EDUARDO COSTA JACOBBER

**PROPOSTA E IMPLEMENTAÇÃO DE UMA INTERFACE PARA
MOTORES DE JOGOS INTERATIVA E CENTRADA NO USUÁRIO**

São Paulo
2007

EDUARDO COSTA JACOBBER

**PROPOSTA E IMPLEMENTAÇÃO DE UMA INTERFACE PARA
MOTORES DE JOGOS INTERATIVA E CENTRADA NO USUÁRIO**

Dissertação apresentada à
Escola Politécnica da Universidade de
São Paulo para obtenção do título de
Mestre em Engenharia.

São Paulo
2007

EDUARDO COSTA JACOBBER

**PROPOSTA E IMPLEMENTAÇÃO DE UMA INTERFACE PARA
MOTORES DE JOGOS INTERATIVA E CENTRADA NO USUÁRIO**

Dissertação apresentada à
Escola Politécnica da Universidade de
São Paulo para obtenção do título de
Mestre em Engenharia.

Área de Concentração:
Engenharia Elétrica – Departamento de
Computação e Sistemas Digitais

Orientador:
Prof. Livre-Docente
Romero Tori

São Paulo
2007

Este Exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com anuência de seu orientador.

São Paulo, 16 de Maio de 2007

Assinatura do autor

Assinatura do orientador

FICHA CATALOGRÁFICA

Jacober, Eduardo Costa

**Proposta e implementação de uma interface para motores de jogos interativa e centrada no usuário / E.C. Jacober. – ed.rev. -- São Paulo, 2007.
175 p.**

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

**1.jogos eletrônicos 2.Usabilidade de software 3.Interação usuário-computador 4.Ambientes de programação
I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.**

Para meu filho,
Rafael

AGRADECIMENTOS

A Deus pela inspiração,
a meus pais pelo apoio e
a minha esposa pela compreensão.

Ao amigo e orientador

Prof. Dr. Romero Tori não somente por concordar com a iniciativa, como também por encorajar a procura de disciplinas relacionadas ao tema em outras unidades da USP, dentre inúmeras outras decisões importantes que moldaram essa dissertação.

Este trabalho somente foi possível com o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, através da bolsa de pesquisa do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo – PCS/EPUSP pelo período de Março de 2004 a Março de 2005; e de meus pais, Fernando e Maria Celia.

O processo de testes de usabilidade e projeto participatório utilizando protótipo em papel ocorreu entre 17 de Agosto e 10 de Outubro de 2006. Os testes foram realizados nas dependências do Interlab-USP e somente foram possíveis graças à dedicação e ao apoio dos seguintes participantes:

Usuários voluntários para as sessões de teste:

Dennis Akihiko Tanaka; Felipe Kamakura; Rodrigo Petriche; William Colen

“Core Team” (Desenvolvedores, CPU, GPU e Observadores):

Eduardo Costa Jacober; João Luiz Bernardes Jr.; Ricardo Nakamura

Observadores e Desenvolvedores:

Alexandre Nascimento Tomoyose; Caue Fiedler; Daniel Calife;

Roberto Cezar Bianchini; Yuri Correa

Produção e Apoio:

Prof. Dr. Romero Tori

Um brinde a:

Alberto e Mariana; Takuo, Simone e Henrique; Pequeno Timothem, Premê e Tubaína; Mumu, Marrento e Mala; Zé Colméia e Mara; Gene, Steven e Ian; Elias, Jefferson, Gonzaga e todo o pessoal do Senac; Tita, Cleber, Francisquinho e toda minha família; Homer, Al e Katia; Fábio, Caio e Nino; Roger, Edison e Felipe; Mario Seabra; Agaevz; Alessandro, Diego, Evellyn, Thais, Ademir, Gabriel, Pedro, Leandro, Paulo, Maurício, David, Thomazini e todos os (des)orientados; Ricardo, Roberto, João e todo o pessoal do Interlab; Mario, Yoshi e Link; Shimabuko, Elizângela, Jairo, Lester e todo o pessoal da Prolan; Rodrigo, Juliana, Dulce, Neuto, Vera, Procópio, Eduardo e Juliana; Kassatu Maru, Cuca e a maçã; Joaquim, Rodo, Gabriel, Bicho, Mena, Murilim e Os Pirata.

Deus quis que os homens naturalmente tivessem todas as formas de alegria para
que pudessem suportar os desgostos e tribulações da vida, quando lhes
sobreviessem.

Por isso os homens procuraram muitos modos de realizar com plenitude tal alegria e
criaram diversos jogos que os divertissem.

(Dom Afonso X, o sábio)

RESUMO

Um motor de jogo é uma ferramenta que facilita a criação de jogos eletrônicos garantindo a reutilização de *softwares*. Por sua origem tecnológica, os motores de jogo atuais, tanto comerciais quanto acadêmicos, tendem a primar pelo número de funcionalidades específicas, em detrimento da usabilidade, o que amplia o período de tempo necessário para seu aprendizado. Com o objetivo de melhor adaptar um motor de jogo didático às necessidades pedagógicas, este trabalho propõe uma interface interativa para motores denominada **Game Wrench** e criada através de métodos de Interação Humano-Computador. Como se tratam de áreas multidisciplinares, é também apresentado um embasamento teórico sobre os diversos aspectos envolvidos, a começar pela adoção de uma definição precisa para o conceito de jogo. O cenário específico para o qual este ambiente de programação (motor e interface) foi desenvolvido é composto pelos alunos da disciplina Computação Gráfica da Escola Politécnica da USP, que programam sobre um motor de jogo didático chamado **enJine**. Através de um protótipo em papel, testes de usabilidade foram realizados com parte desses alunos, o que gerou diversas diretrizes para a programação final da interface. Uma versão preliminar dessa interface usando plataforma **Java** foi desenvolvida, gerando diversos resultados e contribuições ao longo do processo, como artefatos de projeto (*personas*, cenários), ferramentas de desenvolvimento (estrutura **GAML**, classes para **enJine**, *widgets* especializados) e diretrizes sobre a metodologia utilizada (avaliação heurística, prototipagem em papel, teste de usabilidade).

Palavras-chave: Interação Humano-Computador. Ambiente de Programação. Motor de Jogo. Teste de Usabilidade. Jogos Eletrônicos. Ensino e Aprendizagem.

ABSTRACT

Game engines are tools to aid the development of computer and video games through software reuse. Once born in a technological environment, present game engines, either commercial or not, are normally worried more about the number of implemented features than usability issues. These characteristics may extend the period of learning-time. To address the goal of better adapting a didactic game engine to pedagogical needs, this study proposes an interactive interface for engines called **Game Wrench**, following Computer-Human Interface methods. Being multidisciplinary fields, a theoretical base on the related subjects is presented, starting with a precise definition for the concept of gaming. Undergraduate students of Escola Politecnica da USP enrolled in a Computer Graphics course where they develop games using a didactic engine called **enJine**, compose the specific scenario for which this programming environment was developed. Using paper prototypes, usability tests were conducted with some of these students, generating a list of guidelines for implementing this interface. A preliminary **Java** version of the interface was made available, generating results and contribution throughout the design process, such as design artifacts (personas, scenarios), programming tools (**GAML** specification, **enJine** classes, specialized widgets) and guidelines for the methodology adopted (heuristic evaluation, paper prototypes, usability tests).

Keywords: Computer-Human Interaction. Development Environment. Game Engine. Usability Test. Electronic Game. Teaching and Learning.

LISTA DE ILUSTRAÇÕES

Figura 2-1 - Descobertas arqueológicas relacionadas a jogos.....	18
Figura 2-2 - Jogo conhecido por Senat, precursor do Gamão, encontrado no Antigo Egito.....	18
Figura 2-3 - Modelo de Walther	20
Figura 2-4 - Screenshots de produtos-jogo com o tema Tênis.....	32
Figura 2-5 - Caso 1: Criação da Interface do Motor de Jogo após sua codificação ...	56
Figura 2-6 - Caso 2: Criação de Motor de Jogo otimizado para o Game Designer	57
Figura 3-1 - Documentos produzidos, atividades realizadas e softwares criados e utilizados como metodologia. Setas indicam seqüência e interdependência das tarefas.....	61
Figura 4-1 - Screenshot do jogo do museu, mod-game utilizando o Unreal Engine. .	76
Figura 4-2 - Screenshot de FootBot, produto-jogo criado sobre o enJine.....	79
Figura 4-3 - Tela Inicial do Game Maker v.6.1	81
Figura 4-4 - Tela Inicial do 3D Game Studio v.A6.....	83
Figura 4-5 - Protótipo em papel.	86
Figura 6-1 - Estrutura em Schema XSD da GAML para armazenar informação de sistemas de jogo.....	97
Figura 6-2 - Estrutura em Schema XSD da GAML para armazenar informação de jogos.	98
Figura 6-3 - Estrutura em Schema XSD da GAML para armazenar informação de campeonatos.	99
Figura 6-4 - Estrutura em Schema XSD da GAML para armazenar informação de um projeto de produto-jogo.....	100
Figura 7-1 - Diagramação Visual do Game Wrench implementado com a plataforma Java Swing.....	106
Figura 7-2 - Detalhe de Yellow Helper e Red Tooltip sendo testados.	106
Figura 7-3 - Screenshot do Game Wrench durante teste utilizando GAML.	107
Figura 7-4 - Detalhe do código enJine gerado pelo Game Wrench durante o teste.	107

LISTA DE TABELAS

Tabela 2.1 - Priorização dada pelas regras do jogo a outros elementos de produtos-jogo.	26
Tabela 2.2 - Comparação entre as capacidades de humanos e sistemas digitais.....	41
Tabela 2.3 - Exemplo de distribuição de atividades produtivas no ciclo de vida de um jogo eletrônico comercial.	54
Tabela 4.1 - Projeto de Tarefas para o cenário do protótipo em papel.....	74

LISTA DE ABREVIATURAS E SIGLAS

<i>API</i>	<i>Application Programming Interface</i>
<i>CG</i>	<i>Computação Gráfica</i>
<i>CN-V</i>	<i>Comunicação Não-Verbal</i>
<i>DOM</i>	<i>Document Object Model</i>
<i>EPUSP</i>	<i>Escola Politécnica da Universidade de São Paulo</i>
<i>FPS (gênero de jogo)</i>	<i>FirstPerson Shooter</i>
<i>FPS (métrica)</i>	<i>Frames Per Second</i>
<i>GAML</i>	<i>GAme Markup Language</i>
<i>GML</i>	<i>Game Maker Language</i>
<i>GUI</i>	<i>Graphical User Interface</i>
<i>IA</i>	<i>Inteligência Artificial</i>
<i>IDE</i>	<i>Integrated Development Environment</i>
<i>IHC</i>	<i>Interação Humano-Computador</i>
<i>Interlab</i>	<i>LABoratório de tecnologias INTERativas</i>
<i>MMORPG</i>	<i>Massively Multiplayer Online RPG</i>
<i>MOD, mod-games</i>	<i>game MODifications</i>
<i>NPC</i>	<i>Non-Player Character</i>
<i>POV, viewpoint</i>	<i>Point Of View</i>
<i>RPG</i>	<i>RolePlaying Game</i>
<i>RTS</i>	<i>RealTime Strategy</i>
<i>SAX</i>	<i>Simple API for XML</i>
<i>SME</i>	<i>Subject Matter Experts</i>
<i>TI</i>	<i>Tecnologia da Informação</i>
<i>VP</i>	<i>Victory Points</i>
<i>XML</i>	<i>eXtensible Markup Language</i>
<i>XSD</i>	<i>XML Schema Definition</i>

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Motivação e Objetivos.....	15
1.2	Estrutura	16
2	CONCEITUAÇÃO E DEFINIÇÃO DO PROBLEMA.....	18
2.1	Conceito de Jogo: Uma Definição	18
2.2	Aplicações do Modelo.....	22
2.2.1	Exemplos: Características e Elementos que Permitem as Transgressões	22
2.2.1.1	Primeira transgressão	22
2.2.1.2	Segunda transgressão	23
2.2.2	Elementos de Jogo e Outros Elementos em Produtos-Jogo.....	26
2.2.3	Elementos de Jogo em Outras Mídias	27
2.2.4	Narrativa e Ludologia.....	29
2.2.5	Jogo, Ritual e Mídias Lineares.....	29
2.2.6	Jogo e Arte	30
2.2.7	Simulação vs. Emulação.....	31
2.2.8	Sorte, Caos e Aleatoriedade.....	32
2.3	Desenvolvimento de Jogos Eletrônicos.....	33
2.3.1	Complexidade Crescente	33
2.3.2	O Motor de Jogo	34
2.3.3	Motores Comerciais	35
2.3.4	Motores Não-Comerciais.....	36
2.3.5	Modificações de Jogos	37
2.4	Interação Humano-Computador.....	38
2.4.1	Modelo de Comunicação	39
2.4.2	Humanos e Computadores.....	39
2.4.3	Multidisciplinaridade de IHC	43
2.4.4	Usabilidade	43
2.5	Etapas e Atividades de um Projeto de Interação.....	44
2.5.1	O Desenvolvimento de Software Tradicional.....	45
2.5.2	O Projeto de Interação e a Engenharia de Software.....	45
2.5.3	O Projeto Orientado por Tarefas.....	46
2.5.4	O Projeto de Interação: Uma Definição	47
2.5.5	Projeto de Interação Orientado por Metas.....	48
2.5.6	A Engenharia de Usabilidade.....	48
2.5.7	Prototipagem em papel	49
2.5.8	Avaliação de Interfaces	50
2.6	Usuários de um Motor de Jogo	51
2.6.1	O Projetista de Jogos	52
2.6.2	Classificação de Usuários	52
2.6.2.1	Universo Corporativo.....	55
2.6.2.2	Universo Serious Games.....	58
2.6.2.3	Universo amador	59
3	METODOLOGIA	60
3.1	Pesquisa Bibliográfica.....	61
3.2	Projeto de Interação.....	62

3.3	Análise Comparativa	63
3.4	Estudo de Usabilidade com Prototipagem em Papel.....	63
3.4.1	Entrevistas com Stakeholders	64
3.4.2	Recrutamento de Usuários	64
3.4.3	Projeto de Tarefas.....	64
3.4.4	Prototipagem em papel	65
3.4.5	Teste de Usabilidade de Voz-Alta Simplificado.....	65
3.5	Protótipo Computacional da Interface Game Wrench	66
4	DESENVOLVIMENTO	67
4.1	Projeto de Interação.....	67
4.1.1	Escopo do Projeto de Interação.....	67
4.1.2	Conceito do Produto.....	68
4.1.3	Cenários	68
4.1.3.1	Definição de usuários específicos	69
4.1.3.2	Criação de Personas.....	69
4.1.3.2.1	Validação da hipótese de personas	70
4.1.3.2.2	Metas dos Usuários	71
4.1.3.2.3	Escopo e escolha de usuário-foco	71
4.1.3.3	Definição de tarefas específicas	72
4.1.3.3.1	O jogo a ser criado	72
4.1.3.3.2	Projeto de tarefas	72
4.1.3.3.3	Definição do contexto	73
4.2	Análise Comparativa.....	75
4.2.1	Jogo do Museu	75
4.2.2	enJine	78
4.2.3	Motores de jogo com Interface de Desenvolvimento.....	80
4.2.3.1	Game Maker	80
4.2.3.2	3D Game Studio	83
4.3	Estudo de Usabilidade com Protótipo em Papel.....	84
4.3.1	Criação do protótipo.....	84
4.3.2	Sessões de Teste.....	87
4.3.3	Observações dos Usuários	88
4.3.4	Conclusões e Diretrizes a Partir dos Testes.....	88
4.3.5	Considerações sobre o processo:.....	89
5	GAME WRENCH – ESPECIFICAÇÃO DE AMBIENTE INTERATIVO PARA MOTORES DE JOGOS	91
5.1	Requisitos de Usuário.....	91
5.2	Requisitos de Usabilidade	91
5.3	Requisitos Funcionais.....	92
5.4	Requisitos Não-Funcionais.....	93
5.5	Requisitos de Dados	93
5.6	Requisitos de Ambiente.....	94
6	IMPLEMENTAÇÃO	95
6.1	Game Markup Language	95
6.2	Game Wrench v0.1 em Java Swing	100
6.2.1	O padrão Model-View-Controller e o Java Swing.....	101
6.2.2	Implementação do Yellow Helper / Orange Alert	101
6.2.3	Caso exemplo: Gerar código do enJine através da interface	102
7	RESULTADOS	104
8	CONCLUSÃO	109

APÊNDICE A – Relatório do Estudo de Usabilidade com Auxílio de Protótipo em Papel..	114
APÊNDICE B – GAML definido em XML Schema e exemplos	126
APÊNDICE C – Projeto de Interação para interface de Motor de Jogo Genérico.....	133
APÊNDICE D – Análise Comparativa através de Avaliação Heurística	151
APÊNDICE E – Questionários para Validação de Personas relacionadas a Motor de Jogo .	158
ANEXO A – Heurísticas de Usabilidade	167
GLOSSÁRIO.....	169
REFERÊNCIAS	172

1 INTRODUÇÃO

1.1 *Motivação e Objetivos*

Em 2002, o Laboratório de Tecnologias Interativas da Escola Politécnica da USP, Interlab*, iniciou um trabalho de pesquisa para verificar a viabilidade de utilização de um motor de jogo comercial para produção de jogos eletrônicos (*serious games*) para educação.

O projeto escolheu o motor **Unreal Engine** (usado em **Unreal Tournament**, **Deus Ex**, **America's Army**, **Rainbow Six** entre outros) para testes que consistiram no desenvolvimento de um protótipo de jogo *single-player* situado em um museu.

A partir das conclusões obtidas nesse projeto, resolveu-se criar um motor próprio – **enJine** (enJine, 2006)¹, desenvolvido levando-se em consideração necessidades específicas do ambiente acadêmico, sem as restrições funcionais e de prazos comuns em projetos comerciais.

Com esse intuito de criar um motor de jogo que seja mais acessível para esse ambiente acadêmico, é interessante que se apliquem as práticas de Interação Humano-Computador, através de um projeto de interação, criando-se um ambiente interativo para o acesso a suas funcionalidades. Esse trabalho tem a intenção de diminuir as dificuldades relacionadas à atividade de desenvolvimento de um jogo eletrônico que são encontradas na utilização de um motor, através da especificação de uma interface de alta usabilidade para esses softwares. Isto permitirá, por um lado, diminuir o tempo de aprendizado de tal ferramenta, e por outro, aumentar o espectro de usuários que poderão utilizar-se a contento do mesmo.

Mais detalhadamente, será desenvolvida a especificação de interface humano-computador para um motor de jogo, tendo como foco os usuários da área de

* Interlab - Laboratório de Tecnologias Interativas. Departamento de Computação e Sistemas Digitais, Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo.

educação. Essa especificação de interface será aplicada ao motor de jogo **enJine**, que tem como intenção a produção de jogos eletrônicos como atividade de aprendizagem, e que vem sendo aplicado a aulas de Computação Gráfica em cursos de graduação (TORI et al, 2006)².

A comunicação entre esses dois módulos de software (interface e motor) deverá ter baixo acoplamento, o que permitirá maior flexibilidade para a aplicação das técnicas aqui desenvolvidas a outros projetos e aplicações.

Os objetivos principais dessa dissertação são:

- Desenvolver uma especificação para ambiente interativo de alta usabilidade para que um grupo de usuários específico (alunos e professores da disciplina Computação Gráfica da EPUSP) possa acessar as funcionalidades de um motor de jogo, através da metodologia de projeto de interação;
- Aplicar esta interface ao motor de jogos **enJine**, desenvolvido pelo Interlab – Laboratório de Tecnologias Interativas.

Entre seus objetivos secundários temos:

- Atender aos requisitos para motor de jogo voltado ao meio acadêmico levantados durante os projetos do museu e **enJine e detalhados nas Seções 4.2.1 e 4.2.2**;
- Constituir um ambiente de desenvolvimento que permita aumentar o formalismo e previsibilidade nos projetos de jogos eletrônicos, enquanto se mantém flexível o suficiente para permitir o interfaceamento de outras *Application Programming Interfaces (APIs)* e de motores de terceiros;
- Diminuir o tempo de aprendizado e aumentar a eficiência de um desenvolvedor de jogos junto a um motor de jogos.

1.2 Estrutura

Esta seção descreve a estrutura dos próximos capítulos desta dissertação.

O capítulo 2 apresenta a pesquisa teórica e os conceitos relacionados que formam a base para o desenrolar do projeto. Essa parte da dissertação é de extrema importância, pois está relacionada com os objetivos de promover maior formalismo e evitar ambigüidades. É o ponto de partida para qualquer processo de desenvolvimento. Todas as definições adotadas, bem como anglicismos e termos

técnicos estão relacionados ao final da dissertação no Glossário. Descreve ainda pesquisa em forma de Projeto de Interação com um foco mais amplo, de entender o conceito do produto “motor de jogo” e o universo de usuários do mesmo.

O capítulo 3 apresenta a metodologia aplicada em busca dos objetivos desse trabalho, ou seja, as tarefas executadas como passos em direção a essas metas. Descreve como foram aplicados os métodos, encontrados na literatura e empregados na indústria, para obtenção de melhorias na interface para o caso específico de se criar a interface para o motor **enJine** sendo utilizado com um grupo de usuários específico: alunos da disciplina de graduação Computação Gráfica.

O capítulo 4 detalha primeiramente o estudo necessário para a construção de cenários que orientam a construção de um primeiro protótipo de baixa-tecnologia (prototipagem – construção de protótipos – em papel), suficiente para a condução de testes de usabilidade.

O capítulo 5 resume toda a especificação resultante do processo descrito anteriormente, em forma de uma lista de requisitos.

O capítulo 6 descreve o processo de criação da interface otimizada para o motor **enJine** utilizando-se da plataforma de software escolhida para isto, **Java Swing** e *Extensible Markup Language (XML)*.

O capítulo 7 apresenta os resultados obtidos e comenta as conclusões sobre o processo do projeto de interface.

Por fim, o capítulo 8 apresenta a conclusão e uma última etapa de avaliação, listando os assuntos associados que não foram tratados nesse trabalho ou por estarem fora do escopo dessa proposta ou por inviabilidade de execução dentro do prazo estipulado.

2 CONCEITUAÇÃO E DEFINIÇÃO DO PROBLEMA

Uma definição do conceito de jogo não é tarefa simples: exige interpretações multidisciplinares e é dificultada por questões culturais, pela falta de uma taxonomia consistente e por traduções conflitantes.

Apesar de tais dificuldades, neste capítulo, este problema será abordado, procurando-se uma definição satisfatória para este conceito e aplicando a mesma no intuito de responder questões frequentemente relacionadas com a natureza dos jogos, com exemplos de produtos-jogo de complexidade e tecnologia distintos.

2.1 Conceito de Jogo: Uma Definição

Os jogos acompanham os seres humanos constantemente ao longo de suas vidas, desde a sua origem. De fato, descobertas arqueológicas de brinquedos e objetos relacionados a jogos datam de antes de 2000aC (Figura 2-1 e Figura 2-2).

Figura 2-1 - Descobertas arqueológicas relacionadas a jogos.

Figura 2-2 - Jogo conhecido por Senat, precursor do Gamão, encontrado no Antigo Egito.

Em “Homo Ludens”, Huizinga (2004, cap.1, p.12)³ constata que o jogo:

Ornamenta a vida, ampliando-a, e nessa medida torna-se uma necessidade tanto para o indivíduo, como função vital, quanto para a sociedade, devido ao sentido que encerra, à sua significação, a seu valor expressivo, a suas associações espirituais e sociais, em resumo, como função cultural.

Apesar de lançar luz sobre a importância deste tema, este texto mistura em sua argumentação tanto esse conceito mais genérico de jogo, quanto suas manifestações mais específicas, como a diversão dos animais e crianças pequenas, os “jogos de adulto”, a música, o teatro, o culto religioso e o processo jurídico.

Isto é resultado em parte do fato de muitos idiomas, ao contrário do Português, não terem palavras distintas para descrever as atividades de jogar, apostar, brincar, interpretar e tocar. Como o jogo e o interesse pelo mesmo existem em todas as culturas (por serem características humanas), surgiram inúmeras traduções complexas e às vezes incorretas.

Um exemplo dessa problemática é apresentado em “Rules of Play” (SALEN; ZIMMERMAN, 2004)⁴, onde são comparadas sete definições de jogo (a saber, de Parlett, Abt, Huizinga, Caillois, Sutis, Crawford, Costikyan e Avedon/Sutton-Smith) e apresentada uma própria.

Adicionalmente, esse livro apresenta duas formas corriqueiras de entender o relacionamento dos conceitos de jogo e diversão/brincadeira como igualmente úteis:

1. Jogo (*game*) como subconjunto de Brincadeira (*play*)
2. Brincadeira como subconjunto de Jogo.

Ora, se ambas as formas forem igualmente válidas, chegar-se-á à conclusão contrária do que se deseja estabelecer de que jogo e brincadeira são o mesmo conceito. Mas como se verá adiante, trata-se de conceitos distintos, ainda que intrinsecamente relacionados.

Nesse estudo foi utilizado o modelo em que Walther (2003)⁵ propõe uma separação dos conceitos de realidade (o que não é diversão – *the other of play*), diversão (*playing*) e jogo (*gaming*), através de uma modelagem sistêmica. A Figura 2-3 ilustra essa separação.

Figura 2-3 - Modelo de Walther

A partir da realidade (ou normalidade) é feita uma primeira transgressão, restringindo-se espaço e tempo e chegando-se ao estado de diversão (ou fantasia, brincadeira). Ou seja, a brincadeira ocorre dentro de espaço e tempo delimitados e restritos. Walther define também como modo de diversão (*play-mode*) atividades que transpõem freqüentemente a barreira que separa esses dois estados.

O termo transgressão ajuda a ressaltar o fato de tratar-se de uma mudança na forma de pensar e encarar um objeto, região no espaço ou período de tempo. Alguns conceitos relacionados com o estado de diversão são: imersão, imaginação e virtualidade.

Nos jogos comerciais, as equipes artísticas (roteiro, artes gráficas, trilha sonora, realidade virtual) estão interessadas em garantir essa imersão, ou seja, manter o jogador dentro do estado de diversão.

Uma vez dentro do estado de diversão, faz-se uma transgressão de segunda ordem restringindo-se possibilidades e criando-se uma estrutura, chegando-se ao estado de jogo; ou seja, o jogo depende de o jogador familiarizar-se com as regras, de

modo a progredir em sua estrutura. O modo de jogo (*game-mode*) compreende atividades que oscilam entre os estados de diversão e de jogo.

Alguns conceitos relacionados com o modo de jogo são: jogabilidade, máquina de estados, projeto de fases, administração de recursos, regras, estratégias, objetivos e competição.

Nos jogos comerciais, as equipes de programação (motor de jogos, *scripts*), de projeto de fases e de jogo (*level designers*, *game designers*) estão preocupadas em manter, para o jogador, a estrutura dentro do estado de jogo.

Com base nas informações apresentadas acima, serão adotadas as seguintes definições para este trabalho:

Diversão ou **estado de diversão**: uma transgressão sobre a realidade, libertando determinado objeto, pessoa ou lugar das suas regras convencionais por um determinado período de tempo.

Jogo ou **estado de jogo**: uma transgressão sobre a diversão, acrescentando-se a essa elementos não-convencionais de estrutura e regras.

Quanto à programação de jogos eletrônicos, as dificuldades existem porque não só os conceitos de diversão e de jogo estão mais próximos do ser humano do que dos sistemas digitais, como fazem parte da própria essência humana. Ou seja, como seres humanos e sistemas digitais são entidades muito distintas entre si e o jogo é uma atividade exclusiva de seres humanos, existe uma complexidade ainda maior para transformar esses conceitos humanos em modelos computacionais. Talvez isso explique a estranha ausência desses elementos em motores de jogo atuais. Em compensação, os sistemas digitais são muito mais indicados para obedecer, calcular e seguir regras e criar modelos complexos e estruturados, ambos essenciais para permitir o estado de jogo.

2.2 Aplicações do Modelo

Esse modelo e definições são, primeiramente, essenciais para o desenvolvimento adequado de um motor de jogo, ou seja, é impossível pensar numa plataforma de desenvolvimento de jogos sem compreender claramente o que é jogo.

Também, servem de ferramenta para entender questões complexas recorrentes na literatura relacionada a jogos.

2.2.1 Exemplos: Características e Elementos que Permitem as Transgressões

Analisando os seguintes exemplos será mais fácil entender o modelo proposto: o jogo de xadrez, o jogo de futebol e um jogo de computador *First Person Shooter (FPS)*.

2.2.1.1 Primeira transgressão

Consiste em libertar objetos, pessoas, espaços e períodos de tempo específicos das características do “real”.

No caso do **xadrez**, separa-se um espaço específico (o tabuleiro 8x8), e objetos específicos (as peças) que serão destinados ao jogo. Durante esta transgressão se diz “Isso é um tabuleiro de xadrez. Essa peça é um bispo etc.”.

O tempo específico acordado entre os jogadores é o período compreendido entre o “início do jogo” até o momento denominado “fim de jogo”. Parece simples, mas, se por acaso, entre esses dois momentos, um dos jogadores resolver não dedicar seu raciocínio ao “jogo”, estará quebrando a primeira transgressão, voltando à “realidade”, e impedindo o prosseguimento do jogo (interrompendo o tempo de jogo e causando uma quebra da etiqueta do jogo).

No caso do **futebol**, separa-se um espaço específico (a quadra ou campo – superfície delimitada pelas linhas de fundo e laterais; os “gols” – superfícies delimitadas pelas traves e a linha de fundo), pessoas (os jogadores de linha, os

goleiros, os árbitros, os técnicos, os reservas), e objetos específicos (a bola, o som do apito, os pés dos jogadores de linha). O tempo de jogo é definido como: duas etapas de aproximadamente 45 minutos, com um intervalo de 15 minutos entre elas, iniciadas e terminadas pelo apito do árbitro. É o momento em que se separam os times, se diz “você é do meu time, você começa no gol, você fica no banco, etc.”.

No caso do **FPS**, cria-se todo um “ambiente virtual” (ou seja, sintetizado através de parâmetros, e diferente do real). Esse ambiente virtual é composto de “mapas” - o equivalente dos tabuleiros - e objetos virtuais (como obstáculos, portas, chaves, armas, projéteis e NPCs – *non-player characters*).

O ambiente virtual será visualizado através de um “*viewpoint*” mapeado em uma janela ou tela de computador. Esse *viewpoint* poderá ser controlado através dos dispositivos de interface do computador (teclado, *mouse*, *joystick*).

Cria-se um objeto chamado “avatar” que representará o jogador nesse mundo.

O tempo de jogo será determinado pela “condição de fim” de jogo (normalmente, o valor da energia do avatar do jogador ou de todos os avatares de um time chegando a zero).

Elementos característicos:

A partir destes exemplos, pode-se listar alguns elementos que permitem a primeira transgressão a um “produto-jogo”:

- Participantes – Essenciais, pois são os responsáveis pela transgressão para a diversão. Por enquanto – transgressão de primeira ordem – não precisam ser necessariamente humanos adultos;
- Espaço de jogo – Campo, Tabuleiro, Mapa;
- Interação e Atuadores – Objetos utilizados pelo jogador como forma de modificar o jogo: peças, partes do corpo, objetos virtuais;
- Tempo de jogo – Para a primeira transgressão, basta diferenciar entre “agora estamos jogando” e “agora não estamos mais”, mas nos exemplos acima vimos que a condição de fim (uma das regras) permite especificar melhor esse período de tempo.

