mais simples dentre os existentes – marcadores feitos de papel, por exemplo –, o que torna o uso desta tecnologia muito prático, analisando o custo unitário para cada jogador.

A configuração da câmera utilizada para testes é a seguinte:

Tabela 4 - Propriedades da câmera a ser utilizada

Propriedade	**Valor**
Modelo	CMS-V11
Lente	F = 3.0mm
Foco	Operação manual
Contração, Brilho e Contraste	Ajuste automático
Balanço da cor branca	Ajusta automático
Taxa de quadros	640x480 (VGA) - 15 quadros por segundo 352x288 (CIF) - 30 quadros por segundo
Intensidade de luz mínima	30 Lux
Interface de comunicação	USB 1.1

Analisando o custo para o jogador, podemos chegar aos seguintes dados:

Tabela 5 - Custos para Jogador

Recurso	**Utilização**	**Valor**	**Observação**
Câmera de vídeo	Câmera utilizada para a captura dos movimentos	R$ 123,00	-
Máquina Virtual Java (J2SE)	Inclui as bibliotecas necessárias para execução do jogo e o funcionamento das demais bibliotecas	-	-
Biblioteca Java3D	Biblioteca necessária para o tratamento e exibição na tela dos gráficos tridimensionais	-	-
API DirectX	API necessária para o funcionamento do Java3D	-	Este recurso é obtido juntamente com o Microsoft Windows XP
Jogo em Realidade Aumentada	Jogo desenvolvido	-	-

Analisando esta tabela, podemos concluir que esta solução de jogos usando realidade aumentada é muito barata se comparada com, como foi mencionado no inicio deste relatório, as soluções da Nintendo – Wii, cujo preço é 250 dólares apenas o console – e da Sony – PlayStation 3, cujo preço é 500 dólares apenas o console, sem jogos. Caso o jogador já possua um computador que tenha uma configuração mínima para execução destes jogos, esta solução é muito mais vantajosa, já que o sistema operacional, em mais de 80% dos usuários domésticos do Brasil, é o Windows XP (IDG, 2005), ou seja, este custo também pode ser descartado (sobrando assim somente o custo da câmera). Além disso, todos os componentes necessários para o jogador (computador, webcam, sistema operacional) são relativamente comuns e tem diversos usos, não só em diversos tipos de jogo como em outras atividades, o que constitui um ponto a favor dessa solução em comparação com jogos que utilizam interfaces especificas, úteis somente para um jogo ou uma classe bastante restrita de jogos, como exemplos que foram citados anteriormente nesse texto (como por exemplo, as guitarras do jogo "Guitar Hero" (GUITARHERO)).

8 Testes e Avaliação

Conforme detalhado anteriormente, o projeto constitui-se de duas fases: implementação da interface enJine e jARToolKit e o desenvolvimento do jogo e, desta maneira, os testes também foram definidos e divididos da mesma forma.

Além disto, logo no início do projeto o grupo definiu que o plano de testes estaria em função dos requisitos estabelecidos no início do projeto e, apesar de já detalhados anteriormente, segue uma breve citação e resumo dos mesmos:

- O enJine deve suportar o desenvolvimento de jogos utilizando Realidade Aumentada (detecção dos marcadores e calibração automática de suas posições, assim como sua associação à reprodução dos objetos 3D no ambiente real situado ao fundo da tela) e, além disto, deve reproduzir músicas no formato MP3;
- Garantia da usabilidade da solução adotada e do jogo;
- Garantia da confiabilidade, desempenho e suportabilidade da solução adotada.

8.1 1ª Fase

Assim, no primeiro período, conforme já citado no item relacionado ao cronograma, o grupo realizou intensos estudos sobre as alternativas das

ferramentas que auxiliam na implementação da realidade aumentada e, após a escolha da biblioteca jARToolKit, testes para validação técnica foram definidos, no qual o tempo de resposta fornecido pela ferramenta, assim como a mudança dos seus valores de saída em função das variáveis de ambiente, seriam analisados e aferidos em relação aos requisitos não funcionais citados anteriormente (confiabilidade e desempenho).

Além disto, durante esta fase, foi implementada a interface entre o jARToolKit e o enJine, já detalhada, para permitir que este fosse capaz de suportar o desenvolvimento de jogos utilizando Realidade Aumentada e, para validá-la, protótipos foram criados e serviram de prova de conceito dos pacotes já implementados. A seguir é descrito um detalhamento desses testes.

8.1.1 Testes com jARToolKit

Como já dito, com o jARToolKit, foram feitos alguns testes para validar a sua utilização no projeto e conseqüentemente permitir a avaliação de cada estilo do jogo de acordo com as suas limitações e capacidades. Para tal, foi montado um aparato, como mostrado na figura 53, onde o marcador foi preso em uma linha puxada por um motor, fazendo com que a câmera tentasse capturar o padrão contido no marcador em movimento e o programa gerasse um objeto virtual sobre ele. As configurações de câmera utilizadas estão descritas na tabela 6. O teste foi realizado para verificar qual a velocidade máxima com que o marcador pode se mover diante da câmera.

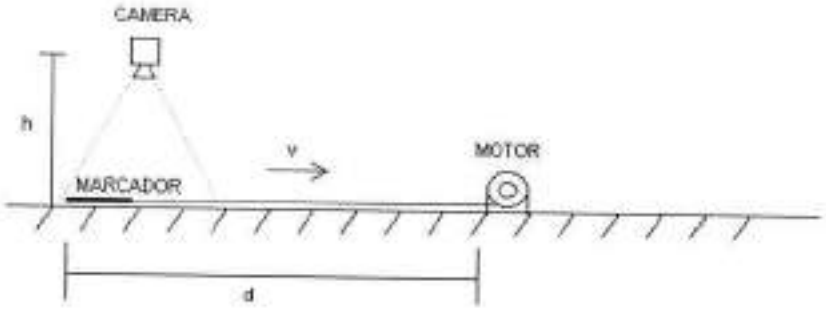


Figura 53 - Aparato utilizado para testar a velocidade do marcador em relação à sua distância da câmera

Tabela 6 - Configurações de Câmera Utilizadas

Resolução	Taxa de Captura
320 x 240	30 quadros (frames por segundo)
640 x 480	15 quadros (frames por segundo)

A configuração do computador utilizado nesse teste foi um Pentium M de 1,73 GHz, com 512 MB de memória RAM e placa de vídeo da Intel on-board com memória compartilhada. A versão do Java utilizado foi a 5.0 Update 5 e a do Java3D foi a 1.3.1.

Porém, os resultados foram muito desapontadores, pois com a câmera a 1 metro de altura, a maior velocidade obtida com resultados satisfatórios foi de 0,1 m/s, ou seja, uma movimentação muito lenta. Além disso, notou-se que a minima variação da iluminação também influiu nos resultados nos testes, como pode ser comprovado pela presença ou não de pessoas próximas do marcador, mesmo que a olho nu a iluminação sobre este não fosse alterada. Como a configuração do computador utilizada é bastante recente, chegou-se a uma conclusão inicial que a causa do problema fosse o jARToolKit, o qual por sua vez foi implementado sobre

uma versão bastante antiga do ARToolKit. Para melhor ilustrar a comparação, a data da última versão do jARToolKit, e conseqüentemente do ARToolKit utilizado, é indicada como sendo de junho de 2004, enquanto que a última versão do ARToolKit é de junho de 2006. A partir desses testes, foram discutidos mais seriamente alguns planos de contingência, entre os quais podemos citar a criação de uma interface própria entre o ARToolKit mais recente com o Java. Porém, dois fatores mostraram que o jARToolKit pode ser utilizado no projeto:

- A utilização de computadores com placas de vídeo dedicadas ao invés dos com placas de vídeo on-board. A execução do programa nos computadores instalados no laboratório da Microsoft no CCE mostrou a possibilidade da movimentação do marcador com uma maior velocidade. A configuração desse computador é a seguinte: Pentium 4 de 3,40 GHz com HyperThreading, 1 GB de memória RAM e processador gráfico ATI FireGL V3100 com 128 MB de memória dedicada;
- A utilização de versões mais recentes das ferramentas e bibliotecas como, por exemplo, a versão 5.0 Update 8 do Java e a versão 1.4.0 do Java3D. Isso pôde ser comprovado pela instalação dessas versões no computador onde o teste descrito anteriormente foi realizado, fazendo com que fosse possível a movimentação do marcador com uma maior velocidade. Esse fator contribuiu de forma decisiva para a utilização do jARToolKit no projeto.

8.1.2 Testes com a Interface entre jARToolKit e enJine

Os testes de integração foram realizado de duas maneiras:

- Teste de entrada de comandos através dos marcadores usando jogos já desenvolvidos com o enJine;
- Criação do primeiro jogo (protótipo) que levasse em conta a nova forma de jogabilidade.

No primeiro teste foi utilizado o jogo "City Run", jogo de corrida desenvolvido por alunos do curso de computação gráfica e disponível no site do enJine. Nele, a jogabilidade foi alterada para aceitar comandos provenientes da leitura de um marcador no *stream* de vídeo. Dessa forma, o carro pode ser manipulado através da movimentação de um marcador para cima, baixo, esquerda e direita, desconsiderando-se, portanto, a movimentação em profundidade e as rotações.

Um dos resultados interessantes dos testes com esse protótipo foi a presença de muitos elementos gráficos (neste caso todo o cenário) na tela pareceu atrapalhar o controle por parte do usuário, por encobrir o vídeo e conseqüentemente a percepção da posição do marcador na tela.

Além disso, foi observada a necessidade da inversão (horizontal) da imagem proveniente da câmera, para que o usuário atue como se estivesse diante de um espelho, melhorando assim a usabilidade.

Em relação ao movimento do marcador, quando este se movia para cima, o enJine detectava este movimento, mas sua amplitude não era armazenada, ou seja, o carro movia-se para frente sempre de uma distância fixa. Identificou-se neste instante, a necessidade de interpretar e associar o movimento do marcador de

diversas maneiras: posição absoluta do marcador, posição relativa ao marcador, posição relativa ao deslocamento do marcador (em relação à posição anterior). Observe na figura seguinte uma tela deste teste realizado.

Figura 54 - Teste utilizando adaptações para um jogo já criado no enJine

Enfim, com este primeiro teste, foi possível identificar os seguintes fatos:

- Validação da detecção de movimento do marcador e associá-los, de fato, à determinada ação no jogo;
- Validação da inserção do *stream* de vídeo na tela do jogo juntamente com o restante do cenário;
- Identificação da necessidade de inversão da imagem proveniente da câmera;
- Identificação da necessidade da criação de diferentes interpretações do movimento do marcador (absoluto, relativo a ele, relativo ao seu deslocamento);
- Identificação da superposição do cenário virtual sobre o *stream* de vídeo e possível incômodo do usuário em relação à falta de visibilidade do movimento do marcador realizado por ele.

Já para o segundo protótipo, criou-se um jogo similar ao Pong, escolhido devido à necessidade de poucos elementos na tela e a facilidade da implementação dos mesmos. O seu funcionamento consiste em movimentar o marcador para cima e para baixo, fazendo com que os cubos, representando as plataformas que no jogo original rebatem a bola, movam-se na direção correspondente e com o objetivo de acertar a bola para devolvê-la ao adversário.

Nos testes com esse protótipo, os objetivos foram verificar o uso de mais de um marcador simultaneamente e o aprendizado na criação de jogos no enJine. Apesar das regras do jogo não terem sido completamente implementadas – a contagem dos pontos, por exemplo –, o objetivo de movimentar os objetos com marcadores diferentes foi atingido. Aqui também se testou a rotação através dos valores obtidos da matriz, porém elas não possuem nenhum efeito prático nas regras do jogo implementado até então.

Vale a pena citar que neste protótipo já fora implementado a associação do objeto em relação à posição absoluta do marcador que, neste caso se deu apenas no eixo y (note na figura 55), já que as plataformas devem estar fixas em suas posições x.

Ainda nesse segundo teste, notou-se a dificuldade de se trabalhar com a movimentação do jogador em profundidade em relação à tela, pois uma pequena variação na distância já influencia, por exemplo, na colisão dos objetos. Porém, esse é um problema que pode ser resolvido através do Game Design.

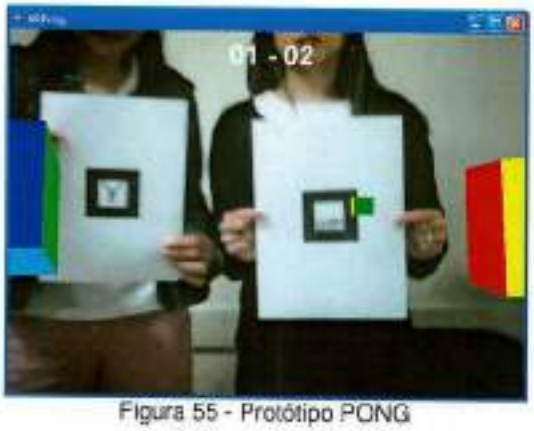


Figura 55 - Protótipo PONG

Desta forma, neste segundo teste, foi possível identificar os seguintes fatos:

- Validação das diferentes interpretações implementadas para o movimento do marcador (neste caso, o objeto acompanha o valor da coordenada y da posição do marcador);
- Validação da utilização de mais de um marcador na tela;
- Validação da associação da rotação do marcador ao objeto;
- Identificação da dificuldade de tratar o movimento do marcador em relação à coordenada z para a realização de colisões durante o desenvolvimento do jogo.

Além disto, o grupo notou a importância do teste de profundidade da câmera que não fora realizado até o momento, que consiste em testar o comportamento do objeto 3D em função da variação do marcador sobre o eixo z (profundidade). Para isto, o grupo criou um teste semelhante à demonstração do jARToolKit (figura 10) onde um "color cube" deve acompanhar o marcador e o enJine deve ser capaz de, a partir da matriz de posição fornecida pelo jARToolKit, interpretar o movimento do marcador e o conseqüente deslocamento do cubo.