2.2.1.2 Segunda transgressão

Consiste em acrescentar estrutura e regras de jogo a esses objetos, pessoas, espaços e períodos de tempo.

A estrutura de um jogo pode incluir definições, a configuração inicial e a mecânica do jogo. Definições têm forte ligação com a primeira transgressão, mas são acrescentadas de formalismo durante a segunda transgressão. A mecânica do jogo inclui regras como a ordem dos jogadores, os turnos, as ações, os objetivos (*Victory Points* - VPs) e a condição de fim.

Sistemas computacionais são ferramentas muito boas para implementação de jogos por sua capacidade de processamento de algoritmos e modelos de alto nível como máquinas de estados.

Ao analisar o caso do **xadrez** verifica-se:

Exemplos de definições e termos específicos:

- Jogada *en passant*;
- Jogada *roque*;
- Captura;
- Cheque;
- Cheque-mate etc.

A configuração inicial:

- Cada peça possui seu lugar inicial no tabuleiro,
- As brancas têm a “vez” de jogar.

A mecânica do jogo:

- Movimentação das peças;
- Formas de “captura”;
- Os jogadores se alternam movimentando uma peça (ação) por vez;
- Limite de tempo por jogada, etc.

Os objetivos dos dois jogadores são iguais:

- Encurralar a peça “rei” do adversário.

As condições de fim são as seguintes:

- Se em uma jogada configurar um “cheque-mate” o jogo termina com um vencedor;
- Se em uma jogada sobrarem só os “reis” o jogo termina com empate;
- Se um mesmo movimento repetir-se por três jogadas consecutivas, o jogo termina com empate;

Ao analisar o caso do **futebol** verifica-se:

Exemplos de definições e termos específicos:

- Campo;
- Impedimento;
- Juiz;
- Gol;
- Goleiro;
- Linhas do campo, etc.

A configuração inicial:

- Decide-se qual o campo inicial;
- Decide-se quem dá a saída com a bola;
- Cada jogador inicia a partida em seu respectivo campo;
- O juiz marca o início do tempo de jogo com um apito.

A mecânica do jogo:

- Os jogadores de linha levam a bola em direção ao gol adversário;

- Se a bola entrar em um gol, marca-se um *VP* para o time adversário e reinicia-se o jogo do meio do campo;
- Se um jogador deixar a bola sair do campo, a posse de bola passa ao time adversário, a partir da linha do campo onde saiu a bola etc;

Os objetivos dos dois times são iguais:

- Maior número de *VPs*.

A condição de fim é a seguinte:

- Ao final da partida (apito do juiz), o vencedor é quem tem maior número de *VPs*; Em caso de empate, não há vencedor;

Ao analisar o caso do **FPS** verifica-se:

Exemplos de definições e termos específicos:

- *frag*;
- *camper/sniper*;
- *power-up*;
- *bots* etc.

A configuração inicial:

- Escolha do mapa;
- Escolha dos times;
- Criação de *bots* para equilibrar os times;
- Posição inicial dos jogadores (definida no mapa).

A mecânica do jogo:

- Jogadores devem eliminar jogadores adversários reduzindo sua energia a zero ao acertá-los com projéteis;
- Jogadores podem pegar armas, munição e *power-ups* disponíveis no cenário ou derrubados por jogadores eliminados;

Os objetivos dos dois times são iguais:

- Maior número de *frags*.

A condição de fim pode ser:

- Por tempo de jogo;
- Por número de *frags*;
- Por obtenção de objetivos específicos (*capture the flag, dominance* etc.)

Elementos característicos

A partir desses exemplos, pode-se listar alguns elementos que permitem a segunda transgressão a um jogo:

- Definição formal de termos específicos;
- Configuração inicial;
- Mecânica do jogo;
- Turnos;
- Ações;
- Objetivos;
- Condição de fim.

2.2.2 Elementos de Jogo e Outros Elementos em Produtos-Jogo

Como visto acima, os elementos que cuidam das transgressões de primeira e segunda ordem em um produto podem ser considerados como elementos de jogo desse produto. Analisando-se os produtos a que se chama casualmente de “jogo” *, conclui-se que raramente esses terão apenas elementos de jogo.

Fazendo essa análise, de posse da definição de jogo, descobrir-se-á que:

- Alguns não realizam a segunda transgressão e portanto não são jogos, e seriam mais bem definidos como brinquedos ou simulações;
- Outros têm muitos elementos importantes além de jogo, como “tema”, “arte”, “esporte”, “competição”, “violência”, “aposta”, “sorte”, “destreza” que podem, inclusive, serem muito mais destacados do que os elementos de jogo (de primeira e segunda transgressões).

Quanto a esses outros elementos de um produto-jogo, cabe ao projetista de jogo (*game designer*) descobrir como dissociá-los (criando “jogos abstratos”), ou ressaltá-los, fortalecendo as ligações entre eles e o jogo, através das regras.

A Tabela 2.1 apresenta alguns exemplos de balanceamento desses elementos.

Tabela 2.1 - Priorização dada pelas regras do jogo a outros elementos de produtos-jogo.

	Tema ou Estória	CN-V†	Dedicação	Esporte	Competição	Violência	Aposta	Destreza	Sorte
Boxe	Baixa	Média	Alta	Alta	Alta	Alta	Baixa	Alta	Baixa
FPS	Alta	Baixa	Baixa	Baixa	Alta	Alta (embora indireta)	Baixa	Alta	Baixa
Futebol	Baixa	Média	Média	Alta	Alta	Baixa	Baixa	Alta	Baixa
Mímica	Média	Alta	Média	Baixa	Baixa	Baixa	Baixa	Alta	Baixa
Par ou Ímpar	Baixa	Média	Baixa	Baixa	Alta	Baixa	Alta	Baixa	Alta
Pôquer	Baixa	Alta	Alta	Baixa	Alta	Baixa	Alta	Baixa	Alta
RPG	Alta	Média	Média	Baixa	Baixa	Baixa	Baixa	Baixa	Média
Xadrez	Média	Baixa	Alta	Baixa	Alta	Baixa	Média (Qual peça se “arrisca” a cada jogada)	Baixa	Baixa

* A partir de agora serão utilizados os termos **jogo** para se referir ao conceito definido no item 2.1, e **produto-jogo** para se referir a produtos comerciais ou atividades que possuem elementos de diversão ou jogo, e são comercializados como “jogos”.

† Comunicação Não-Verbal

Importante notar como as regras do jogo podem desfavorecer ou ressaltar qualquer um desses elementos; Isto constitui uma ferramenta poderosa para o projetista, principalmente se esse estiver buscando utilizá-lo como forma de aprendizado ou treinamento.

Outro detalhe a notar, é que a ligação entre o jogador e um desses elementos é ressaltada pela sua importância na mecânica do jogo: para o jogador, a importância de um desses elementos é tanto maior quanto mais diretamente estiver relacionada com suas regras.

2.2.3 Elementos de Jogo em Outras Mídias

Elementos de jogo podem ser encontrados em diversos tipos de mídia. Serão analisadas as relações entre os produtos-jogo e o cinema.

A interatividade do cinema

Se a indústria do cinema tem responsabilidade apenas no “caminho de ida” da comunicação, ou seja, de passar a mensagem do autor para os espectadores, isso não exclui completamente a chance do retorno.

Pode-se considerar mais interessante um filme que, de tão mal feito, gere muitos comentários a seu respeito, do que um filme mediano, efêmero e sem repercussão alguma.

Apesar de o cinema não ter um retorno tão rápido quanto mídias mais folhetinescas como os programas de televisão ou um **blog**, o retorno é feito através da bilheteria, dos comentários que o filme gera e da venda de mercadorias relacionadas, incluindo o próprio filme para exibição doméstica. Nesse sentido, a crítica de cinema torna-se um dos melhores “caminhos de volta”, completando o ciclo, trazendo a resposta do espectador aos produtores e autores, e trazendo elementos de interatividade ao cinema.

Não é, portanto, como poderia se supor *a priori*, apenas a interatividade, ou a sua falta, que distingue o produto-cinema do produto-jogo.

Jogo no cinema

Na série de filmes *Star Wars* (LUCAS, 1977)⁶, quando o expectador lê a frase “*A long time ago in a galaxy far, far away...*”^{*} e é surpreendido pelos acordes da trilha sonora, a transgressão de primeira ordem é induzida com maestria.

A partir daquele instante, o expectador foi transportado da realidade para um outro mundo, onde as possibilidades são infinitas.

Quando a narrativa prossegue, e elementos da estrutura desse outro mundo começam a ficar mais claros (existem vários planetas com seres vivos, existem outros seres vivos além dos humanos, existe um elemento chamado “força” que controla esse mundo etc.), então a transgressão de segunda ordem começa a tomar forma, e o expectador começa a sair do “modo de diversão” e entrar no “modo de jogo”.

Portanto, pode-se inferir que, assim como os elementos de som já estavam presentes no chamado cinema mudo (CHION, 1994)⁷, elementos de jogo já podem estar presentes nessa (ou em qualquer outra) obra ficcional, mesmo que uma centena de produtos-jogo com a marca “*Star Wars*” não fossem criados.

Conclui-se que, com base na definição de jogo apresentada, mais do que a interatividade, um conjunto de regras que define a estrutura de um mundo virtual é o que separa a diversão do jogo.

Apesar de ambos, produto-cinema e produto-jogo, conterem esses elementos, uma diferença no enfoque entre as duas transgressões - se a intenção é manter o expectador no modo de diversão ou no modo de jogo - diferencia um do outro.

^{*} “Há muito tempo, numa galáxia muito, muito distante...”

2.2.4 Narrativa e Ludologia

Existe uma disputa no meio acadêmico entre os que defendem o estudo de jogos a partir de suas características narrativas (*storytelling*) ou lúdicas (*ludology*).

Segundo o modelo adotado, essa discussão torna-se simples: a narrativa de um produto-jogo é um elemento que ajuda a primeira transgressão, necessária para possibilitar a diversão, mas não-suficiente para o estado de jogo.

Mas a discussão é interessante e muito fértil. Por exemplo, Juul (2005)⁸ analisa as semelhanças e diferenças das duas formas de comunicação digital (*storytelling* e *games*). O melhor argumento que apresenta para diferenciar narrativa e jogo é:

- “O tempo acontece de forma diferente nos jogos e nas narrativas”

Essas mídias contêm três “linhas do tempo”: tempo da história (*story time*), tempo da narração (*discourse time*) e tempo da leitura (*reading/viewing time*).

Aí está um grande diferencial entre o que é (contado+assistido), para o que é (programado+jogado). O argumento é que em um jogo existe sincronismo entre tempo de narração e tempo de leitura causado pela interatividade (o tempo da narração é sempre o presente).

2.2.5 Jogo, Ritual e Mídias Lineares

A vida social do ser humano apresenta diversas atividades regidas por regras. Da mesma forma, existem diversas outras formas de comunicação e entretenimento (mídias). O relacionamento entre o Jogo e Rito é abordado tanto por Huizinga (1938)³, quanto por Lévi-Strauss (1966)⁹.

Fica claro que nem tudo pode ser considerado jogo, e que para fazer essa distinção, basta aplicar o modelo de jogo. Se forem identificados elementos das transgressões de primeira e segunda ordem, confirma-se o jogo.

Mas existe uma diferenciação temporal entre esses conceitos:

- Uma mídia linear (como livro, revista, cinema etc.) sempre começa, se desenvolve e termina da mesma forma, independente do leitor ou de sua intervenção.
- Jogo começa da mesma forma, se desenvolve diferentemente segundo regras fortes através da intervenção do jogador e termina de forma imprevista. Apesar de uma instância de um jogo (partida) poder terminar em derrota (ou acabar antes do fim da estória), o jogo pode ser reiniciado quantas vezes forem desejadas, na intenção de obter-se a vitória (ou chegar-se ao fim da estória).
- Rito começa da mesma forma, se desenvolve diferentemente segundo regras fortes através da intervenção dos participantes, e termina da forma prevista e desejada (apesar de existirem outras inúmeras possibilidades de finais previstas nas regras).

2.2.6 Jogo e Arte

Outra preocupação recorrente no meio acadêmico é definir se jogos podem ser considerados como uma forma de arte.

Sem dúvida, e cada vez mais, produtos-jogo utilizam-se de uma grande produção dos profissionais do meio artístico.

Mas, para se declarar que o jogo seria uma forma artística, seriam necessárias definições claras desses dois conceitos – jogo e arte. Apesar de se estar a meio caminho através da definição de jogo, uma definição de arte (também não-trivial) não faz parte do escopo deste trabalho.

Apesar disso, se arte for encarada tanto como a dedicação a uma técnica (como nas artes marciais, artesanais, etc.), ou como uma forma de comunicação não-verbal entre artista e expectadores, pode-se assumir que o jogo, pelo menos, pode colaborar ou ser utilizado para esses dois fins.

2.2.7 Simulação vs. Emulação

Qualquer produção multimídia pode servir-se das técnicas de simulação e de emulação em sua construção.

A simulação é obtida através de um processo de síntese, onde parâmetros são passados pelo autor ao sistema, e este reproduz resultados a partir de um modelo e regras preestabelecidas.

Por outro lado, a emulação é obtida através de um processo de aquisição (*sampling*) de uma apresentação, primeiramente capturando-se informações de forma análoga ao mundo real (sinais analógicos). Depois dessa captura, esse sinal pode ser digitalizado (permitindo processamento e transmissão de forma mais eficiente e resistente a ruídos) e comprimido (permitindo transmissão e armazenagem mais eficiente). Para sua reprodução, os sistemas precisam apenas conhecer o protocolo de codificação e suprir as necessidades não triviais de processamento e fluxo de dados.

Dadas as suas características, os sistemas simulados precisam de muita tecnologia para atingir níveis de realismo convincentes (sua percepção pelo ser humano é normalmente de algo artificial); são dependentes do equipamento onde são reproduzidos, mas permitem modificações e experimentação, inclusive durante o tempo de execução.

Como permitem essa interatividade, as simulações costumam ser confundidas, ou automaticamente relacionadas com jogos.

Ao contrário, o equilíbrio entre simulação e emulação é uma tarefa muito importante para o projetista de um jogo. Um bom jogo independe do grau de realismo que sua simulação alcança, mas muitas vezes elementos emulados permitem uma melhor primeira transgressão por se aproximarem mais do real.

O modelo de jogo mostra que sempre que alguém está em estado de jogo, necessariamente encontra-se afastado do real, portanto, a aproximação da realidade pode acabar inclusive tirando o jogador do estado de jogo. Esse fato está

relacionado com o conceito da suspensão da descrença (*willing suspension of disbelief*) (COLERIDGE, 1817, cap.14 p.314)¹⁰ onde o expectador, para fins de entretenimento, aceita como real algo que sabe ser imaginário.

Esse conceito acaba por implicar que, quanto mais uma apresentação multimídia se aproxima do real, mais o expectador estará propenso a sair do modo de diversão (retornando ao real) por pequenas imperfeições no sistema – a exigência de realismo aumenta. Por exemplo, os jogadores atuais de títulos como **Outlaw Tennis** ou **Wii Sports** estão muito mais propensos a questionar a simulação física ou a inteligência artificial dos *NPCs* do jogo, do que os jogadores contemporâneos de **Pong** em aceitar que retângulos se movendo na tela da televisão representavam uma bola e raquetes.

Figura 2-4 - Screenshots de produtos-jogo com o tema Tênis. Esq.: Pong (1972) da Atari para plataforma arcade. Centro: Outlaw Tennis (2005) da Hypnotix para plataformas Xbox e Playstation2. Dir.: Wii Sports (2006) da Nintendo para plataforma Wii.

Esses aspectos de equilíbrio entre emulação e simulação e do contexto de diversão também diferenciam os jogos eletrônicos da simulação científica, onde a precisão e confiabilidade são sempre mais importantes que desempenho, interatividade e satisfação subjetiva do expectador.

2.2.8 Sorte, Caos e Aleatoriedade

Como visto na Tabela 2.1, as regras dos produtos-jogo *Role-Playing Game (RPG)*, par ou ímpar e pôquer priorizam o elemento “sorte” mais do que os outros exemplos.

O excesso de prioridade ao elemento sorte costuma ser visto como ponto fraco em um jogo, mas seu maior ou menor uso está a cargo do *game designer* no momento de criação e depende da preferência do jogador no momento da escolha do jogo.

Essa sorte pode vir de dois fatores: caos e aleatoriedade.

Aleatoriedade em um produto-jogo é o fator que impede o jogador de identificar a próxima resposta deste jogo em um determinado momento, e é introduzida de diversas formas como, por exemplo, lançamento de dados, embaralhamento de cartas, habilidade dos adversários.

Caos em um produto-jogo é o fator que impede ao jogador identificar, a partir da situação atual e das ações possíveis, qual será a situação futura em um determinado número de rodadas. Quanto mais caótico o jogo, mais difícil será desenvolver uma estratégia para esse jogo.

2.3 Desenvolvimento de Jogos Eletrônicos

2.3.1 Complexidade Crescente

Blow (2004)¹¹, em seu artigo “*Game development: harder than you think*”^{*}, avalia como a complexidade no desenvolvimento de um “jogo eletrônico” evoluiu drasticamente para chegar ao nível atual. O produto conhecido por jogo eletrônico, que há apenas uma década poderia ser inteiramente concebido, desenvolvido e comercializado por apenas uma pessoa ou equipe de programadores, hoje conta com orçamentos milionários e equipes de dezenas de pessoas, das mais diversas áreas de competência.

Como em qualquer software que atinge tal nível de complexidade, existe a necessidade de reaproveitar-se parte do trabalho realizado em projetos futuros. Para

^{*} Desenvolvimento de Jogos: mais difícil do que você pode imaginar

isso cria-se uma separação entre o que é específico em um projeto e o que pode ser reutilizado em outros.

2.3.2 O Motor de Jogo *

Adotou-se chamar de motor de jogo a parte do software que cria a infra-estrutura na qual um jogo é desenvolvido e que pode ser reaproveitada para criar jogos similares.

Isto é do interesse tanto da empresa de desenvolvimento, que terá mais agilidade a criação de novos produtos e flexibilidade na alocação (e eventual substituição) de suas equipes, quanto de outras empresas, que podem adquirir tais motores de jogo como soluções já testadas e consolidadas.

Tori, R. et al (2004)¹² cita como alguns dos possíveis componentes de um motor de jogo: renderização, inteligência artificial, comunicação em rede, entrada de usuário, simulação física (com detecção e tratamento de colisão) e multimídia (com ambientação de som e música).

Nos jogos tradicionais (não-eletrônicos) é comum encontrar “sistemas de jogo” que não têm função isolados, mas que permitem criar diversos produtos-jogo apenas variando-se as regras e alguns outros elementos. Por exemplo, pode-se ter inúmeros jogos que utilizem o mesmo tabuleiro 8x8 e peças do xadrez, inúmeros outros que utilizam os baralhos de 52 cartas, as peças do dominó duplo-6, etc. Um motor de jogos pode ser visto de forma análoga a esses sistemas de jogos, com a diferença de ser constituído por um conjunto de módulos de *software* em uma plataforma computacional.

Para este trabalho, e de acordo com o projeto de interação proposto, pode-se definir:

* A expressão *game engine* pode ser traduzida para “motor de jogo” ou “máquina de jogo”. Para essa dissertação será adotada a primeira alternativa por ressaltar o aspecto de um *game engine* ser parte de um software maior, o jogo eletrônico, e de não poder operar sozinho.

Motor de jogo: um conjunto de módulos de um jogo eletrônico que podem ser reaproveitados e compartilhados por outros *softwares*.

Com base no exposto anteriormente, uma ferramenta baseada em motor de jogo deverá, dentre outras preocupações, garantir as transgressões de primeira e segunda ordem, e deverá apresentar ao usuário pelo menos os seguintes elementos identificados acima:

- Jogadores;
- Espaço de jogo;
- Interação e Atuadores;
- Tempo de jogo;
- Definições e termos específicos;
- Configuração Inicial;
- Mecânica do jogo;
- Turnos;
- Ações;
- Objetivos;
- Condição de fim.

2.3.3 Motores Comerciais

O uso de motores comerciais é comum no caso de uma empresa que tenha licença de uso de material artístico (de filmes, seriados, quadrinhos etc.) e deseja utilizá-la na criação de um jogo com o mínimo de desenvolvimento envolvido. Apesar disso, de posse do motor, está apta a realizar modificações mais estruturais, permitindo a criação de jogos bastante diferentes do original.

Embora esse reaproveitamento de código seja desejado e até tenha sido explorado comercialmente, a atividade das empresas desenvolvedoras dos jogos de última geração normalmente é atrelada a exigências de orçamentos e prazos, o que muitas vezes impede que a componentização do *software* seja realmente tratada com cuidado (NAKAMURA et al, 2003)¹³.

Como resultado, a eficaz utilização de tais motores de jogo por um usuário não familiarizado com o desenvolvimento original do jogo exige um estudo preliminar de

toda a estrutura dessas bibliotecas antes de esse usuário tornar-se eficiente na utilização das mesmas.

Tal situação pode ser grave a ponto de um programador demorar mais tempo para aprender como se utiliza um motor de jogo a contento, do que levaria se ele fosse começar o desenvolvimento do princípio. Ou seja, a curva de aprendizado de um motor de jogo pode ser tão lenta, que acaba contradizendo a própria razão de existir de tais ferramentas.

Quanto ao ambiente de desenvolvimento e sua interface, pouca atenção é dada a fatores como facilidade de uso, prevenção de erros e ajuda sincronizada.

2.3.4 Motores Não-Comerciais

Existem também iniciativas de criação de motores de jogo por parte da comunidade acadêmica ou das comunidades de *software* livre. Tais iniciativas normalmente têm prazos de entrega menos rígidos, o que pode permitir maior dedicação à padronização, estrutura e documentação de seus motores. A exploração de novas funcionalidades e um grande cuidado com otimização também são características desses projetos.

Em contrapartida, é comum existir menos incentivo financeiro e grande oscilação de responsabilidades dentro da equipe de desenvolvimento. A divulgação, controle de versões e um suporte ágil são aspectos críticos às vezes negligenciados por tais ferramentas. Nesse caso, a interface de desenvolvimento costuma ser até mais hermética, uma vez que está sendo vista dentro de um universo de especialistas.

2.3.5 Modificações de Jogos

Uma tendência que surge com a maioria dos jogos recentes é o reaproveitamento de um jogo na forma de modificações, ou “*mod-games*”.

Por não exigir muita habilidade de programação, tal técnica é a preferida por usuários iniciantes ou de colaboradores de áreas não-técnicas, criando algumas vezes produtos mais interessantes e criativos do que os originais.

Existem diversas formas de alterar um jogo comercial, dependendo do nível de intervenção permitido pelos desenvolvedores e da compatibilidade e facilidade do uso das ferramentas fornecidas. Basicamente, o que caracteriza um *mod-game* como tal é exatamente QUEM irá realizar tais modificações. Além disso, em um *mod-game* é mantido um vínculo com o jogo original, ou seja, pouca modificação é feita na estrutura e mecânica do jogo.

Diferentemente de motores de jogo, onde as modificações são feitas por empresas de desenvolvimento interessadas em criar um jogo a partir de outro jogo já estabelecido, em geral, no caso de *mod-games*, as mudanças são feitas pelos usuários finais (ou seja, jogadores), que procuram aumentar sua experiência de jogo com customizações e alterações na dinâmica do jogo. Fica claro que isto configura um público bem mais heterogêneo e com um processo menos elaborado ou profissional.

Dessa forma, o que costumeiramente acaba ocorrendo no universo dos *mod-games* é apenas uma atuação na primeira transgressão, nos quesitos de imersão e fantasia, no estado de diversão. Alterações mais profundas, na transgressão que caracteriza o estado de jogo (sua estrutura e suas regras), exigem ferramentas fornecidas pelo estúdio de jogo, e exclusivas para usuários com conhecimentos técnicos avançados.

Para essa dissertação, será adotada a seguinte definição para *mod-games*:

Mod-Game: Jogo eletrônico criado a partir de produto-jogo comercial completo, utilizando-se de ferramentas fornecidas pelo estúdio do jogo.

Um resultado esperado com o projeto de interação desenvolvido para essa dissertação é aproximar o uso de um motor de jogo, normalmente restrito a programadores avançados, do grupo de usuários típicos de ferramentas de modificação de jogos, mais heterogêneo e tecnicamente menos especializado.

A forma de se atingir esse resultado é otimizando-se a interface de tais ferramentas.

2.4 Interação Humano-Computador

Interação Humano-Computador é uma área multidisciplinar de estudo. Assim como as disciplinas humanas com as quais se relaciona, utiliza-se de teorias de alto nível, princípios e diretrizes (SHNEIDERMAN; PLAISANT, 2005)¹⁴ para fins de aplicação e pesquisa.

Teorias de alto nível buscam a aplicação de um modelo que possa representar bem o objeto de estudo. Quanto melhor o modelo, mais aspectos do objeto de estudo podem ser inferidos através dele.

Ao contrário das teorias de alto nível, que partem da abstração, os princípios são uma compilação de conselhos provenientes da experiência de profissionais e estudiosos da área, que atuaram em muitos projetos com diversos níveis de complexidade.

Diretrizes (boas-práticas ou *guidelines*) são listas de recomendações voltadas à implementação de funcionalidades bastante específicas. Por exemplo, diretrizes para elaboração de menus podem ajudar profissionais que estão construindo um sistema baseado nesse estilo interativo pela primeira vez a não cometer erros comuns nessa atividade.

2.4.1 Modelo de Comunicação

Para explicar o escopo da Interação Humano-Computador, pode-se utilizar um modelo de comunicação. Para essa dissertação, será utilizado o modelo de Shannon (1949)¹⁵.

Segundo esse modelo, existem duas entidades distintas: o emissor e o receptor. Uma mensagem (informação) é transmitida do emissor para o receptor através de um meio de comunicação (padrão de mídia) e um canal (dependente de tecnologia).

Partindo desse modelo pode-se definir Interação Humano-Computador (IHC):

Interação Humano-Computador: comunicação bidirecional (ou seja, as entidades alternam-se nos papéis de emissor e receptor) entre duas entidades distintas: um ser humano e um sistema digital.

Vê-se, portanto que o estudo da Interação Humano-Computador não pode limitar-se à interface (ou seja, o canal de comunicação), mas engloba aspectos relacionados às duas entidades envolvidas, (que são bastante distintas entre si), e à maneira como recebem, processam e enviam informações. Esse é o principal motivo da multidisciplinaridade do assunto.

2.4.2 Humanos e Computadores

Das duas entidades em questão, ironicamente, a de que se tem mais informações e controle é o sistema digital. Afinal, uma vez que é projetado pelo ser humano, pode ter seu funcionamento e operação completamente entendidos.

Historicamente, o relacionamento com a tecnologia segue um padrão quanto à importância e o foco das pesquisas: quando a tecnologia é novidade, poucas pessoas estão capacitadas a utilizá-la. O foco está na tecnologia, em entendê-la e

dominá-la. As aplicações podem ainda nem estar definidas, a ciência avança e os investimentos são consideráveis.

Conforme as aplicações da tecnologia surgem, há intenção de aproximá-la de usuários comuns, cada vez menos capacitados. O foco migra para o ser humano, a sociedade avança e procura-se reduzir os custos e com isso democratizar tal tecnologia.

Nesse processo, a relevância do estudo de interação humano-computador aumenta.

Um equívoco usualmente cometido no meio desse caminho é pensar que a melhor forma de se aumentar a familiaridade das pessoas com os sistemas computacionais seria simular habilidades humanas em tais sistemas. Essa antropomorfização dos sistemas computacionais esteve bastante presente nas obras artísticas das décadas de 1970-80, época em que surgiram os computadores pessoais, videogames e todo tipo de dispositivo eletrônico de consumo. Foi a época em que se queria ter computadores mais “amigáveis” (*user-friendly*). Se o usuário estranha um sistema digital, procura-se deixar o sistema digital com a cara do ser humano que o utiliza. Aparece um computador assassino em “2001 – Uma Odisséia no Espaço” (KUBRICK; CLARK, 1968)¹⁶, robôs cômicos e com personalidades fortes em “Guerra nas Estrelas” (LUCAS, 1977)⁶, e um carro temperamental em “Supermáquina” (LARSON, 1982)¹⁷. Esses são só alguns exemplos de uma idéia muito freqüente nas obras de ficção científica dessa época. Uma crítica bem-humorada a esse conceito é feita, contemporaneamente, pelo robô Marvin da série “O Guia do Mochileiro das Galáxias” (ADAMS, 2005)¹⁸, possuidor de todo o conhecimento do universo, mas, ao mesmo tempo e ironicamente, de personalidade melancólica.

Esses exemplos mostram que, por um lado, esse conceito induz a sistemas de pouca serventia. Sistemas digitais e humanos são entidades bastante distintas entre si. Possuem qualidades e restrições características – ver Tabela 2.2. Tentar transformar um sistema digital em algo mais humano significa desprezar as qualidades características desse e transformá-lo em, no máximo, uma imitação limitada do outro. E, se a parte mais complexa da comunicação entre humano e computador é justamente o ser humano, dificilmente se obterá uma melhora transformando o computador em mais humano. Seguindo esse raciocínio, sistemas

digitais são ferramentas para os seres humanos e, portanto devem ser tratados como tais e não como amigos, companheiros, heróis ou rivais.

Por outro lado, autores como Reeves e Nass (2003)¹⁹ afirmam que é impossível tratar computadores e outros sistemas digitais exclusivamente como ferramentas, porque a forma com que o ser humano comunica-se com uma mídia é transformando-a em algo real em seu imaginário (modelo mental).

Tabela 2.2 - Comparação entre as capacidades de humanos e sistemas digitais.

Vantagens de seres humanos	Vantagens de sistemas computacionais
Captar estímulos sutis	Captar estímulos fora do alcance do ser humano
Filtrar estímulos em ambientes ruidosos	Contar ou medir quantidades físicas
Reconhecer de padrões em situações variadas	Armazenar quantidades grandes de informação
Captar eventos incomuns ou imprevistos	Monitorar eventos, principalmente os não-freqüentes
Lembrar princípios e estratégias	Responder rapidamente aos estímulos
Basear-se na experiência e adaptar as decisões conforme a situação	Lembrar quantidades detalhadas de informação
Selecionar alternativas em caso de falha na primeira estratégia adotada	Processar dados quantitativos de forma pré-especificada
Generalizar a partir de observação	Utilizar princípios gerais
Agir em emergências e situações novas	Atuar de forma repetitiva
Aplicar princípios para a resolução de problemas variados	Executar força física extrema
Fazer avaliação subjetiva	Executar muitas funções ao mesmo tempo
Desenvolver novas soluções	Manter operação durante excesso de carga
Priorizar tarefas	Manter a performance por longo tempo
Adaptar-se fisicamente a mudanças de situação	
(Sanders and McCormick, 1993) apud (SHNEIDERMAN; PLAISANT, 2005, pág. 79, Box 2.3, tradução nossa)¹⁴	

A questão é que computadores e sistemas digitais atualmente, e cada vez mais, acumulam funcionalidades. Ora servem como ferramentas para criação (como no uso de um editor de textos), ora como sistemas de missão-crítica (como no controle de sistemas de energia), ora como meio de informação (como no uso de um navegador de internet), ora como meio de comunicação (como no uso de um gerenciador de mensagens instantâneas) e ainda ora como entretenimento (como no uso de um reproduutor de mídia).

Essas diferentes funcionalidades indicam uma maior necessidade de cada aplicação específica com relação aos quantificadores de usabilidade (Seção 2.4.4). Cabe ao projetista da interface saber maximizá-los quando possível e saber como dar prioridades nas situações de compromisso.

Uma questão semelhante é quanto ao uso de metáforas (não apenas as que antropomorfizam os sistemas). A princípio, metáforas facilitam o processo de aprendizado de algo estranho ao usuário sugerindo analogia com algo que esse usuário já conheça. Boas metáforas, aquelas que cumprem bem esse papel, devem ser exploradas e utilizadas sempre que possível. Mas o uso de metáforas apresenta problemas e não é indispensável para uma boa interação.

Os problemas com metáforas são (COOPER; REIMANN, 2003)²⁰:

- Metáforas não são infinitamente extensíveis. Conforme uma aplicação aumenta em complexidade, manter a analogia para as novas funcionalidades pode começar a influenciar negativamente na interação;
- Metáforas podem aumentar a distância entre o modelo mental do usuário e o modelo estrutural do sistema. As metáforas não possuem apenas características análogas ao sistema, mas outras características que não têm equivalência nesse. Isso pode levar os usuários a imaginar propriedades e funcionalidades que o sistema não possui;
- Metáforas perdem a funcionalidade se não houver tradução. Para usuários de sistemas fora da língua onde foram concebidos, pode ocorrer de aprenderem primeiro como o sistema funciona, para depois descobrir a tradução do termo da metáfora para seu idioma. Nesse momento, a metáfora torna-se dispensável. Algo semelhante pode ocorrer sempre que a bagagem sócio-cultural do usuário for muito diferente daquela do projetista.

2.4.3 Multidisciplinaridade de IHC

Partindo-se do modelo de interação, pode-se verificar que, para completar um simples ciclo de interação, passa-se por disciplinas tão distintas quanto Cognição Humana, Fisiologia, Eletrônica de Controle e Processamento de Dados.

Shneiderman (2003)²¹ afirma que o mundo vive uma transição de um paradigma de tecnologia antigo (*“old computing”*), onde o foco estava naquilo que os computadores podem fazer, para uma nova tecnologia (*“new computing”*), onde o foco está em o que as pessoas poderiam fazer com eles.

E, para atender às necessidades dos usuários - usuários esses cada vez mais diversificados e menos interessados em computação - os desenvolvedores de tecnologia necessitam adaptar-se ao modelo da nova tecnologia, desenvolvendo novas competências.

2.4.4 Usabilidade

Usabilidade é uma medida que serve para comparar produtos similares (concorrentes ou versões de um mesmo produto) quanto à sua capacidade de atender às necessidades de um usuário específico.

Usabilidade: Medida da facilidade com que a interface de um produto (geralmente orientada a *software*) permite atender às necessidades de seus usuários.

Para realizar essa medida, pode-se utilizar quantificadores (SHNEIDERMAN; PLAISANT, 2005)¹⁴ como os listados a seguir:

- **Tempo de Aprendizado:** tempo necessário para um usuário novato (primeiro contato com o produto) se tornar produtivo com a ferramenta (tornar-se um usuário mediano).

- **Retenção ao longo do tempo:** após determinado período de tempo sem utilizar a ferramenta, o (inverso de) quanto deverá ser reaprendido pelo usuário por causa de esquecimento.
- **Taxa de erros do usuário:** que parcela do trabalho executado por um usuário mediano por período específico de tempo deverá ser refeito de forma aprimorada.
- **Desempenho do usuário:** quanto trabalho pode ser realizado por um usuário mediano em um período específico de tempo.
- **Satisfação subjetiva:** o quanto os usuários da ferramenta se declaram satisfeitos com a mesma.

Embora o ideal seja maximizar todos esses quantificadores para qualquer produto, normalmente tem-se uma situação de compromisso, onde decisões de projeto podem aumentar um deles em detrimento dos demais.

Dessa forma, uma das primeiras decisões de um projeto de interação é justamente a identificação de quais são os quantificadores mais importantes para o produto em questão e usuários-alvo.

2.5 Etapas e Atividades de um Projeto de Interação

No intuito de realizar a concepção, desenvolvimento e refinamento de interfaces de software, a seguir serão listadas as diversas metodologias encontradas na literatura sobre projeto de interação, com suas características e atividades programadas.

2.5.1 O Desenvolvimento de Software Tradicional

Por muito tempo empresas adotaram e ainda hoje adotam o modelo de desenvolvimento em cascata (*waterfall*), em que o projeto de software é segmentado em etapas e nenhum trabalho é iniciado antes que a etapa anterior termine.

Além de essa interdependência entre processos ser muito pouco eficiente, as etapas relacionadas com o desenvolvimento da interface do sistema costumavam ser relegadas para os últimos estágios do processo, normalmente quando muito investimento já havia realizado e os prazos se tornavam muito estreitos. Este procedimento impedia qualquer modificação de maior alcance. O resultado, constantemente, eram produtos finais com baixíssima usabilidade e aceitação pelos usuários finais.

Outro erro usual de projeto, principalmente na área de tecnologia, consiste em se partir das soluções e buscar problemas para serem resolvidos por elas. Dessa forma, é comum acabar escolhendo uma solução muito complexa para um problema simples; ou pior, pode-se tentar resolver um problema que não existe, criando-se outros durante o processo.

2.5.2 O Projeto de Interação e a Engenharia de Software

Durante o desenvolvimento de um software, o ideal é que existam essas duas equipes (engenharia de software e de interação) trabalhando paralelamente, desde o início do processo.

É comum que a equipe de interação entre no processo somente no final do desenvolvimento, quando a maioria dos requisitos já foi desenvolvido e precisa-se de alguém para criar “as telas do sistema”. Nesse caso, pouco pode ser feito para transformar esse produto em algo desejável ou mais útil para o usuário.

A equipe de Engenharia de Software e a equipe de Interação são distintas porque têm preocupações e objetivos distintos. Mas devem atuar paralelamente durante o desenvolvimento, trocando informações constantemente, de forma a identificar problemas rapidamente, economizando tempo e trabalho dos desenvolvedores. Alguns dos objetivos de Engenharia de Software são (SHNEIDERMAN; PLAISANT, 2005)¹⁴ :

- Funcionalidade adequada;
- Confiabilidade;
- Disponibilidade;
- Segurança;
- Integridade dos dados;
- Padronização;
- Integração;
- Consistência;
- Portabilidade;
- Respeito a prazos e orçamentos.

2.5.3 O Projeto Orientado por Tarefas

O desenvolvimento de software tradicional é nitidamente orientado pelos requisitos de sistema, bem ao estilo do *old computing* discutido em 2.4.3.

Lewis e Rieman (1994)²² propõem a mudança de foco das funcionalidades do sistema para as tarefas dos usuários. Sugerem um método obedecendo às seguintes etapas:

1. Identificar usuários potenciais;
2. Identificar suas pretensões com o sistema;
3. Conversar diretamente com eles, incluindo-os no processo produtivo (*Participatory Design*):
 - Levantar exemplos concretos e detalhados de tarefas;
 - **O que** a tarefa faz e não **como**;

- Tarefas específicas;
 - Usuários específicos;
 - Conferir entradas, saídas e transições das tarefas.
4. Formalização das descrições de tarefas;
 5. Criação de Cenário (onde se determina **como**);
 6. Representação dos cenários através de *storyboard*.

É importante notar como o método acima preocupa-se com a interação logo nas primeiras etapas do processo de desenvolvimento e com o constante relacionamento com usuários reais do sistema ao longo do projeto do software.

2.5.4 O Projeto de Interação: Uma Definição

Preece; Rogers e Sharp (2002)²³ definem o projeto de interação como:

Projeto de Interação: a elaboração de produtos para auxiliar as pessoas em suas vidas pessoais e em suas carreiras.

Essencialmente, envolve quatro atividades básicas:

- Identificação das necessidades e estabelecimento de requisitos;
- Desenvolvimento de projetos alternativos que atendam aos requisitos;
- Construção de versões interativas dos projetos, de forma a serem mostrados, discutidos e avaliados;
- Avaliação do que está sendo construído durante o processo.

Existem três características-chave do processo:

- Usuários devem ser envolvidos durante o projeto;
- Metas de experiência de usuário e de usabilidade específicas devem ser identificadas, claramente documentadas e acordadas no início do projeto;
- Iteração sobre as quatro atividades é inevitável.

2.5.5 Projeto de Interação Orientado por Metas

Cooper e Reimann (2003)²⁰ definem sua metodologia de Projeto de Interação Orientado por Metas como três objetivos:

- Entender os desejos, necessidades, motivações e contexto dos usuários;
- Entender requisitos e restrições técnicas, de negócio e do campo de aplicação;
- Tradução desse conhecimento em planos para artefatos cuja forma, conteúdo e comportamento sejam úteis, usáveis e desejáveis, enquanto economicamente viáveis e tecnicamente factíveis.

O processo para atingir tais objetivos passa pelas seguintes atividades:

- Pesquisa (usuários e campo da aplicação);
- Modelagem (usuários e contexto de uso);
- Definição de Requisitos (definição de necessidades do usuário, do negócio e técnicas);
- Definição de Estrutura (estrutura e fluxo de projeto);
- Refinamento (comportamento, forma e conteúdo).

2.5.6 A Engenharia de Usabilidade

Mayhew (1999)²⁴ vai além do projeto de interação e cria uma metodologia para a maximização da usabilidade em sistemas digitais que não se restringe às etapas de projeto, mas se estende durante todo o processo; anteriormente, na análise de requisitos, e posteriormente, durante a implantação dos sistemas.

[Nielsen 1989b] apud (NIELSEN, 1994)²⁵ apresenta um modelo chamado Engenharia de Usabilidade Barata (*Discount Usability Engineering*), baseado em três tarefas principais:

- Criação de cenários;
- Teste de voz-alta simplificado (*simplified thinking aloud test*);
- Inspeção através de avaliação heurística;

Esse método permite que o projeto de interação possa ser realizado por um número pequeno de avaliadores e com pouca participação de usuários reais, o que resulta sempre em custos operacionais e logísticos menores. Dessa forma, muito pode ser inferido com pouco investimento de tempo e orçamento.

2.5.7 Prototipagem em papel

Uma maneira de assegurar que o projeto de interação possa acontecer nas primeiras iterações de projeto, mesmo para produtos inovadores, é a utilização de protótipos.

A primeira consideração a respeito de protótipos é quanto ao número de funcionalidades implementadas (sua largura) e o nível de interação permitido a cada uma delas (sua profundidade): o protótipo deverá estar completo o suficiente para atender um cenário específico que se queira testar.

A prototipagem usando rascunhos em papel é um método rápido que permite várias versões e mudanças durante avaliação (inspeção ou teste de usabilidade). Snyder (2003)²⁶ detalha uma metodologia para aplicação de protótipos em papel em teste de usabilidade do tipo de voz-alta, desde justificativas, pontos fortes e fracos, projeto de tarefas e criação dos *widgets*, até a condução do teste e relatório. É uma atividade em grupo, envolve usuários reais, permite o projeto participativo e a descoberta de um grande número de informações importantes em pouco espaço de tempo. As etapas dessa metodologia são as seguintes:

1. *Kick-off meeting*
2. Recrutamento de usuários
3. Projeto de tarefas
4. Construção do protótipo
5. Teste-piloto
6. Aplicação dos testes
7. Consolidação das informações

Outro fator importante é quanto ao uso desses protótipos para validação com clientes. Protótipos devem ser utilizados somente como artefatos de projeto, nunca sendo reaproveitados ou pós-desenvolvidos em produtos finais. Essa prática é bastante comum em sistemas de software, mas a prototipagem em papel garante que isso não aconteça (por usarem mídias físicas diferentes).

2.5.8 Avaliação de Interfaces

A atividade do projetista de interação deve ser cíclica, de forma que os resultados obtidos na fase de avaliação sejam rapidamente utilizados num refinamento do projeto, e novas versões de protótipos.

Como se trata de uma medida, a usabilidade é utilizada como instrumento de avaliação dos sistemas digitais.

Dentre as inúmeras formas e métodos possíveis para medir a usabilidade de um sistema, pode-se classificá-los em dois grupos: testes de usabilidade e inspeção de usabilidade. A diferença está na presença de usuários reais do sistema.

Métodos como os testes de usabilidade (com a presença de usuários reais) são mais indicados em fases mais avançadas do projeto, para avaliar produtos antes de seu lançamento ou comparar produtos concorrentes. São indicados para se obter ajuda no desenvolvimento ou medir o desempenho dos usuários. Deve-se ter a preocupação com a validade e confiabilidade das sessões de testes conduzidas. Devido aos custos envolvidos no processo, são normalmente feitas gravações de áudio e vídeo dessas sessões de teste, que ficam disponíveis para análise posterior.

Métodos de inspeção de usabilidade (onde avaliadores experientes representam usuários reais) podem ser utilizados nas fases iniciais do projeto, até mesmo sem nada ter sido programado. Geralmente produzem uma lista de diretrizes a serem seguidas pela equipe de desenvolvedores.

Inspeção por Avaliação Heurística

Avaliação heurística é um método de inspeção de usabilidade onde os avaliadores seguem um cenário (composto de um usuário específico – persona – uma sequência de tarefas específica em um contexto – momento e lugar – específico), procurando identificar problemas de interação.

A cada etapa, se considerarem que o usuário teria algum problema de interação, devem anotá-lo, associando-o a uma entre dez heurísticas de usabilidade (reproduzidas no Anexo A).

Numa segunda fase, os relatórios individuais dos avaliadores devem ser discutidos para se chegar a um relatório final, com todos os problemas identificados, associados às heurísticas quebradas e à um grau de severidade.

2.6 Usuários de um Motor de Jogo

É comum o desenvolvedor não conhecer o usuário que utilizará o produto que desenvolve. Costuma-se pensar que qualquer pessoa possa ser usuário de um sistema, ou que o próprio desenvolvedor é um bom exemplo de tal usuário.

Admitir como usuário “qualquer pessoa” é o mesmo que não identificar nenhuma característica, e, portanto, não ter nenhuma informação a respeito desses usuários. Considerar o próprio desenvolvedor como exemplo, assume, além de uma série de características típicas de programadores, um completo conhecimento do código do sistema. Ambas atitudes estão equivocadas e levarão a produtos indesejáveis.

Portanto, um grande objetivo para um projetista de interação é conhecer o usuário de seu produto, de forma a identificar suas metas e necessidades, e adequar seu produto às mesmas.

2.6.1 O Projetista de Jogos*

A priori, antes do projeto de interação, já é possível identificar o projetista de jogos como um usuário importante de motor de jogos. Trata-se de uma função dentro de uma empresa desenvolvedora de jogos eletrônicos responsável pela concepção e condução de projetos de produtos-jogo.

Dentre as funções sob sua responsabilidade estão:

- Desenvolvimento da mecânica do jogo;
- Descrição de requisitos;
- Balanceamento de medidas e de recursos dos objetos de jogo;
- Desenvolvimento do layout e padrão visual do ambiente de jogo;
- Desenvolvimento do tema;

Boa parte das competências exigidas para tal função vem das ciências humanas e, portanto, estes profissionais não possuem necessariamente embasamento teórico forte nas disciplinas de computação.

2.6.2 Classificação de Usuários

Para o efetivo reconhecimento dos usuários de motor de jogo, será feita a identificação e listagem dos diversos tipos de indivíduos afetados pelo sistema. A intenção disso é um levantamento de todo universo do sistema (o motor de jogo), de modo a evitar o esquecimento de usuários não-triviais e um melhor entendimento do contexto para que está se desenvolvendo. Assim permitindo a visualização do universo de usuários afetados pelo sistema desenvolvido, criando uma visão generalista, possibilitando posterior recorte para o subconjunto de maior interesse ao

* O *Game Designer*, como é conhecido esse profissional, costuma criar erroneamente alguma ligação com as atividades de *Web Designer* ou *Designer Gráfico*, apesar de terem competências e atividades distintas.

projeto sem, no entanto, desconsiderar nenhuma parte afetada, e chegando assim a uma escolha mais válida.

Para isto serão listados inicialmente os usuários, classificando-os como usuários primários, secundários ou terciários (Eason (1987)) *apud* (PREECE; ROGERS; SHARP, 2002)²³.

Usuários primários são aqueles que utilizam diretamente o sistema.

Usuários secundários dividem-se em dois grupos: aqueles que usam o sistema indiretamente, e usuários esporádicos, que sempre precisam lembrar seu funcionamento.

Usuários terciários não utilizam o sistema diretamente ou indiretamente, mas estão relacionados e são importantes para o mesmo. Também são divididos em dois grupos: aqueles que são afetados pelo sistema, e aqueles que participam do processo de escolha ou compra do produto (*stakeholders*).

Vale ressaltar que essa divisão não é excludente, podendo um indivíduo ora se comportar como usuário de um tipo, ora de outro tipo.

No caso específico dos motores de jogo, foi constatado mais de um universo de usuários possíveis: os universos corporativo, *serious games* e amador.

Para o universo corporativo, a identificação desses usuários foi feita através do capítulo 5 de Bethke (2003)²⁷, que realiza um levantamento das posições de trabalho em uma empresa de desenvolvimento de jogos. Essas funções foram listadas em uma tabela, acrescentando informações sobre para qual tipo de empresa esse profissional pode atuar, suas competências exigidas e, por fim, a classificação como tipo de usuário de um motor de jogo (Tabela 2.3).

Na falta de uma referência bibliográfica igualmente detalhada, foram feitas descrições e considerações sobre os universos *serious games* e amador.

Tabela 2.3 - Exemplo de distribuição de atividades produtivas no ciclo de vida de um jogo eletrônico comercial. Destaques em azul para relacionamento mais forte com motor de jogo.

Função ou atividade	Tipo de Empresa	Competência	Classificação
<i>Game Designer: Visionary</i>	Desenvolvimento	Gerência	Secundário
<i>Game Mechanics Designer</i>	Desenvolvimento	Projetista	Primário ou Secundário
<i>Level/Mission Designer</i>	Desenvolvimento	Projetista	Primário ou Secundário
<i>Story and Dialog Writer</i>	Desenvolvimento	Projetista	Secundário
<i>Lead Programmers and Technical Directors</i>	Desenvolvimento	Programador	Primário ou Secundário
<i>Game Mechanics Programmer</i>	Desenvolvimento	Programador	Primário
<i>3D Graphics Programmer</i>	Desenvolvimento	Programador	Primário
<i>Artificial Intelligence Programmer</i>	Desenvolvimento	Programador	Primário
<i>User Interface Programmer</i>	Desenvolvimento	Programador	Primário
<i>Audio Programmer</i>	Desenvolvimento	Programador	Primário
<i>Tools Programmer</i>	Desenvolvimento	Programador	Desenvolvedor*
<i>Mission/Level Editor Programmer</i>	Desenvolvimento	Programador	Primário
<i>Network Programmer</i>	Desenvolvimento	Programador	Primário
<i>Art Director</i>	Desenvolvimento	Gerência	Terciário
<i>Concept Artist</i>	Desenvolvimento	Projetista	Terciário
<i>2D Artist/ Interface Designer</i>	Desenvolvimento	Artista	Primário
<i>3D Modeler</i>	Desenvolvimento	Artista	Secundário
<i>Character Modeler</i>	Desenvolvimento	Artista	Secundário
<i>Texture Artist</i>	Desenvolvimento	Artista	Secundário
<i>Animator/Motion Capture Studio</i>	Desenvolvimento	Artista	Secundário
<i>Storyboarder</i>	Desenvolvimento	Projetista	Primário
<i>Voice-Overs</i>	Desenvolvimento	Artista	Secundário
<i>Sound Effects</i>	Desenvolvimento	Projetista	Secundário
<i>Music</i>	Desenvolvimento	Artista	Secundário
<i>Line Producer</i>	Desenvolvimento	Apoio	Terciário
<i>Associate Producer</i>	Desenvolvimento	Gerência	Terciário
<i>Studio Head/Executive Producer</i>	Desenvolvimento	Gerência	Terciário
<i>Producer</i>	Desenvolvimento	Gerência	Terciário
<i>QA Lead</i>	Produção	Jogo e Tecnologia	Terciário
<i>QA Main Team</i>	Produção	Jogo	Terciário
<i>QA Multiplayer Team</i>	Produção	Jogo	Terciário
<i>QA Fresh Teams</i>	Produção	Jogo	Terciário
<i>Compatibility Team</i>	Produção	Tecnologia	Terciário
<i>Localization Team</i>	Produção	Idioma e Cultura	Primário ou Secundário
<i>Beta Testers</i>	Consumidor	Jogo	Terciário
<i>Beta Testing Program Manager</i>	Desenvolvimento	Relações Públicas	Terciário
<i>Business Development Executive</i>	Produção	Negócio	Terciário
<i>Publisher CEO and President</i>	Produção	Negócio	Terciário
<i>Studio Heads</i>	Desenvolvimento e Produção	Negócio	Terciário
<i>Lawyers</i>	Desenvolvimento e Produção	Direitos autorais, legislação	Terciário, em alguns casos específicos

(continua)

* essa é a função de quem desenvolve o motor de jogo, e portanto, não é considerado usuário

Tabela 2.3 - Exemplo de distribuição de atividades produtivas no ciclo de vida de um jogo eletrônico comercial. Destaques em azul para relacionamento mais forte com motor de jogo.

Função ou atividade	Tipo de Empresa	Competência	Classificação
<i>Licensing</i>	Produção	Direitos autorais; Contratos	Terciário
<i>Sales Executive</i>	Distribuição	Negociação; Comunicação	Terciário
<i>Sales Force and Retail Purchasing Agents</i>	Distribuição	Negociação; Comunicação	Terciário
<i>Press Relations Manager</i>	Distribuição	Comunicação	Terciário
<i>Trade Shows</i>	Distribuição	Comunicação	Terciário
<i>Marketing</i>	Distribuição	Comunicação	Terciário
<i>Hardcore Fans</i>	Consumidor	Jogo e tecnologia	Terciário
<i>Manual</i>	Distribuição	Comunicação	Terciário
<i>Strategy Guide</i>	Distribuição	Comunicação	Terciário
<i>Console Manufacturers</i>	Licenciamento e Produção	Jogo e tecnologia; Contratos	Terciário
<i>Hardware Representatives</i>	Produção	Tecnologia	Terciário
<i>Patches and updates</i>	Suporte	Jogo e tecnologia; Comunicação	Primário
<i>FAQ e Forums</i>	Suporte	Comunicação	Terciário
<i>Customer Service</i>	Suporte	Comunicação	Terciário
<i>Gamemasters</i>	Operação	Jogo; Comunicação	Primário ou Secundário
(BETHKE, 2003) ²¹			(conclusão)

(Eason (1987)) apud (PREECE; ROGERS; SHARP, 2002)²³

2.6.2.1 Universo Corporativo

A partir da Tabela 2.3, identificou-se como importantes para o desenvolvimento os seguintes usuários desse universo:

- *Game Mechanics designer/programmer;*
- *Level designer/programmer;*
- *Art Director;*
- *Localization Team;*
- *Patches and updates;*
- *Gamemasters;*

Esse conhecimento será utilizado para criar um modelo desse universo comercial.

O universo de desenvolvimento de jogos comerciais envolve vários softwares, seus desenvolvedores e seus usuários. Um motor de jogo é apenas um desses softwares. Em Figura 2-5 e Figura 2-6 esquematiza-se esse contexto, assim evitando confusão entre os mesmos.

O projeto de interação de um motor de jogo pode ser utilizado para dois casos: no primeiro e mais comum, o motor de jogo já existe no todo ou em parte, e deseja-se criar uma interface para o mesmo; em um segundo caso mais ideal, a equipe de desenvolvimento ainda está na fase de projeto do motor de jogo e deseja aproximá-lo das necessidades de seus usuários.

Figura 2-5 - Caso 1: Criação da Interface do Motor de Jogo após sua codificação

No primeiro caso, ilustrado pela Figura 2-5, temos:

- dois tipos de desenvolvedores (o programador responsável pelo motor, e o engenheiro de usabilidade ou designer, responsável pela interface);
- um “desenvolvedor secundário”, o engenheiro de software, que entrega o projeto com as especificações e requisitos para as equipes de programação de motor;

- uma vez que a interface apenas mapeia as funcionalidades implementadas no motor de jogo, dificilmente um *game designer* sem noções de programação conseguirá utilizá-lo a contento (será um usuário secundário), e portanto, solicitará a outros programadores (usuários primários) a criação do jogo;
- apesar de não estarem ilustrados na figura, todos os responsáveis pelo conteúdo artístico do jogo são usuários secundários do motor do jogo, da mesma forma que o *game designer*, através dos programadores de jogo;
- como usuários terciários, tem-se o produtor, que financia o desenvolvimento dos jogos esperando um retorno financeiro; os jogadores, que não usam o motor de jogo, mas são afetados por esse, através dos jogos produzidos através dele; e o gerente de Tecnologia da Informação (TI), que influencia sua compra e é responsável pela instalação inicial do motor.

Figura 2-6 - Caso 2: Criação de Motor de Jogo otimizado para o Game Designer

No segundo cenário, ilustrado na Figura 2-6, percebe-se as seguintes diferenças:

- O engenheiro de usabilidade ou *interface designer*, atuando também como projetista (“desenvolvedor secundário”), agora

participa do processo de desenvolvimento muito mais cedo, em paralelo com o engenheiro de software, na confecção das especificações e requisitos do sistema;

- Dessa forma, o desenvolvedor do motor de jogo pode criar uma ferramenta que possui não apenas a infra-estrutura tecnológica necessária para a criação de jogos, mas uma linguagem compreensível pelo usuário final da ferramenta, o *game designer*;
- O *game designer* passa a ser usuário primário do motor de jogo. Esse pode até continuar contando com programadores para a implementação do jogo, mas não mais por restrição, uma vez que a ferramenta está orientada a atender as suas necessidades, de acordo com o seu modelo mental;
- Com a estrutura otimizada, os investimentos do produtor são mais eficientemente alocados e podem até ser reduzidos;
- Com uma ferramenta mais adequada às suas necessidades, o *game designer* pode produzir mais e melhores jogos, podendo gerar um retorno financeiro maior através dos jogadores.

2.6.2.2 Universo Serious Games

À medida que a tecnologia de ponta é testada e aplicada por jogos eletrônicos, esta passa a ser atraente para desenvolvedores de outras aplicações. Esses usuários conhecem muito bem o problema que precisam resolver e descobrem em algum jogo eletrônico uma solução que julgam suficiente ou até mais elaborada que suas necessidades.

Um problema nesse universo pode ser a ilusão de que, uma vez que jogos são para entretenimento, podem ser uma solução muito mais barata que as aplicações comerciais industriais. Pode causar desapontamento descobrir que o custo e tempo de desenvolvimento de um jogo podem ser iguais ou superiores ao de qualquer outro *software*.

Outro problema é o fato de os jogos terem como meta principal atender a uma interação em tempo real que garanta imersão e “jogabilidade” adequada, muitas vezes em detrimento de precisão – um compromisso entre emulação e simulação. Dependendo da “aplicação séria” isto pode ser um limitante.

Nesse universo, podem ser utilizados os mesmos diagramas da Figura 2-5 e Figura 2-6, mas observando diferenças nas características do *Game Designer*. No caso de *serious games*, o projetista não tem formação ou experiência com desenvolvimento de jogos, mas possui conhecimento técnico especializado na área do negócio a que deseja aplicar os recursos de um motor de jogo.

Importante notar que o produto final nesse caso não poderá ser automaticamente considerado um jogo eletrônico, ou mesmo brinquedo interativo.

2.6.2.3 Universo amador

No caso de um desenvolvedor amador, esse projetista tem muito pouca formação ou experiência técnica, mas muita experiência como jogador (com os produtos finais) e deseja implementar sua própria versão deles.

Normalmente são os usuários que têm mais tempo livre para se dedicar ao desenvolvimento de um jogo, mas ao mesmo tempo o menor embasamento teórico e grande taxa de distração (possibilidade de desistência ao encontrar as primeiras dificuldades).

3 METODOLOGIA

As tarefas realizadas são derivadas dos métodos de projeto de interação descritos em 2.5, sempre priorizando as alternativas que se adaptam melhor às condições de curto prazo e baixo orçamento.

Inicialmente, durante um primeiro contato com o universo dos motores, a metodologia adotada para o planejamento do ambiente de programação a ser aplicada ao **enJine (Game Wrench)** foi baseada na metodologia “*Discount Usability Engineering*”, criada por Jakob Nielsen (1994)²⁵, incluindo a confecção de cenários e avaliação heurística. Nesse momento, procurou-se estabelecer uma série de diretrizes voltadas para um grupo de usuários mais amplo. Os resultados dessa fase estão apresentados no Projeto de Interação no Apêndice C.

Com base nos resultados assim obtidos, o foco foi concentrado no atendimento às necessidades do motor **enJine** que atualmente é utilizado por alunos em disciplinas como Computação Gráfica de cursos de computação (TORI et al., 2006)¹². Ou seja, reduziu-se bastante a gama de usuários e permitiu-se contato direto com os mesmos. Nesse momento foi seguida uma metodologia análoga, apresentada em Snyder (2003)²⁶, que inclui entrevistas com *stakeholders*, projeto de tarefas, prototipagem em papel e teste “*Thinking Aloud*” simplificado. Os resultados dessa fase são apresentados no Relatório de Testes no Apêndice A.

A Figura 3-1 apresenta um diagrama com as atividades realizadas e resultados produzidos durante esse processo, com a ordem e relacionamento entre elas.

Figura 3-1 - Documentos produzidos, atividades realizadas e softwares criados e utilizados como metodologia. Setas indicam seqüência e interdependência das tarefas.

3.1 Pesquisa Bibliográfica

Como base para o desenvolvimento, foi feita pesquisa bibliográfica sobre os temas da dissertação, apresentada no Capítulo 2.

O termo jogo possui conotações diferentes nos diversos idiomas encontrados na literatura e suas respectivas traduções, o que pode causar muita confusão. Qualquer

estudo sobre esse tema deve se iniciar com um esclarecimento do pesquisador sobre qual definição de jogo está adotando para o seu trabalho. Por esse motivo, tal definição já foi apresentada anteriormente na Seção 2.1.

Da mesma forma, apresentou-se a conceituação de Motor de Jogo (o produto sobre o qual deseja-se melhorar a interação) e sua diferenciação com *mod-games*.

Assim como diversão e jogo, outras áreas de conhecimento são abordadas nessa dissertação, como Interação Humano-Computador, Usabilidade e Engenharia de Software, bem como as técnicas de projeto de interface e interação.