Com este teste, foi possível identificar que a calibração anteriormente realizada, em que os valores de translação da matriz de posição eram adaptados ao plano de referência, era funcional apenas para jogos 2D, ou seja, apesar dos objetos serem tridimensionais, as ações dos jogos só ocorrem numa profundidade fixa (z=0). Desta forma, o grupo buscou novas soluções que permitissem a criação de jogos 3D para que atendesse o requisito principal do projeto: utilização realidade aumentada no jogo.

A solução adotada, conforme já detalhada em "projeto e implementação" foi a criação de uma classe estendida da entidade câmera da biblioteca jARToolkit utilizando as mesmas configurações, como a sua posição, orientação. Deste modo o teste do "color cube" foi executado novamente e validado com sucesso. Contudo, até o momento, é possível apenas criar uma instância de câmera por jogo, limitando, portanto, a funcionalidade do enJine de criar várias instâncias da mesma e posicioná-las em diversos pontos. Essa alternativa adotada, apesar de limitante, atinge o escopo do projeto e seu aprimoramento está visado em trabalhos futuros.

8.2 2ª Fase

Durante a segunda fase, o grupo implementou o jogo escolhido (we A.R. Dancin'), detalhado no item "Projeto e Implementação", por meio da divisão do mesmo em módulos, que foram definidos através da semelhança das funções executadas em cada estado (tela) do jogo. Estes foram individualmente validados e,

por último, a integração. Além disto o grupo visou avaliar a usabilidade do jogo, como poderá ser visto no tópico adiante.

8.2.1 Testes com Módulos do Jogo

Em cada módulo, o grupo buscou implementar primeiramente suas principais funcionalidades, e por último, inserir os objetos 3D do jogo. Desta forma, os testes foram realizados, de início, com "*Color Cubes*" que não necessitam de iluminação, facilitando assim sua implementação.

Validadas as funcionalidades de cada estado do jogo (baseado no diagrama de atividades e casos de uso), o grupo utilizou ainda figuras 2D, figura 56, para testar o seu desempenho baseado no tempo de resposta do jogo e que foi comparado com os resultados obtidos da integração do jogo e utilizando os objetos 3D finais detalhados adiante.

Figura 56 - Imagens 2D Utilizadas

8.2.2 Testes da Integração do Jogo

Validados todos os módulos implementados, o grupo inseriu os objetos 3D (mão e luva) e integrou todos os estados do jogo. Para validação do mesmo, cada integrante do grupo testou o jogo desde seu início, e aferiu com sucesso os resultados seguindo novamente o diagrama de navegação.

Figura 57 - Jogador e Marcadores 3D

Apesar de o grupo esperar um tempo de resposta maior em relação aos estados com figuras 2D, esta diferença não pôde ser detectada pelo jogador, isso porque as figuras 3D utilizadas são modelos simples constituídos de poucos polígonos, conforme visto na figura 57.

8.2.3 Testes de Usabilidade

Para testes de avaliação de usabilidade do projeto, o grupo sentiu a necessidade de outras pessoas testarem o jogo, para verificar o modo como estes usuários reagiriam numa primeira instância. Estes testes realizados com diferentes pessoas encontram-se em anexo no CD no formato de vídeo ".mpeg".

9 Considerações Finais

Através dos protótipos implementados – prova de conceito (PONG) e o jogo (weA.R.dancin') – pôde-se verificar que a metodologia definida e largamente implantada foi fator de sucesso e permitiu que a solução adotada ao enJine fosse adequada para o desenvolvimento de jogos utilizando Realidade Aumentada.

Isto porque, conforme detalhado nas atividades programadas, ao longo de todo o primeiro semestre, o grupo buscou estudar e analisar intensamente todas as alternativas existentes para a implementação do projeto e encontrar a melhor alternativa para adaptar o escopo do mesmo aos limites impostos ao grupo: custo, prazo e recursos.

Restando, portanto, para o segundo período, a implementação da solução escolhida assim como a realização de testes para validá-la durante a implementação que, desta forma, possibilitou o surgimento de interações durante o desenvolvimento que garantiam o controle e monitoramento do tempo disponível e das tarefas ainda a serem implementadas. Conforme já dito, neste período o grupo também realizou a escolha do jogo e o seu desenvolvimento que concretizou de fato o sucesso da solução adotada.

Uma atividade que não estava programada, porém foi muito importante para o projeto, foi a criação de um artigo sobre o assunto, o qual foi publicado no II Workshop de Aplicações de Realidade Virtual, que aconteceu em Recife, na Universidade Federal de Pernambuco, no mês de novembro de 2006. O artigo publicado pelo grupo foi "Integração do JarToolkit com Engine Didático para a Criação de Jogos Usando Realidade Aumentada". O artigo foi escrito em conjunto

com os orientadores deste projeto, e foi apresentado pelo Professor Doutor Romero Tori. O mesmo encontra-se no apêndice 4.

Por fim, o desenvolvimento de novos jogos em realidade aumentada utilizando essas ferramentas e explorando novos projetos e formas de interação também é, obviamente, uma importante possibilidade de trabalhos futuros, tanto para a área de jogos e entretenimento quanto para outras áreas, como mencionado anteriormente.

10 Trabalhos Futuros

Durante a implementação do projeto, também foi possível identificar outras funcionalidades, não contidas até o momento, que melhorariam a performance ou, pelo menos, forneceriam novas maneiras para o desenvolvimento dos jogos utilizando o enJine, mas que não foram desenvolvidas devido ao fim do prazo estipulado e à prioridade dada, pelo grupo, ao atendimento dos requisitos estabelecidos no início do projeto, tornando-as, assim, metas para trabalhos futuros.

Dentre as funcionalidades identificadas, algumas estão relacionadas às novas funções internas do enJine, ou seja, criação de uma nova versão da ferramenta que possibilite, ao desenvolvedor de jogos, a escolha da remoção de fundo da imagem da câmera para integrar a imagem do jogador, obtida em tempo real, ao jogo como o próprio avatar do jogador ou personagem principal do jogo. Ou seja, ao contrário do exemplo dado no *weA.R.dancin'*, no qual os objetos 3D virtuais são incluídos no ambiente real do jogador, este seria imerso num cenário do jogo, dando-lhe a sensação de estar em outro local, através de algoritmos de remoção do fundo da stream de vídeo.

Outra possibilidade ainda relacionada ao enJine é a possibilidade de renderizar os elementos reais com profundidade (atualmente a *stream* de vídeo é uma imagem de fundo, sempre oclusa pelos objetos virtuais), de forma que possam fazer oclusão de objetos virtuais. Isso poderia inclusive reduzir o problema de muitos elementos na cena do jogo atrapalharem a interação devido à oclusão de elementos reais.

Vale a pena lembrar, que na solução final adotada, o grupo limitou a possibilidade de criar várias instâncias de câmeras no jogo pelo enJine, devido à alternativa incorporada para a calibração da câmera. Desta forma, como trabalho futuro, seria ainda interessante o aprimoramento da calibração implementada e que fora descartada anteriormente.

Em relação à ferramenta jARToolKit utilizada, após o intenso estudo de sua arquitetura e a análise feita para identificar a solução mais plausível para permitir a interoperabilidade desta com o enJine através das novas classes criadas, foi possível identificar fontes de melhorias na própria arquitetura do jARToolKit, como, por exemplo, a interface desta com o java3D (implementada na classe "jartoolkit for java3D") que poderia ser removida e implementada no enJine para otimizar seu desempenho.

Já em relação ao jogo *weA.R.dancin'* criado, no qual foi possível de fato implementar o nível de dificuldade do jogo, seleção de músicas, criação e consulta do ranking das melhores pontuação, assim como a navegação entre essas telas, o grupo identificou outras funcionalidades que, posteriormente, poderiam ser incluídas no mesmo, principalmente, em relação à introdução de novas músicas ao jogo.

Como já dito, o jogo utiliza um arquivo com extensão ".arm", que armazena todos os eventos (mãos e pés) que aparecerão no decorrer da música e que devem ser "apanhados" durante do jogo para promover a pontuação. Caso o jogador queira adicionar uma nova música ao jogo, é necessário que ele saiba o instante da música em que um evento ocorre, o número identificador deste evento, o número identificador da posição da tela e ainda editar estes dados como um arquivo texto. Desta maneira, nota-se que a criação de uma nova música e seus eventos (".arm") não é intuitiva e não existe nenhuma ferramenta que facilite a edição deste arquivo.

Portanto, a criação de uma funcionalidade no jogo que permitisse o usuário carregar uma nova música e, enquanto esta é executada, ele possa, por meio da câmera, exibir o marcador que deseja utilizar e a posição na tela onde o evento deverá aparecer naquele instante da música e, automaticamente o arquivo ".arm" seja gerado, melhoraria sensivelmente esta interface.

Além disto, agora com a possibilidade de execução de arquivos de música do formato ".mp3" no enJine, seria ainda adequado o estudo deste padrão e a forma como a música é nele armazenada, para tentar, assim, identificar a possibilidade de carregar esses mesmos eventos configurados no arquivo ".arm" num campo de informação do ".mp3". Buscando, dessa forma, a remoção deste arquivo de configuração da música no jogo (.arm). Vale a pena notar, entretanto, que estas funcionalidades apesar de muito úteis, não são fundamentais à existência do jogo e, por este motivo, não foram priorizadas durante o desenvolvimento.

REFERÊNCIAS

ABRAGAMES. Plano Diretor da Promoção da Indústria de Desenvolvimento de Jogos Eletrônicos no Brasil. Disponível em <http://www.abragames.com.br>;

AMIRE. ARToolKit for AMIRE. Disponível em <http://mtd.fh-hagenberg.at/depot/graphics/artoolkit/index.html>;

APLUS. ARToolkit Plus. Disponível em <http://studierstube.org/handheld_ar/artoolkitplus.php>;

ARSTUDIO. AR Studio. Disponível em <http://www.cv.iit.nrc.ca/research/ar/arstudio.html>;

ARTAG. ARTag. Disponível em <http://www.cv.iit.nrc.ca/research/ar/artag/>;

ARTOOLKIT. ARToolkit 2.33 manual. Disponível em: <http://www.lamce.ufrj.br/grva/realidade_aumentada>;

AZUMA, R. T. A Survey of Augmented Reality. Teleoperators and Virtual Environments, vol. 6, p. 355-385, ago. 1997;

BASTOS, N., Arquitetura para Dispositivos Não-Convencionais de Interação Utilizando Realidade Aumentada: Um Estudo de Caso, fev 2005;

BERNARDES, J. Descrição do enJine e seu uso com Realidade Aumentada. São Paulo, 2005;

BICHO, A. L. et al. Programação Gráfica 3D com OpenGL, Open Inventor e Java 3D. Revista Eletrônica de Iniciação Científica (REIC), vol. II, n. I, mar 2002;

CARVALHO, M. M.; RABECHINI JR., R., Construindo Competências para Gerenciar Projetos, São Paulo, 2005;

DART. DART: The Designer's Augmented Reality Toolkit. Disponível em <http://www.cc.gatech.edu/ael/projects/dart.html>;

enJine. Engine para Jogos Online em Java. Disponível em: <http://enjine.incubadora.fapesp.br>;

FERREIRA, A. P. Utilização do MDA em conjunto com ODP em soluções distribuídas. Editora Atlas, 1ed, 2006;

Game Engine. Disponível em: <http://en.wikipedia.org/wiki/Game_engine>;

GAMEHERO. Game Hero, jogo para PlayStation 2 desenvolvido pela REDOCTANE e publicado pela Activision. Disponível em: <http://www.guitarherogame.com/>;

GAMESPOT. Site de notícias sobre jogos. Disponível em <http://www.gamespot.com>;

GEIGER, C. et al. JARToolkit – A Java Binding for ARToolkit. Augmented Reality Toolkit, The First IEEE International Workshop. 2002;

GILLEANES T. A. Guedes, UML - Uma Abordagem Prática, Novatec Editora, 2004;

IDG NOW!. Reportagem sobre Sistemas Operacionais no Brasil, Revista IDG NOW!, mai 2005;

INFOEXAME. Reportagem sobre Estagnação do Mercado de Jogos Mundial, Revista Info Exame, fev 2006;

INTERLAB3D. Ferramenta de suporte ao ensino de Computação Gráfica desenvolvida em Java. Disponível em: <http://interlab3d.incubadora.fapesp.br/portal>;

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J., The Unified Modeling Language User Guide. Addison Wesley, 1999;

JARTOOLKIT. jarToolkit. Disponível em <http://www.c-lab.de/jartoolkit/>;

JOSÉ NETO, J.; KOMATSU, W., Compiladores de gramáticas descritas em notação de Wirth Modificada, Anais EPUSP, Serie B: Engenharia de Eletricidade, São Paulo, 1988;

JUNIOR, O. B. de C., Garantia de Qualidade de Software, Faculdade de Administração, Contabilidade e Informática da Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2003;

MILGRAM, P.; KISHINO, F. A Taxonomy of Mixed Reality Visual Displays, IEICE Transactions on Information Systems, dez 1994, Special Issue on Networked Reality;

OI PROJECT. The OpenIllusionist Project. Disponível em <http://www.openillusionist.org.uk/>;

OPENCV. OpenCV. Disponível em <http://www.intel.com/technology/computing/opencv/>;

OPENVIDIA. openVidia. Disponível em <http://openvidia.sourceforge.net/>;

PMBOK. A Guide to the Project Management Body of Knowledge (PMBOK Guide), Project Management Institute, 2000;

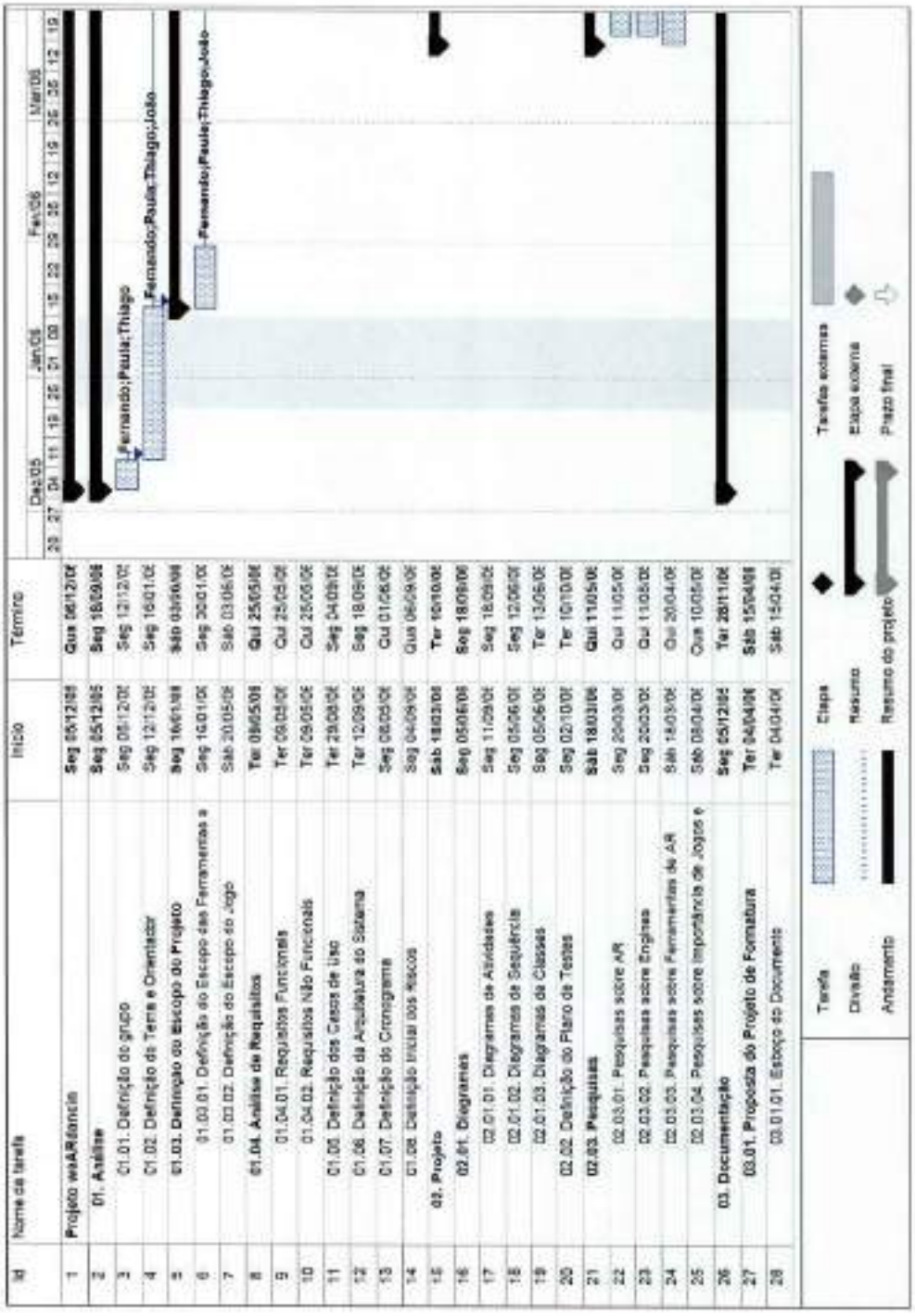
PRESSMAN, R. S., Engenharia de Software. MCGraw-Hill, 5ed., 2002;

ROGERRABBIT. Informações sobre o filme "Who Framed Roger Rabbit". Disponível em <http://www.bsospirit.com/comentarios/whoframedrogerrabbit.php>;

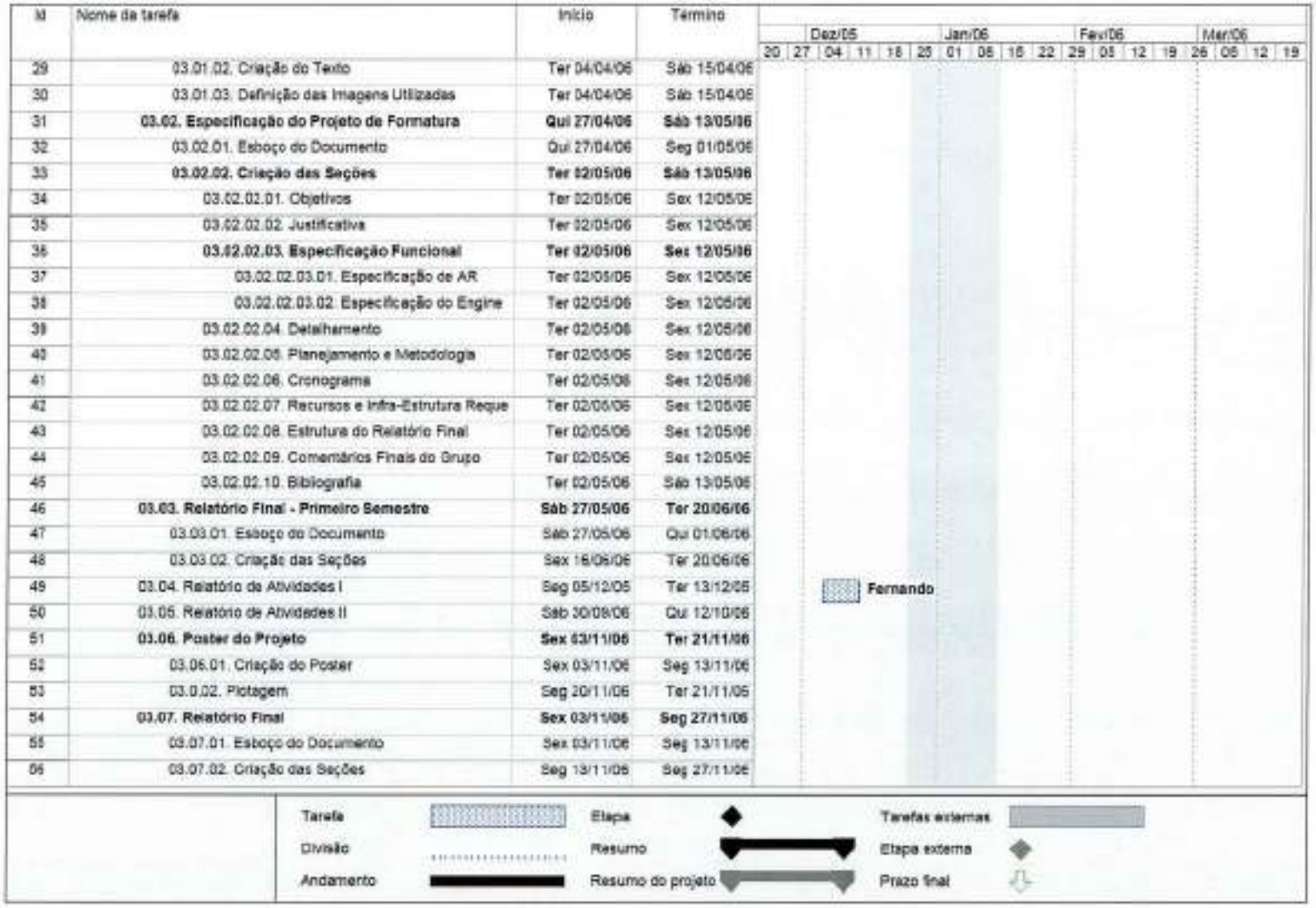
SILVA, R. J. M., WAGNER, G. N., RAPOSO, A. B., GATTASS, M., Experiência de Portais em Ambientes Arquitetônicos Virtuais, VI Simpósio de Realidade Virtual, Ribeirão Preto, 2003;

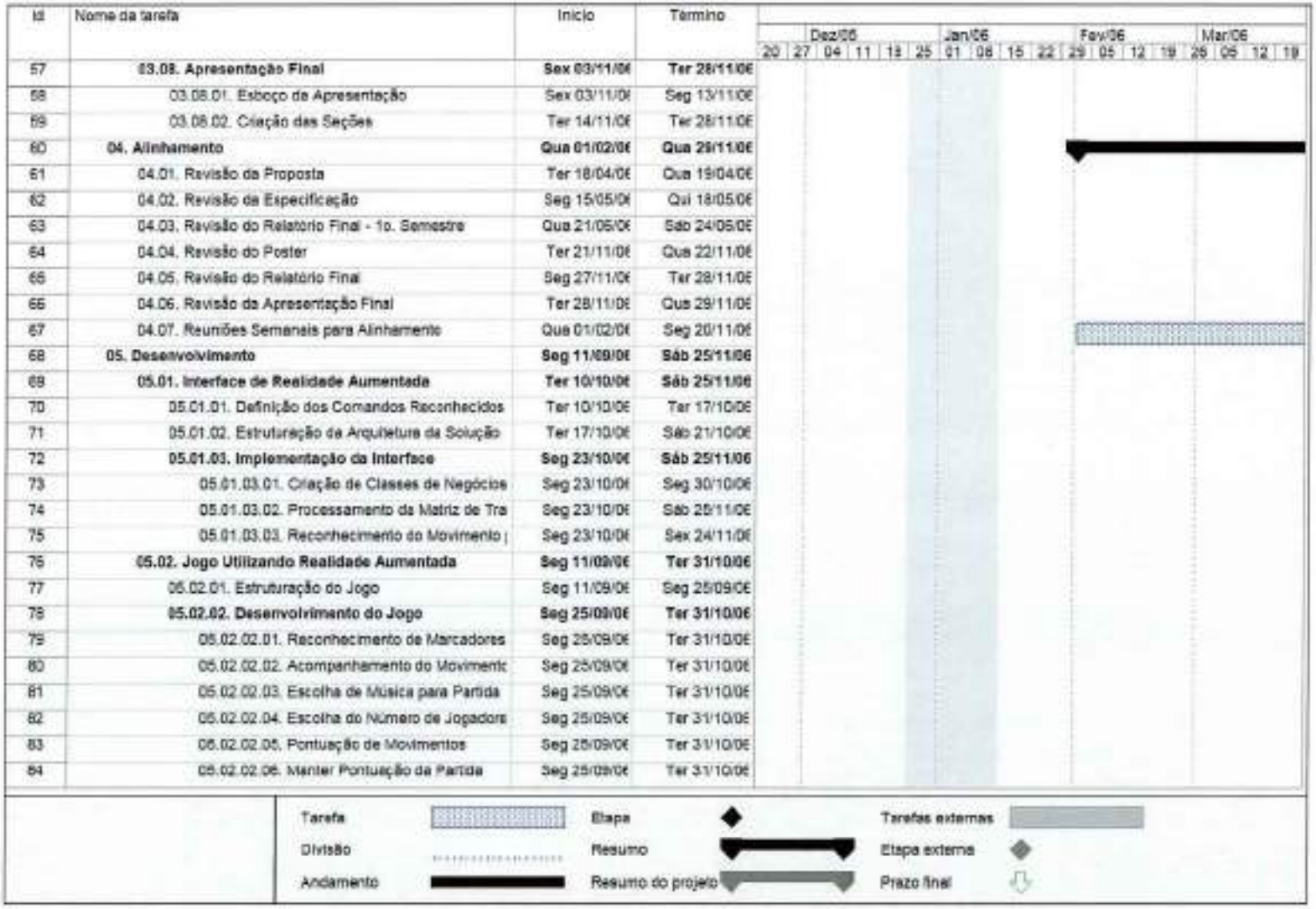
VALENTE, L., Representação de Cenas Tridimensionais: Grafo de Cenas, Instituto de Computação , Universidade Federal Fluminense, 2004.