3.2 Projeto de Interação

Para a primeira parte do projeto, foram executadas atividades com o intuito de entender o produto motor de jogo e seus usuários:

1. **Conceito do produto.** Responder à questão “O QUE é o produto?”. Para isso é possível utilizar-se de respostas complementares (às perguntas ONDE – contexto – e PARA QUE – funcionalidade), mas sem recair em restrições tecnológicas ou soluções prontas (COMO).
2. **Classificação de usuários.** Pesquisar o(s) universo(s) de potenciais usuários do produto motor de jogo de forma a partir-se de uma base mais abrangente ao se fazer as seleções e escolhas para o projeto.
3. **Criação de *Personas*.** Selecionar os usuários-foco do projeto, listar características relevantes e criar *personas* em número suficiente para representá-los. Estas características podem ser orientadas por modelos formais ou informais de pessoas, como sócio-culturais, cognição, astrologia e personagens de ficção.
4. **Identificação de metas.** Através de *personas* representativas, é mais simples a identificação das metas dos usuários reais que essas representam.

5. **Tabela de tarefas por *personas*.** Identifica quais tarefas são prioritárias para qual tipo de usuário, através da frequência com que ocorrem e grau de criticidade.
6. **Criação de Cenários.** Através da informação da tabela, selecionar usuários específicos realizando tarefas específicas. Para completar as narrativas dos cenários, acrescenta-se contexto: tempo e lugar em que ocorrem.

3.3 *Análise Comparativa*

Fez-se um levantamento de características encontradas em algumas soluções de motor de jogo, verificando pontos positivos e negativos e traduzindo-os na forma de requisitos.

Primeiramente, foram analisadas as experiências feitas no Interlab, tanto com um motor comercial, quanto ao desenvolver um motor acadêmico próprio.

Fez-se também uma análise sobre dois motores comerciais que apresentam interface gráfica (*GUI*) na tentativa de aproximar o uso de motores de usuários menos técnicos. O método de inspeção de usabilidade por avaliação heurística foi empregado para essa análise sobre os softwares **Game Maker v.6.1** e **3D GameStudio v.A6**. Essas duas ferramentas possuem interface visual para organização de recursos artísticos, criação de conteúdo e acesso às funcionalidades de um motor de jogo proprietário.

3.4 *Estudo de Usabilidade com Prototipagem em Papel*

Esta etapa envolveu entrevistas com *Stakeholders*, recrutamento de usuários, projeto de tarefas, construção de protótipos em papel, e teste de usabilidade.

3.4.1 Entrevistas com Stakeholders

Foram realizadas entrevistas com os principais envolvidos na programação do **enJine**. Nessas reuniões, discutiu-se o estado atual do projeto, definiu-se o usuário típico do sistema, listaram-se tarefas comuns, discutiram-se preocupações e pontos críticos.

3.4.2 Recrutamento de Usuários

Durante as entrevistas com *stakeholders*, definiu-se que, ao contrário do que vinha sendo vislumbrado, o grupo de usuários-foco deste projeto deveria ser composto pelos alunos cursando a disciplina de Computação Gráfica que vêm usando o **enJine** para desenvolver um produto-jogo como atividade da disciplina.

Uma vez estabelecido esse aluno como usuário-foco principal, foi feita durante uma das aulas dessa turma uma exposição sobre os conceitos de IHC, projeto de interação e prototipagem. Quatro dos alunos presentes se apresentaram como voluntários para as sessões de teste.

3.4.3 Projeto de Tarefas

A partir das tarefas levantadas através das entrevistas com os *stakeholders*, foram escolhidas as que estão relacionadas com as fases iniciais de planejamento para fazer parte do cenário a ser seguido nos testes com os protótipos em papel.

Isso é importante para a preparação do protótipo, sendo que o mesmo precisará estar “operacional” apenas o necessário para percorrer esse cenário.

3.4.4 Prototipagem em papel

Prototipagem em papel é uma técnica de baixa tecnologia para avaliação de interfaces de software que é muito valiosa nas etapas iniciais de desenvolvimento, antes de qualquer programação ser executada. Permite mudanças rápidas e fáceis na interface, a uma alta taxa de interação com os avaliadores.

O conjunto inicial de representações de “telas” e “menus” só precisa estar “operacional” ao ponto de permitir a representação da execução das tarefas necessárias para os cenários.

Maiores detalhes sobre os protótipos e sua aplicação são apresentadas no Apêndice A.

3.4.5 Teste de Usabilidade de Voz-Alta Simplificado

As sessões dos testes foram marcadas para o horário após as aulas dos alunos, um por vez, durante uma semana. As sessões foram programadas para durar de uma a duas horas. Foi solicitado que não conversassem entre si sobre os testes enquanto todos não os tivessem completado. As sessões foram gravadas em vídeo para futura referência.

O time de examinadores dividiu-se nas funções de “facilitador”, “computador”, “organizador” e “observador”. Entre cada sessão de testes, as questões levantadas pelos observadores foram discutidas, documentadas e criadas as modificações relevantes, já implementadas nas sessões seguintes.

3.5 *Protótipo Computacional da Interface Game Wrench*

Com base no aprendizado com os testes, nas diretrizes advindas dos requisitos colhidos durante todo o projeto e na última versão do protótipo em papel, começou-se a programar esta interface com o auxílio de ferramentas de software.

Para essa primeira versão computacional, escolheu-se a plataforma **Java** com a *API Swing* como infra-estrutura. Desta forma, foi possível visualizar como tal interface poderá se comportar quando for totalmente desenvolvida, em futuras versões.

Ainda como infra-estrutura computacional, definiu-se um esquema *XML (XSD)*, que possibilita estruturar um arquivo-texto com informações de alto nível sobre jogos. Esta estrutura foi batizada de *Game Markup Language (GAML)*.

Devido a suas características de linguagem de marcação, pode ser utilizada para armazenar as informações coletadas pela interface, para transformar tais informações em código através do motor de jogo, ou mesmo para efetuar a comunicação entre estes módulos de forma padronizada. Ainda pode servir para o armazenamento de instâncias de jogos e campeonatos.

4 DESENVOLVIMENTO

4.1 *Projeto de Interação*

As etapas do projeto de interação, que inicialmente servem para estabelecer um contato com o universo dos motores de jogo e depois como base para os testes de usabilidade com usuários reais, estão descritas a seguir.

4.1.1 **Escopo do Projeto de Interação**

Dentre as atividades executadas para atingir as metas dessa dissertação estão o projeto de interação do sistema digital denominado motor de jogos e a aplicação do resultado desse projeto ao motor de jogos **enJine**¹, desenvolvido pelo Interlab.

Pela figura 1-1 de Cooper e Reimann (2003)²⁰, vê-se que o processo de desenvolvimento de *softwares* tem evoluído de uma situação onde alguns desenvolvedores/programadores cuidam de todo o ciclo de vida do software, desde sua concepção, implementação até a comercialização e suporte, para outra situação mais eficiente onde gerentes cuidam da concepção e viabilidade, em seguida os projetistas (*design*) asseguram a funcionalidade, para somente então os desenvolvedores passarem à codificação.

No meio termo dessa evolução, existe um caso muito comum em que os projetistas de interface recebem um produto praticamente pronto, com a incumbência de “criar a interface” para esse produto. Apesar de tal situação ser indesejável, uma vez que pouca modificação estrutural pode ser realizada com base em descobertas do processo do projeto de interação, muitas vezes é inevitável.

Sendo o ambiente **Game Wrench** uma camada de abstração acima dos motores de jogo, sua especificação pode servir tanto como orientação para o desenvolvimento de um motor de jogo novo, quanto para o acesso estruturado às funcionalidades de um ou mais motores existentes. Essas duas possibilidades já foram mais elaboradas no item 2.6.2.1.

4.1.2 Conceito do Produto

O projeto de interação deve começar com uma especificação clara e objetiva do produto em questão.

É importante definir, logo de início, sobre O QUE se está projetando. Essa definição deve, a priori, estar clara e ser consenso entre todos os envolvidos nos processos de projeto e desenvolvimento.

Uma vez definido qual o conceito do produto, considerações sobre o contexto ONDE se encaixa e funcionalidade PARA QUE é normalmente utilizado ajudam a esclarecer esse conceito.

Para tanto, no projeto de interação foi utilizada a definição levantada na dissertação (capítulo 2.3.2), apresentado um histórico mais detalhado sobre o termo, e considerados os três universos de usuários.

4.1.3 Cenários

Cenários são narrativas compostas de usuários específicos executando uma série de tarefas específicas em um contexto (tempo e lugar) específico.

4.1.3.1 Definição de usuários específicos

Inicialmente, são identificados os usuários mais importantes para o projeto de interação em questão e consideradas quais seriam as possíveis formas de obter maiores informações de indivíduos representantes de tais grupos de usuários.

4.1.3.2 Criação de Personas

Existem dois enfoques para uma pesquisa com usuários reais: quantitativo (estatísticas levantadas sobre um espaço amostral representativo) e qualitativo (informações aprofundadas sobre indivíduos representativos). Essas duas maneiras de realizar uma pesquisa são complementares, cada uma permite obter um tipo de informação, uma visão do mesmo tema por pontos de vista distintos. Recomenda-se alternar os dois estilos.

Pesquisas quantitativas (estatísticas) são importantes, mas para as questões que se deseja levantar, de âmbito subjetivo, dificilmente trazem resultados satisfatórios. Informações importantes como quais são as metas dos usuários, se perguntadas diretamente, poderão não ser válidas por vários motivos:

- O usuário real pode simplesmente não saber quais são suas metas;
- O usuário real pode, apesar de saber suas metas, não saber expressá-las de forma adequada;
- O usuário real pode saber suas metas, saber expressá-las corretamente mas, ainda assim, não querer ou não poder colaborar informando-as.

Dessa forma, é mais fácil criar modelos de usuários (*personas*). Se os modelos forem bons, as conclusões inferidas sobre os mesmos também serão válidas. É mais fácil trabalhar com modelos do que com pessoas reais.

São criados modelos de usuários, com os quais são feitas as suposições do projeto. Essas *personas* são modelos de um grupo representativo dos usuários do sistema.

Existem diversas formas de se modelar as pessoas; alguns são métodos formais baseados em teorias das ciências humanas, outras são métodos informais de tradição cultural. Todas são válidas no sentido de criar modelos capazes de representar com a maior fidedignidade possível os grupos de usuários em questão.

Para as *personas* criadas no projeto de interação, foram utilizadas:

- Características físicas (como sexo, idade, aparência etc.);
- Características de personalidade e cognição – através dos métodos Astrologia, Estilos Cognitivos (RIDING; RAYNER, 1998)²⁸ e Dramática (PHILIPS; HUNTLEY, 1996)²⁹;
- Características Sócio-culturais (como profissão, estado civil etc.);
- Habilidades e competências (como fluência em línguas, informática etc.).

4.1.3.2.1 Validação da hipótese de *personas*

Uma vez definidas as *personas*, deve-se criar meios de confirmação e caracterização dessas através da pesquisa qualitativa, capazes de gerar modelos validados e bem precisos. Esses modelos permitirão inferir as metas dos usuários por eles representados.

Baseado no capítulo 4 de Cooper e Reimann (2003)²⁰ sobre pesquisa qualitativa com usuários, foram preparados questionários específicos para *subject matter experts*, *stakeholders*, compradores e usuários diretos, adaptados para seus equivalentes em relação aos motores de jogo: *game designers*, produtores, gerentes de TI e programadores. Ainda foram criados dois questionários adicionais para maior abrangência: jogador e programador de motores. Esses questionários encontram-se reproduzidos no Apêndice E.

4.1.3.2.2 Metas dos Usuários

A criação desses modelos de usuários ajuda a identificar as metas dos grupos de usuários que representam. Essas metas são classificadas como metas de vida, metas de experiência e metas fins.

- Metas de Vida são os objetivos que o usuário tem, independente da atividade que executa, ou produto que utiliza.
- Metas de Experiência identificam como o usuário gostaria de sentir-se ao utilizar o produto.
- Metas Fins mostram quais objetivos o usuário tem ao adquirir e utilizar um produto.

4.1.3.2.3 Escopo e escolha de usuário-foco

Feito esse estudo, escolheu-se um desses tipos de usuários para aprofundar o desenvolvimento com foco no mesmo.

Como o Interlab possui o projeto **enJine** de um motor voltado às necessidades acadêmicas, procurou-se atender seus usuários.

Inicialmente pensou-se no professor com intenção de criar jogos sérios, para explicar conteúdo e aumentar o interesse em suas disciplinas. Mas, como o **enJine** está sendo utilizado atualmente diretamente por alunos na disciplina de Computação Gráfica (CG), estes foram os usuários escolhidos, inclusive pela facilidade de comunicação com os mesmos, para se obter informações sobre sua experiência com o motor e incluí-los, através de suas sugestões, nas atividades de projeto (*participatory design*).

4.1.3.3 Definição de tarefas específicas

4.1.3.3.1 O jogo a ser criado

A definição do tipo de jogo em criação pela ferramenta direciona o tipo de tarefas que mais comumente os usuários precisarão executar através dela.

Com as informações anteriormente levantadas foi possível definir um gênero de jogo para o qual o motor de jogo mais se adapta: os jogos com fortes componentes de simulação, como por exemplo, um jogo de *pinball*.

Para as atividades com usuários (testes de usabilidade), foi utilizado o caso real do jogo que cada um desenvolveu. Os parâmetros passados como requisitos pela disciplina de CG foram (TORI et al, 2006)²:

- O ambiente deverá conter um elemento tridimensional estático de complexidade geométrica e influência na dinâmica do jogo (como colisão, por exemplo);
- O ambiente deverá conter um elemento tridimensional dinâmico, com partes que se movem relativamente (como rodas em um automóvel ou articulações em um robô);
- A câmera não poderá ser fixa em um ponto de vista (*POV*) apenas (deverá seguir um objeto ou chavear entre mais de um *POV*, por exemplo);
- O jogo deverá ter código para manipulação de malhas e polígonos (como geração de terrenos ou animação).

4.1.3.3.2 Projeto de tarefas

As seguintes tarefas foram levantadas através das entrevistas com os *stakeholders*:

Definir o Jogo:

- Definir personagem e jogo;
- Definir Etapas do jogo;
- Objetivo, Cenário e Personagem Dinâmico, Interação Monitor+Teclado

Programar:

- Modelar personagens e cenários;
- Montar personagem – partes do personagem (grafo de cena);
- Carregar modelos no **enJine**;
- Iluminação;
- Câmeras;
- Tratar Input do personagem;
- Caso o jogo requeira, Colisões;
- Tratar Input do personagem;
- Mostradores na tela: *Overlay*;
- *Gameplay*: Tratar interação entre objetos do jogo;
- Tratar interação entre objetos do jogo;

Refinamento:

- Criar fases;
- Som;
- “Filminhos” de *gameplay* e *trailers*;

Foram escolhidas as tarefas relacionadas com as fases iniciais de planejamento para fazer parte do cenário em questão. Estas tarefas foram tabuladas segundo modelo encontrado em Snyder (2003)²⁶ (Tabela 4.1).

4.1.3.3.3 Definição do contexto

Para completar a definição do cenário, é preciso acrescentar-lhe um contexto: condições temporais e de ambiente onde o cenário se desenrola.

Definiu-se como tal o primeiro contato do aluno com a ferramenta, após receber o arquivo do *website* e instalá-lo em seu ambiente de desenvolvimento pessoal. A primeira reunião de *brainstorm* do grupo já fora realizada, bem como a divisão de tarefas. O aluno estava sozinho, em sua casa após o período normal de suas aulas (por volta de 18h00).

Tabela 4.1 - Projeto de Tarefas para o cenário do protótipo em papel. A notação encontrada junto aos passos indica os respectivos widgets em papel.

Tarefa #	1-0001
Nome	Criar Projeto
Grupo	Planejamento
Meta e resultados:	Definir características do jogo a ser criado. Mínimo necessário: um nome, uma data de release, um gênero ou um exemplo de jogo similar.
Entrada:	Lista resumida de gêneros de jogo com exemplos;
Pressupostos:	Usuário (aluno) está de posse da lista de requisitos do projeto-jogo proposto (incluindo datas de entrega); Equipes já definidas.
Passos:	<p><1></p> <p>criar novo projeto: <V-1></p> <p>ícone modo produtor: <U-I-1>;<U-L-1>;<P-1></p> <p>digitar o nome: <P-2>;<U-I-1></p> <p>definir data de entrega: <P-3>;<TP-2>;<2></p> <p>Escolher gênero:</p>
Tempo de expert	
Instruções para Usuário	
Notas	
Tarefa #	1-0002
Nome	Definir Personagem
Grupo	Planejamento
Meta e resultados	Definir características estáticas e dinâmicas do personagem/avatar controlado pelo jogador. Mínimo necessário: um nome, um <i>placeholder</i> , lista de ações (movimento, combate, atuação, etc).
Entrada:	Lista resumida de tipos de ações comuns
Pressupostos:	Projeto criado no sistema; Gênero definido <1-0001>
Passos:	<p><2></p> <p>Projetar jogo: <V-2></p> <p>ícone modo projeto: <P-TP-0>; <3></p> <p>Objetos:<V-2.2></p> <p>ícone objetos: <P-TP-2>; <P-L-2>; <P-4></p> <p>digitar nome: <P-5></p> <p>editar atributos: <P-TP-2></p> <p>editar ações:</p>
Tarefa #	1-0003
Nome	Definir Etapas
Grupo	Planejamento
Meta e resultados	Definir macroetapas do jogo. Mínimo necessário: Criar diagrama contendo, pelo menos, tela de abertura, jogo e tela de encerramento.
Entrada:	Lista resumida de fases comuns de um produto-jogo
Pressupostos:	Projeto criado no sistema

(continua)

Tabela 4.1 - Projeto de Tarefas para o cenário do protótipo em papel. A notação encontrada junto aos passos indica os respectivos widgets em papel.

Tarefa #	1-0004
Nome	Programar Objetivo do Jogo
Grupo	Planejamento
Meta e resultados	Definir condição de fim. Mínimo necessário: Associar um atributo a <i>Victory Points (VPs)</i> e criar condição para fim de jogo, critérios de desempate.
Entrada:	Definição de <i>VPs</i>
Pressupostos:	Objetos criados e atributos associados <1-0006> <1-0001>
Tarefa #	1-0005
Nome	Definir Cenário
Grupo	Planejamento
Meta e resultados	Definir espaço(s) de jogo. Mínimo necessário: Criar um objeto "cenário", contendo áreas de jogo + um esboço visual desse cenário.
Entrada:	Explicação do escopo da tarefa (ver notas)
Notas	Essa tarefa não envolve criação visual do cenário, apenas definição de espaços lógicos, p.ex: gols, casas, <i>checkpoints</i> , etc
Tarefa #	1-0006
Nome	Definir classes de objetos de jogo
Grupo	Planejamento
Meta e resultados	Definir objetos e seus atributos. Mínimo necessário: Definir todos os principais atributos do objeto avatar.
(SNYDER, 2003) ²⁶ (conclusão)	

4.2 Análise Comparativa

4.2.1 Jogo do Museu

Nakamura et al. (2003a)¹³ descrevem a experiência de se criar um jogo educativo utilizando-se um motor comercial.

Figura 4-1 - Screenshot do jogo do museu, mod-game utilizando o Unreal Engine.

O produto-jogo é um *adventure* em ambiente virtual tridimensional, visto em primeira pessoa, representando um museu de arte. O jogador precisa inicialmente escolher e instalar quadros para a próxima exposição. Durante esse processo, descobre que uma das obras foi roubada; precisa encontrar o ladrão e recuperar o quadro, antes da abertura da exposição.

O motor escolhido para o desenvolvimento foi o *Unreal Engine*, devido ao fato de possuir uma linguagem de *script* orientada a objetos, similar a C++ ou Java, chamada *UnrealScript*.

Dentre as dificuldades encontradas na implementação desse projeto, as principais foram as seguintes (NAKAMURA et al., 2003a)¹³:

- Compatibilidade do motor com ferramentas de modelagem 3D – não havia uma ferramenta integrada para esse fim, nem mesmo conversores confiáveis para os formatos mais conhecidos. Também

não estavam disponíveis bibliotecas com modelos prontos utilizáveis nesse projeto;

- A linguagem *UnrealScript*, na tentativa de atender requisitos característicos de motor (estados dos objetos, sincronização de estados, etc.), resultou em uma linguagem complexa, de difícil leitura e aprendizado.
- Apesar disso, a especificação da linguagem *UnrealScript* é menos formal, e mais suscetível a erros de compilação, tornando-se pouco confiável.
- Outro problema com a *UnrealScript* é que sua estrutura de classes viola princípios básicos do paradigma de orientação a objetos, como encapsulamento e herança.
- Por fim, foi também detectada deficiência na flexibilidade em troca de eventos e mensagens entre objetos de jogo.

A partir dessa análise, criou-se uma série de requisitos desejáveis para o desenvolvimento de um motor voltado para o meio acadêmico e o desenvolvimento de jogos com fins educativos:

- Deverá atuar como um *framework* para aplicações, fornecendo uma infraestrutura personalizável, tal como a do desenvolvimento de *mod-games*, aumentando assim o acesso a usuários com menos capacitação de técnica e de programação;
- Deverá possuir ferramentas auxiliares confiáveis para geração de conteúdo (como mapas, modelos visuais, *scripts* de “Inteligência Artificial” (IA) etc.);
- Deverá possuir uma biblioteca de recursos básicos pré-existentes e prontos para utilização (como modelos visuais, trechos de som, imagens, *scripts* de IA, etc.), orientados ao meio educacional (como equipamento de sala de aula, laboratório, profissionais do ensino, alunos etc.).
- Deverá implementar um mecanismo flexível para troca de eventos e mensagens entre objetos de jogo, uma vez que ambientes de simulação requerem diversas formas de interação complexas entre objetos virtuais.
- No caso de incluir uma linguagem de *script*, essa deverá ser de alto nível, ter mecanismos robustos de detecção, informação e correção de erros, bem

como facilidade de depuração. Deve também ter uma implementação rígida para evitar comportamentos inesperados.

4.2.2 enJine

Baseado na experiência do jogo do Museu com motores comerciais, o Interlab iniciou um projeto para construção de um motor próprio a partir das ferramentas disponíveis. Essa proposta de desenvolvimento de motor tem por característica estar inserida no ambiente acadêmico, e por premissa os seguintes objetivos (NAKAMURA et al., 2003b)³⁰:

- Pesquisar o desenvolvimento e operação de um motor de jogo, assim como seus requisitos;
- Estudar diversas arquiteturas de motor de jogo, de forma a acelerar o desenvolvimento;
- Avaliar métodos de Engenharia de Software e sua aplicação na criação de jogos e motores.

Além dos requisitos levantados durante o projeto Jogo do Museu, acrescentaram-se os seguintes para esse projeto:

- Deverá ter suporte à comunicação em rede com entrega de dados confiável e não confiável;
- Deverá ter suporte à replicação de objetos entre computadores conectados em rede;
- Deverá fornecer serviços de gráficos tridimensionais, som e entrada de usuário;
- Deverá fornecer uma camada de abstração para os serviços acima, de forma que o desenvolvedor do jogo seja capaz de trabalhar em entidades no domínio do jogo em vez de entidades de baixo nível;
- Deverá ter implementação desassociada de bibliotecas de programação de baixo nível, permitindo a troca dessas bibliotecas sem afetar a interface do motor.

Figura 4-2 - Screenshot de FootBot, produto-jogo criado sobre o enJine.

De forma a permitir um desenvolvimento mais otimizado, tal motor foi desenvolvido em paralelo com um primeiro projeto de jogo sobre o mesmo. O motor foi batizado **enJine** e o primeiro jogo a utiliza-lo, **FootBot**.

A plataforma inicial de desenvolvimento escolhida para ambos foi **Java**, por suas características com relação à comunicação em redes, paralelismo (*multi-threading*), independência de plataforma, orientação a objetos e suporte a gráficos 3D (através da **API Java3D**).

As primeiras etapas do projeto foram relacionadas com a parte estrutural do motor, preocupando-se com características muito importantes, como distinção clara entre lógica de jogo e representação gráfica, desacoplamento de bibliotecas de baixo nível (como o **Java3D**), troca de mensagens de alto nível entre objetos de jogo, separação clara entre **enJine** e **FootBot**.

Entretanto, nessa fase, não houve uma preocupação maior com um ambiente de programação otimizado para usuários estudantes de computação. É com o projeto **Game Wrench** que inicia-se esse processo de aproximação com seus usuários.

4.2.3 Motores de jogo com Interface de Desenvolvimento

Utilizando-se o método de avaliação de inspeção de usabilidade por heurísticas, foram analisadas duas ferramentas de *software* que seguem o mesmo princípio de criar um ambiente integrado interativo (interface+motor) para desenvolvimento de jogos: **Game Maker v.6.1** e **3D Game Studio v.A6**.

Os resultados dessa primeira iteração de avaliação do projeto, obtidos utilizando-se os personagens e cenários desenvolvidos, encontram-se no Anexo D.

Dois problemas comuns encontrados em ambas as ferramentas foram:

- Necessitar de tutoriais externos para aprendizado da ferramenta;
- Procurar “esconder” a parte de programação do usuário iniciante, desvinculando a programação de sua linguagem de script da interface gráfica principal.

As observações feitas sobre esse processo geraram novos requisitos para o sistema.

4.2.3.1 Game Maker

Com a ferramenta **Game Maker**, um usuário não-avançado consegue rapidamente passar suas idéias para um jogo de computador. É muito útil para jogos simples (como demonstrações e jogos para propaganda) e dentro da fase de prototipagem de projetos maiores.

A licença de software do **Game Maker** possui uma versão gratuita, que permite boa parte da funcionalidade necessária, e uma versão registrada não muito cara (de US\$20 para um usuário até US\$330 para um número ilimitado de usuários). O software está facilmente disponível e tem baixos requisitos de hardware. Mas há a restrição de possuir somente versão para plataforma **Windows**.

Figura 4-3 - Tela Inicial do Game Maker v.6.1

A ferramenta possui as seguintes características:

- Programação visual, orientada a eventos, que permite reduzir o tempo de aprendizado de gramática da linguagem e reduzir a taxa de erros de sintaxe; Em compensação, isto implica em uma limitação de eficiência. A diagramação de um número muito grande de ícones também tende a ser confusa. Problemas freqüentes são relacionados com o fato de a ferramenta focar na apresentação do jogo, escondendo aspectos da lógica do jogo. São resolvidos pela criação e tratamento de variáveis na *Game Maker Language* (GML).
- A rapidez de implementação permite o desenvolvimento incremental e interativo. Existe suporte à combinação entre jogos, através do recurso “merge game”;
- Os requisitos de *hardware* do **Game Maker** são poucos, uma vez que tem seu foco em jogos simples, bidimensionais. Mas existem problemas de motor,

relacionados à detecção de colisão e performance, que não podem ser resolvidos, uma vez que o código não é aberto.

- Boa documentação, com manual disponível através do menu *Help*, tutoriais e exemplos disponíveis no *site* oficial (www.gamemaker.nl). Apesar da disponibilidade do manual do *software* traduzido para o Português brasileiro, os tutoriais são em inglês e boa parte do aprendizado acontece através dos tutoriais específicos (plataforma, tiro, labirinto, etc.), que permanecem no original.
- *GML* é a linguagem de programação do **Game Maker**. Apesar de relativamente poderosa, seria melhor adotar uma linguagem não-proprietária, como **C**, **Java** ou **Python**, e criar uma biblioteca de funções específicas do **Game Maker**, permitindo aos usuários, que precisem desses recursos, reaproveitamento do conhecimento prévio e adquirido.

Ao instalar o *software*, o primeiro contato que o usuário tem com a ferramenta é a tela inicial apresentada na Figura 4-3. Apesar da proposta de ser um *software* intuitivo, essa tela inicial não oferece nada para orientar o usuário nesse primeiro uso.

O usuário que inicia o processo de exploração da interface está sujeito a cometer diversos erros até entender os conceitos de salas, objetos e eventos da ferramenta. É somente através de recursos separados do ambiente, ou seja, dos tutoriais encontrados no *site*, dos exemplos encontrados no diretório de instalação e do manual apresentado em janela separada, que a aprendizagem realmente acontece.

A implementação de programação visual levou a interface a se segmentar em barras de ícones separadas por abas. Essa solução é complicada, pois obriga o usuário a não somente decorar o significado de dezenas de ícones, mas também memorizar em qual aba o ícone desejado se encontra. Outra desvantagem é a constante alternância entre essas abas.

Alguns recursos de interface comuns também fazem falta, como:

- A possibilidade de *drag-and-drop* da área de recursos para a área de trabalho;
- O *double-click* nas pastas de recursos, em busca de recursos pré-existentis disponíveis;
- A capacidade de *undo* através de atalhos tradicionais como CTRL-Z;

4.2.3.2 3D Game Studio

A ferramenta **3D Game Studio** também integra a funcionalidade de um motor de jogo em uma interface visual que agrega o controle de recursos multimídia e a criação de jogos através de manipulação direta.

Como trata-se de um motor para jogos tridimensionais, o grau de complexidade é extremamente aumentado para essa finalidade. Por exemplo, basta acrescentar alguma complexidade à cena em criação que o tempo de compilação passa de alguns segundos para vários minutos.

Figura 4-4 - Tela Inicial do 3D Game Studio v.A6

Os problemas são similares aos detectados no **Game Maker**, mas agravados pelas dificuldades intrínsecas do 3D. A interface inicial é ainda menos informativa, o que pode ser conferido na Figura 4-4. A possibilidade exploratória é quase nula, e tarefas comuns como aplicação de texturas a faces de objetos estão relativamente “escondidas” na ferramenta. Ou seja, a dependência de tutoriais externos para o aprendizado da interface é ainda maior.

A estrutura de comunicação entre tarefas executadas passa pela compilação, armazenamento e importação de arquivos. Esse fluxo de trabalho não é otimizado, incluindo “gargalos” e essa dificuldade é agravada pelo uso de nomenclatura não-familiar ao usuário (*WMB, map entity* etc.).

4.3 Estudo de Usabilidade com Protótipo em Papel

4.3.1 Criação do protótipo

A partir das informações levantadas em fases anteriores, o protótipo em papel foi criado, com os seguintes princípios em mente:

- Resolver o problema relacionado à dependência de tutoriais externos encontrado nos softwares analisados, integrando recursos de tutorial dentro da interface de desenvolvimento, evitando, no entanto, recair em sobreposição de conteúdo de janelas;
- Destacar os elementos de jogo no projeto do produto-jogo auxiliado pelo motor assim como o controle e acompanhamento de recursos e projetos, mesmo porque são elementos fundamentais estranhamente relegados em motores atuais;
- Aumentar o desempenho dos usuários automatizando tarefas de programação, mas tomando o cuidado de evitar a tendência de ocultar código do usuário, mesmo porque se trata de um motor utilizado para aprendizado de técnicas de programação de Computação Gráfica e Jogos Eletrônicos.

A largura de um protótipo está relacionada com a quantidade de funcionalidades do produto final que estarão aparentes para os testes. Sua profundidade indica dentre essas tarefas aparentes, quão interativas elas se apresentam.

Por exemplo, um “Menu” de um protótipo pode apresentar três dentre cinco opções que o produto final terá. Esta é a largura deste menu dentro do protótipo. Se apenas

duas destas três opções estão habilitadas para abrir caixas de diálogo no protótipo e receber entradas do usuário, esta é sua profundidade.

Esta definição de escopo do protótipo é importante e ainda mais acentuada em protótipos em papel, pois os “*widgets*” em papel deverão ser construídos com base nesta informação.