Apêndice 1 - Gráfico de Gantt do Projeto



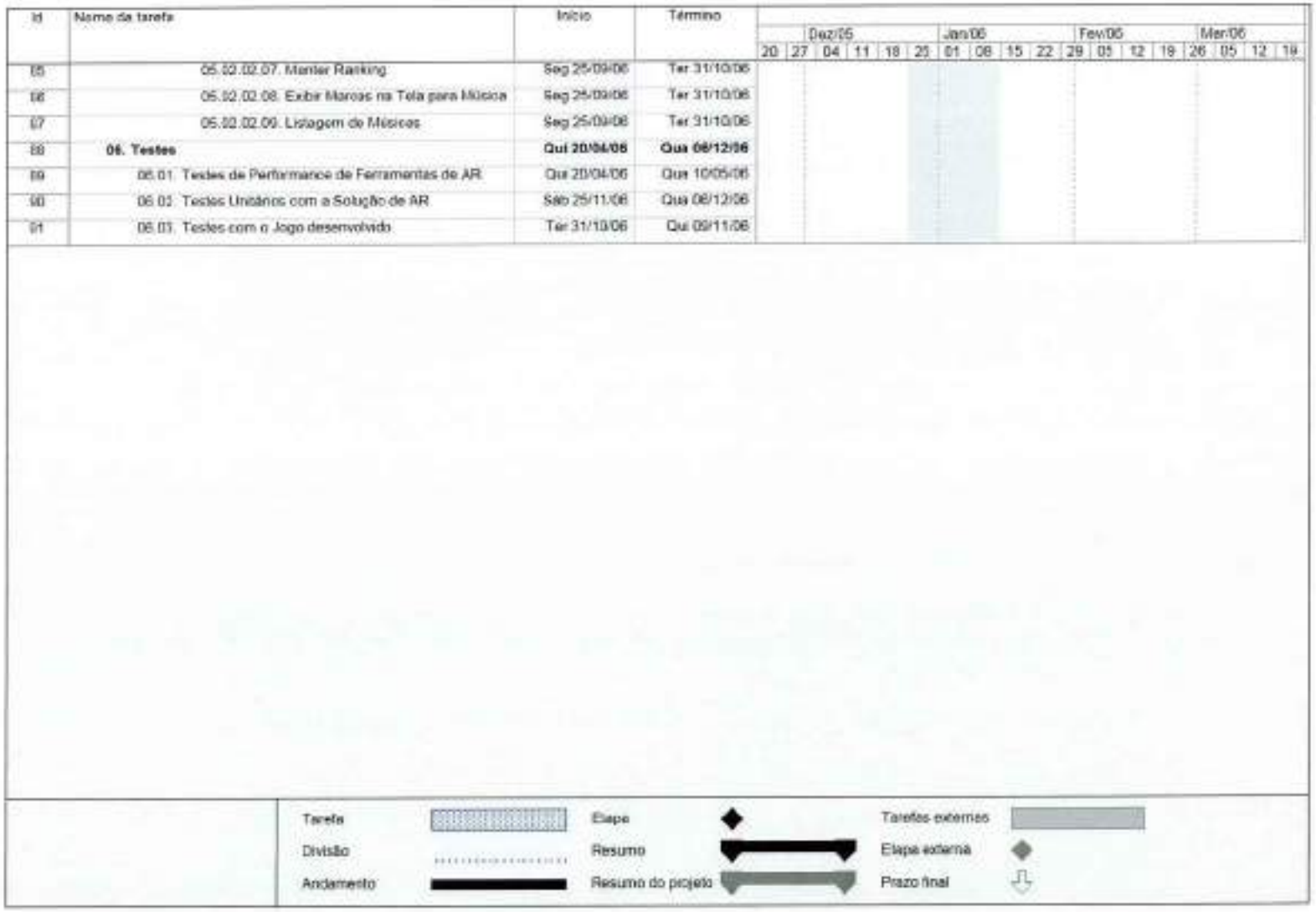
Id	Nome da tarefa	Início	Término
29	03.01.02. Criação do Texto	Ter 04/04/06	Sáb 15/04/06
30	03.01.03. Definição das Imagens Utilizadas	Ter 04/04/06	Sáb 15/04/06
31	**03.02. Especificação do Projeto de Formatura**	**Qui 27/04/06**	**Sáb 13/05/06**
32	03.02.01. Esboço do Documento	Qui 27/04/06	Seg 01/05/06
33	**03.02.02. Criação das Seções**	**Ter 02/05/06**	**Sáb 13/05/06**
34	03.02.02.01. Objetivos	Ter 02/05/06	Sex 12/05/06
35	03.02.02.02. Justificativa	Ter 02/05/06	Sex 12/05/06
36	**03.02.02.03. Especificação Funcional**	**Ter 02/05/06**	**Sex 12/05/06**
37	03.02.02.03.01. Especificação de AR	Ter 02/05/06	Sex 12/05/06
38	03.02.02.03.02. Especificação do Engine	Ter 02/05/06	Sex 12/05/06
39	03.02.02.04. Detalhamento	Ter 02/05/06	Sex 12/05/06
40	03.02.02.05. Planejamento e Metodologia	Ter 02/05/06	Sex 12/05/06
41	03.02.02.06. Cronograma	Ter 02/05/06	Sex 12/05/06
42	03.02.02.07. Recursos e Infra-Estrutura Requ	Ter 02/05/06	Sex 12/05/06
43	03.02.02.08. Estrutura do Relatório Final	Ter 02/05/06	Sex 12/05/06
44	03.02.02.09. Comentários Finais do Grupo	Ter 02/05/06	Sex 12/05/06
45	03.02.02.10. Bibliografia	Ter 02/05/06	Sáb 13/05/06
46	**03.03. Relatório Final - Primeiro Semestre**	**Sáb 27/05/06**	**Ter 20/06/06**
47	03.03.01. Esboço do Documento	Sáb 27/05/06	Qui 01/06/06
48	03.03.02. Criação das Seções	Sex 16/06/06	Ter 20/06/06
49	03.04. Relatório de Atividades I	Seg 05/12/05	Ter 13/12/05
50	03.05. Relatório de Atividades II	Sáb 30/09/06	Qui 12/10/06
51	**03.06. Poster do Projeto**	**Sex 03/11/06**	**Ter 21/11/06**
52	03.06.01. Criação do Poster	Sex 03/11/06	Seg 13/11/06
53	03.0.02. Plotagem	Seg 20/11/06	Ter 21/11/06
54	**03.07. Relatório Final**	**Sex 03/11/06**	**Seg 27/11/06**
55	03.07.01. Esboço do Documento	Sex 03/11/06	Seg 13/11/06
56	03.07.02. Criação das Seções	Seg 13/11/06	Seg 27/11/06

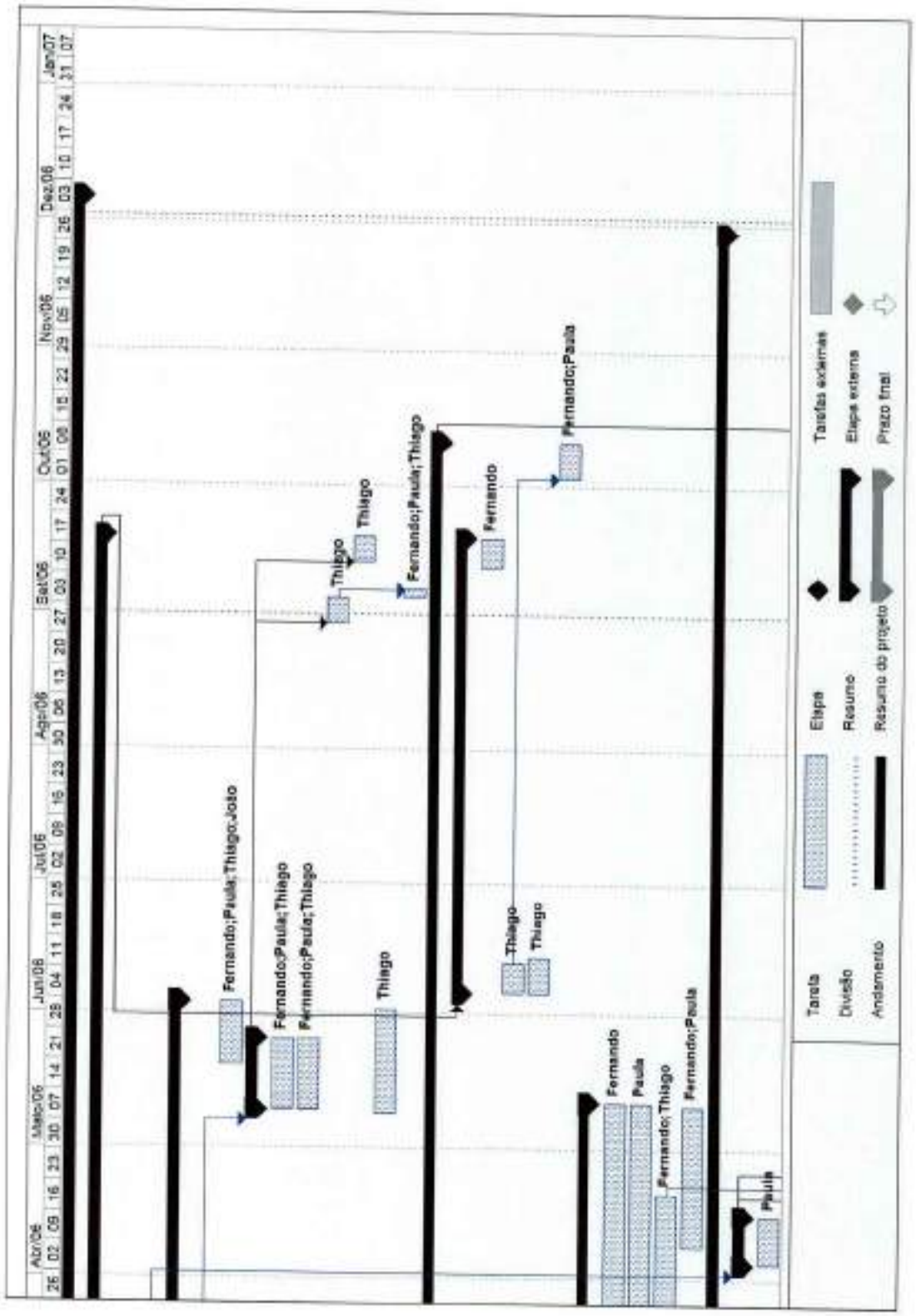


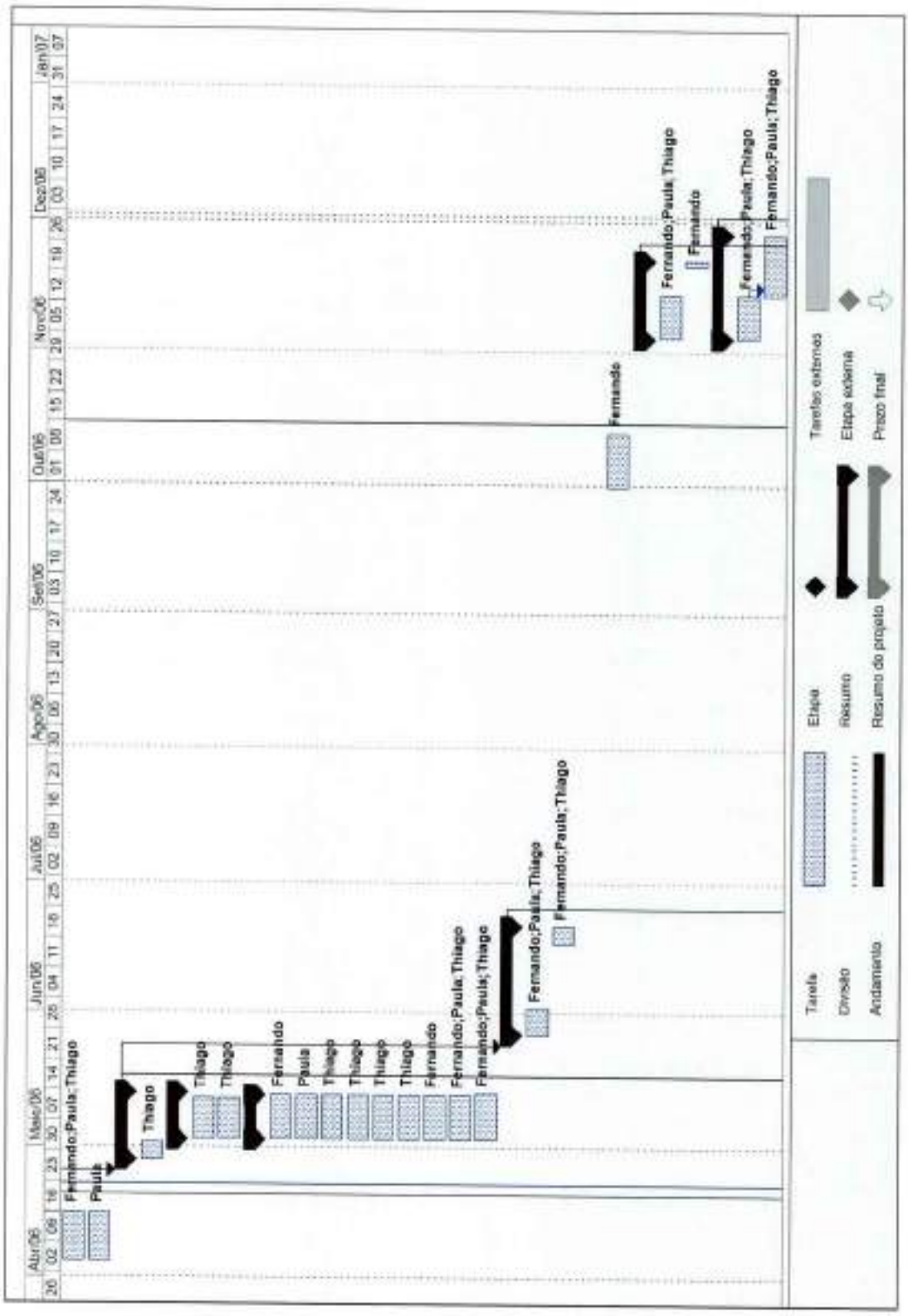


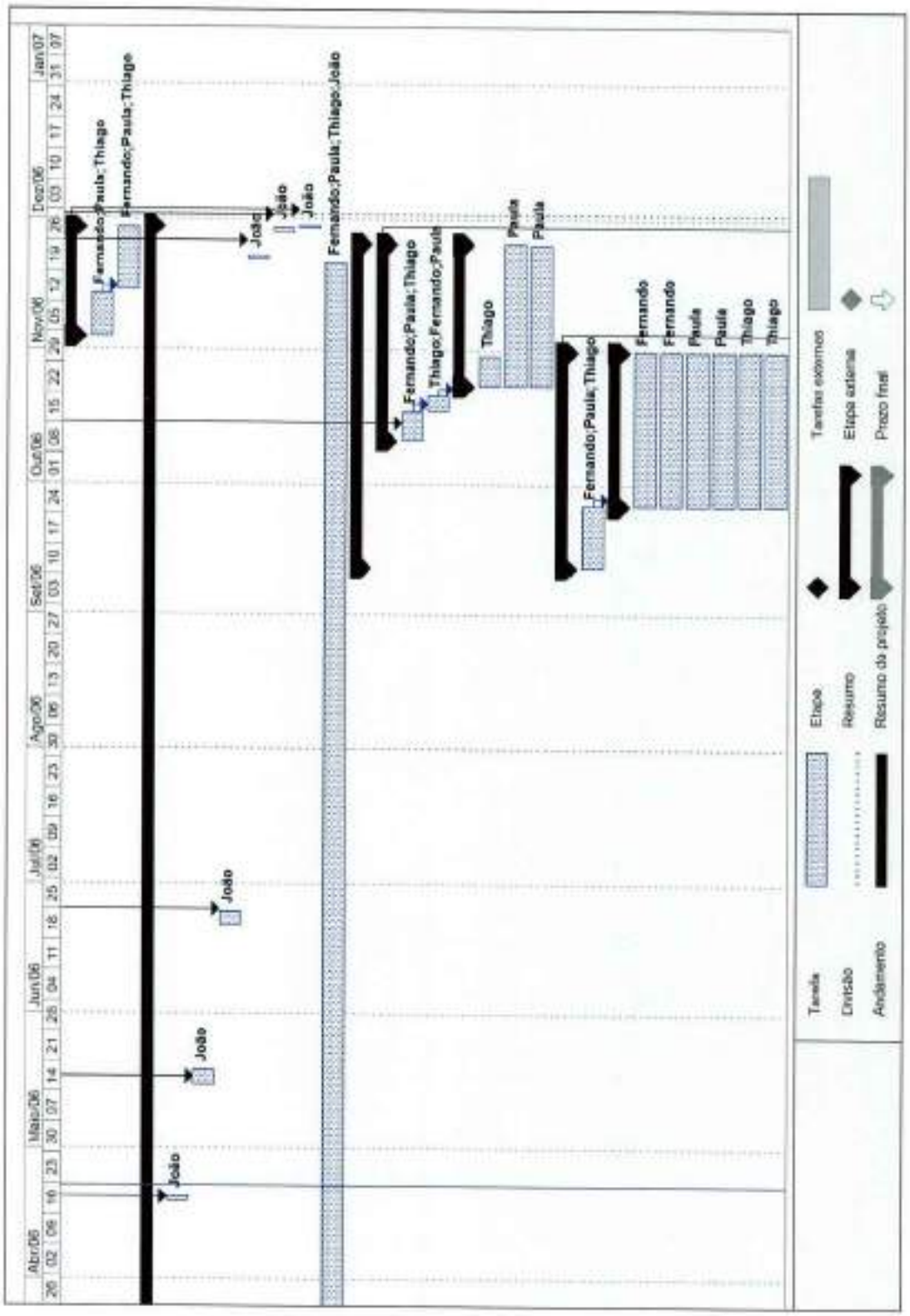
Id	Nome da tarefa	Início	Término
57	**03.08. Apresentação Final**	**Sex 03/11/06**	**Ter 28/11/06**
58	03.08.01. Esboço de Apresentação	Sex 03/11/06	Seg 13/11/06
59	03.08.02. Criação das Seções	Ter 14/11/06	Ter 28/11/06
60	**04. Alinhamento**	**Qua 01/02/06**	**Qua 29/11/06**
61	04.01. Revisão da Proposta	Ter 18/04/06	Qua 19/04/06
62	04.02. Revisão da Especificação	Seg 15/05/06	Qui 18/05/06
63	04.03. Revisão do Relatório Final - 1o. Semestre	Qua 21/06/06	Sáb 24/06/06
64	04.04. Revisão do Poster	Ter 21/11/06	Qua 22/11/06
65	04.05. Revisão do Relatório Final	Seg 27/11/06	Ter 28/11/06
66	04.06. Revisão da Apresentação Final	Ter 28/11/06	Qua 29/11/06
67	04.07. Reuniões Semanais para Alinhamento	Qua 01/02/06	Seg 20/11/06
68	**05. Desenvolvimento**	**Seg 11/09/06**	**Sáb 25/11/06**
69	**05.01. Interface de Realidade Aumentada**	**Ter 10/10/06**	**Sáb 25/11/06**
70	05.01.01. Definição dos Comandos Reconhecidos	Ter 10/10/06	Ter 17/10/06
71	05.01.02. Estruturação da Arquitetura da Solução	Ter 17/10/06	Sáb 21/10/06
72	**05.01.03. Implementação da Interface**	**Seg 23/10/06**	**Sáb 25/11/06**
73	05.01.03.01. Criação de Classes de Negócios	Seg 23/10/06	Seg 30/10/06
74	05.01.03.02. Processamento da Matriz de Tra	Seg 23/10/06	Sáb 25/11/06
75	05.01.03.03. Reconhecimento do Movimento j	Seg 23/10/06	Sex 24/11/06
76	**05.02. Jogo Utilizando Realidade Aumentada**	**Seg 11/09/06**	**Ter 31/10/06**
77	05.02.01. Estruturação do Jogo	Seg 11/09/06	Seg 25/09/06
78	**05.02.02. Desenvolvimento do Jogo**	**Seg 25/09/06**	**Ter 31/10/06**
79	05.02.02.01. Reconhecimento de Marcadores	Seg 25/09/06	Ter 31/10/06
80	05.02.02.02. Acompanhamento do Movimento	Seg 25/09/06	Ter 31/10/06
81	05.02.02.03. Escolha de Música para Partida	Seg 25/09/06	Ter 31/10/06
82	05.02.02.04. Escolha do Número de Jogadores	Seg 25/09/06	Ter 31/10/06
83	05.02.02.05. Pontuação de Movimentos	Seg 25/09/06	Ter 31/10/06
84	05.02.02.06. Manter Pontuação da Partida	Seg 25/09/06	Ter 31/10/06

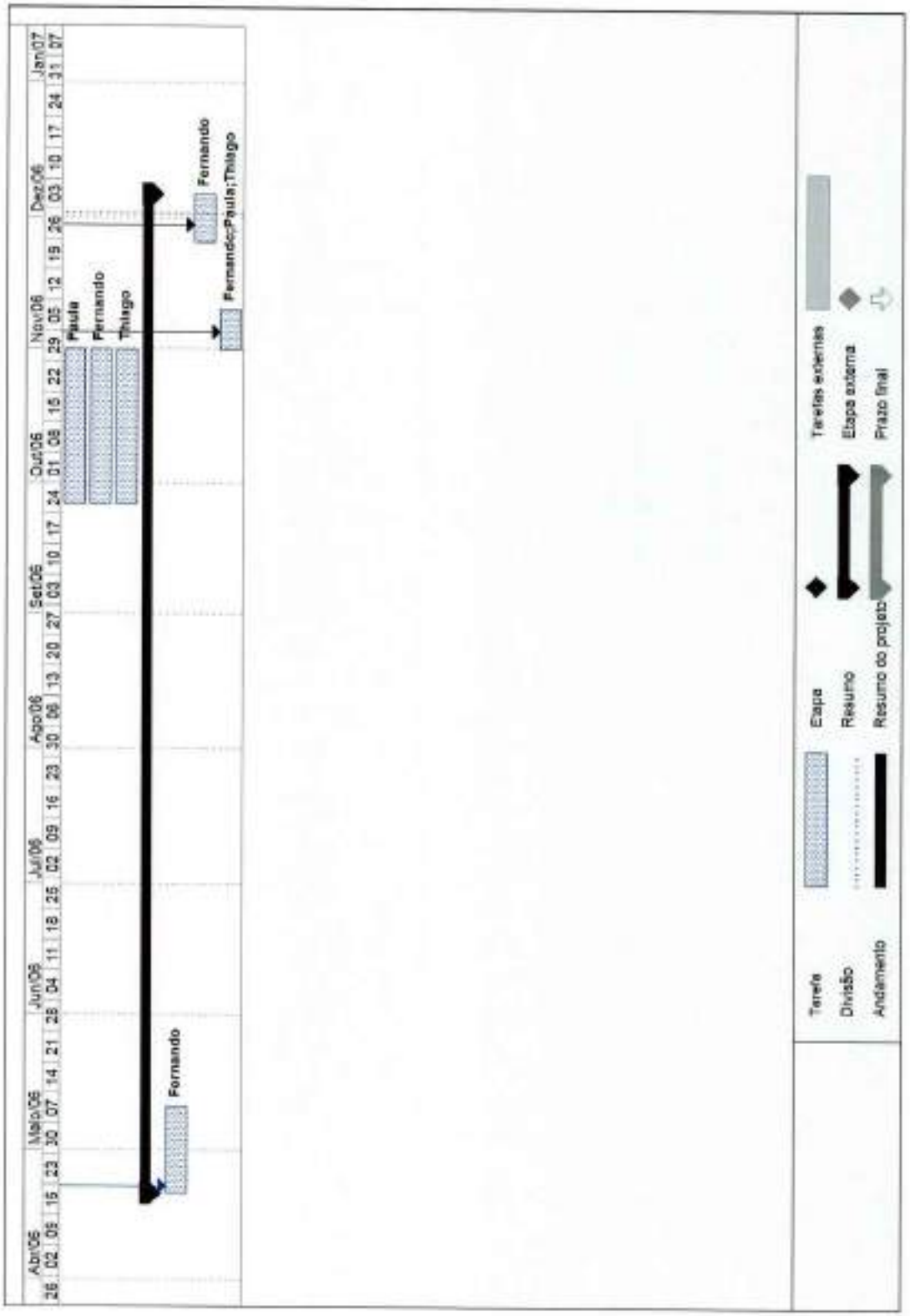
Id	Nome da tarefa	Início	Término
05	05.02.02.07. Manter Ranking	Seg 25/09/06	Ter 31/10/06
06	05.02.02.08. Exibir Marcas na Tela para Música	Seg 25/09/06	Ter 31/10/06
07	05.02.02.09. Listagem de Músicas	Seg 25/09/06	Ter 31/10/06
08	**06. Testes**	**Qui 20/04/06**	**Qua 06/12/06**
09	06.01. Testes de Performance de Ferramentas de AR	Qui 20/04/06	Qua 10/05/06
10	06.02. Testes Unitários com a Solução de AR	Sáb 25/11/06	Qua 06/12/06
11	06.03. Testes com o Jogo desenvolvido	Ter 31/10/06	Qui 09/11/06




Abr/06
Maio/06
Jun/06
Jul/06
Ago/06
Set/06
Out/06
Nov/06
Dez/06
Jan/07
Fernando;Paula;Thiago;João
Fernando;Paula;Thiago
Fernando;Paula;Thiago
Thiago
Thiago
Thiago
Fernando;Paula;Thiago
Fernando
Fernando;Paula
Thiago
Thiago
Fernando
Paula
Fernando;Thiago
Fernando;Paula
Paula
Tarefas externas
Etapa externa
Prazo final
Tarefa
Divisão
Andamento
Etapa
Resumo
Resumo do projeto


Abr/06
Mai/06
Jun/06
Jul/06
Ago/06
Set/06
Out/06
Nov/06
Dez/06
Jan/07
Fernando;Paula;Thiago
Paula
Thiago
Fernando
Fernando;Paula;Thiago
Fernando;Paula;Thiago
Fernando;Paula;Thiago
Fernando
Fernando;Paula;Thiago
Tarefa
Divisão
Andamento
Etapa
Resumo
Resumo do projeto
Tarefas externas
Etapa externa
Prazo final


Abr/06
Mai/06
Jun/06
Jul/06
Ago/06
Set/06
Out/06
Nov/06
Dez/06
Jan/07
João
Fernando;Paula;Thiago
Fernando;Paula;Thiago;João
Fernando;Paula;Thiago
Thiago;Fernando;Paula
Thiago
Paula
Fernando;Paula;Thiago
Fernando
Paula
Thiago
Tarefa
Divisão
Andamento
Etapa
Resumo
Resumo do projeto
Tarefas externas
Etapa externa
Prazo final


Abr/06
Mai/06
Jun/06
Jul/06
Ago/06
Set/06
Out/06
Nov/06
Dez/06
Jan/07
Fernando
Paula
Fernando
Thiago
Fernando
Fernando;Paula;Thiago
Tarefa
Divisão
Andamento
Etapa
Resumo
Resumo do projeto
Tarefas externas
Etapa externa
Prazo final

Apêndice 2 - Cronograma do Projeto

Atividade	*Início*	*Final*	*Predecessor*	*Responsável*
Projeto weARdancin	05/12/05	29/11/06		
01. Análise	05/12/05	18/09/06		
01.01. Definição do grupo	05/12/05	12/12/05		Fernando;Paula;Thiago
01.02. Definição do Tema e Orientador	12/12/05	16/01/06	3	Fernando;Paula;Thiago;João
01.03. Definição do Escopo do Projeto	16/01/06	18/09/06	4	
01.03.01. Definição do Escopo das Ferramentas a Serem Desenvolvidas	16/01/06	30/01/06		Fernando;Paula;Thiago;João
01.03.02. Definição do Escopo do Jogo	04/09/06	18/09/06		Fernando;Paula;Thiago;João
01.04. Análise de Requisitos	07/08/06	15/09/06	6	
01.04.01. Requisitos Funcionais	07/08/06	23/08/06		Fernando;Paula;Thiago
01.04.02. Requisitos Não Funcionais	31/08/06	15/09/06		Fernando;Paula;Thiago
01.05. Definição dos Casos de Uso	14/08/06	18/08/06	8	Thiago
01.06. Definição da Arquitetura do Sistema	14/08/06	18/08/06	8	Thiago
01.07. Definição do Cronograma	21/08/06	13/09/06		Thiago
01.08. Definição Inicial dos Riscos	28/08/06	30/08/06	11	Fernando;Paula;Thiago
02. Projeto	20/03/06	10/10/06		
02.01. Diagramas	19/09/06	27/09/06	2	
02.01.01. Diagramas de Atividades	19/09/06	26/09/06		Fernando
02.01.02. Diagramas de Seqüência	19/09/06	26/09/06		Thiago
02.01.03. Diagramas de Classes	19/09/06	27/09/06		Thiago
02.02. Definição do Plano de Testes	02/10/06	10/10/06	18	Fernando;Paula
02.03. Pesquisas	20/03/06	11/05/06		
02.03.01. Pesquisas sobre AR	20/03/06	11/05/06		Fernando
02.03.02. Pesquisas sobre Engines	20/03/06	11/05/06		Paula
02.03.03. Pesquisas sobre Ferramentas de AR	20/03/06	20/04/06		Fernando;Thiago
02.03.04. Pesquisas sobre Importância de Jogos e AR	10/04/06	10/05/06		Fernando;Paula
03. Documentação	19/12/05	28/11/06		
03.01. Proposta do Projeto de Formatura	05/04/06	17/04/06	4	
03.01.01. Esboço do Documento	05/04/06	17/04/06		Paula
03.01.02. Criação do Texto	05/04/06	17/04/06		Fernando;Paula;Thiago
03.01.03. Definição das Imagens Utilizadas	05/04/06	17/04/06		Paula
03.02. Especificação do Projeto de Formatura	28/04/06	13/05/06	27	
03.02.01. Esboço do Documento	28/04/06	02/05/06		Thiago
03.02.02. Criação das Seções	02/05/06	13/05/06		
03.03. Relatório Final - Primeiro Semestre	27/05/06	20/06/06	31	
03.03.01. Esboço do Documento	27/05/06	01/06/06		Fernando;Paula;Thiago
03.03.02. Criação das Seções	16/06/06	20/06/06		Fernando;Paula;Thiago