Tendo construído o cenário e projeto de tarefas apresentados em 4.1.3, um primeiro protótipo em papel foi construído, seguindo as orientações de Snyder (2003)²⁶. Importante ressaltar uma transgressão significativa a essas orientações no que diz respeito à precisão de tamanho, que, segundo a bibliografia, não é necessária e pode atrapalhar. Mas, como se procurou estudar o efeito de sobreposição de telas e espaços, o protótipo foi criado sobre uma estrutura inicial em papel milimetrado, para se ter noção de espaço em *pixels* que cada parte ocuparia. (Figura 4-5).

Para atender o princípio da integração de tutoriais ao *Integrated Development Environment (IDE)*, evitando sobreposição, foram reservados espaços fixos na interface para cada funcionalidade, com base em um espaço inicial mínimo de 1024x768 *pixels* (padrão XGA, bastante razoável para o padrão de ambiente computacional atual). São eles:

- Área de Diálogo: “*Yellow Helper / Orange Alert*”. 240x120. fixa. sup.esq.
- Área de Recursos. Esq.
- Barra de ícones: para info. / navegação. Centro sup.
- Área de Trabalho principal. Centro.
- Área de *preview*. 320x240. fixa. sup. dir.
- Área de Informação. Dir.
- (Barra de Menu) opcional
- (Barra de status) opcional

Figura 4-5 - Protótipo em papel. A estrutura em papel milimetrado permite uma noção de pixels utilizados por cada espaço projetado. Sobre esta estrutura, widgets de papel adesivo representando painéis, botões, tooltips e campos de entrada são sobrepostos de forma a permitir a interatividade com o protótipo.

Os espaços “*Yellow Helper / Orange Alert*” e “*Red Tooltips*” (“*pop-ups*” semitransparentes), atuarão como “*wizards*”, somente enquanto o usuário seguir a sequência indicada, não segurando o foco, nem impedindo ações avançadas.

Ao mesmo tempo, para o projetista do jogo as atividades e tarefas de alto nível e as de programação estão em esferas separadas, dispensando exibição simultânea na tela. Para isso, foram criados modos de operação com visões independentes para cada espaço reservado, alternadas através da barra de ícones.

Como se trata de um motor utilizado para projetos acadêmicos, apesar de apresentar facilitadores visuais, deverá evitar “esconder” a parte de programação e código do usuário. Primeiramente, porque é impossível criar qualquer jogo mais complexo sem este aspecto, depois porque se tratam de usuários com interesse no

aprendizado de programação. A eficiência da ferramenta no processo de aprendizado foi objeto de estudo das sessões de teste.

Vislumbrando um paradigma de interação mais moderno, o protótipo inicial não apresenta elementos tradicionais como a barra de menu ou a barra de status. Tendo em mente que a ferramenta deverá se encarregar de garantir automaticamente a integridade de informações do trabalho do usuário, a real necessidade de tais elementos foi outro objeto de estudo durante as sessões de teste.

4.3.2 Sessões de Teste

Antes dos testes com usuários reais, o “*core team*”, grupo que conduziu as sessões de teste fez um teste-piloto com um pesquisador convidado. Verificou-se a importância de se preparar uma introdução aos testes através de um informativo e um roteiro impressos (Apêndice A).

A divisão de tarefas entre os examinadores é feita da seguinte forma:

- O “facilitador” ficou encarregado do diálogo com o usuário, recebendo-o, conduzindo o teste, decidindo o grau de informação que pode ser passada em caso de dúvidas, e procurando “ligar o *verbose-mode* do usuário”, ou seja, fazê-lo dizer o que está pensando;
- O “computador” ficou encarregado da atualização das telas, ou seja, fazer a troca de *widgets* em papel. Uma técnica empregada foi a de afastar o protótipo do usuário durante esta atualização, e empurrar de volta quando estivesse pronta para a próxima interação do usuário. Essa dinâmica foi explicada ao usuário, e sempre que este indicava a próxima interação, o facilitador perguntava sobre suas expectativas sobre a resposta do sistema àquela ação; ao retornar, novamente o usuário era questionado sobre o atendimento a essas expectativas;
- Um examinador pode ser o “organizador”, responsável pela organização dos diversos *widgets* em papel, de forma a agilizar a interação;

- O(s) “observador(es)” ficou(ficaram) encarregado(s) de registrar todas informações relevantes sobre o que ocorre durante as sessões de teste. Ao final de cada sessão, numa reunião somente com os examinadores, todas observações assim registradas foram discutidas de forma a documentar e criar modificações ou alternativas possíveis ao protótipo antes da próxima sessão de teste.

Foram realizadas quatro sessões de testes com usuários distintos de aproximadamente uma hora e meia, sendo que uma ocorreu em duas partes em dias consecutivos.

4.3.3 Observações dos Usuários

As seguintes funcionalidades e sugestões de solução foram elaboradas durante os testes, com auxílio dos usuários:

- Inclusão de informações dos marcos (nome e data) na linha do tempo;
- Campo de informação sobre os ícones de modos da interface (produção, projeto e desenvolvimento);
- Campo para associação de teclas às ações e movimentos de personagens do jogo, em estrutura em árvore e diagrama do dispositivo físico (teclado e mouse);
- Melhorias na área de *preview*, com auxílio de eixos de coordenadas e botão de Play;
- Inclusão de visualização de código na área de *preview*;

4.3.4 Conclusões e Diretrizes a Partir dos Testes

- A ausência dos elementos tradicionais Barra de Menu e Barra de *Status* causa desconforto, mas, por enquanto, sua presença não foi essencial para

execução de nenhuma das tarefas propostas. Em compensação, é preciso acrescentar ações de Desfazer (*undo*), Refazer (*redo*) e Missão cumprida (parte do *save*). Outras funcionalidades interessantes que poderão ser acrescentadas futuramente são a de auto-geração de documentos e impressão e o *preview* do jogo (desde que não acarrete em perda de desempenho do sistema para o programador);

- A tarefa inicial (<1-0001> Criar Projeto) não é familiar aos usuários. Eles não fizeram isso formalmente em sua experiência com o **enJine**. É uma tarefa boa para apresentar o comportamento da interface, seus modos, ajudas e visões. É, portanto, também onde surgem dúvidas sobre estas informações. Nem todos os usuários reconheceram este funcionamento naturalmente, e a forma de informá-los deve ser aprimorada. Se a interface começar no modo produção, será uma ação a menos, mas prorroga o contato com este comportamento;
- Os estilos dos usuários variaram bastante: uns seguiram mais os auxílios (*helper*), outros seguiram a área de recursos ou a área de informação. Isso comprova uma flexibilidade boa desta interface e de suas áreas reservadas predefinidas. Em compensação, sempre haverá espaços superutilizados competindo por *pixels* dos espaços subutilizados: espera-se reduzir este problema com a adoção de ícones/setas de “minimizar” ou “esconder”. Em geral, os usuários terminaram o teste com uma noção correta da função de cada espaço;
- A interface “funciona” bem, encoraja exploração e guia o usuário em tarefas de planejamento que antes eram feitas informalmente, como o controle do projeto e organização de elementos de jogo e recursos multimídia;

4.3.5 Considerações sobre o processo:

- Alguns usuários tendem a querer “explorar a interface” no começo do teste. O facilitador deve se conter neste momento, mas tentando guiar o usuário para o foco do teste, lembrando da lista de tarefas;

- Devido ao horário dos testes, procurou-se ter petiscos durante as sessões, o que as deixou com um clima descontraído muito positivo.
- A metáfora gráfica/visual é um ponto crítico. Os *widgets* em papel têm cores que ora representam funcionalidade, ora não.
- É importante preparar previamente uma série de “campos de entrada”, os “X” de *check-box*, etc., pois serão usados com frequência, agilizando assim a interação com o protótipo.

5 GAME WRENCH* – ESPECIFICAÇÃO DE AMBIENTE INTERATIVO PARA MOTORES DE JOGOS

Esse capítulo resume na forma de requisitos os resultados coletados durante toda a análise feita nos capítulos anteriores.

5.1 *Requisitos de Usuário*

- U1 - Usuário é um aluno cursando disciplina de graduação sobre Computação Gráfica ou Jogos Eletrônicos;
- U2 - Recebeu orientação sobre o projeto de jogo eletrônico: a ser entregue ao final do quadrimestre, em grupo e valendo nota;
- U3 - O grupo já se reuniu e chegou a algumas decisões sobre como será o jogo (sessão de *brainstorming*).

5.2 *Requisitos de Usabilidade*

- Q1 – Prioridade para tempo de aprendizado (evitar a necessidade de tutoriais ou manual externo ao motor) e desempenho do usuário (um usuário médio deverá ser capaz de criar um jogo simples- como “**pacman**” ou “**tetris**” – em uma sessão de desenvolvimento);
- Q2 – Identificar as metas do usuário através de “*wizards*” e manter controle sobre o andamento do projeto;

* Tool, usually operated by hand, for tightening bolts and nuts.

"**wrench**." *Britannica Concise Encyclopedia* from Encyclopædia Britannica Premium Service.
<<http://www.britannica.com/ebc/article?tocId=9382969>>
[Accessed June 2, 2005].

O termo *wrench* foi escolhido por analogia às chaves e ferramentas utilizadas para acesso especializado e eficiente às partes de motores automotivos.

- Q3 – Sugestão de escolhas como próximos passos em direção às metas.
- Q4 – Permitir ao desenvolvedor do jogo trabalhar em entidades no domínio do jogo em vez de entidades de baixo nível;

5.3 *Requisitos Funcionais*

- F1 – Deverá implementar um mecanismo flexível para troca de eventos e mensagens entre objetos de jogo,
- F2 – Suporte a diversas formas de interação complexas entre objetos virtuais;
- F3 – Deverá fornecer serviços de gráficos tridimensionais, som e entrada de usuário;
- F4 – Encarregar-se de garantir automaticamente a integridade de informações do trabalho do usuário, como por exemplo, a auto-gravação do progresso do usuário nos seus projetos;
- F5 – Interface dividida em visões nos modos: Produção, Projeto e Desenvolvimento;
- F6 – Controlar elementos de jogo e recursos;
- F7 – Manter separados lógica de jogo e apresentação visual, mas visivelmente ligados;
- F8 – Documentação automática para produtores, desenvolvedores e jogadores, através de “*wizards*”;
- F9 – Configuração visual de máquinas de estado e *scripts* para o comportamento de objetos do jogo;
- F10 – Aplicar detecção e correção de erros para evitar comportamento imprevisível advindo de ambigüidade nas definições;
- F11 – Suporte a trabalho colaborativo multiusuário;
- F12 – Usar a definição formal de jogo como modelo de desenvolvimento;

5.4 Requisitos Não-Funcionais

- NF1 – Deverá atuar como um “*framework*” para aplicações, fornecendo uma infra-estrutura customizável, interoperável com outros motores, *APIs* e ferramentas de terceiros;
- NF2 – Deverá possuir ferramentas auxiliares confiáveis para geração de conteúdo (como mapas, modelos visuais, *scripts* de IA, edição 3D, etc.);
- NF3 – Deverá possuir uma biblioteca de recursos básicos pré-existentes e prontos para utilização, (como modelos visuais, trechos de som, imagens, *scripts* de IA, etc.) orientados ao meio educacional (como equipamento de sala de aula, laboratório, profissionais do ensino, alunos, etc.).
- NF4 – Incluir uma linguagem de *script* de alto nível, com mecanismos robustos de detecção, informação e correção de erros, bem como facilidade de depuração, com uma implementação rígida para evitar comportamentos inesperados;
- NF5 – Deverá ter implementação desassociada de bibliotecas de programação de baixo nível, permitindo a troca dessas bibliotecas sem afetar a interface do motor;
- NF6 – Deverá ter suporte a comunicação em rede com entrega de dados confiável e não confiável;
- NF7 – Deverá possuir *display* de resolução mínima XGA (1024x768).

5.5 Requisitos de Dados

- D1 – Deverá ter suporte a replicação de objetos entre computadores conectados em rede;
- D2 – Suporte a importação de imagens adquiridas;
- D3 – Suporte a imagens vetoriais;
- D4 – Suporte a modelos 3D sintetizados;
- D5 – Suporte a arquivos de animação 3D;

- D6 – Usar *XML* para manter os dados de projeto, de forma a permitir interoperação do “*framework*” com bibliotecas e ferramentas de terceiros.

5.6 Requisitos de Ambiente

- A1 – Ambiente computacional padrão.

6 IMPLEMENTAÇÃO

Esse capítulo trata das soluções específicas adotadas para uma primeira implementação da interface **Game Wrench**.

6.1 *Game Markup Language*

A linguagem *Game Markup Language* (**GAML**) é o modo com o qual pretende-se fazer a ligação entre os conceitos de alto nível de jogo que foram estudados e o modelo estrutural de um motor de jogo, independentemente da linguagem de programação na qual este tenha sido desenvolvido.

Conforme apresentado na Seção 3.5, a linguagem **GAML** utiliza a padronização *XML Schema Language (XSD)* para definir uma estrutura de dados de alto nível para definir jogos em geral.

Sendo um padrão *XML* e estando suas regras definidas em um esquema *XSD*, esse padrão permite que um arquivo assim validado seja lido facilmente por qualquer aplicativo ou linguagem de programação. Também possibilita a leitura e edição humana de forma simplificada. Isso torna a linguagem **GAML** uma boa forma de comunicação entre o ambiente de desenvolvimento **Game Wrench** e qualquer motor ou *API* necessários para implementação dos jogos ali definidos.

A definição da **GAML** em *XSD* está reproduzida no Apêndice B.

Nesta implementação, o **Game Wrench** irá salvar as informações de projeto de jogo no formato **GAML**. Será então desenvolvido um aplicativo que verificará as informações pertinentes desse arquivo e as converterá em código de programação, **Java** no caso, utilizando o **enJine**. Isto ajudará a comprovar a facilidade de comunicação entre o **Game Wrench** e motores de jogo através da **GAML**.

De forma similar, outras *APIs* ou motores de jogo poderão facilmente seguir o mesmo procedimento, aprimorando cada vez mais o sistema, e criando um suporte para novas versões e funcionalidades.

Tendo o objetivo de guardar informações de alto nível de um projeto de jogo, o formato **GAML** pode criar arquivos complementares para fins distintos.

A Figura 6.1 mostra de forma gráfica (utilizando o *plugin WTP - Web Tools Plugin* - do *IDE Eclipse*) a estrutura de um arquivo **GAML** para armazenar informações ou definir um Sistema de Jogo. Para documentar um baralho normal de 52 cartas, basta criar um `<Token>` com duas `<Dimensão.Token>` : com `nome="naipe"` e `nome="número"`. Para melhorar o sistema, inclui-se ainda um `<Universo>` contendo um `<Ambiente>` com três `<ÁreaDeJogo>` : `maço`, `mesa` e `descarte` (ver arquivo `Baralho52.xml` no Anexo B). Para acrescentar dois coringas a esse sistema, basta editá-lo para que `<Dimensão.Token>` `naipe` e `número` do `<Token>` tenham o elemento `dinâmico="true"`, e que `<ÁREADEJOGO>` `maço` e `descarte` possuam elemento `valorMáximo="54"` (ver arquivo `Baralho54.xml` no Apêndice B) .

A partir de um sistema de jogo, diversos jogos podem ser definidos, com algumas modificações e acrescentando-se jogadores e regras. A Figura 6-2 - mostra a estrutura da **GAML** para a definição de jogos. Para documentar um jogo de truco, precisa-se fazer referência ao `SistemaDeJogo Baralho52`, e acrescentar informações para dois elementos `<Equipe>` de dois `<Jogador>` cada e o `<LivroDeRegras>` do jogo.

Figura 6-1 - Estrutura em Schema XSD da GAML para armazenar informação de sistemas de jogo.

A Figura 6-3 - mostra que o GAML ainda pode ser utilizado para armazenar diversas instâncias de jogos - <Partida> - dentro de uma estrutura chamada <campeonato>. Para documentar um campeonato de truco entre 256 equipes, pode-se fazer referência ao Jogo Truco, acrescentar informação dos elementos <equipe>, sua pontuação e até mesmo cada <movimento> realizado pelos jogadores em cada <turno>.

A Figura 6-4 mostra a estrutura do GAML para manter atualizado um projeto de produto-jogo sendo desenvolvido. Faz-se referência a um Jogo, e adiciona-se ao projeto <nome>, <estado> de finalização e as datas importantes <MarcoIntermediário>, incluindo a <DataDeEntregaFinal>.

Figura 6-2 - Estrutura em Schema XSD da GAML para armazenar informação de jogos.

Figura 6-4 - Estrutura em Schema XSD da GAML para armazenar informação de um projeto de produto-jogo

Para a integração do **enJine** com o **GAML**, foram implementadas classes para fazer a comunicação (leitura, manipulação e escrita) com arquivos XML de forma simplificada. Como o **Game Wrench** irá modificar a estrutura deste arquivo com frequência, foram utilizadas classes que utilizam *Document Object Model (DOM)* para estas operações.

É necessário, portanto, certo cuidado com arquivos muito grandes, complexos ou recursivos que podem ocupar toda a memória do sistema. Para tais casos, alternativas usando *Simple API for XML (SAX)* seriam mais indicadas.

6.2 Game Wrench v0.1 em Java Swing

Existem diversas bibliotecas utilizadas para a especificação e programação de interfaces em computadores (**QT** para **Linux**, **XUL** para **Firefox**, **SWT** para **Eclipse**, **AWT** e **Swing** para **Java**). Escolheu-se a **API Swing** do **Java** por suas

características de independência de plataforma e de sistema operacional e controle sobre o resultado final (a **AWT**, por exemplo, faz requisições ao sistema operacional – SO para os seus *widgets*, cada SO usa o seu padrão visual para o desenho na tela). A **Swing** acompanha qualquer instalação de **Java** para ambiente *desktop* (*Java Standard Edition – JSE* e *Java Enterprise Edition – JEE*).

6.2.1 O padrão Model-View-Controller e o Java Swing

O **Java Swing** é inspirado em um padrão de projeto (*design pattern*) bastante utilizado, o *Model-View-Controller*. Ou seja, a separação dos objetos de software em componentes relacionados a sua lógica/funcionalidade, sua aparência e a interface com outras entidades (como o usuário).

Como se tratam de objetos visuais de interface com o usuário, seus projetistas verificaram ser inviável a separação entre apresentação e interação com usuário, recaindo numa arquitetura chamada de Modelo Separável – *Separable Model Architecture* (FOWLER, 1994)³¹.

Esta arquitetura permite ao programador, por um lado, trabalhar com programação a partir de modelos, onde diversos *widgets* visuais podem consultar uma mesma fonte de informação e manterem-se sincronizados; e por outro lado, a capacidade de escolha e troca de projeto visual (*pluggable look-and-feel*).

6.2.2 Implementação do Yellow Helper / Orange Alert

As características da *API Swing* facilitam a programação do comportamento desejado para tutoriais integrados do **Game Wrench**.

Para cada tutorial, pode-se criar um modelo contendo os passos de cada tarefa, acompanhando o comportamento do usuário, com informações como: o que este já realizou das tarefas programadas; se ele está seguindo as recomendações do tutorial; e se deve ser alertado sobre algo.

Todos os elementos da interface podem dessa forma acessar este modelo para saber como devem se apresentar ao usuário (por exemplo, se um *Red Tooltip* associado a um campo de entrada deverá estar visível, ou qual deve ser a cor e o texto apresentado no *Yellow Helper/Orange Alert*).

Na versão inicial atual, o comportamento desse *widget* especializado segue a estrutura de um *wizard*, com apenas um caminho possível e controlado por um botão “*Next*”, mas a intenção é que na versão final possa ter um comportamento mais próximo a uma máquina de estados, com mais de uma tarefa possível a cada estado, assim como ocorreu com o protótipo em papel no teste de usabilidade.

6.2.3 Caso exemplo: Gerar código do enJine através da interface

Para exemplificar a funcionalidade da interface **Game Wrench**, o comportamento dos tutoriais integrados e a comunicação com outros módulos de software através do **GAML**, será realizado o teste apresentado a seguir.

1. Iniciar o **Game Wrench** e analisar sua aparência.
2. O usuário será conduzido pelo *Yellow Helper* a mudar de modo de visão para produção e incluir informação do nome de seu projeto de produto-jogo;
3. Neste momento, o **Game Wrench** deverá carregar um modelo vazio de **GAML** e renomeá-lo com o nome escolhido para o projeto. Visualizar esse arquivo;
4. Ao usuário, ainda pelo *Yellow Helper* será solicitado mudar para modo de visão projeto e criar um objeto de jogo. O usuário nomeará, por exemplo, um objeto “*SilverBall*” e associará um modelo 3D *placeholder* “esfera”;
5. O sistema deverá acrescentar automaticamente ao arquivo **GAML** estas informações. Visualizar novamente o arquivo;
6. Classes Java que se utilizam do **enJine** deverão ser capazes de ler o **GAML** e escrever código automaticamente para criação deste objeto dentro de um mundo tridimensional de jogo. Executar tal aplicativo e visualizar esse código;

7. Opcionalmente, o item acima poderá ser automatizado, integrado ao **Game Wrench** e o código gerado apresentado para visualização na Interface sem, contudo ter opção de edição.

7 RESULTADOS

A pesquisa bibliográfica e adoção da definição de Walther para o conceito de diversão e jogo foram fundamentais para o embasamento desse trabalho, e permitiram uma melhor compreensão das características do produto jogo eletrônico, do produto motor de jogo, bem como seu relacionamento com outros conceitos e o projeto de interação para o ambiente **Game Wrench**.

Esta contribuição foi exemplificada através da aplicação desta definição para atacar problemas tradicionais relacionados ao tema, como seu relacionamento com outros tipos de mídia e sua diferenciação de outros conceitos como brincadeira, produtos-jogo, esporte, etc.

O levantamento das posições de trabalho na indústria de jogos eletrônicos norte-americana e o detalhamento através da classificação de usuários primários, secundários e terciários permitiram uma visão abrangente do universo de desenvolvimento de jogos comerciais. Ou seja, possibilitaram uma visão generalista, sobre a qual se pode fazer um recorte mais confiável e válido.

O trabalho gerou questionários desenvolvidos com base nas diretrizes apresentadas em (COOPER; REIMANN, 2003)²⁰ que podem ser úteis em outras pesquisas com finalidades similares, sendo instrumento para uma pesquisa qualitativa. Esses artefatos permitem ainda a obtenção de informação privilegiada e comprovação de hipóteses sobre esses usuários. Esses questionários encontram-se no Anexo E.

As *personas* e cenários criados são outros artefatos de projeto e podem ser modificados, aprimorados e reutilizados em projetos de software. Os resultados obtidos com a utilização dos mesmos em inspeções de usabilidade sobre ferramentas similares foram diretamente utilizados na especificação e confecção do protótipo da interface **Game Wrench**.

O método de inspeção de usabilidade Avaliação Heurística permitiu realizar uma análise comparativa sobre as ferramentas **Game Maker** e **3D Game Studio**. Estas

ferramentas foram escolhidas por serem interfaces gráficas para acesso às funcionalidades de seus motores de jogo. Dentre os problemas levantados, a necessidade de tutoriais separados do ambiente e a tentativa de evitar o uso de programação de código na interface, foram aspectos que serviram de diretrizes para a criação do **Game Wrench**.

O teste de usabilidade com protótipo em papel foi uma atividade muito produtiva, essencialmente coletiva, de onde foram levantadas diretrizes diretamente com os usuários-foco. Tais informações, como a melhor forma de ter-se visualização prévia de resultados na interface, quais notificações são necessárias para o usuário, qual a sequência lógica de ações, dificilmente seriam conseguidas através de qualquer outro método de desenvolvimento (encontrados na Engenharia de Software, por exemplo).

Durante a fase de protótipo, aconselha-se a não ponderar o esforço de desenvolvimento envolvido na tarefa de transformar o protótipo em produto final. Mas, uma vez tendo cumprido seu papel, o protótipo precisa ser descartado, quando essa grande barreira aparece. Para que a interface operacional do **Game Wrench** pudesse representar o comportamento de seu protótipo em papel, uma boa dose de arquitetura de software foi necessária. As soluções encontradas para realizar essa tarefa incluem o uso de *XML* e da estrutura *separable model architecture* do **Java Swing**.

O **GAML** mostrou-se como uma solução adequada para os requisitos do **Game Wrench**. Auxiliado pela programação visual do **WTP** do **Eclipse**, foi agregando diversos aspectos, desde a definição de sistemas de jogo, os elementos necessários para um jogo, até o controle de recursos e de projetos de jogo. Como é *XML*, é legível e permite edição por seres humanos, além de torna-se flexível para qualquer biblioteca de programação interessada em acessar seu conteúdo. O acoplamento entre interface e código pode ser feito através desses arquivos. As classes desenvolvidas com esse intuito para o **enJine** exemplificam esta comunicação e a viabilidade dessa arquitetura. O acréscimo de funcionalidades para versões mais robustas fica dependente de prazos futuros e pode contar com esforços de um grupo maior de desenvolvedores.

A versão preliminar implementando o **Game Wrench** utilizando **Java Swing** e **enJine v3.0** foi utilizada para realizar testes sobre o caso exemplo descrito na Seção 6.2.3.

A Figura 7-1 apresenta a diagramação visual do sistema. Tal diagramação necessita de um tratamento mais requintado, mas é importante manter o fato de o *Yellow Helper* chamar a atenção do usuário ao iniciar o sistema pela primeira vez, de forma a oferecer a opção de seguir um tutorial para reconhecimento de sua funcionalidade.

Figura 7-1 - Diagramação Visual do Game Wrench implementado com a plataforma Java Swing.

A Figura 7-2 mostra o detalhe da funcionalidade de *Yellow Helper* e *Red Tooltip* para apresentar um tutorial integrado à interface de desenvolvimento sendo testada. Em versões futuras, a navegação pelo tutorial poderá ser menos linear, quando o botão *Next* deverá ser substituído por opções de navegação mais sofisticada.

Figura 7-2 - Detalhe de Yellow Helper e Red Tooltip sendo testados.

Figura 7-3 - Screenshot do Game Wrench durante teste utilizando GAML.

Figura 7-4 - Detalhe do código enJine gerado pelo Game Wrench durante o teste.

Seguindo com o teste do caso exemplo, um arquivo GAML vazio foi utilizado como base para inserir as informações de nome de projeto (“Pinball”), objeto de jogo (“SilverBall”) e *placeholder* (“Esfera”). Este arquivo é apresentado para o usuário na forma de um componente **JTree** na área de *preview* do **Game Wrench** (ver Figura 7-3).

Esta mesma área de *preview*, organizada em abas pelo componente **JTabbedPane**, é utilizada para apresentar (de forma não editável) o código **enJine** gerado automaticamente pelo **Game Wrench** utilizando as informações encontradas no **GAML** do projeto (Ver Figura 7-4). Em versões futuras, esta funcionalidade poderá ser feita de forma automática, sem precisar do botão “Gerar Código enJine”.

Essa versão preliminar do **Game Wrench** pode ser encontrada junto ao **enJine** no site incubadora.fapesp.br/projects/enjine.

Uma vez realizados os primeiros testes e o caso exemplo descrito na Seção 6.2.3, a intenção é utilizar o **Game Wrench** junto ao **enJine** com uma turma de alunos da disciplina de Computação Gráfica em 2008, quando poderá ser verificada a real eficácia do mesmo no processo de aprendizado junto aos principais usuários. Para isso, um grupo de desenvolvedores precisará ser alocado ao projeto pelo Interlab.

Além dessa dissertação e seus apêndices, este trabalho gerou um *short paper* para o congresso **SBGames2006** (JACOBBER; TORI, 2006)³² e um *full paper* para o congresso **ICECE2007** (JACOBBER; TORI, 2007)³³.

8 CONCLUSÃO

Como base teórica para o trabalho dessa dissertação foi realizada pesquisa sobre três temas principais: conceituação de jogo, metodologias de projeto de interação e identificação de usuários de motor de jogo (Seção 2).

A complexidade do tema exigiu um aprofundamento sobre a conceituação de “jogo”. A partir de um estudo sobre diversas vertentes (às vezes até divergentes) que procuram explicar o conceito, adotou-se a de Walther, que se mostrou a mais adequada para tomar como ponto de partida deste trabalho.

Metodologias de projeto de interação precisaram ser estudadas, pois esta dissertação trata de preocupações teóricas e atividades de projeto relativamente recentes no meio de desenvolvimento de software, ainda mais no mundo dos jogos eletrônicos. Foi preciso conhecer muitas propostas e diretrizes para aplicar o que foi considerado coerente para o escopo desse trabalho.

Conhecendo o autor o mundo dos motores de jogo principalmente pelo ponto de vista de desenvolvedor/programador, foi igualmente importante buscar informações do universo de possíveis usuários dessa ferramenta de software. A partir daí, o recorte e a escolha de um tipo de usuário bastante específico como “público-alvo” se tornam muito mais confiáveis, válidos e ricos de informação.

Os resultados, representados pela lista de requisitos que compõem a especificação do **Game Wrench** somente foram possíveis através do processo de Projeto de Interação.

Seria muito improvável que os mesmos fossem identificados ou inferidos através de simples especulação, imaginação ou bom-senso. Da mesma forma, este processo é complementar aos requisitos do processo de Engenharia de Software, por terem focos diferentes.

A metodologia descrita não é definitiva ou à prova de falhas, mas foi ágil e eficiente para o caso deste projeto, ao mesmo tempo em que passou por diversas técnicas encontradas na literatura, permitindo este estudo; pode ser aplicada de forma similar a outros produtos de *software*; e contribuiu de forma a evitar problemas que, se detectados em estágios mais avançados de desenvolvimento, seriam desastrosos em termos de investimento de recursos e de tempo.

Uma versão preliminar em **Java Swing** da interface **Game Wrench** permitiu avaliar suas principais características e relacionamento com os demais componentes como o motor **enJine** e está disponível junto à distribuição do **enJine**, hospedada na incubadora da FAPESP no site incubadora.fapesp.br/projects/enjine.

De uma forma geral, o valor maior desse trabalho encontra-se no processo, no desenvolvimento de competências e habilidades dentro do Interlab através da aplicação das práticas de Interação Humano-Computador, do que no estado atual dos produtos desenvolvidos.

Resgate dos Objetivos

Para desenvolver a especificação de alta usabilidade descrita no Capítulo 5, primeiramente realizou-se um estudo sobre os diversos métodos de projeto de interação atuais, definiu-se e estudou-se o grupo de usuários específico.

Este objetivo principal foi alcançado através de atividades baseadas nos métodos Avaliação Heurística, Projeto Participativo, entrevistas com *stakeholders*, Prototipagem em Papel e Teste *Thinking Aloud* Simplificado. Dentro do universo escolhido, alunos da disciplina Computação Gráfica da Escola Politécnica da USP, foi fundamental a participação voluntária de quatro destes alunos, o que representa aproximadamente dez por cento desta turma.

A aplicação de uma interface seguindo tal especificação ao motor de jogo **enJine** fica dependente do trabalho de um grupo com as competências adequadas (dentre elas uma diagramação visual mais adequada). Entretanto, tal objetivo principal foi comprovado através de uma prova de conceito sobre a aplicação de tais

funcionalidades utilizando-se a plataforma **Java Swing** no caso exemplo descrito na Seção 6.2.3.

Apesar de nem todos requisitos levantados nos projetos anteriores serem resolvidos com o **Game Wrench**

, eles foram compilados e incluídos na lista apresentada no Capítulo 5. A maior contribuição neste sentido foi realizada ao estabelecer como foco os usuários principais e suas metas e tarefas reais ao utilizar um motor de jogo para o aprendizado de Computação Gráfica.

O **Game Wrench** pode contribuir para aumentar o formalismo nos projetos de jogos eletrônicos ao incluir em sua estrutura os elementos identificados como essenciais para um produto ser considerado jogo. Tal estrutura foi definida formalmente por **XSD** no **GAML**.

A sua previsibilidade pode ser aumentada com o **Game Wrench** uma vez que este leva em consideração as três visões por que todo projeto de software precisa passar: produção, projeto e desenvolvimento.

A flexibilidade é garantida novamente pelo **GAML**. As informações de projeto levantadas com o usuário através da interface são armazenadas em arquivos **XML**. Por ser um padrão forte e multiplataforma, a leitura de arquivos **XML** é facilmente implementada por qualquer linguagem de programação, permitindo o interfaceamento bastante desacoplado com qualquer *API* ou motor de terceiros que necessite de tais informações para geração de código específico.

O desenvolvimento do **Game Wrench** teve como diretriz diminuir o tempo de aprendizado e aumentar a eficiência do programador do **enJine** sem, no entanto, esconder a parte de programação do desenvolvimento de jogos, como foi o caso identificado nas ferramentas observadas durante a análise comparativa. A intenção é automatizar tarefas repetitivas, gerar código automaticamente, mas permitir e incentivar a otimização deste código para as necessidades específicas de cada projeto de jogo. O cumprimento eficaz de tal objetivo somente poderá ser comprovado quando o **Game Wrench** for efetivamente aplicado em disciplinas que usem o **enJine**.

Trabalhos Relacionados

Desde o início desse trabalho, outras ferramentas de desenvolvimento de jogos surgiram, atualizaram-se ou chegaram ao conhecimento, mas não puderam ser analisadas a contento, como foram o **Game Maker** e o **3D Game Studio**.

Dentre estas, destacam-se ferramentas orientadas ao ensino, como:

- **Phrogram (Kids Programming Language v2.0)** – ver página *The Phrogram Company* em phrogram.com;
- **Simple Academic Game Engine** – ver página em larc.csci.unt.edu/sage;
- **Multi-user Programming Pedagogy for Enhancing Traditional Study (M.U.P.P.E.T.S.)** – ver página *The Muppets Project @ RIT* em muppets.rit.edu;

e a plataforma de desenvolvimento híbrida PC/console **XNA** (ver página *XNA developer center* em msdn.microsoft.com/XNA).

Para um trabalho que começa agora, recomenda-se incluir tais ferramentas para uma análise comparativa mais atual e completa.

Trabalhos Futuros

Como continuidade ao trabalho aqui apresentado, além da implementação e avaliação de uma versão completa e funcional do **GameWrench** para **enJine**, vislumbram-se várias possibilidades, entre elas as seguintes:

- O foco deste projeto foi a interface entre um usuário humano e um sistema digital, melhorando a comunicação entre os dois, também conhecido como *Front-End* do sistema. Através da especificação elaborada nesse trabalho, pode-se criar a estrutura de ligação entre este *Front-End* e diversos módulos ou sistemas independentes, ou seja, uma arquitetura de alto nível para a interface entre este *Front-End* e os demais sistemas que constituem um Motor de Jogos.

Foi mostrado como criar essa ligação com um caso específico – o motor **enJine**. A aplicação a outros sistemas e módulos ficou fora do escopo deste trabalho.

- O método utilizado no projeto de interação inclui uma escolha de aplicação e usuários específicos para os quais foram voltados o projeto e o desenvolvimento. O mesmo método, bem como os artefatos produzidos (*personas*, cenários, protótipo, etc.), poderão ser aplicados a outra população de usuários, e até mesmo outro sistema computacional sendo desenvolvido.
- Traduzir / Internacionalizar o **GAML**. O **GAML** surgiu para atender uma necessidade local desta dissertação, mas pode se tornar uma ferramenta útil e poderosa para diversas outras aplicações. Um primeiro passo para sua divulgação reside em traduzi-lo para a língua inglesa.
- Aplicar o **Game Wrench** para facilitar o uso e aprendizado do **enJine** nas aulas de Computação Gráfica ou outras disciplinas afins.

APÊNDICE A – Relatório do Estudo de Usabilidade com Auxílio de Protótipo em Papel

Game Wrench

Testes de Usabilidade com Protótipo em Papel

Relatório da Metodologia

Outubro/2006

Teste de Usabilidade – técnicas empregadas

Discount Usability Engineering (Nielsen, 1994)

Participatory Design

Thinking Aloud Testing

Paper Prototyping (Snyder, 2003)

Atividades

Cronograma:

A tabela a seguir representa a lista de tarefas envolvidas no processo, junto à sua estimativa de tempo. É importante ressaltar que a criação do protótipo seguiu requisitos que já vinham sendo estudados há mais tempo, através de outras atividades da dissertação de mestrado do autor.

Etapas		Tempo	Descrição
17 e 23/08	Kickoff Meeting	3 horas	discutir metas, riscos e preocupações; consenso no perfil de usuário determinar grupo principal marcar cronograma
Quarta 30/08	Recrutamento de Usuários	2 a 3 semanas	agendar sessões de testes com usuários dentro do perfil
Segunda 04/09	Projeto de tarefas	3 a 5 horas	Listar tarefas a serem testadas
Segundas	Criação do protótipo e walk-throughs	3 a 5 dias	Listar elementos de interface necessários para o protótipo Dividir trabalho Executar os walkthroughs periódicos Teste Piloto
Segundas e Sábados, exceções antes das 16h	Testes de Usabilidade e Refinamento Iterativo	4 meios períodos	Executar os testes (1-2 horas) Listar problemas verificados Revisar o Protótipo
	Priorização de problemas e plano de ação	3 horas	Priorizar problemas não resolvidos Discutir problemas principais e buscar soluções Criar plano de ação para atacar problemas acompanhar problemas
	Comunicação de resultados	Depende	Escrever sumário dos 10 principais problemas Publicar resultado na Intranet Escrever relatório Apresentar resultados (reunião) Criar vídeo de walkthrough Criar especificação para interface

Entrevistas com Stakeholders

Foram realizadas entrevistas com os principais envolvidos na programação do enJine. Nestas reuniões, discutiu-se o estado atual do projeto, definiu-se o usuário típico do sistema, listou-se tarefas comuns, discutiu-se preocupações e pontos críticos:

Entrevista #1

Data: 17/8/2006

Participantes:

Eduardo Jacober (Usabilidade)

Caue (desenvolvedor)

Yuri (desenvolvedor)

Entrevista #2

Data: 17/8/2006

Participantes:

Eduardo Jacober (Usabilidade)

Romero Tori (produtor)

Entrevista #3

Data: 23/8/2006

Participantes:

Eduardo Jacober (Usabilidade)

Ricardo (desenvolvedor)

João (desenvolvedor)

Romero (produtor)

Estado atual do Projeto:

O enJine encontra-se na versão nominal 3.0. Desde a versão 1.0, ocorreram as seguintes modificações:

1. Foco em jogos SinglePlayer;
2. Capacidade de Overlay (HUD);
3. Som estéreo;
4. Capacidade de “picking-collision” (usado em posicionamento em jogo de skate);
5. Loader de modelos .X (com animação, integração com 3DS e Maya);
6. Modificações em Input e Colisão.
7. Arquitetura concentrada na entidade Game Object
8. Desenvolvimento através de Game States
9. A arquitetura de comunicação em rede será remodelada para Cliente-Servidor futuramente

Foi utilizado por alunos na disciplina de Computação Gráfica do Prof. Romero, gerando diversos jogos exemplo.

Perfil do usuário:

Durante o desenvolvimento do enJine (até o momento), não houve preocupação em definir o perfil de usuário. Para estes desenvolvedores, o usuário é este aluno do curso de CG, mas pode se estender a qualquer um que queira fazer um jogo sem ter muita experiência em computação ou em game design (pois ajuda a lembrar de tudo que é preciso ter em um jogo). Além destes, também aqueles que já conhecem Java ou querem aprender.

A idéia inicial, antes destas reuniões, era a de orientar para o Professor desenvolvendo objetos de aprendizagem para suas disciplinas. Como atualmente o uso do enJine está vinculado à disciplina de CG, o foco mudou para o Aluno que desenvolve um jogo como projeto para esta disciplina.

Mudando o foco para ferramenta para o aluno:

- Trata-se de uma ferramenta intermediária para o aluno aprender a programação básica, antes de entrar com o código mais pesado (algo como prototipagem);
- Neste primeiro momento, a interface deverá facilitar para o usuário atual (aluno de CG), depois evoluindo para usuários leigos;
- Perguntar para estes alunos se a interface ajuda ou atrapalha o aprendizado desta disciplina (ou seja, talvez a “ajuda” da ferramenta elimine o esforço que permitiria o aluno aprender);
- Ter por objetivos familiarizar, ser ambiente de desenvolvimento e ajudar a visualização de um ambiente de jogo, assim como o Interlab3D tem por objetivo familiarizar com grafo de cena, etc.

Da experiência com a turma:

- O primeiro contato é através de dois tutoriais disponíveis no site da incubadora e, agora, com os exemplos já desenvolvidos; Este tutorial é muito simples, seria bom ter tutoriais para as diversas partes do projeto (mais focados);
- A maioria dos grupos não terminou com um “jogo acabado”. Nenhum grupo continuou o projeto após o término da disciplina;
- Existem problemas de desempenho – não recomendam para o “desenvolvedor profissional”;
- Em compensação, o código está bastante claro, sem “mágicas”, facilitando o aprendizado através de exemplos;
- Os javadocs não estão prontos, e fazem falta;

Concluindo, nesta fase, será orientado para o perfil deste aluno de CG, que conhece programação orientada a objetos, tem noções de computação gráfica e Java3D (passou pelo *Art of Illusion* e pelo Interlab3D) e está se familiarizando com o enJine.

Em fases futuras, pode-se realizar o mesmo trabalho para estender o foco para o Professor, o Game Designer corporativo ou outros tipos de usuário.

Metas, riscos e preocupações:

Desafios específicos encontrados pelos grupos:

- Problemas na importação de alguns formatos de som – alguns formatos não permitem controle de volume pelo Java;
- Funcionalidade de *picking* do exemplo “Ladeira abaixo”;
- Problema no carregamento de texturas de objetos modelados no Art of Illusion – pode ser no *loader* de .OBJ ou no AoI;

Não facilitar estas coisas, deixar programação “na raça”:

- Manipular o grafo de cena
- Manipular iluminação
- Manipular câmera
- Transformações Geométricas

Atenção para conceitos de ENGFW

Atenção para conceitos de CG e Jogos:

- grafo de cena

Podem ser Tarefas Visuais:

- Fazer animações: importar do 3DS
- Mecanismos de Triggers no mapa
- Modelo de mapa – colisão com mapa
- Bounding boxes
- Instanciamento de classes (código automático)

São diretrizes para aumentar produtividade, portanto não quer dizer que os usuários não saibam programar

Futuramente, pode ser utilizado para aumentar o escopo do enJine para ensinar turmas de Game Design.

Projeto de tarefas:

Definir o Jogo:

- Definir personagem e jogo;
- Definir Etapas do jogo;
- Objetivo, Cenário e Personagem Dinâmico, Interação Monitor+Teclado

Programar:

- Modelar personagens e cenários;
- Montar personagem – partes do personagem (grafo de cena);
- Carregar modelos no enJine;
- Iluminação;
- Câmeras;
- Tratar Input do personagem;
- Caso o jogo requeira, Colisões;
- Tratar Input do personagem;
- Mostradores na tela: *Overlay*;
- *Gameplay*: Tratar interação entre objetos do jogo;
- Tratar interação entre objetos do jogo;

Refinamento:

- Criar fases;
- Som;
- Filminhos de *gameplay*: trailers;

Tarefa #	1-0001
Nome	Criar Projeto
Grupo	Planejamento
Meta e resultados:	Definir características do jogo a ser criado. Mínimo necessário: um nome, uma data de release, um gênero ou um exemplo de jogo similar.
Entrada:	Lista resumida de gêneros de jogo com exemplos;
Pressupostos:	Usuário (aluno) está de posse da lista de requisitos do projeto-jogo proposto (incluindo datas de entrega); Equipes já definidas.
Passos:	<1>
	criar novo projeto: <V-1>

	ícone modo produtor: <U-I-1>;<U-L-1>;<P-1>
	digitar o nome: <P-2>;<U-I-1>
	definir data de entrega: <P-3>;<TP-2>;<2>
	escolher gênero:
Tempo de expert	
Instruções para Usuário	
Notas	
Tarefa #	1-0002
Nome	Definir Personagem
Grupo	Planejamento
Meta e resultados	Definir características estáticas e dinâmicas do personagem/avatar controlado pelo jogador. Mínimo necessário: um nome, um placeholder, lista de ações (movimento, combate, atuação, etc).
Entrada:	Lista resumida de tipos de ações comuns
Pressupostos:	Projeto criado no sistema; Gênero definido <1-0001>
Passos:	<2>
	Projetar jogo: <V-2>
	ícone modo projeto: <P-TP-0>; <3>
	Objetos:<V-2.2>
	ícone objetos: <P-TP-2>; <P-L-2>; <P-4>
	digitar nome: <P-5>
	editar atributos: <P-TP-2>
	editar ações:
Tarefa #	1-0003
Nome	Definir Etapas
Grupo	Planejamento
Meta e resultados	Definir macro-etapas do jogo. Mínimo necessário: Criar diagrama contendo, pelo menos, tela de abertura, jogo e tela de encerramento.
Entrada:	Lista resumida de fases comuns de um produto-jogo
Pressupostos:	Projeto criado no sistema
Tarefa #	1-0004
Nome	Programar Objetivo do Jogo
Grupo	Planejamento
Meta e resultados	Definir condição de fim. Mínimo necessário: Associar um atributo a VPs (victory points) e criar condição para fim de jogo, critérios de desempate.
Entrada:	Definição de VPs
Pressupostos:	Objetos criados e atributos associados <1-0006> <1-0001>
Tarefa #	1-0005
Nome	Definir Cenário
Grupo	Planejamento
Meta e resultados	Definir espaço(s) de jogo. Mínimo necessário: Criar um objeto "cenário", contendo áreas de jogo + um esboço visual deste cenário.
Entrada:	Explicação do escopo da tarefa (ver notas)
Notas	Esta tarefa não envolve criação visual do cenário, apenas definição de espaços lógicos, p.ex: gols, casas, checkpoints, etc
Tarefa #	1-0006

Nome	Definir classes de objetos de jogo
Grupo	Planejamento
Meta e resultados	Definir objetos e seus atributos. Mínimo necessário: Definir todos os principais atributos do objeto avatar.
Tarefa #	1-0007
Nome	Definir método de tratamento de colisões
Grupo	Planejamento
Meta e resultados	Identificar as necessidades do jogo quanto a detecção e tratamento de colisões. Mínimo necessário: identificar quais objetos serão marcados <collidable> e que tipo de filtro(s) será(ão) utilizado(s).
Tarefa #	2-0001
Nome	Modelar personagem/avatar
Grupo	Desenvolvimento
Tarefa #	2-0002
Nome	Modelar cenário
Grupo	Desenvolvimento
Tarefa #	2-0003
Nome	Carregar/importar modelos ao engine
Grupo	Desenvolvimento
Tarefa #	2-0004
Nome	Aplicar controles ao personagem/avatar
Grupo	Desenvolvimento
Tarefa #	2-0005
Nome	Criar iluminação para a cena
Grupo	Desenvolvimento
Tarefa #	2-0006
Nome	Criar e manipular câmeras
Grupo	Desenvolvimento
Tarefa #	2-0007
Nome	Aplicar tratamento de colisão
Grupo	Desenvolvimento
Tarefa #	2-0008
Nome	Criar Overlay com mostradores na tela
Grupo	Desenvolvimento
Tarefa #	2-0009
Nome	Atuação entre objetos do jogo
Grupo	Desenvolvimento
Tarefa #	3-0001
Nome	Tela de créditos
Grupo	Refinamento

Construção do Protótipo

Princípios e Hipóteses da interface:

- Evitar sobreposição de janelas, reservando previamente espaço para cada tipo de interação. Estes espaços são:
 - Área de Diálogo “Yellow Helper”. 240x120. fixa. sup.esq.
 - Área de Recursos. Esq.
 - Barra de escolhas com ícones e info / navegação. Centro sup.
 - Área de Trabalho principal. Centro.
 - Área de Preview. 320x240. fixa. sup. Dir.
 - Área de Informação. Dir.
 - (Barra de Menu)*
 - (Barra de status)*
- Integrar o efeito de tutoriais no ambiente de desenvolvimento, sem incomodar os usuários avançados. Esta será a função do Yellow Helper/Orange Alert e Red Tooltips (popups semitransparentes), atuando como “wizards”;
- Atividades e tarefas de alto-nível são tão importantes quanto às de programação; Interface dividida em visões nos modos produção, projeto e desenvolvimento;
- Apesar de apresentar facilitadores visuais, evitar “esconder” a programação do usuário, primeiramente porque é impossível para qualquer jogo mais complexo, segundo porque se tratam de usuários com interesse no aprendizado de programação. Tentar aumentar a produtividade, diminuir a curva de aprendizado, sem, no entanto prejudicar a aprendizagem de Computação Gráfica ou Projeto de Jogos;

Largura e profundidade do protótipo:

Foi decidido concentrar no primeiro grupo de tarefas (planejamento), por questões de foco, tempo das sessões, e por serem as tarefas que têm menos intersecção com o que o enJine já faz atualmente (estas tarefas são feitas informalmente, fora do ambiente de programação).

Sessões de Testes

Roteiro para os testes de interface

Abaixo está reproduzida uma lista de considerações entregue ao usuário antes de cada sessão como introdução à atividade:

Game Wrench - enJine

- **O objeto dos testes é a interface, não o usuário.** Este estudo sendo realizado é para identificarmos falhas em um sistema em processo de desenvolvimento. Os erros identificados são resultados de falha no processo de interação entre usuário e sistema;
- **Seja impiedoso.** Apesar de conhecer os desenvolvedores e seu esforço para produzir um bom sistema, não hesite em informar qualquer detalhe que incomode ou atrapalhe a comunicação com o sistema. Estas sugestões são muito valiosas, serão registradas e priorizadas pelos desenvolvedores e transformar-se-ão em melhorias no sistema;

* Será verificada durante os testes a real necessidade destes elementos

- **Confidencialidade.** Este teste está sendo feito sobre um projeto em andamento. Contamos com sua discrição para não divulgar informações sobre o sistema antes de sua distribuição oficial (data de lançamento).
- **A participação é voluntária e as gravações são opcionais.** Os testes serão gravados por equipamento de vídeo para referência futura. Você poderá encerrar sua participação neste teste a qualquer momento e poderá solicitar o desligamento da gravação de áudio ou vídeo caso se sinta desconfortável.
- **O anonimato é garantido.** As informações levantadas a partir dos testes quando divulgadas sempre serão desassociadas dos nomes dos usuários.
- **Perguntas durante os testes.** Fique à vontade para fazer qualquer pergunta, mas nem todas serão respondidas. Parte do teste envolve analisar a capacidade do sistema de responder às respostas do usuário sem uso de ajuda externa. Portanto, o nível de interferência do condutor dos testes será controlado.
- **Protótipo.** Estes testes serão feitos sobre um protótipo, ou seja, não está totalmente funcional. A exploração da interface é encorajada, mas procure não se incomodar se diversas ações forem respondidas com “ainda não disponível”.
- **Protótipo em papel.** Para estes testes será utilizado um protótipo em papel. Um dos examinadores representará o “computador”, estando encarregado de emular o processamento e modificações no estado do sistema. Dependendo da interação, o tempo para executar estas modificações poderá variar consideravelmente.
- **Verbose mode.** Na medida do possível, diga em voz alta tudo o que pensar durante o teste, sem censura. Em particular, antes de qualquer interação, diga qual esperaria ser o resultado de suas ações, e após o processamento do “computador”, diga se atendeu suas expectativas. Diga também em que momento identificou ter completado cada uma das tarefas programadas.

Roteiro de acompanhamento

Outro documento utilizado nas sessões para orientar e registrar informação dos testes:

Nome:

Jogo:

Data:

Início/Final:

Pressupostos:

- Usuário está cursando disciplina de graduação sobre Computação Gráfica e Jogos Eletrônicos.
- Recebeu orientação sobre o projeto de jogo eletrônico a ser entregue ao final do quadrimestre, em grupo e valendo nota.
- O grupo já se reuniu e chegou a algumas decisões sobre como será o jogo (sessão de *brainstorming*).

- O software sendo representado já foi adquirido e previamente instalado, mas esta sessão representa o primeiro contato e uso.

Tarefas programadas:

- <1-0001> Criar Projeto
- <1-0002> Definir Personagem
- <1-0004> Programar Objetivo do Jogo
- <1-0005> Definir Cenário

Dados das Sessões

Teste Piloto:

Usuários: Roberto
 Data: 24/09/2006
 CPU: João
 GPU: Eduardo
 Observador: Ricardo

Sessão #1:

Usuário: William Colen
 Companheiro de Grupo: Caue
 Descrição do jogo:

- “Caça-Bichinho de pelúcia” Didático
- Pegar figura geométrica com braço mecânico

Data: 26/09/2006
 Tempo Total: 1h20
 Tarefas Concluídas: <1-0001> 30” + <1-0002> 50”
 Tarefas encaminhadas:
 CPU: Ricardo
 GPU: Eduardo
 Observadores: Alexandre e Daniel

Sessão #2:

Usuário: Rodrigo Petriche
 Companheiro de Grupo: Tatiana
 Descrição do jogo:

- “LEGO Adventure”
- Cenário
- Bichinho (avatar)
- Interação com objetos

Data: 27/09/2006
 Tempo Total: 43”
 Tarefas Concluídas: <1-0001> 19” + <1-0004> 24”
 Tarefas encaminhadas:
 CPU: João
 GPU: Eduardo
 Observadores: Ricardo, Romero

Sessões #3-4:

Usuário: Felipe Kamakura

Companheiro de Grupo: Luiz

Descrição do jogo:

- “Elephant Wars”
- Por turnos
- Multiplayer no mesmo computador
- Elefantes atiram coisas entre si

Data: 27-28/09/2006

Tempo Total: 48”

Tarefas Concluídas: <1-0001> 20” + <1-0002> 28”

Tarefas encaminhadas:

CPU: João / Ricardo

GPU: Eduardo

Observadores: Ricardo / João

Sessão #5

Usuário: Dennis

Companheiro de Grupo: Diogo

Descrição do jogo:

- “Futebol”:
- 6 jogadores por time;
- jog vs. Comp. ou jog1 vs. Jog2.;
- regras simplificadas;
- física simplificada

Data: Seg, 02/10/06, 12h00

Tempo Total: 1h24”

Tarefas Concluídas: <1-0001> 37” <1-0002> 47”

Tarefas encaminhadas: <1-0004> <1-0005>

CPU: Ricardo

GPU: Eduardo

Observadores: -

Considerações e Planejamento Futuro***Diretrizes a partir dos testes:***

- A ausência de elementos tradicionais como Barra de Menu, Barra de Status etc. causa desconforto, mas, por enquanto, não foi essencial para execução de nenhuma das tarefas propostas; Em compensação, é preciso acrescentar ações de Undo, Redo e Missão cumprida. Outra funcionalidade interessante é a de auto-geração de documentos e impressão; Mais outra: preview do jogo;
- A tarefa inicial (<1-0001> Criar Projeto) não é comum para os usuários. Eles não fizeram isso formalmente em sua experiência com o enJine. É uma tarefa boa para apresentar o comportamento da interface, seus modos, ajudas e visões. É, portanto, também onde surgem dúvidas sobre estas informações. Nem todos os usuários reconheceram este funcionamento naturalmente, e uma boa forma de informá-los deve ser aprimorada. Se a interface começar no modo produção, será uma ação a menos, mas prorroga o contato com este comportamento;

- Os estilos dos usuários variaram bastante, o que comprova uma flexibilidade boa desta interface e suas áreas reservadas pré-definidas. Em compensação, sempre haverá espaços sobre-utilizados competindo por pixels dos espaços subutilizados; Em geral, os usuários terminaram o teste com uma noção correta da função de cada espaço;
- A interface “funciona” bem, encoraja exploração e guia o usuário em tarefas de planejamento que antes eram feitas informalmente;

Considerações sobre o processo:

- Em geral, alguns usuários tendem a querer “explorar a interface” no começo do teste. O facilitador deve se conter neste momento, mas tentando guiar o usuário para o foco do teste, lembrando da lista de tarefas;
- Devido ao horário dos testes, pensou-se em petiscos para “enganar o estômago”. O uso deste recurso deixou as sessões com um “ar de jogo de tabuleiro”, muito positivo. Sugestão para sempre ter algo parecido. Lembrar de deixar as bebidas em lugar separado (afinal é um protótipo em PAPEL...), trazer guardanapos e escolher somente petiscos não gordurosos.
- A metáfora gráfica/visual é um ponto crítico. Os post-its têm cores que hora representam funcionalidade, ora não.
- É importante preparar uma série de “campos de entrada” previamente, pois serão usados com frequência, o mesmo para X de check-box, etc.

Participatory Design

As seguintes funcionalidades e sugestões de solução foram elaboradas durante os testes, com auxílio dos usuários:

- Inclusão de informações dos marcos (nome e data) na linha do tempo;
- Campo de informação sobre os ícones de modos da interface (produção, projeto e desenvolvimento);
- Campo para associação de teclas às ações e movimentos de personagens do jogo, em estrutura em árvore e diagrama do dispositivo físico (teclado e mouse);
- Melhorias na área de preview, com auxílio de eixos de coordenadas e botão de Play;
- Inclusão de visualização de código na área de preview;

APÊNDICE B – GAML definido em XML Schema e exemplos

```

GaML.xsd 5/2/2007
1 <?xml version="1.0" encoding="UTF-16"?>
2 <xsd:schema targetNamespace="GameWrench"
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:qam1="GameWrench">
4   <xsd:element name="Campeonato">
5     <xsd:complexType>
6       <xsd:sequence>
7         <xsd:element name="Estado" type="qam1:Estado" />
8
9         <xsd:element name="Equipe" minOccurs="2"
10           maxOccurs="unbounded" type="qam1:Equipe.compet" />
11
12         <xsd:element name="Partida" type="qam1:Partida"
13           maxOccurs="unbounded" />
14       </xsd:sequence>
15     </xsd:complexType>
16   </xsd:element>
17
18   <xsd:simpleType name="Estado">
19     <xsd:restriction base="xsd:string">
20       <xsd:enumeration value="Preparação" />
21       <xsd:enumeration value="Em andamento" />
22       <xsd:enumeration value="Finalizado" />
23     </xsd:restriction>
24   </xsd:simpleType>
25
26   <xsd:element name="Partida" type="qam1:Partida" />
27
28   <xsd:complexType name="Partida">
29     <xsd:sequence>
30       <xsd:element name="Estado" type="qam1:Estado" />
31       <xsd:element name="Rodada" maxOccurs="unbounded"
32         minOccurs="0">
33         <xsd:complexType>
34           <xsd:sequence>
35             <xsd:element name="Turno" maxOccurs="unbounded"
36               minOccurs="0">
37               <xsd:complexType>
38                 <xsd:sequence>
39                   <xsd:element name="Movimento"
40                     type="qam1:Movimento" maxOccurs="
41                     "unbounded" minOccurs="1" />
42                 </xsd:sequence>
43               </xsd:complexType>
44             </xsd:element>
45           </xsd:sequence>
46         </xsd:complexType>
47       </xsd:element>
48       <xsd:element ref="qam1:Jojo" />
49     </xsd:sequence>
50   </xsd:complexType>
51
52   <xsd:complexType name="Jojo">
53     <xsd:sequence>
54       <xsd:element name="Nome" type="xsd:string" />
55       <xsd:element name="Equipe" type="qam1:Equipe"
56         maxOccurs="unbounded" minOccurs="1" />
57
58       <xsd:element name="Sistema" type="qam1:SistemaDeJojo"
59         maxOccurs="unbounded" minOccurs="1" />
60
61       <xsd:element name="Regras" type="qam1:LivroDeRegras" />
62

```


GaML.xsd

5/2/2007

```

63     </xsd:sequence>
64 </xsd:complexType>
65
66 <xsd:complexType name="Equipe">
67     <xsd:sequence>
68         <xsd:element name="Jogador" type="xsd:string"
69             maxOccurs="unbounded" minOccurs="1" />
70     </xsd:sequence>
71 </xsd:complexType>
72
73 <xsd:complexType name="Equipe.compet">
74     <xsd:complexContent>
75         <xsd:extension base="qaml:Equipe">
76             <xsd:sequence>
77                 <xsd:element name="Pontos" type="xsd:int" />
78                 <xsd:element name="Colocação"
79                     type="xsd:positiveInteger" />
80             </xsd:sequence>
81         </xsd:extension>
82     </xsd:complexContent>
83 </xsd:complexType>
84
85 <xsd:complexType name="Definição">
86     <xsd:sequence>
87         <xsd:element name="Termo" type="xsd:string" />
88         <xsd:element name="Classe" type="xsd:string" maxOccurs="1"
89             minOccurs="0" />
90         <xsd:element name="Sinônimo" type="xsd:string"
91             maxOccurs="unbounded" minOccurs="0" />
92         <xsd:element name="Explicação" type="xsd:string"
93             maxOccurs="unbounded" minOccurs="1" />
94         <xsd:element name="Exemplo" type="xsd:string"
95             maxOccurs="unbounded" minOccurs="0" />
96     </xsd:sequence>
97 </xsd:complexType>
98
99 <xsd:complexType name="Token">
100     <xsd:sequence>
101         <xsd:element name="NewElement" type="xsd:string" />
102     </xsd:sequence>
103 </xsd:complexType>
104
105 <xsd:complexType name="SistemaDeJogo">
106     <xsd:sequence>
107         <xsd:element name="Universo">
108             <xsd:complexType>
109                 <xsd:sequence>
110                     <xsd:element name="Ambiente"
111                         maxOccurs="unbounded" minOccurs="1">
112                         <xsd:complexType>
113                             <xsd:sequence>
114                                 <xsd:element name="AreaDeJogo"
115                                     maxOccurs="unbounded" minOccurs="
116                                     1" type="qaml:AreaDeJogo" />
117                             </xsd:sequence>
118                         </xsd:complexType>
119                     </xsd:element>
120                     <xsd:element name="Njog_min"
121                         type="xsd:int" maxOccurs="1" minOccurs="1">
122                     </xsd:element>
123                 </xsd:sequence>