03.04. Relatório de Atividades I	28/08/06	05/09/06		Fernando
03.05. Relatório de Atividades II	30/09/06	12/10/06		Fernando
03.06. Pôster do Projeto	19/12/05	13/11/06		
03.06.01. Criação do Pôster	03/11/06	13/11/06		Fernando;Paula;Thiago
03.0.02. Plotagem	19/12/05	20/12/05		Fernando
03.07. Relatório Final	03/11/06	27/11/06		
03.07.01. Esboço do Documento	03/11/06	13/11/06		Fernando;Paula;Thiago
03.07.02. Criação das Seções	13/11/06	27/11/06	55	Fernando;Paula;Thiago
03.08. Apresentação Final	03/11/06	28/11/06		
03.08.01. Esboço da Apresentação	03/11/06	13/11/06		Fernando;Paula;Thiago
03.08.02. Criação das Seções	14/11/06	28/11/06	58	Fernando;Paula;Thiago
04. Alinhamento	02/02/06	29/11/06		
04.01. Revisão da Proposta	18/04/06	19/04/06	27	João
04.02. Revisão da Especificação	15/05/06	18/05/06	31	João
04.03. Revisão do Relatório Final - 1o. Semestre	21/06/06	24/06/06	46	João
04.04. Revisão do Pôster	13/11/06	14/11/06	51	João
04.05. Revisão do Relatório Final	27/11/06	28/11/06	54	João
04.06. Revisão da Apresentação Final	28/11/06	29/11/06	57	João
04.07. Reuniões Semanais para Alinhamento	02/02/06	21/11/06		Fernando;Paula;Thiago;João
05. Desenvolvimento	31/07/06	31/10/06		
05.01. Interface de Realidade Aumentada	31/07/06	21/09/06		
05.01.01. Definição dos Comandos Reconhecidos pelo Engine	31/07/06	07/08/06	15	Fernando;Paula;Thiago
05.01.02. Estruturação da Arquitetura da Solução	07/08/06	11/08/06	70	Thiago;Fernando;Paula
05.01.03. Implementação da Interface	11/08/06	21/09/06	71	
05.01.03.01. Criação de Classes de Negócios	11/08/06	18/08/06		Thiago
05.01.03.02. Processamento da Matriz de Transformação	18/08/06	21/09/06		Paula
05.01.03.03. Reconhecimento do Movimento pelo enJine	18/08/06	21/09/06		Paula
05.02. Jogo Utilizando Realidade Aumentada	11/09/06	31/10/06		
05.02.01. Estruturação do Jogo	11/09/06	25/09/06		Fernando;Paula;Thiago
05.02.02. Desenvolvimento do Jogo	25/09/06	31/10/06	77	Fernando;Paula;Thiago
05.02.02.01. Reconhecimento de Marcadores	25/09/06	31/10/06		Fernando
05.02.02.02. Acompanhamento do Movimento Executado	25/09/06	31/10/06		Fernando
05.02.02.03. Escolha de Música para Partida	25/09/06	31/10/06		Paula
05.02.02.04. Escolha do Número de Jogadores	25/09/06	31/10/06		Paula
05.02.02.05. Pontuação de Movimentos	25/09/06	31/10/06		Thiago
05.02.02.06. Manter Pontuação da Partida	25/09/06	31/10/06		Thiago
05.02.02.07. Manter Ranking	25/09/06	31/10/06		Paula
05.02.02.08. Exibir Marcas na Tela para Música	25/09/05	31/10/06		Fernando

05.02.02.09. Listagem de Músicas	25/09/06	31/10/06		Thiago
06. Testes	20/04/06	09/11/06		
06.01. Testes de Performance de Ferramentas de AR	20/04/06	10/05/06	24	Fernando
06.02. Testes Unitários com a Solução de AR	22/09/06	02/10/06	69	Fernando
06.03. Testes com o Jogo desenvolvido	31/10/06	09/11/06	76	Fernando;Paula;Thiago

Apêndice 3 - Tutorial para Desenvolvimento de Jogos de Realidade Aumentada com o EnJine

Introdução

O presente documento visa mostrar as etapas necessárias para a implementação de um jogo que utilize os conceitos de Realidade Aumentada através do enJine integrado com a biblioteca jARToolKit. O exemplo considera a utilização de uma câmera de video (webcam) conectada ao computador, além do conhecimento prévio da criação de jogos com o enJine.

Primeiramente, será apresentada uma visão geral da arquitetura. A estrutura de um jogo possui uma estrutura similar a apresentada na figura 1, onde é possível identificar três etapas principais: a inicialização, o loop principal e a finalização. As quais ocorrem seqüencialmente. A inicialização e a finalização ocorrem somente uma vez e envolvem a alocação e liberação dos recursos utilizados pelo jogo, incluindo-se os recursos do Java3D e do jARToolKit. O uso do Java3D torna o laço principal um conjunto de múltiplos threads que são executados em paralelo. A ilustração da figura 1 mostra essas threads seqüencialmente. Nela, pode-se perceber que o laço principal atualiza o estado do jogo, enquanto que os métodos do Java3D apenas lêem os dados a serem renderizados na tela. Os métodos referentes ao jARToolKit são executados na etapa de leitura de entradas, para a obtenção da imagem fornecida pela câmera. Através dela, são identificados os marcadores utilizados pelo jogo e obtido o stream de vídeo a ser exibido na tela do jogo.

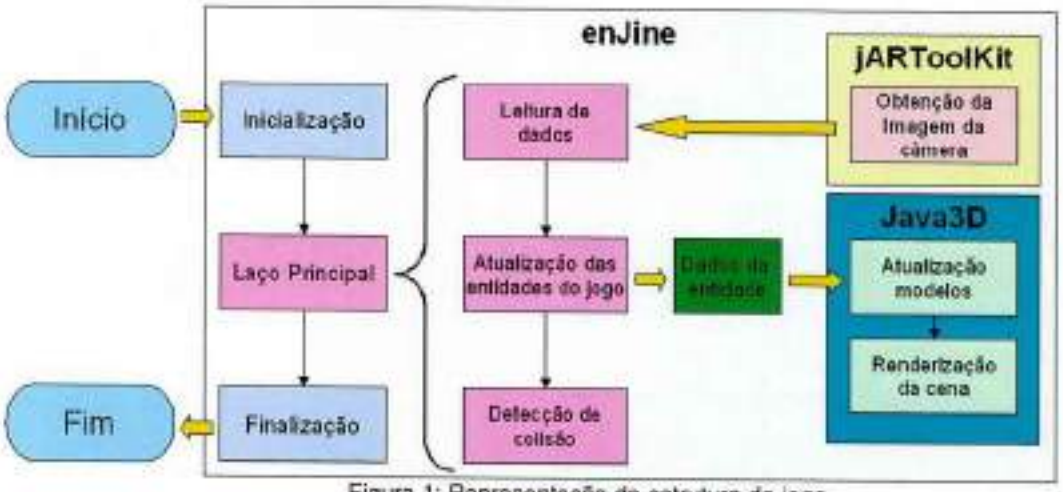


Figura 1: Representação da estrutura do jogo

Como exemplo, será criado uma demonstração onde um Cubo irá acompanhar o marcador. Esse exemplo é bem básico, visando apenas destacar as etapas fundamentais que diferenciam a implementação do jogo que usa Realidade Aumentada de um jogo comum. Nos exemplos dos códigos que serão exibidos, essas etapas estão marcadas com um comentário "AR". Recomenda-se também o uso de uma IDE para facilitar o desenvolvimento.

Criação de um novo jogo

Para a criação de um jogo básico, são necessárias algumas etapas básicas descritas a seguir:

- **Criação da classe que representa o jogo:**

1. Criar a classe "MyGame". A classe criada deverá estender a classe "ARGame", encontrada no pacote "interlab.engine.framework". Essa classe terá o atributo que referencia o objeto a ser criado e o método "main", responsável pela execução do jogo. Além desses, é recomendável a criação dos métodos de inicialização e de execução, tal como feito nesse exemplo;

2. O método "main" deverá alocar uma instância da classe "MyGame", para que ela possa ser inicializada e executada;

3. O método "initialize" terá os seguintes passos:
   3.1. Criação do objeto que faz referência ao jogo;
   3.2. Chamada do método "initialize" do objeto alocado, passando como parâmetros o título do jogo e o caminho e o nome do arquivo com as definições da câmera;
   3.3. Chamar a instância do "InputManager", através do método disponível no objeto;
   3.4. Definir o número de marcadores a serem utilizados no jogo a partir do método estático "setNumInstances" do ARInput (localizado no pacote interlab.engine.io.input);
   3.5. Chamar o método estático "setInstance" do ARInput, passando como parâmetros o índice, o caminho e o nome do arquivo referente ao marcador, o tipo de comportamento do objeto em relação à identificação do marcador e o tipo de movimento a ser reconhecido. Essa etapa deverá ser repetida de acordo com o número de marcadores definidos na etapa anterior. Assim, os valores válidos para o índice estão entre o 0 (zero) e o número de marcadores alocados. O tipo de comportamento define se o objeto que será controlado pelo marcador deverá continuar aparecendo na tela (true) ou não (false) caso o marcador não seja identificado em uma cena. O tipo de movimento indica como o marcador vai ser interpretado pelo jogo, sendo definidos três tipos:
      3.5.1. ***ABSOLUTE***: O objeto irá acompanhar o movimento do marcador na tela, mais indicado para o uso em relação à Realidade Aumentada;
      3.5.2. ***RELATIVE_CAMERA***: O objeto irá mover de acordo com o movimento do marcador em relação a uma posição inicial fixa, geralmente o centro da tela;
      3.5.3. ***RELATIVE_TIME***: O objeto irá mover de acordo com o movimento feito em relação à posição do instante anterior.

3.6. Chamar o método estático "registerDevices" do ARInput, passando como parâmetro a referência do objeto InputManager.

4. O método "run" terá os seguintes passos:
   4.1. Inicialização do listener de identificação dos marcadores na cena, através do método "startTimerTask" do objeto do jogo. O parâmetro a ser passado é o número de quadros provenientes do vídeo que deverão ser lidos por segundo;
   4.2. Chamar o método "mainLoop" do objeto do jogo, passando como parâmetro o estado inicial do jogo representado por um objeto "InGame", que será criado e detalhado adiante.

Exemplo da classe a ser criada:

```
// Classe MyGame - arquivo MyGame.java
import interlab.engine.framework.ARGame;
import interlab.engine.ic.input.ARInput;
import interlab.engine.ic.input.InputManager;

public class MyGame extends ARGame {

	private MyGame game;

	public void initialize() {
		game = new MyGame();
		game.initialize("Tutorial", "data/camera_para.dat"); //**AR**

		InputManager im = game.getInput();
		ARInput.setNumInstances(1); //**AR**
		try {
			ARInput.setInstance(0, "data/pattern/patt.kanji", false,
				ARInput.MOVEMENT_TYPE_ABSOLUTE); //**AR**
		} catch(Exception e) {
			e.printStackTrace();
		}
		ARInput.registerDevices(im); //**AR**
	}

	public void run() {
		game.startTimerTask(10); //**AR**
		game.mainLoop(new InGame());
	}

	public static void main(String[] args) {
		MyGame tutorial = new MyGame();
		tutorial.initialize();
		tutorial.run();
		System.exit(0);
	}
}
```

- **Criação dos objetos utilizados no jogo:**

Para o exemplo, é necessária somente a criação da classe "Cube". Aqui, o processo não apresenta muitas diferenças em relação a qualquer jogo implementado no enJine, porém será citado com caráter ilustrativo.

- Criar a classe "Cube", que deverá estender a classe "GameObject" do enJine (pacote interlab.engine.core), com os atributos referentes a posição e ângulos de rotação. Os métodos principais são os referentes ao acesso ("get" e "set") aos atributos. Também é necessário definir a construtora;
- Criar a classe "CubeUpdater", que estende a classe "Updater" (interlab.engine.core) e será responsável pela atualização do objeto "Cube" de acordo com os movimentos do marcador. Ela deverá conter como atributos a referência ao objeto "Cube" e os movimentos que serão obtidos do marcador (InputActions). É possível obter os seis graus de liberdade existentes (translação e rotação nos três eixos geométricos). Aqui é necessário criar os métodos de acesso aos atributos, a construtora e implementar os métodos abstratos da classe "Updater";
- Criar a classe "CubeView", que estende a classe "Viewable" (interlab.engine.core) e será responsável pelo nó de visualização do cubo que será processado pelo Java3D. Aqui é necessário criar os métodos de acesso aos atributos, a construtora e implementar os métodos abstratos da classe "Viewable".