```

```

GaML.xsd 5/2/2007
125         </xsd:complexType>
126     </xsd:element>
127     <xsd:element name="Token">
128         <xsd:complexType>
129             <xsd:sequence>
130                 <xsd:element name="Atuador" type="boolean" />
131
132                 <xsd:element name="Dimensão.Token"
133                     type="gaml:Dimensão" maxOccurs="unbounded"
134                     minOccurs="0" />
135             </xsd:sequence>
136         </xsd:complexType>
137     </xsd:element>
138 </xsd:sequence>
139 </xsd:complexType>
140
141 <xsd:complexType name="Dimensão">
142     <xsd:sequence>
143         <xsd:element name="Nome" type="xsd:string" />
144         <xsd:element name="Tipo" type="xsd:string" fixed="Discreta" />
145     </xsd:sequence>
146     <xsd:element name="Dinâmico" type="xsd:boolean"
147         default="false">
148         <xsd:annotation>
149             <xsd:documentation>
150                 Indica se o valor desta dimensão pode mudar
151                 durante o jogo
152             </xsd:documentation>
153         </xsd:annotation>
154     </xsd:element>
155     <xsd:element name="ValorMínimo" type="xsd:string"
156         maxOccurs="1" minOccurs="0" />
157     <xsd:element name="ValorMáximo" type="xsd:string"
158         maxOccurs="1" minOccurs="0" />
159     <xsd:element name="ValorPadrão" type="xsd:string"
160         maxOccurs="1" minOccurs="0" />
161     <xsd:element name="ValorDiscreto" type="xsd:string"
162         maxOccurs="unbounded" minOccurs="0" />
163 </xsd:sequence>
164 </xsd:complexType>
165
166 <xsd:complexType name="LivroDeRegras">
167     <xsd:sequence>
168         <xsd:element name="Definição" type="gaml:Definição"
169             maxOccurs="unbounded" minOccurs="0" />
170
171         <xsd:element name="Regra" type="xsd:string" />
172         <xsd:element name="Tempo">
173             <xsd:complexType>
174                 <xsd:sequence>
175                     <xsd:element name="Início" type="xsd:dateTime" />
176
177                     <xsd:element name="Etapa" type="xsd:duration"
178                         maxOccurs="unbounded" minOccurs="0" />
179
180                     <xsd:element name="Final" type="xsd:dateTime" />
181                 </xsd:sequence>
182             </xsd:complexType>
183         </xsd:element>
184         <xsd:element name="ConfiguraçãoInicial" type="xsd:string" />
185     </xsd:sequence>

```

GaML.xsd

5/2/2007

```

186         <xsd:element name="CondiçãoDeFim" type="xsd:string"
187             maxOccurs="unbounded" minOccurs="1" />
188     </xsd:sequence>
189 </xsd:complexType>
190
191     <xsd:complexType name="Movimento">
192         <xsd:sequence>
193             <xsd:sequence>
194                 <xsd:element name="Jogador" type="xsd:string" />
195                 <xsd:element name="Ação" type="xsd:string" />
196                 <xsd:element name="Momento" type="xsd:duration" />
197             </xsd:sequence>
198         </xsd:sequence>
199     </xsd:complexType>
200
201     <xsd:element name="Joqo" type="gaml:Joqo" />
202
203     <xsd:complexType name="ÁreaDeJogo">
204         <xsd:sequence>
205             <xsd:element name="Nome" type="xsd:string" />
206             <xsd:element name="Dimensão.Área" type="gaml:Dimensão"
207                 maxOccurs="unbounded" minOccurs="1" />
208         </xsd:sequence>
209     </xsd:complexType>
210
211     <xsd:element name="SistemaDeJogo" type="gaml:SistemaDeJogo" />
212
213     <xsd:complexType name="Projeto">
214         <xsd:sequence>
215             <xsd:element name="Nome" type="xsd:string" minOccurs="1"
216                 maxOccurs="1"></xsd:element>
217             <xsd:element name="Estado" type="gaml:Estado" minOccurs="1"
218                 maxOccurs="unbounded"></xsd:element>
219             <xsd:element name="MarcoIntermediário" type="gaml:Marco"
220                 minOccurs="1" maxOccurs="1"></xsd:element>
221             <xsd:element name="DataDeEntregaFinal" type="gaml:Marco"
222                 minOccurs="1" maxOccurs="1"></xsd:element>
223             <xsd:element ref="gaml:Joqo" minOccurs="1" maxOccurs="1"
224                 maxOccurs="1"></xsd:element></xsd:sequence>
225         </xsd:complexType>
226
227     <xsd:complexType name="Marco">
228         <xsd:sequence>
229             <xsd:element name="Nome" type="xsd:string" minOccurs="1"
230                 maxOccurs="1"></xsd:element>
231             <xsd:element name="Data" type="xsd:date" minOccurs="1"
232                 maxOccurs="1"></xsd:element>
233             <xsd:element name="Estado" type="gaml:Estado"></xsd:element>
234             <xsd:element name="Recurso" type="xsd:string"></xsd:element>
235         </xsd:sequence>
236     </xsd:complexType>
237
238     <xsd:element name="Projeto" type="gaml:Projeto"></xsd:element>
239
240 </xsd:schema>

```

Baralho52.xml

5/2/2007

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <gwl:SistemaDeJogo xmlns:gwl="GameWrench" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
   "GameWrench GaML.xsd">
3    <Universo>
4      <Ambiente>
5        <AreaDeJogo>
6          <Nome>Mão</Nome>
7          <Dimensão.Área>
8            <Nome>Ordem (de cima para baixo)</Nome>
9            <Tipo>Sequência</Tipo>
10           <ValorMínimo>1</ValorMínimo>
11           <ValorMáximo>52</ValorMáximo>
12           <Dinâmico>true</Dinâmico>
13         </Dimensão.Área>
14       </AreaDeJogo>
15       <AreaDeJogo>
16         <Nome>Mesa</Nome>
17         <Dimensão.Área>
18           <Nome>Nome</Nome>
19           <Tipo>Discreta</Tipo>
20           <Dinâmico>false</Dinâmico>
21         </Dimensão.Área>
22       </AreaDeJogo>
23       <AreaDeJogo>
24         <Nome>Descarte</Nome>
25         <Dimensão.Área>
26           <Nome>Nome</Nome>
27           <Tipo>Discreta</Tipo>
28           <Dinâmico>false</Dinâmico>
29         </Dimensão.Área>
30       </AreaDeJogo>
31     </Ambiente>
32   </Universo>
33   <Token>
34     <Atuador>true</Atuador>
35     <Dimensão>
36       <Nome>Naipes</Nome>
37       <Tipo>Discreta</Tipo>
38       <Dinâmico>false</Dinâmico>
39       <ValorDiscreto>Ouros</ValorDiscreto>
40       <ValorDiscreto>Espadas</ValorDiscreto>
41       <ValorDiscreto>Copas</ValorDiscreto>
42       <ValorDiscreto>Paus</ValorDiscreto>
43     </Dimensão>
44     <Dimensão>
45       <Nome>Número</Nome>
46       <Tipo>Sequência</Tipo>
47       <ValorMínimo>1</ValorMínimo>
48       <ValorMáximo>13</ValorMáximo>
49       <Dinâmico>false</Dinâmico>
50     </Dimensão>
51   </Token>
52 </gwl:SistemaDeJogo>
53

```

Baralho54.xml

5/2/2007

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <gwl:SistemaDeJogo xmlns:gwl="GameWrench" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
   "GameWrench GaML.xsd">
3    <Universo>
4      <Ambiente>
5        <AreaDeJogo>
6          <Nome>Maço</Nome>
7          <Dimensão.Area>
8            <Nome>Ordem (de cima para baixo)</Nome>
9            <Tipo>Sequência</Tipo>
10           <ValorMínimo>1</ValorMínimo>
11           <ValorMáximo>54</ValorMáximo>
12           <Dinâmico>true</Dinâmico>
13         </Dimensão.Area>
14       </AreaDeJogo>
15       <AreaDeJogo>
16         <Nome>Mesa</Nome>
17         <Dimensão.Area>
18           <Nome>Nome</Nome>
19           <Tipo>Discreta</Tipo>
20           <Dinâmico>false</Dinâmico>
21         </Dimensão.Area>
22       </AreaDeJogo>
23       <AreaDeJogo>
24         <Nome>Descarte</Nome>
25         <Dimensão.Area>
26           <Nome>Ordem (de cima para baixo)</Nome>
27           <Tipo>Sequência</Tipo>
28           <Dinâmico>false</Dinâmico>
29           <ValorMínimo>1</ValorMínimo>
30           <ValorMáximo>54</ValorMáximo>
31         </Dimensão.Area>
32       </AreaDeJogo>
33     </Ambiente>
34   </Universo>
35   <Token>
36     <Atuador>true</Atuador>
37     <Dimensão>
38       <Nome>Naipes</Nome>
39       <Tipo>Discreta</Tipo>
40       <Dinâmico>true</Dinâmico>
41       <ValorDiscreto>Ouros</ValorDiscreto>
42       <ValorDiscreto>Espadas</ValorDiscreto>
43       <ValorDiscreto>Copas</ValorDiscreto>
44       <ValorDiscreto>Paus</ValorDiscreto>
45     </Dimensão>
46     <Dimensão>
47       <Nome>Número</Nome>
48       <Tipo>Sequência</Tipo>
49       <ValorMínimo>1</ValorMínimo>
50       <ValorMáximo>13</ValorMáximo>
51       <Dinâmico>true</Dinâmico>
52     </Dimensão>
53   </Token>
54 </gwl:SistemaDeJogo>
55

```

APÊNDICE C – Projeto de Interação para interface de Motor de Jogo Genérico

Game-Wrench

Projeto de Interação para Especificação da Interface a ser utilizada em um Game Engine

I. Introdução

Este projeto de interação tem por objetivo ser um guia para o desenvolvedor-programador de jogos durante a tarefa de criação de um game engine.

Procura criar uma orientação para aproximar as funcionalidades implementadas e requisitos tecnológicos de um game engine ao modelo mental e às metas que um usuário típico possa ter com relação a este produto.

Portanto, este projeto não é um produto de software, mas sim um ferramental para desenvolvedores destes softwares.

Espera-se que, como resultado da utilização destas recomendações dentro de um projeto de um game engine, sejam produzidos softwares que ajudem as atividades dos game designers e maximizem seu potencial criativo.

II. Conceito do Produto

Game Engine

Conceito:

Um jogo eletrônico é um *software* destinado ao entretenimento.

Chama-se *game engine* a parte do jogo eletrônico que pode ser reaproveitada e compartilhada por outros *softwares*.

Identifica-se com tabuleiros, peças e baralhos tradicionais como sua contrapartida da era mecânica: qualquer um destes *sistemas* permite que inúmeros jogos sejam projetados utilizando-se deles.

Contexto:

Como qualquer *software*, em sua história, os jogos eletrônicos foram primeiramente projetados, desenvolvidos, testados e até comercializados por poucas pessoas, ou pequenas equipes.

Conforme a indústria foi crescendo, os projetos tornando-se mais complexos e inovadores, os orçamentos aumentando e os prazos reduzindo, equipes de dezenas de pessoas de diversas áreas foram sendo envolvidas no processo.

Neste cenário, flexibilidade, modularidade e reaproveitamento de código são muito importantes para tais desenvolvedores.

Apesar de ser interessante a idéia de um Game Engine genérico, capaz de produzir qualquer tipo de jogo, na sua implementação prática, tal engine não permitiria otimizações para um tipo específico de jogo, e encareceria o custo computacional de todos os jogos. Portanto, para este projeto será feita a escolha de um gênero ou classe de jogos para o qual o engine será voltado. Esta escolha será feita com base nas informações levantadas sobre os usuários do sistema.

Existem três universos a considerar para identificar os usuários de um game engine.

Primeiramente temos o universo *corporativo*, ou seja, game houses e empresas de desenvolvimento de software, com equipes e funções específicas e projetos com prazos e orçamentos bastante definidos.

Temos também o universo dos chamados *serious games*, onde profissionais buscam utilizar o ferramental dos jogos eletrônicos em aplicações fora do entretenimento. Por exemplo, professores utilizando jogos para transmitir conhecimento para seus alunos, ou psicólogos procurando tratar as fobias de seus pacientes.

Temos ainda o universo *amador*, constituído de pessoas interessadas em jogos, cuja paixão pelo tema motiva uma dedicação a um projeto de jogo em seu tempo livre.

Funcionalidade:

Um *game engine* permite que o trabalho de desenvolvimento de um jogo eletrônico seja reaproveitado em projetos futuros da mesma empresa, ou mesmo seja comercializado para desenvolvedores de terceiros.

É sobre um *game engine* que as regras de um jogo específico são implementadas.

Com base exposto na dissertação, uma ferramenta baseada em motor de jogo deverá, dentre outras funcionalidades, garantir as transgressões de primeira e segunda ordem, e deverá apresentar ao usuário, pelo menos os seguintes elementos identificados:

- Participantes (jogadores);
- Espaço de jogo;
- Interação e Atuadores;
- Tempo de jogo;
- definição formal de termos específicos;
- configuração inicial;
- mecânica do jogo;
- turnos;
- ações;
- objetivos;
- condição de fim.

III. Usuários

Lista e classificação de usuários

Universo Corporativo

Do ponto de vista comercial, um jogo eletrônico é um produto como qualquer outro software ou sistema digital. Envolve, portanto, empresas (de desenvolvimento, produção, comercialização, distribuição, suporte, etc.), as condições do mercado, e o público-alvo (consumidores).

O capítulo 5 de “Game Development and Production” elenca uma grande lista de atividades que percorre todas estas áreas do desenvolvimento

Classificação de Usuários:

Usuários primários:

- utilizam diretamente o sistema.

Usuários secundários:

- usam o sistema indiretamente, ou;
- usuários esporádicos, sempre precisam relembrar de seu funcionamento.

Usuários terciários:

não utilizam o sistema direta ou indiretamente.

- são afetados pelo sistema, ou;
- participam do processo de escolha ou compra do produto.

de um jogo eletrônico. A tabela abaixo resume esta lista*, associando a cada posto o tipo de empresa onde este profissional pode atuar, as competências desejadas e a sua classificação como usuário de um game engine.

Não é necessário que cada uma destas funções seja ocupada por uma pessoa específica ou todas elas sejam executadas pela mesma empresa ou grupo, normalmente o que acontece é um profissional atender por mais de uma destas responsabilidades, e haver terceirização de algumas funções.

Ainda, não é necessário que um projeto de jogo específico envolva todas estas atividades, e projetos mais complexos podem envolver atividades não listadas nesta tabela.

Função ou atividade	Tipo de Empresa	Competência	Classificação
Game Designer: Visionary	Desenvolvimento	Gerência	Secundário
Game Mechanics Designer	Desenvolvimento	Projetista	Primário ou Secundário
Level/Mission Designer	Desenvolvimento	Projetista	Primário ou Secundário
Story and Dialog Writer	Desenvolvimento	Projetista	Secundário
Lead Programmers and Technical Directors	Desenvolvimento	Programador	Primário ou Secundário
Game Mechanics Programmer	Desenvolvimento	Programador	Primário
3D Graphics Programmer	Desenvolvimento	Programador	Primário
Artificial Intelligence Programmer	Desenvolvimento	Programador	Primário
User Interface Programmer	Desenvolvimento	Programador	Primário
Audio Programmer	Desenvolvimento	Programador	Primário
Tools Programmer	Desenvolvimento	Programador	Desenvolvedor†
Mission/Level Editor Programmer	Desenvolvimento	Programador	Primário
Network Programmer	Desenvolvimento	Programador	Primário
Art Director	Desenvolvimento	Gerência	Terciário
Concept Artist	Desenvolvimento	Projetista	Terciário
2D Artist/ Interface Designer	Desenvolvimento	Artista	Primário
3D Modeler	Desenvolvimento	Artista	Secundário
Character Modeler	Desenvolvimento	Artista	Secundário
Texture Artist	Desenvolvimento	Artista	Secundário
Animator/Motion Capture Studio	Desenvolvimento	Artista	Secundário
Storyboarder	Desenvolvimento	Projetista	Primário
Voice-Overs	Desenvolvimento	Artista	Secundário
Sound Effects	Desenvolvimento	Projetista	Secundário
Music	Desenvolvimento	Artista	Secundário
Line Producer	Desenvolvimento	Apoio	Terciário
Associate Producer	Desenvolvimento	Gerência	Terciário
Studio Head/Executive Producer	Desenvolvimento	Gerência	Terciário
Producer	Desenvolvimento	Gerência	Terciário
QA Lead	Produção	Jogo e Tecnologia	Terciário
QA Main Team	Produção	Jogo	Terciário

* Os nomes dos cargos foram preservados no original em inglês para evitar traduções errôneas. O paralelo com o mercado nacional ficará a cargo do leitor.

† esta é a função de quem desenvolve o game engine, e portanto, não é considerado usuário

Função ou atividade	Tipo de Empresa	Competência	Classificação
QA Multiplayer Team	Produção	Jogo	Terciário
QA Fresh Teams	Produção	Jogo	Terciário
Compatibility Team	Produção	Tecnologia	Terciário
Localization Team	Produção	Idioma e Cultura	Primário ou Secundário
Beta Testers	Consumidor	Jogo	Terciário
Beta Testing Program Manager	Desenvolvimento	Relações Públicas	Terciário
Business Development Executive	Produção	Negócio	Terciário
Publisher CEO and President	Produção	Negócio	Terciário
Studio Heads	Desenvolvimento e Produção	Negócio	Terciário
Lawyers	Desenvolvimento e Produção	Direitos autorais, legislação	Terciário, em alguns casos específicos
Licensing	Produção	Direitos autorais; Contratos	Terciário
Sales Executive	Distribuição	Negociação; Comunicação	Terciário
Sales Force and Retail Purchasing Agents	Distribuição	Negociação; Comunicação	Terciário
Press Relations Manager	Distribuição	Comunicação	Terciário
Trade Shows	Distribuição	Comunicação	Terciário
Marketing	Distribuição	Comunicação	Terciário
Hardcore Fans	Consumidor	Jogo e tecnologia	Terciário
Manual	Distribuição	Comunicação	Terciário
Strategy Guide	Distribuição	Comunicação	Terciário
Console Manufacturers	Licenciamento e Produção	Jogo e tecnologia; Contratos	Terciário
Hardware Representatives	Produção	Tecnologia	Terciário
Patches and updates	Suporte	Jogo e tecnologia; Comunicação	Primário
FAQ e Forums	Suporte	Comunicação	Terciário
Customer Service	Suporte	Comunicação	Terciário
Gamemasters	Operação	Jogo; Comunicação	Primário ou Secundário

Tabela 3 a: Exemplo de distribuição de atividades produtivas no ciclo de vida de um jogo eletrônico comercial. Destaques em azul.

Desta tabela, é possível identificar as seguintes funções (sejam elas de usuários primários, secundários ou terciários) que possuem mais forte ligação/dependência com os game engines:

- Tools Programmer: por ser o responsável pelo desenvolvimento das ferramentas de programação, que agrupadas compõe o game engine e sua interface;
- Game Mechanics designer/programmer; Level designer/programmer: por serem os principais usuários diretos do engine;
- Art Director: todo o conteúdo artístico deverá ser digitalizado e carregado no jogo dentro dos requisitos impostos pela tecnologia. Cabe a este profissional garantir a padronização do trabalho das equipes sob sua direção;

- Localization Team; Patches and updates; Gamemasters: O trabalho destas equipes de suporte e operação de um jogo pode ser tão simples quanto trabalhar com uma tabela, ou tão complicado como analisar um código assembly, dependendo de como o engine se preocupou com estes assuntos;
- Licensing: Uma vez tendo um engine de sucesso, é esta equipe que realizará a venda do engine para outras empresas;
- Trade shows, Hardcore fans e Console manufacturers: Novos engines são notícia, orientam a indústria para futuros desenvolvimentos em hardware e atraem público a feiras e eventos;

Universo “Serious Games”

Desde sua edição de 2004, um dos maiores eventos sobre jogos eletrônicos, a Game Developer’s Conference (GDC) tem um capítulo separado – Serious Games Summit - somente para o tema de serious games. É um exemplo de como esta tendência está forte atualmente e tem chamado a atenção da comunidade mundial de desenvolvedores.

Tomados os devidos cuidados, existem vários casos de sucesso da aplicação de *game engines* para resolver problemas sérios como educação, visualização científica, treinamento e até mesmo tratamentos psiquiátricos.

Universo Amador

Estes usuários são grandes entusiastas de games e apreciadores das novas tecnologias que aparecem a cada novo lançamento. Costumam ter suas atenções tomadas pelo *game* que estão jogando no momento e aproveitam qualquer oportunidade para personalizar e modificar estes jogos. Sonham em um dia publicar o melhor jogo do mundo, enquanto isso não acontece, experimentam todo game novo que aparece.

Escopo e Levantamento de informações

Para este projeto de interação estão sendo considerados como mais importantes e, portanto prioritários os seguintes tipos de usuários:

Do universo comercial:

- Game Mechanics designer/programmer;
- Level designer/programmer;
- Art Director;
- Localization Team;
- Patches and updates;
- Gamemasters;

Além destes, acrescentamos os usuários de serious games:

- Jogos para Educação;

Os usuários amadores não farão parte da prioridade deste projeto.

Procedimento

Quantificadores de Usabilidade:

- **Tempo de Aprendizado** (*learnability*):

tempo necessário para um usuário novato tornar-se um usuário mediano.

- **Retenção** (*memorability*):

após um tempo sem utilizar a ferramenta, quanto deverá ser re-aprendido pelo usuário.

- **Taxa de erros** (*error rate*):

que parcela do trabalho precisa ser refeito.

- **Desempenho do usuário** (*efficiency*):

quanto trabalho é realizado em um período de tempo.

- **Satisfação subjetiva** (*satisfaction*):

o quanto os usuários declaram-se satisfeitos com o sistema.

Identificou-se as seguintes formas com as quais se pode obter informações sobre estes usuários eficientemente:

- Pesquisa qualitativa, orientada através pelos questionários conforme Anexo E.
- O critério de escolha dos entrevistados, conforme descrito abaixo.

Critério de seleção

Profissionais de empresas:

- Com mais de um jogo comercial publicado;
- Desenvolvimento de jogo tanto através de Engines terceirizados, quanto desenvolvimento interno;
- Atua como programador de engine (usuário primário) ou Game Designer (usuário secundário-indireto).

Serious Games:

- Professores Universitários;
- Ministrando disciplinas através de Ensino à Distância

Gênero de Jogos melhor atendido

A partir deste recorte, identificou-se como classe, ou gênero, de jogos para o qual o game engine sendo produzido destina-se: aqueles jogos cujo **componente de simulação** seja predominante (o que permitirá melhores resultados para o universo serious games).

Um exemplo são os jogos que simulam jogos eletro-mecânicos como Pinball.

Quantificadores de Usabilidade

Da mesma forma, podemos priorizar os seguintes quantificadores de usabilidade, de forma a ajudar o processo decisório do projeto:

- Tempo de aprendizado
- Desempenho do usuário

Modelagem de usuários

Com os personagens descritos a seguir, espera-se estar representando a contento uma grande parte dos usuários de um *game engine*, de forma a otimizar o processo de projeto.

Estes personagens inicialmente propostos serão continuamente validados e corrigidos pelos resultados obtidos através dos questionários.

Juca Politreco

Representa:

Serious Games - Jogos para Educação: aluno;

Características Físicas:

Sexo: masculino; Idade: 25 anos;

Outros: cabelo comprido, óculos grossos.

Características de Personalidade e Cognição:

Cognição: Visual e Especialista;

Arquétipo Protagonista:**Propósito:**

- Conhecimento (aquilo que se considera verdade);
- Realidade *Actuality* (como as coisas são, realidade objetiva);

Avaliação:

- Provado (algo é correto pelos padrões relevantes);
- Efeito (resultado específico forçado por uma causa);

Motivação:

- Consideração (Pesa os prós e contras);
- Busca (vontade para perseguir);

Metodologia:

- Certeza (Determinação de que algo é absolutamente verdade);
- Pró-atividade (Ter a iniciativa).

Signo: Aquário;

Arquétipo: Protagonista (caminha em direção construtiva aos objetivos);

Características Sócio-Culturais:

Profissão: aluno da Poli, entre estágios;

Apelido: "Politreco" foi o apelido que ganhou dos seus colegas de colegial quando entrou na faculdade;

Estado civil: Solteiro, mora com os pais;

Habilidades e Competências:

Dirige automóveis; Inglês intermediário; Computação; Conhece tudo sobre *hard rock* dos anos 90;

Metas de Vida:

- Terminar a faculdade e receber o tão sonhado diploma de Engenharia;
- Trabalhar na Microsoft ou IBM;
- Comprar seu primeiro carro;
- Criar um jogo eletrônico de sucesso;

Metas de Experiência:

- Aplicar seus conhecimentos de programação e ver os resultados se transformarem em um produto comercial;

Metas Fins:

- Deve utilizar a ferramenta como um projeto para a disciplina de Computação Gráfica que está cursando;
- Usar modelos 3D e sons com tema de sua banda favorita;
- Entregar um jogo funcional "nota dez" ao final do semestre e, sendo assim, continuar o projeto para se tornar um produto comercial posteriormente;

Mr. Ello Maroto

Representa:

Game Mechanics designer/programmer;
Level designer/programmer;
Art Director;

Características Físicas:

Sexo: masculino
Idade: 50 anos
Outros: não aparenta a idade

Características de Personalidade e Cognição:

Estilo Cognitivo: Visual e Holista
Signo: Câncer
Arquétipo-Base: Protagonista

Características Sócio-Culturais:

Profissão: Lead Game Designer da Game in the Box
(Estúdio de Games de médio porte)

Arquétipo Contagonista:**Propósito:**

- Desigualdade (desbalanceamento, injustiça ou falta de igualdade);
- Especulação (Extensão de possibilidades para o futuro);

Avaliação:

- Determinação (Conclusão baseada em evidências circunstanciais);
- Interminável (Continua sem interrupção);

Motivação:

- Tentação (A urgência de receber benefícios imediatos sem importar-se com as possíveis consequências);
- Sabotagem (Minando os esforços alheios);

Metodologia:

- Produção (Processo que determina potencial);
- Reavaliação (Revisão de uma situação ou circunstâncias)

Estado civil: Solteiro, sem filhos.

Habilidades e Competências:

Mantem-se informado sobre todas as novas tecnologias, apesar de considerar programação uma atividade secundária;
É um desenhista invejável e costuma usar diagramas e gráficos para orientar suas equipes.

Metas de Vida:

- Obter liberdade criativa para seus jogos junto aos produtores;
- Ter seu portfólio de jogos considerado como obra artística;

Metas de Experiência:

- Que a ferramenta não restrinja sua criatividade;
- Aumentar sua capacidade de comunicação com sua equipe de desenvolvedores e artistas;

Metas Fins:

- Criar storyboards;
- Criar esboço da interface do jogo;
- Organizar e visualizar objetos de jogo (game content);
- Manipular mapas;
- Adicionar comportamentos (IA) a objetos e personagens;
- Representar graficamente as interdependências entre objetos e recursos de um jogo (mecânica do jogo);
- Fazer o “ajuste fino” da mecânica do jogo;

Rubio Egghead
Representa:

Localization Team;
Patches and updates;
Gamemasters;

Características Físicas:

Sexo: masculino
Idade: 40 anos
Outros: Careca;

Fica vermelho quando está nervoso;

Características de Personalidade e Cognição:

Estilo Cognitivo: Auditivo e Especialista
Signo: Touro
Arquétipo-Base: Contagonista

Características Sócio-Culturais:

Profissão: Gerente de TI da Game in the Box
Estado civil: Casado, com 3 filhas entre 9 e 12 anos.

Habilidades e Competências:

Certificação de Administrador de Banco de Dados Oracle e
Administrador de Redes NovellTM

Metas de Vida:

- Comprar o carro do ano
- Ir com a família para conhecer a Disney World

Arquétipo Guardiã:**Propósito:**

- Igualdade (balanceamento, justiça, estabilidade);
- Projeção (extensão de possibilidades para o futuro);

Avaliação:

- Expectativa (Conclusão como o eventual efeito de uma causa particular);
- Encerramento (chegar a uma conclusão);

Motivação:

- Consciência (obter prazer imediato ou benefício por consequências futuras);
- Apoio (assintência indireta dada aos esforços de outros).

Metodologia:

- Redução (Método de raciocínio que determina probabilidade);
- Avaliação (Verificação de uma situação ou circunstâncias).

Metas de Experiência:

- Garantir que o software não lhe comprometa;
- Saber executar tarefas não-triviais;

Metas Fins:

- Traduzir os diálogos e menus do jogo
- Conferir quantidade de features implementadas por suas equipes (relatórios *to-do* e *bug-tracking*)
- Adicionar conteúdo dentro da mecânica do jogo
- Poder expulsar jogadores online infratores

Prof. Emmet Xavier aka Prof. Pardal
Representa:

Serious Games: Jogos para Educação;

Características Físicas:

Sexo: masculino

Idade: 40 anos

Outros: careca.

Características de Personalidade e Cognição:

Cognição: Visual e Holista

Signo: Peixes

Arquétipo: Guardiã (professor ou ajudante, serve de padrão moral)

Características Sócio-Culturais:

Profissão: professor do ensino médio (Física)

Apelido: Prof. Pardal (conhecido, mas não-declarado)

Estado civil: Divorciado, 1 filho de 17 anos, mora com a mãe, passa os Domingos com o pai.

Habilidades e Competências:

Inglês

Computação

Uso de mídias não-convencionais para o ensino

Metas de Vida:

- Conseguir admiração de seus alunos, ser o paraninfo na formatura da turma;
- Aproveitar as férias escolares para fazer viagens internacionais;
- Ter mais tempo “de qualidade” com seu filho

Metas de Experiência:

- Ferramenta “sólida” de desenvolvimento rápido e resultados de qualidade;
- Programação semelhante ao seu repertório (Pascal e um pouco de VB)

Metas Fins:

- Criar jogos simples para ilustrar conceitos das aulas de Física Newtoniana

Arquétipo Emoção:**Propósito:**

- Desejo (obter algo melhor);
- Auto-conhecimento (consciência da própria existência).

Avaliação:

- Instinto (conhecimento baseado em evidências insuficientes);
- Teste (experimento para determinar a validade de algo).

Motivação:

- Emoção (sensação emocional sobre a situação);
- Incontrolada (sem direção e irrestrita).

Metodologia:

- Possibilidade (algo pode ser verdadeiro);
- Proteção (esforço para prevenir intervenções).

Buba Zeraueca **Representa:**

Art Director;

Características Físicas:

Sexo: masculino

Idade: 31 anos

Outros: Cabelo armado

Características de Personalidade e Cognição:

Estilo Cognitivo: Visual

Signo: Virgem

Arquétipo-Base: Emoção

Características Sócio-Culturais:

Profissão: Diretor artístico da Game in the Box

Estado civil: Solteiro

Habilidades e Competências:

Modelagem 3D;

Discurso;

Representações e Impersonificações;

Metas de Vida:

- Visitar Paris;
- Ser famoso;
- Morar em um loft espaçoso, com vista urbana;

Metas de Experiência:

- Estar “sintonizado” com a ferramenta
- Poder ajustar o resultado até os mínimos detalhes

Metas Fins:

- Criar padrões de estilo
- Transformar desenhos 2D em modelos 3D
- Aplicar texturas a modelos 3D

Joãozinho Longneck **Representa:**

Gamemasters;

Voice-Over; Story/Dialog Writer

Características Físicas:

Sexo: masculino

Idade: 25 anos

Outros: Nariz grande, pele e cabelos oleosos

Características de Personalidade e Cognição:

Estilo Cognitivo: Especialista

Signo: Touro

Arquétipo-Base: Sidekick

Arquétipo Sidekick:**Propósito:**

- Ordem (uma disposição padronizada);
- Inércia (continuação de um estado ou processo);

Avaliação:

- Precisão (estar dentro das margens de erro);
- Resultado (produto de um processo; ramificações de um efeito específico)

Motivação:

- Fé (aceitar algo como certo sem provas);
- Apoio (assistência indireta aos esforços alheios);

Metodologia:

- Dedução (método de raciocínio que determina certeza);
- Aceite (Decisão de permitir, tolerar, adaptar-se ou não se opor).