Exemplo das classes a serem criadas:

```
//Classe Cube - arquivo Cube.java
import javax.vecmath.Vector3f;
import interlab.engine.core.GameObject;

public class Cube extends GameObject {

	private Vector3f position;
	private float rotA;
	private float rotB;
	private float rotC;

	public Cube() {
		position = new Vector3f(0.0f, 0.0f, 0.0f);
	}
```

```
        // OBS: Criar os "get"s e "set"s.
        public Vector3f getPosition(){return position;}
        public float getRotA(){return rotA;}
        public float getRotB(){return rotB;}
        public float getRotC(){return rotC;}

        public void setPosition(Vector3f position) {this.position = position;}
        public void setRotA(float rotA) {this.rotA = rotA;}
        public void setRotB(float rotB) {this.rotB = rotB;}
        public void setRotC(float rotC) {this.rotC = rotC;}
}

// Classe CubeUpdater - arquivo CubeUpdater.java
import javax.vecmath.Vector3f;
import interlab.engine.core.Game;
import interlab.engine.core.Updater;
import interlab.engine.io.input.InputAction;

public class CubeUpdater extends Updater {

        private Cube parent;
        private InputAction posX;
        private InputAction posY;
        private InputAction posZ;
        private InputAction rotA;
        private InputAction rotB;
        private InputAction rotC;

        public CubeUpdater(Cube p) {
                super();
                parent = p;
                posX = new InputAction(0, 0, "positionX");
                posY = new InputAction(1, 0, "positionY");
                posZ = new InputAction(2, 0, "positionZ");
                rotA = new InputAction(3, 0, "rotationA");
                rotB = new InputAction(4, 0, "rotationB");
                rotC = new InputAction(5, 0, "rotationC");
        }

        public void update(Game g, float delta) {
                //Método abstrato a ser implementado.
                float x, y, z;
                x = posX.getIntensity();
                y = posY.getIntensity();
                z = posZ.getIntensity();
                parent.setPosition(new Vector3f(x, y, z));

                parent.setRotA(rotA.getIntensity());
                parent.setRotB(rotB.getIntensity());
                parent.setRotC(rotC.getIntensity());
        }

        public InputAction getPosX() {return posX;}
        public InputAction getPosY() {return posY;}
        public InputAction getPosZ() {return posZ;}
        public InputAction getRotA() {return rotA;}
        public InputAction getRotB() {return rotB;}
        public InputAction getRotC() {return rotC;}
}

// Classe CubeView - Arquivo CubeView.java
import javax.media.j3d.Appearance;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import com.sun.j3d.utils.geometry.ColorCube;
import interlab.engine.core.Viewable;

public class CubeView extends Viewable {

        private Cube parent;
        private BranchGroup root;
        private TransformGroup position;
```

```
    private TransformGroup rotA;
    private TransformGroup rotB;
    private TransformGroup rotC;

    public CubeView(Cube p) {
        super();
        parent = p;
        root = new BranchGroup();
        position = new TransformGroup();
        position.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        rotA = new TransformGroup();
        rotA.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        rotB = new TransformGroup();
        rotB.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
        rotC = new TransformGroup();
        rotC.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

        ColorCube cc = new ColorCube(20.0f);
        cc.setAppearance(new Appearance());

        rotA.addChild(cc);
        rotB.addChild(rotA);
        rotC.addChild(rotB);
        position.addChild(rotC);
        root.addChild(position);
    }

    public BranchGroup getView() {
        //Método abstrato a ser implementado.
        return root;
    }

    public void updateView(long delta) {
        //Método abstrato a ser implementado.
        Transform3D t3d = new Transform3D();
        t3d.setIdentity();
        t3d.rotX(parent.getRotA());
        rotA.setTransform(t3d);

        t3d.setIdentity();
        t3d.rotY(parent.getRotB());
        rotB.setTransform(t3d);

        t3d.setIdentity();
        t3d.rotZ(parent.getRotC());
        rotC.setTransform(t3d);

        t3d.setIdentity();
        t3d.setTranslation(parent.getPosition());
        position.setTransform(t3d);
    }
}
```

- **Criação da classe que representa o estado do jogo:**

Para o exemplo, a classe "InGame", herdeira da "GameState" (interlab.engine.core) representa o estado do jogo que representa o "mainLoop" do jogo.

- Definir os atributos do jogo nesse estado. Nesse caso, existe somente o cubo;

- Implementar o método "initialize", definindo-se o espaço de colisão, a alocação dos objetos referentes ao cubo (CubeUpdater e CubeView), fazer o binding entre os InputActions do CubeUpdater com os sensores do ARInput e alocar o nó do grafo de cena responsável pelo reconhecimento dos marcadores na imagem obtida da câmera;
- Implementar o método "cleanup", o qual libera os recursos quando o jogo é encerrado. É importante remover o nó referente ao grafo de cena de reconhecimento dos marcadores. Na prática, esse método não é chamado no exemplo, mas é uma boa prática a ser adotada.

Exemplo da classe a ser criada.

```
// Classe InGame - arquivo InGame.java
import javax.media.jsd.BoundingSphere;
import interlab.engine.collision.SimpleCollisionManager;
import interlab.engine.collision.Subspace;
import interlab.engine.core.Game;
import interlab.engine.core.GameState;
import interlab.engine.io.graphics.GameView;
import interlab.engine.io.input.ARInput;
import interlab.engine.io.input.InputManager;
import interlab.engine.io.input.InputSensor;

public class InGame extends GameState {

        private Cube cubeObj;

        public void initialize(Game game) {
                MyGame g = (MyGame)game;
                SimpleCollisionManager scm = g.getCollider();
                BoundingSphere bs = new BoundingSphere();
                bs.setRadius(Double.POSITIVE_INFINITY);
                Subspace sp = new Subspace(bs);
                scm.addSubspace(sp);

                InputManager im = g.getInput();
                InputSensor[] directions;

                cubeObj = new Cube();
                CubeView v = new CubeView(cubeObj);
                cubeObj.setViewable(v);
                CubeUpdater u = new CubeUpdater(cubeObj);

                directions = ARInput.getInstance(0).getSensors(); //**AR**
                for(int i = 0; i < directions.length; i++) {
                        if(directions[i].getSensorCode() == ARInput.AREVENT_AXES_X) {
                                im.bind(directions[i], u.getPosX());
                        }
                        else if(directions[i].getSensorCode() == ARInput.AREVENT_AXES_Y) {
                                im.bind(directions[i], u.getPosY());
                        }
                        else if(directions[i].getSensorCode() == ARInput.AREVENT_AXES_Z) {
                                im.bind(directions[i], u.getPosZ());
                        }
                        else if(directions[i].getSensorCode() == ARInput.AREVENT_ROT_X) {
                                im.bind(directions[i], u.getRotA());
                        }
                        else if(directions[i].getSensorCode() == ARInput.AREVENT_ROT_Y) {
                                im.bind(directions[i], u.getRotB());
```

```
                }
                else if(directions[i].getSensorCode() == ARInput.AREVENT_ROT_Z) {
                        in.bind(directions[i], u.getRotC());
                }
        }
        cubeObj.setUpdater(u);

        g.addARBranch(true); //**AR**
    }

    public void cleanup(Game game) {
        //Metodo abstrato a ser implementado.
        MyGame g  =(MyGame)game;
        g.removeAllObjects();
        g.getInput().unbindAll();
        g.removeARBranch(true); //**AR**
    }

    //Os próximos metodo abstratos não são se aplicam a este exemplo.
    public void preStep(Game game, long delta) {}

    public void update(Game game, float delta) {}

    public void postStep(Game game, long delta) {}

}
```

Executando o jogo

Para executar o jogo, a variável de ambiente "CLASSPATH" deve ser corretamente configuradas, indicando onde os pacotes do enJine e do jARToolKit estão localizados. A variável "PATH" também deve ser configurada para o diretório "bin" do jARToolKit, onde estão localizados as bibliotecas "dll"s utilizadas pelo jARToolKit.

Para executar, criar o pacote do jogo chamado tutorial.jar (para facilitar, utilize uma IDE) e, na linha de comando, executar:

java -Dj3d.rend=d3d -jar tutorial.jar

A figura 2 mostra uma imagem do programa-exemplo rodando.

Figura 2: Programa rodando

Apêndice 4 - Artigo Publicado no Segundo Workshop de Aplicações de Realidade Virtual

INTEGRAÇÃO DO JARTOOLKIT COM ENGINE DIDÁTICO PARA A CRIAÇÃO DE JOGOS USANDO REALIDADE AUMENTADA

Fernando Tsuda, Paula M. Hokama, Thiago M. Rodrigues, João L. Bernardes Jr., Romero Tori
Escola Politécnica da Universidade de São Paulo
CEP 05508-900, São Paulo – SP
Brasil
e-mail: fernando.tsuda@gmail.com, paula.hokama@gmail.com, thiagomarcira13@gmail.com, joao.bernardes@poli.usp.br, romero.tori@poli.usp.br

Resumo – *O objetivo do presente trabalho é realizar a integração entre o JARToolkit e o enJine, um engine de jogos didático, visando assim verificar o quanto se facilita o desenvolvimento de jogos com realidade aumentada combinando os recursos fornecidos por essas ferramentas. Como prova de conceito será desenvolvido também um jogo com essa tecnologia. O controle do jogo, que combina elementos reais e virtuais, será realizado com o uso de uma webcam juntamente com marcadores fiduciais passivos, que permitem identificar posições no ambiente real. Para validar a solução em seu estágio atual, pois trata-se de um trabalho em andamento, foram criados dois protótipos e esses resultados parciais são descritos aqui.*

Palavras-Chave – *Engines* de Jogos, JARToolkit, Jogos, Realidade Aumentada.

Abstract – *This work's goal is to integrate JARToolkit and enJine, a didactic game engine, in order to verify how much the development of augmented reality games is facilitated combining the resources of these tools. As a concept proof, a game will be developed using this technology. Initially, the game, which combines real and virtual elements, will have commands that make use of a webcam and passive fiducial markers, that allow the identification of positions in the real world. To validate the solution in its current stage, since this is an ongoing work, two prototypes have been built and these partial results are described here.*

Keywords – Augmented Reality, Game Engines, Games, JARToolkit.

1. INTRODUÇÃO

O objetivo desse trabalho é integrar o JARToolkit e um *engine* de jogos para facilitar o desenvolvimento de jogos que usem realidade aumentada, passando pela análise da viabilidade do uso dessas duas ferramentas para esse tipo de aplicação e quais as preocupações necessárias no processo de game design para tais jogos. O desenvolvimento de um jogo combinando elementos reais e virtuais e com interação através de marcadores fiduciais como prova de conceito também é um objetivo do trabalho, mas atualmente só foram desenvolvidos dois protótipos, pois trata-se de um projeto em andamento.

A realidade aumentada (Augmented Reality ou AR) é uma tecnologia que permite combinar com registro em 3D elementos virtuais e elementos reais, com os quais o usuário pode interagir em tempo real [1]. Para possibilitar essa interação, é comum o uso de dispositivos de custo elevado, como por exemplo, HMDs (Head Mounted Displays) para a exibição da cena com os objetos virtuais, e sensores de posição, como dispositivos GPS (Global Positioning System) ou sensores magnéticos, para a localização do usuário no ambiente. É possível, porém, o uso de outros dispositivos, como *webcams*, monitores convencionais, PDAs ou celulares. Nesse trabalho pretende-se fazer uso de tecnologias de baixo custo e, portanto, de fácil acesso ao público, como *webcams* e monitores de vídeo.

Para o desenvolvimento de aplicações de realidade aumentada, estão disponíveis diversas bibliotecas que fornecem as funcionalidades necessárias, seja de combinação de imagens reais e virtuais, seja de registro em 3D. Entre elas pode-se citar o ARToolkit, o qual permite o desenvolvimento de aplicações em C++ [2], e o JARToolkit, cujas classes permitem o acesso às funções do ARToolkit utilizando-se a linguagem Java [3].

Essa tecnologia pode ser utilizada em uma série de aplicações, como a medicina, a manutenção de prédios e equipamentos e a área de entretenimento em geral. O foco deste trabalho é essa última categoria, mais especificamente a área de jogos eletrônicos.

A área de jogos é interessante para o desenvolvimento de ferramentas de realidade aumentada, pois não é necessária a grande precisão requerida nas outras áreas. Isso permite o uso de equipamentos com menor custo, como *webcams* e computadores desktop. Além disso, a tecnologia permite o desenvolvimento de jogos originais com uma maior imersão do usuário com o ambiente, o que é benéfico para atrair um nicho de usuários que possuem dificuldade em assimilar grandes quantidades de comandos no teclado ou controladores [4]. Esses novos usuários podem aumentar ainda mais o futuramento nessa área, que em 2003 atingiu a marca de 10 bilhões de dólares, ultrapassando assim o lucro obtido com a venda de ingressos pela indústria de cinema de

Hollywood [5]. Jogos também são uma aplicação interessante para novas interfaces como RA pois, devido a seu aspecto lúdico, os usuários tendem a "perdoar" imperfeições nas interfaces em estágio de protótipo e compreender melhor o seu potencial [6].

Para auxiliar no ensino de desenvolvimento de jogos e outras disciplinas de computação (principalmente computação gráfica), o Laboratório de Tecnologias Interativas (Interlab-USP) desenvolveu, utilizando a linguagem Java e a biblioteca Java3D, um *engine* didático, denominado enJine, que fornece classes para representar os elementos presentes em um jogo e com os elementos gráficos organizados na forma de grafo de cena [7]. O uso dessa ferramenta tem aumentado a motivação dos alunos ao aplicar conceitos teóricos no desenvolvimento de jogos. Por ser em código aberto, também possibilita ao aluno compreender o funcionamento de um *engine* de jogos [8]. Esse é o *engine* didático usado nesse trabalho e com o qual o JARToolkit é integrado. Um segundo propósito do enJine é servir como plataforma de testes para novas tecnologias em jogos, visto que sua arquitetura simples (pois tem fins didáticos) e bem conhecida tem se mostrado adequada a esse fim. O presente trabalho se encaixa mais nesse segundo propósito, embora também possa ser usado com fins didáticos, principalmente em disciplinas mais relacionadas a RA.

2. BIBLIOTECAS UTILIZADAS

Inicialmente, foi necessário realizar um estudo das bibliotecas de realidade aumentada existentes. Além dos já mencionados ARToolkit e JARToolkit, realizou-se um estudo sobre as outras bibliotecas de realidade aumentada disponíveis como o ARToolkit Plus [9] e o ARTag [10], bem como a biblioteca de visão computacional OpenCV [11], da Intel. A escolha do JARToolkit ocorreu principalmente devido à sua utilização com a linguagem Java, facilitando assim a integração com o enJine.

O ARToolkit, e conseqüentemente o JARToolkit, funcionam utilizando técnicas de visão computacional, onde uma câmera captura a imagem contendo marcadores passivos, que são identificados na imagem e têm sua posição relativa à câmera calculada a partir de deformações na imagem. A partir dos valores de posição obtidos, é possível criar diversos aplicativos. A utilização mais comum é a renderização de objetos virtuais sobre os marcadores detectados. A figura 1 ilustra o funcionamento com esse uso.

Fig. 1. Aplicativo de RA utilizando o JARToolkit.

O enJine possui diversos pacotes responsáveis pelo tratamento das diferentes partes do jogo como, por exemplo, gráficos, sons, entrada de comandos, lógica do jogo e outros. O módulo responsável pela entrada de comandos do enJine é baseado numa camada de abstração que separa os dispositivos de entrada das correspondentes ações no jogo. Cada dispositivo (como um joystick ou teclado) é representado por uma subclasse "InputDevice" e este possui uma série de objetos "InputSensor" que representam elementos como botões do joystick ou teclas do teclado. E, a fim de vincular estes botões (InputSensor) a uma ação de um jogo (InputAction) como "pular", "mover para direita", por exemplo, existe uma última classe denominada "InputManager". A figura 2 ilustra a relação das classes referentes ao módulo de entrada do enJine.

Esse módulo de entrada é o principal componente do enJine a ser estendido para que haja a integração com o JARToolkit. As entradas da webcam, traduzidas pelo JARToolkit, constituem um "InputDevice", assim como o teclado. Os "InputSensors" podem ser, por exemplo, os movimentos relativos horizontais (dois "InputSensors": para esquerda ou direita) e verticais (para cima ou para baixo), ou então os valores de coordenadas X, Y e Z e das rotações. Nos protótipos desenvolvidos foram usados os movimentos relativos como sensores, mas no jogo final serão usadas posições absolutas.

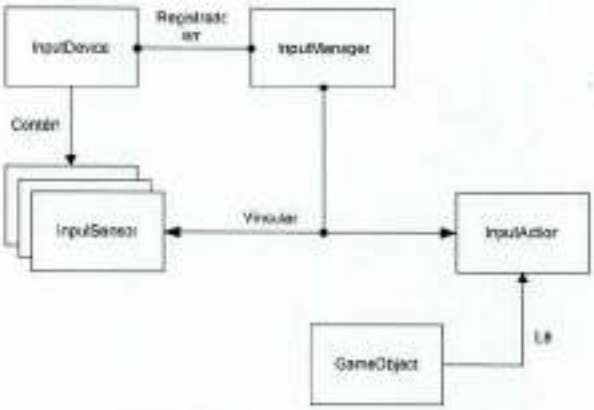


Fig. 2. Representação das entradas no enJine.

Além da extensão da funcionalidade de entrada do enJine, também é necessário estender sua funcionalidade de saída gráfica para que incorpore os objetos reais da cena, provenientes da *stream* de vídeo da webcam.

3. IMPLEMENTAÇÃO E RESULTADOS

O estudo do funcionamento interno do JARToolkit mostrou que a posição e a orientação dos marcadores são armazenados em uma matriz de transformação 4x4, contendo informações sobre a rotação e a translação do marcador (e também de escala, que nesse caso não faz sentido) e que normalmente são usados para realizar a mesma transformação no objeto virtual, dando a sensação ao usuário que o objeto está sobre o marcador.

Inicialmente, foi necessário adicionar o ramo do grafo de cena do JARToolkit responsável pela detecção do marcador na cena no enJine, possibilitando a obtenção da matriz de transformação. Porém, somente a adição desse ramo não é o suficiente para que os valores da matriz sejam reconhecidos como entrada de comandos. Isso foi resolvido com a implementação de uma classe de *middleware*, responsável pela transferência, em intervalos definidos pelo programador, da matriz do grafo de cena para a classe que representa o dispositivo de entrada. O *middleware* também contém as rotinas necessárias para a adição do ramo do grafo do JARToolkit no enJine. A abstração dos dispositivos de entrada feita pelo enJine permitiu definir um dispositivo hipotético, indicada por uma classe derivada do "InputDevice", que representa uma entrada através do uso de uma webcam e dos marcadores passivos. Essa classe faz a leitura da matriz obtida, a partir da qual os valores de translação e os ângulos de rotação são obtidos. Essas informações são então utilizadas para a geração de uma ação no jogo, nos protótipos desenvolvidos.

Além das informações referentes à posição, o JARToolkit também fornece o *stream* de vídeo proveniente da câmera. Para integrá-la ao enJine, foi adicionado ao ramo do grafo de cena do JARToolkit citado anteriormente um outro ramo responsável por essa funcionalidade.

A figura 3 ilustra a interface criada entre o enJine com o JARToolkit. Nessa figura, o *middleware* é responsável pela transferência dos valores da matriz de transformação para o jogo e pela captura das imagens de vídeo que serão exibidos nos jogos juntamente com os objetos virtuais.

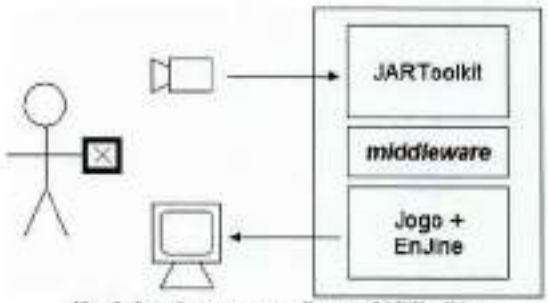


Fig. 3. Interface entre o enJine e o JARToolkit.

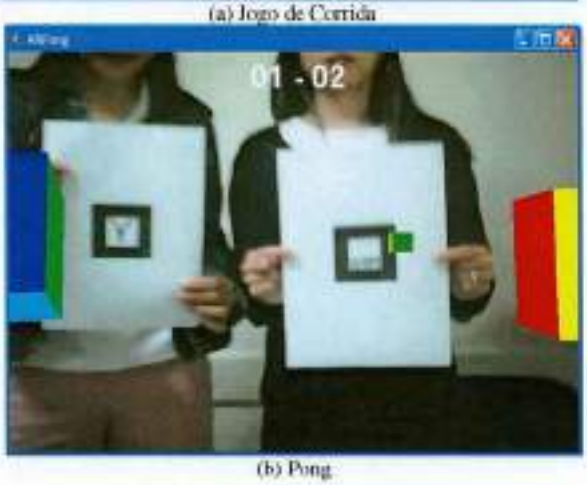
Como forma de se validar e demonstrar os resultados da solução adotada, foram desenvolvidos dois protótipos, testando principalmente a interação através de marcadores e não necessariamente constituindo, nesse estágio, aplicações de realidade aumentada. O primeiro consistiu em modificar a forma de entrada de um jogo previamente desenvolvido no enJine. O segundo mostra o desenvolvimento de uma versão do jogo Pong para ser jogado com o uso dos marcadores fiduciais. Nos protótipos já foi usada a *stream* de vídeo integrada ao grafo de cena, mas somente para testar essa funcionalidade, sem haver um registro entre os objetos reais e os virtuais.

Como primeiro protótipo, utilizou-se um jogo de corrida já existente desenvolvido no enJine, modificando sua forma de entrada de comandos. Nele o carro pode ser manipulado através da movimentação de um marcador para cima, baixo, esquerda e direita, desconsiderando-se, portanto, a movimentação em profundidade e as rotações. Um dos resultados interessantes dos testes com esse protótipo foi que a presença de muitos elementos gráficos (neste caso todo o cenário) na tela pareceu atrapalhar o controle por parte do usuário, por encobrir o vídeo e conseqüentemente a percepção da posição do marcador na tela.

Para o segundo protótipo, decidiu-se criar um jogo similar ao Pong. Ele foi escolhido devido à necessidade de poucos elementos na tela. O seu funcionamento consiste em movimentar o marcador para cima e para baixo, fazendo com que os cubos, representando as plataformas que no jogo original rebatem a bola, movam-se na direção correspondente e com o objetivo de acertar a bola para devolvê-la ao adversário. Nos testes com esse protótipo, o objetivo foi verificar o uso de mais de um marcador simultaneamente. Apesar das regras do jogo não terem sido completamente implementadas, o objetivo de se movimentar os objetos com marcadores diferentes foi atingido. Aqui também se testou a rotação através dos valores obtidos da matriz, porém elas não possuem nenhum efeito prático nas regras do jogo implementado até então. No estágio atual do projeto essa funcionalidade ainda apresenta alguns problemas.

A figura 4 mostra *screenshots* dos protótipos desenvolvidos.

(a) Jogo de Corrida



(b) Pong

Fig. 4. Protótipos Desenvolvidos.

Além dos protótipos, também foi construído um aparato para testar o tempo de resposta do sistema de detecção de posição usando o JARToolkit (antes da integração com o enJine) variando a distância entre o marcador e a câmera e a velocidade de movimentação do marcador. Constatou-se que a velocidade máxima de movimentação em que ainda ocorre a detecção diminui conforme aumenta a distância e que essas velocidade podem ser relativamente baixas para distâncias da ordem de metros. Esse fato influi no *game design* e influirá no jogo final, mas não significa que jogos desenvolvidos com essa tecnologia precisem ter movimentos lentos. Os movimentos podem ser rápidos, por exemplo, desde que o *game design* preveja que os marcadores devam ficar estáticos ou mover-se devagar por pequenos períodos de tempo para permitir sua detecção. Dependendo do *game design* isso pode inclusive ser imperceptível para o usuário.

4. ANÁLISE

A leitura da matriz de transformação obtida do JARToolkit está completamente funcional em relação à translação, porém os valores da rotação ainda não estão sendo reconhecidos corretamente. Isso ocorre pela dificuldade em se obter os ângulos de rotação a partir da matriz dada, onde mais de um resultado pode ser obtido.

Após a criação do primeiro protótipo, verificou-se a necessidade de se inverter horizontalmente a imagem fornecida pela câmera, para que o usuário pudesse atuar como se estivesse em frente a um espelho, melhorando assim a usabilidade. Porém, para atingir esse objetivo, foi necessário modificar internamente uma classe do JARToolkit responsável, entre outras coisas, pelo *stream* de vídeo e pela obtenção das matrizes de transformação.

Apesar de ainda não representarem totalmente aplicações de realizada aumentada, o desenvolvimento das demonstrações mostrou uma limitação dos jogos em relação à quantidade de elementos gráficos virtuais na tela, o que pode poluir a janela do jogo e dificultar a noção de posicionamento do usuário.

Percebeu-se também a influência que as limitações do uso de visão computacional como principal input, em especial a limitação da velocidade de movimentação quando o marcador está distante da câmera, pode ter sobre o game design.

5. CONCLUSÕES E TRABALHOS FUTUROS

Através dos protótipos, pode-se verificar que a solução adotada é adequada para o desenvolvimento de jogos em realidade aumentada.

No momento, além da resolução dos problemas citados anteriormente, estão sendo feitos testes para encontrar funções de calibração da câmera, permitindo assim uma maior precisão na detecção e no posicionamento dos marcadores.

Paralelamente a isso, está sendo desenvolvido um jogo mais complexo usando a solução proposta.

Como um trabalho futuro, pode-se propor a remoção de fundo da imagem da câmera para integrar a imagem do jogador, obtida em tempo real, ao jogo como o próprio avatar do jogador ou personagem principal do jogo.

Outra possibilidade é a renderização dos elementos reais com profundidade (atualmente a *stream* de vídeo é uma imagem de fundo, sempre oclusa pelos objetos virtuais), de forma que possam fazer oclusão de objetos virtuais. Isso poderia inclusive reduzir o problema de muitos elementos na cena do jogo atrapalharem a interação devido à oclusão dos elementos reais.

O desenvolvimento de novos jogos em realidade aumentada utilizando essas ferramentas e explorando novos *designs* e formas de interação também é, obviamente, uma importante possibilidade de trabalhos futuros.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] R.T Azuma. A Survey of Augmented Reality. *Teleoperators and Virtual Environments*, vol. 6, p. 355-385, aug. 1997.

[2] ARToolkit 2.33 manual. Disponível em: http://www.lamse.ufrj.br/grva/realidade_aumentada.

[3] C. Geiger. et al. JARToolkit – A Java Binding for ARToolkit. Augmented Reality Toolkit, *in The First IEEE International Workshop*, 2002.

[4] Mesa Redonda Universo Games 2006, realizada no SENAC-SP, em agosto de 2006.

[5] K. Hause. What to Play Next: Gaming Forecast 1999-2003, *in Conference On Human Factors In Computing Systems*, p. 894-895, 2003.

[6] T. Starner et al. Mind-Warping: Towards Creating a Compelling Collaborative Augmented Reality Game, *in 5th International Conference on Intelligent User Interfaces*, p. 256-259, 2000.

[7] EnJine: Engine para Jogos Online em Java. Disponível em http://enjine.incubadora.fapesp.br.

[8] R. Tori et al. Teaching Introductory Computer Graphics Using Java 3D, Games and Customized Software: a Brazilian Experience, *in International Conference on Computer Graphics and Interactive Techniques SIGGRAPH 2006*, n. 12, 2006.

[9] ARToolkit Plus. Disponível em http://studierstube.org/handheld_ar/artoolkitplus.php.

[10] ARTag. Disponível em http://www.cv.iit.nrc.ca/research/ar/artag/.

[11] OpenCV. Disponível em http://www.intel.com/technology/computing/opencv/.