Características Sócio-Culturais:

Profissão: Gamemaster da Game in the Box

Estado civil: Solteiro

Habilidades e Competências:

Mestre de RPG

Jogos de luta

Mitologia Star Trek

Metas de Vida:

- Ser respeitado na comunidade online
- Conhecer muitas garotas

Metas de Experiência:

- Deixar sua marca no conteúdo criado para os jogos

Metas Fins:

- Gravar diálogos (voz) do jogo
- Escrever um script (roteiro) para os personagens do jogo
- Criar missões / Adicionar conteúdo dentro da mecânica do jogo

IV. Prototipagem**Tabela Usuários x Tarefas**

	Mr. Ello Maroto	Rubio Egghead	Prof. Emmet Xavier	Buba Zeraueca	Joãozinho Longneck
T1: Criar um jogo completo, porém simples			60% / alta	5% / média	
T2: Criar protótipo de jogo ou exemplificar um conceito de jogo	10% / média			5% / média	
T3: Adicionar comportamentos (IA) a objetos e personagens	5% / baixa		5% / baixa		
T4: Manipular mapas	5% / baixa				10% / alta
T5: Organizar e visualizar objetos de jogo (game content)	5% / baixa	25% / alta	20% / baixa	25% / alta	
T6: Criar esboço da interface do jogo	10% / média			5% / alta	
T7: Criar storyboards	10% / média			5% / alta	5% / média
T8: Traduzir os diálogos e menus do jogo		20% / alta			
T9: Expulsar jogadores online infratores		10% / baixa			20% / baixa
T10: Criar missões ou adicionar conteúdo dentro da mecânica do jogo	5% / baixa	15% / alta		5% / baixa	30% / alta

T11: Conferir quantidade de features implementadas por suas equipes (relatórios to-do e bug-tracking)	10% / alta	30% / alta		20% / média	
T12: Criar padrões de estilo	5% / média			5% / alta	
T13: Transformar desenhos 2D em modelos 3D				10% / baixa	
T14: Aplicar texturas a modelos 3D			15% / baixa	10% / baixa	
T15: Gravar diálogos (voz) do jogo					15% / média
T16: Escrever um script (roteiro) para os personagens do jogo	10 % / média			5% / alta	20% / baixa
T17: Representar graficamente as interdependências entre objetos e recursos de um jogo (mecânica do jogo)	10% / média				
T18: Fazer o “ajuste fino” da mecânica do jogo	15% / alta				

Tabela 4 a: Persona x Meta fim. (Porcentagem de tempo de uso do sistema dedicado às tarefas / criticidade das tarefas). Destaques em amarelo indicam relação tarefa/usuário prioritária

Criação de cenários

Cenário 1. Jogo de Física

Personagem:

Emmet Xavier

Contexto:

Parte 1: Sem manual

No meio do Semestre de Física para 1º. Colegial, Prof. Emmet precisa falar de Newton, explicar conceitos como aceleração da gravidade e das experiências de Galileu.

Sendo entusiasta dos métodos não-convencionais de ensino, procura uma ferramenta que possa produzir um objeto de aprendizado para esta parte da matéria, gostaria que fosse um jogo que os alunos pudessem usar em casa e no laboratório do colégio.

Depois de adquirido e instalado o demo da ferramenta numa tarde de Domingo, pretende ter algo para surpreender os alunos na aula de Quinta-Feira.

Parte 2: Tutorial

Após uma primeira exploração com o software, percebe que precisará de ajuda para terminar seu primeiro jogo simples. Procura no site do programa

e encontra alguns tutoriais que podem ajudá-lo. Por sorte, o inglês não será um impecilho para ele.

Na Terça-Feira terá 4 horas livres, quando poderá realizar estes tutoriais e tentar produzir um jogo convincente.

Objetivo:

Criar jogo sobre efeito da gravidade terrestre

Tarefas:

1. Criar “chão”
2. Criar objetos
3. Representar visualmente atributos físicos, associados aos objetos
4. Interface para modificar estes atributos / criar “forças” durante execução
5. Detectar e Tratar colisões entre “chão” e objetos

Cenário 2. Game Studio

Contexto:

Este cenário é composto por diversos eventos interligados.

A empresa Game in the Box é um game studio com alguns anos de existência. Depois de um rápido crescimento, propiciado por alguns projetos pequenos bem sucedidos, conta com aproximadamente 50 colaboradores, dentre eles os personagens em questão.

A empresa acaba de fechar seu primeiro grande contrato, para um jogo relacionado a um programa de televisão, um MMORPG baseado em ambiente 3D, com fortes componentes de simulação. A primeira versão do jogo, um projeto-piloto com capacidade para até 100 jogadores simultâneos deverá ser entregue em 4 meses.

Parte 1: Conteúdo Artístico

Buba Zeraueca tem uma equipe composta de três profissionais, encarregados da criação de conteúdo: criação dos modelos 3D, aplicação de texturas, animação de movimentos, som e efeitos especiais.

Ao receber a documentação do jogo (Initial Treatment), sua primeira missão é criar imagens e textos descritivos para todo conteúdo que deverá ser criado por sua equipe. Ao mesmo tempo, deve definir um conjunto de diretrizes (manual de estilo) para manter o estilo visual entre todos os objetos e personagens do jogo.

Com isso feito, precisa acompanhar de perto o trabalho de sua equipe, para garantir a qualidade final do trabalho que receberá sua assinatura.

Personagem:

- Buba Zeraueca

Objetivos:

- Organizar todo o conteúdo artístico para o projeto-piloto (ambientado em um cenário urbano futurista - cyberpunk);

- Elaborar e manter o manual de estilo do jogo (parte do game design document);

Tarefas:

- Criar um cenário de jogo: estação de metrô;
- Associar um desenho de esboço deste cenário, e alguns comentários;
- Associar a este cenário objetos de cenário (props): Lixeira, Banco, etc.;
- Associar a este cenário: 3 NPCs: Guarda do metrô, Flautista peruano e Velinha com cachorro.
- Associar a este cenário: 7 tipos de inimigos, sendo um deles um mini-boss;
- Associar requisitos tecnológicos a todo este conteúdo: número máximo de polígonos por modelo, paleta de cores permitidas;
- Delegar funções a seus subalternos;
- Gerar documentação e exportar para o Game Design Document;

Parte 2: Crunch Time

Estando a apenas 15 dias do deadline, o sistema registra que 80% do jogo está pronto, mas parece que não está nem na metade. A empresa entrou em “Crunch Mode”, todos trabalhando até altas horas, almoçando e jantando na empresa.

Enquanto Mr. Ello se trancou em sua sala, fazendo o ajuste fino da trama principal do jogo, a equipe de Rubio, da qual faz parte Joãozinho, está encarregada de criar 3 sidequests. Tudo isso dentro do cenário do metrô, já criado, mas com alguns elementos faltando.

Personagem:

- Mr. Ello Maroto

Objetivos:

- Fazer o “ajuste fino” da mecânica do jogo.

Tarefas:

- Modificar uma característica de um inimigo: agilidade e frequência de respawn;
- Verificar através de probabilidades, qual o tempo mínimo, máximo e médio de um jogador atingir nível 50 dentro deste cenário;
- Elaborar mais um tipo de inimigo, determinar suas características, e enviar para a equipe de Buba adequar este personagem ao contexto do cenário.

Personagem:

- Joãozinho Longneck

Objetivos:

- Criar um side-quest no cenário do metrô.

Tarefas:

- Criar flags de 1 a 3;
- Criar diálogo para NPC Velhinha com Cachorro, modificando flags 1 e 3;

- Modificar um objeto Lixeira de prop para NPC;
- Adicionar objeto lógico “carteira vazia” ao NPC lixeira;
- Criar diálogo entre Lixeira e jogador, testando flag 1 e modificando flag 2;
- Adicionar objeto lógico “nota musical” ao NPC Flautista peruano;
- Criar diálogo entre Flautista peruano e jogador, testando flag 3.
- Compilar e executar o jogo;
- Testar o sidequest, usando um personagem padrão;

Personagem:

- Rubio Egghead

Objetivos:

- Conferir andamento dos três side-quests designados a sua equipe

Tarefas:

- Selecionar um side-quest de sua equipe;
- Verificar a porcentagem de funcionalidades implementadas;
- Em caso de atraso, verificar o responsável pelas tarefas;
- Enviar e-mail para o responsável, pedindo relatório do side-quest;

Cenário 3. Aula de Engine

Personagem:

- Juca Politreco

Contexto:

Este é o primeiro contato do aluno com a ferramenta, após receber o arquivo do website e instalá-lo em seu ambiente de desenvolvimento pessoal. A primeira reunião de brainstorm do grupo já foi realizada, bem como a divisão de tarefas. Juca está sozinho em sua casa após o período normal de suas aulas (por volta de 18h00).

Objetivos:

- Criar o projeto no sistema.
- Visualizar os recursos necessários para conclusão do projeto.
- Terminar o semestre com uma versão bem-acabada do jogo, e uma boa nota.

Tarefas:

- Criar projeto
- Definir personagem
- Definir etapas
- Programar objetivo do jogo
- Definir cenário
- Definir classes de objetos do jogo

V. Especificação

Requisitos

Requisitos de usuário:

Características que o usuário deve possuir.

Requisitos de usabilidade:

Limites que os quantificadores de usabilidade devem apresentar.

Requisitos funcionais:

Funcionalidades que o sistema deve ter.

Requisitos não-funcionais:

Propriedades que o sistema deve apresentar.

Requisitos de dados:

Formatos de arquivo, restrições de tamanho e transmissão.

Requisitos de ambiente:

Características externas ao sistema.

Análise Comparativa

Utilizando o método de Avaliação Heurística, foram realizados testes sobre ferramentas de software com proposta semelhante à do Game Wrench: Game Maker v6.1 e Game Studio A6.

Levantamento de Requisitos

Os requisitos listados abaixo foram coletados da literatura e de todo o processo do Projeto de Interação.

Requisitos de usuário:

- Nível de conhecimentos em linguagem de programação entre “básico” e “expert”;
- Fazer parte do universo “comercial” ou “serious games”;

Requisitos de usabilidade:

- Tempo de aprendizado: menor que 1 hora;
- Desempenho: Completar um jogo simples inteiro (um pacman ou tetris, por exemplo) em no máximo 1 hora;

Requisitos funcionais:

- Inferir os objetivos (metas-fim) dos usuários através de wizards e utilizá-las ao longo do projeto;
- “Sugerir” ao usuário alguns próximos passos em direção a seus objetivos, caso este esteja numa situação de impasse;
- Visualização por escopo “Geral / Produção”, “Projeto / Game Design” e “Tarefa / Desenvolvimento” – suporte a multi-usuário;
- Adotar modelo de desenvolvimento focado na definição de jogo, nas características fundamentais de jogo;
- Controlar elementos e recursos de jogo e complementares;
- Separação visual entre lógica e apresentação do jogo;
- Criação automática de documentação para desenvolvedor, produtor e usuário-final (jogador), com auxílio de “Wizards”;
- Mecanismo flexível de troca de eventos e mensagens para interação entre objetos de jogo;
- Ferramenta de programação de alto nível, abstraindo boa parte dos detalhes técnicos, com mecanismos robustos de detecção de erros e debug, e evitar gerar comportamentos indefinidos por ambiguidade na implementação;

Requisitos não-funcionais:

- Atuar como uma plataforma de aplicações (framework), provendo infraestrutura de customização (como as ferramentas de MOD);
- Integrar ferramentas auxiliares, como gerador de mapas e modelos 3D, de forma confiável e com alta usabilidade;
- Possuir bibliotecas de recursos artísticos prontos, como modelos 3D de objetos e personagens, efeitos sonoros, etc;
- Especificação de comandos de alto nível pelos quais será possível a interface flexível com game engines, APIs e ferramentas de terceiros;

Requisitos de dados:

- Auto-save do progresso do usuário;
- Suporte a arquivos de imagem estática adquirida: PNG, JPG;
- Suporte a arquivos de imagem estática vetorial: SVG;
- Suporte a arquivos de modelos 3D: 3DS, X3D, VRML;
- Suporte a arquivos de animação 3D: 3DS;
- Utilizar XML para armazenamento dos projetos de jogo, com criação de parsers responsáveis pela integração entre as decisões de projeto e o acesso às funcionalidades dos game engines;

Requisitos de ambiente:

- Ambiente de programação padrão.

VI. Saiba Mais:***Projeto de Interação***

Cooper, A.; Reimann, R. *About Face 2.0: The essentials of interaction design*, Wiley Publishing, Inc., 2003.

Nielsen, J. *Ten Usability Heuristics*, Página web, Acesso em: 20/03/2006.
Disponível em: http://www.useit.com/papers/heuristic/heuristic_list.html

Modelagem de Personagens

Philips, M.; Huntley, C. *Dramatica: A New Theory of Story*, Third Edition, Screenplay Systems Incorporation, 1996.

Projeto de Jogo: Game Design

Bethke, E. *Game Development and Production*, Wordware Publishing, Inc., 2003

Game Engine

Nakamura, R. *et al. A Practical Study on the Usage of a Commercial Game Engine for the Development of Educational Games* In: Anais do congresso WJogos 2003.

APÊNDICE D – Análise Comparativa através de Avaliação Heurística

Avaliação Heurística

13/03/06

Avaliadores:

Jacober

Thomazini

Ferramenta avaliada:

Game Maker v6.1

Cenário 1**Personagens:**

Emmet Xavier

Contexto:**Parte 1**

No meio do Semestre de Física para 1º. Colegial, Prof. Emmet precisa falar de Newton, explicar conceitos como aceleração da gravidade e das experiências de Galileu.

Sendo entusiasta dos métodos não-convencionais de ensino, procura uma ferramenta que possa produzir um objeto de aprendizado para esta parte da matéria, gostaria que fosse um jogo que os alunos pudessem usar em casa e no laboratório do colégio.

Depois de adquirido e instalado o demo da ferramenta Game Maker v6.1,

numa tarde de Domingo, pretende ter algo para surpreender os alunos na aula de Quinta-Feira.

Parte 2

Após uma primeira exploração com o software, percebe que precisará de ajuda para terminar seu primeiro jogo simples. Procura no site do programa e encontra alguns tutoriais que podem ajudá-lo. Por sorte, o inglês não será um impedimento para ele.

Na Terça-Feira terá 4 horas livres, quando poderá realizar estes tutoriais e tentar produzir um jogo convincente.

Objetivo:

Criar jogo sobre efeito da gravidade terrestre

Tarefas:

1. Criar “chão”
2. Criar objetos
3. Representar visualmente atributos físicos, associados aos objetos
4. Interface para modificar estes atributos / criar “forças” durante execução
5. Detectar e Tratar colisões entre “chão” e objetos

Resumo:

Durante a primeira parte, o Prof. Emmet mal conseguiu terminar a primeira tarefa (criar chão), tendo feito diversas interpretações erradas das funcionalidades do sistema. Por diversas vezes, sentiu falta de funções tipo “Arrastar e Soltar” do painel de recursos (à esquerda da interface) para o painel de edição (centro).

Mas, apesar desta distância cognitiva, sentiu que a ferramenta pode atender seus objetivos, e busca ajuda no arquivo de help, jogos-demo e tutoriais disponíveis.

Na segunda parte, já familiarizado com o estilo interativo da ferramenta, conseguiu chegar bem mais próximo de seu objetivo, a menos da possibilidade de entrar com parâmetros (como aceleração da gravidade, por exemplo) no jogo de forma adequada (ficou “hard-coded”).

Problemas de Usabilidade Detectados:

ID	Contexto	Descrição	Heurísticas Quebradas	Criticidade
1.1	Ao iniciar o software	Existe muito pouca informação sobre o que deva ser feito a seguir, quais são as possibilidades.	H1: visibilidade	Alta
1.2	Ao iniciar o software	Usuário tenta <i>double-click</i> nas pastas de <i>resources</i> e nada acontece.	H7: flexibilidade e eficiência	Média
1.3	Ao clicar em <i>Load Sprite</i> no diálogo <i>sprite properties</i>	É exibido um diretório vazio, que não contem nenhuma imagem. O usuário precisa lembrar onde está instalado o software, na esperança de achar sprites adequados	H6: reconhecimento vs. Lembrança	Baixa
1.4	No diálogo <i>sprite properties</i>	Confusão entre as propriedades <i>transparent</i> e <i>visible</i> .	H4: consistência e padrões	Baixa
1.5	Ao tentar utilizar a <i>sprite</i> recém criada	Confusão entre conceitos de <i>sprite</i> , <i>object</i> e <i>room</i> , e seu inter-relacionamento. Tentativa de <i>drag-and-drop</i> da <i>sprite</i> a área de trabalho não funciona.	H4; H7	Alta
1.6	Ao tentar criar um <i>room</i>	Usuário tenta <i>double-click</i> nas pastas de <i>resources</i> e nada acontece.	H7: flexibilidade e eficiência	Média
1.7	No diálogo <i>object properties</i>	Pouca informação sobre as propriedades dos objetos (<i>visible</i> , <i>solid</i> , <i>depth</i> e <i>persistent</i>) podem levar a interpretações equivocadas	H4: consistência e padrões	Baixa
1.8	No diálogo <i>room properties</i>	A grid do <i>room</i> não cabe na tela padrão, precisando ser maximizada duas vezes.	H8: estética e design minimalista	Baixa
1.9	Ao adicionar um <i>object</i> ao <i>room</i>	Usuário tenta utilizar <i>drag-and-drop</i> ou <i>double-click</i> dos <i>resources</i> para o <i>room</i> . Precisa	H7	Média

		ler as instruções para entender como associar um object ao click esquerdo do mouse.		
1.10	Ao acrescentar objetos por engano no room, e tentativa de correção	Usuário tenta usar atalhos comuns como DEL, ESC, BCKSPC e CTRL+Z, mas não consegue, novamente, precisa ler instruções na tela para descobrir o click direito do mouse, ou botão de <i>undo</i> .	H7; H3: undo e redo	Baixa
1.11	Ao precisar editar o tamanho do sprite para o objeto escolhido	Ao verificar que o sprite associado ao objeto apresenta-se menor do que o desejado, o usuário tenta editá-lo redimensionando diretamente no room, e não consegue. Consegue fazê-lo somente através do botão <i>edit sprite</i> no diálogo <i>object properties</i> .	H7; H6	Alta
1.12	Ao redimensionar um <i>sprite</i> no diálogo <i>sprite editor</i>	Confusão entre as funcionalidades <i>stretch</i> , <i>shift</i> e <i>scale</i> .	H4	Baixa
1.13	Ao redimensionar um <i>sprite</i> no diálogo <i>sprite editor</i>	Para verificar se o tamanho do sprite se adequa ao tamanho do room, várias operações de <i>stretch</i> foram realizadas, alternando-se entre diálogos. Durante o processo, um erro de parâmetro (1000% em vez de 200%) causou lentidão e estouro de memória.	H5: Prevenção de erros; H6; H9: Recuperação de erros	Alta

Tabela 1: Problemas de Usabilidade encontrados em Game Maker v6.1

Avaliação Heurística

27/03/06

Avaliadores:

Jacober

Thomazini

Ferramenta avaliada:

3D GameStudio A6

Cenário 1**Personagens:**

Emmet Xavier

Contexto:**Parte 1**

No meio do Semestre de Física para 1º. Colegial, Prof. Emmet precisa falar de Newton, explicar conceitos como aceleração da gravidade e das experiências de Galileu.

Sendo entusiasta dos métodos não-convencionais de ensino, procura uma ferramenta que possa produzir um objeto de aprendizado para esta parte da matéria, gostaria que fosse um jogo que os alunos pudessem usar em casa e no laboratório do colégio.

Depois de adquirido e instalado o demo da ferramenta Game Maker v6.1, numa tarde de Domingo, pretende ter algo para surpreender os alunos na aula de Quinta-Feira.

Parte 2

Após uma primeira exploração com o software, percebe que precisará de ajuda para terminar seu primeiro jogo simples. Procura no site do programa e encontra alguns tutoriais que podem ajudá-lo. Por sorte, o inglês não será um impedimento para ele.

Na Terça-Feira terá 4 horas livres, quando poderá realizar estes tutoriais e tentar produzir um jogo convincente.

Objetivo:

Criar jogo sobre efeito da gravidade terrestre

Tarefas:

1. Criar “chão”
2. Criar objetos
3. Representar visualmente atributos físicos, associados aos objetos
4. Interface para modificar estes atributos / criar “forças” durante execução
5. Detectar e Tratar colisões entre “chão” e objetos

Resumo:

Parte 1

Ao tentar explorar o uso da ferramenta, conseguiu muito pouco resultado.

As opções apresentadas gratuitamente pela ferramenta na forma de ícones pouco ajudaram nas tarefas em questão, ou por falha de interpretação ou por estar fora do escopo.

Funcionalidades básicas esperadas como aplicação de texturas aos modelos não foram sequer visualizadas na interface.

Parte 2

Na Terça-Feira, Emmet começa o tutorial que descobriu no site da ferramenta, intitulado *wed_tutorial_e.chm*. Dentro deste arquivo, após breve análise, resolve começar pela parte intitulada *LevelDesign_Practice* (Dentre as opções LD_Tutorial, GD_BasicScripts e GD_AI). Apesar de sua expectativa ser seguir o tutorial e ir adaptando às suas necessidades, verifica dificuldades nesse sentido, desistindo desta tentativa.

Depois de uma hora de tutorial, ainda no começo, percebe dois fatos:

- as 3 horas que lhe restam não lhe permitirão terminar o tutorial detalhadamente a tempo;
- os tempos de compilação sobem exponencialmente conforme o nível vai ficando complexo;

Escolhendo pular toda parte do tutorial sobre texturas, acredita resolver o primeiro problema. Desta forma, termina o tutorial com aproximadamente duas horas restantes e parte para confecção de seu jogo/simulação.

O conhecimento adquirido com o tutorial garante uma agilidade grande na construção e modelagem de um cenário 3D, embora encontre dificuldades relacionadas com os diversos tipos de arquivo com que a ferramenta trabalha.

Apesar de estar recente em sua memória, precisou retornar algumas vezes a pontos específicos do tutorial para esclarecer promenores.

Ao procurar um script pronto para fazer os objetos caírem em direção ao solo, não consegue fazer o jogo rodar – provável falha do sistema (bug). Sabe que o tutorial não diz nada sobre isso...

Falta apenas uma hora, está cansado, decide continuar no dia seguinte, se encontrar um tempo livre.

Problemas de Usabilidade Detectados:

ID	Contexto	Descrição	Heurísticas Quebradas	Criticidade
1.1	Ao iniciar o software	Existe apenas uma interface modo “expert”	H10: Ajuda e documentação	Alta
2.1	Ao adicionar uma câmera	Não lembrava qual o procedimento adequado para esta função, precisando consultar o tutorial “first room”	H6:reconhecimento ao invés de lembrança	Baixa
2.2	Ao inserir objetos	Ausência de um assistente para que o usuário acrescente luz, câmera, objetos, sprites, etc	H10: ajuda e documentação; H6: reconhecimento vs. Lembrança	Média

ID	Contexto	Descrição	Heurísticas Quebradas	Criticidade
2.3	Ao tentar enxergar objetos em detalhes	Não foi achado a opção zoom in/out	H6: reconhecimento vs. Lembrança	Média
2.4	Ao gerar objetos, sprites, etc...	O sistema gera objetos com nomes estranhos, não auto explicativos	H4: consistência e padrões; H2: Concordância entre sistema e mundo real	Baixa
2.5	Ao compilar um jogo com certo grau de complexidade	O sistema demora muito para compilar um jogo, dependendo do número de objetos envolvidos. Isso interrompe o fluxo de trabalho: o sistema não oferece uma “compilação light”, nem permite alterações neste íterim. Mas existe a visualização texturizada...	H3: Controle e liberdade do usuário	Alta
2.6	Ao tentar compilar o jogo	Mensagem dizendo que não foi gerado o “WMB”	H5: Prevenção de erros	Média
2.7	Ao tentar compilar o jogo	Não ficou claro se gerar o “WMB” será sempre necessário ou se é possível compilar sem ele (se não for, por que há um botão RUN clicavel?)	H1: Visibilidade do sistema; H2	Média
2.8	Ao inserir objetos que obedecem a scripts	É necessário fechar o projeto, criar o objeto separadamente, salvar como Map Entity, para depois poder abrir o projeto anterior e carregar este ME. Falta integração destas tarefas, e o fato de exigir uma compilação dificulta muito (ver 2.5)	H6; H7: Flexibilidade e eficiência de uso	Alta
2.9	Ao inserir um objeto sólido	“Add Textura” sempre gera uma superfície enorme no mapa. Alterá-la requer paciência	H7: Flexibilidade e eficiência; H6	Média
2.10	Ao criar objeto sólido e sprite	Não óbvio linkar sprite a objeto	H6: Reconhecimento ao invés de lembrança	Média

Tabela 1: Problemas de Usabilidade encontrados em 3D GameStudio v.A6

APÊNDICE E – Questionários para Validação de Personas relacionadas a Motor de Jogo

Observe o diagrama da figura abaixo:

Ele procura representar as etapas de produção e o ciclo de vida de um jogo eletrônico comercial desenvolvido com ajuda de um game engine.

Qual papel, aqui representado, você julga estar mais próximo de sua atividade principal com jogos?

1. Produtor – quem financia, patrocina ou viabiliza o desenvolvimento do jogo eletrônico, buscando um retorno no final do processo, financeiro ou de outra espécie.
2. Game Designer – quem possui a visão do jogo como um todo, projeta e coordena as equipes de desenvolvimento.
3. Programador de Jogos – quem utiliza os recursos do game engine, diretamente ou através de ferramentas, para programar ou acrescentar material artístico ao jogo.
4. Engenheiro de Software ou Projetista de Engine - quem cuida do projeto da programação de base de um jogo, buscando flexibilidade e reaproveitamento de código para uma família de jogos eletrônicos.
5. Programador de Engine – quem programa as bibliotecas de software que viabilizam a infraestrutura de um jogo eletrônico. (ex: ambientes virtuais, simuladores físicos, interatividade, inteligência artificial, som interativo, etc.).
6. Engenheiro de Usabilidade ou Programador de Ferramentas – quem programa ferramentas para o acesso mais prático aos recursos do game engine (ex: editores de fases, etc.).
7. Jogador – quem adquire, experimenta ou testa jogos eletrônicos, em parte ou na sua totalidade.
8. Gerente de TI - quem decide a aquisição de produtos de desenvolvimento de software
9. Outra atividade.

Nome: Empresa:

1. Qual sua visão sobre o produto “game engine” (GE)? O que é um GE? Qual o contexto de sua utilização? Para que serve?

2. Dentro do ciclo de desenvolvimento de jogos eletrônicos em sua instituição, qual a porcentagem do orçamento e homens-hora dedicados ao desenvolvimento de GE?

Orçamento:

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

Homens-Hora:

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

3. Nos projetos de jogos que tenha participado, foi mais comum desenvolver um GE específico, próprio, ou adquirir um produto de terceiros? Qual porcentagem deles utilizou GE de terceiros?

Porcentagem de Terceirização:

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

4. Pela sua experiência, quanto vale o investimento (seja próprio ou de terceiros) em um GE, em relação ao custo total do jogo?

Porcentagem de Investimento:

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

5. Quais recursos são desejados em um GE mas que, por terem um custo muito elevado ou serem tecnologicamente inviáveis, não são utilizados em seus

jogos?

6. Quem são os usuários diretos do GE em sua instituição? Que características possui um destes usuários típicos?

7. Quais são as metas de negócio de sua instituição?

Game Engines: Questionário - Expert		5/2/2007 14:43
<div style="display: flex; justify-content: space-between; margin-bottom: 20px;"><div style="width: 45%;">Nome: <input style="width: 90%;" type="text" value="Seu Nome"/></div><div style="width: 45%;">Empresa: <input style="width: 90%;" type="text" value="Sua empresa/instituição"/></div></div> <div><p>1. Você concorda com o modelo apresentado na figura da página anterior? Está compatível com a realidade das instituições que teve contato? Tem sugestões para transformar este modelo em algo mais próximo do real?</p><div style="border: 1px solid black; height: 80px; margin-top: 5px;"></div></div> <div><p>2. Você usa Game Engines (GEs) diretamente? Ou coordena equipes que os utilizam? ☐ Usa Diretamente ☐ Coordena Equipes</p></div> <div><p>3. Que produto(s) você utiliza (diretamente, ou em sua equipe) para a função de GE?</p><div style="border: 1px solid black; height: 80px; margin-top: 5px;"></div></div> <div><p>4. Quais são os principais problemas encontrados na maioria dos GEs (comerciais ou gratuitos) encontrados atualmente? Como seria possível melhorar estes aspectos?</p><div style="border: 1px solid black; height: 80px; margin-top: 5px;"></div></div> <div><p>5. Quais são as “boas práticas” no processo de utilização de um GE?</p><div style="border: 1px solid black; height: 80px; margin-top: 5px;"></div></div> <div><p>6. Quais são as vantagens e os grandes desafios operacionais da indústria de jogos em geral? E, mais especificamente, no Brasil e na sua região? Existem incentivos das instituições e do governo?</p><div style="border: 1px solid black; height: 80px; margin-top: 5px;"></div></div>		

Game Engines: Questionário - Expert 5/2/2007 14:43

7. Quais canais de informação, referências, FAQs, você considera como essenciais para entender o mundo dos games e o papel do GE? Você mantém ou participa de algum canal de informação para discussão sobre o tema, pública ou particularmente?

Ao pressionar ENVIAR, as respostas serão encaminhadas através de seu programa de email. Caso este não seja seu computador pessoal, é melhor copiar as questões e enviá-las diretamente para eduardo.jacober@gmail.com

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

Game Engines: Questionário - Other 5/2/2007 15:21

Nome: Empresa:

1. Você respondeu OUTRA ATIVIDADE, indicando que não localizou sua atividade no modelo da figura. Como você define sua participação no processo de desenvolvimento de um jogo?
2. Que jogo você está jogando atualmente?
3. Qual o melhor jogo de todos os tempos? (Vale jogos eletrônicos, console, de tabuleiro, cartas, etc.)
4. Quais características fazem este jogo ser tão especial?

5. Você sabe se foi utilizado um Game Engine para construção deste jogo?
☐ Sim, foi. ☐ Não, não foi. ☐ Não sei.
6. Para você, o Game Engine de um jogo faz alguma diferença? Influi na decisão de qual jogo comprar? ☐ Sim ☐ Não

Ao pressionar ENVIAR, as respostas serão encaminhadas através de seu programa de email. Caso este não seja seu computador pessoal, é melhor copiar as questões e enviá-las diretamente para eduardo.jacober@gmail.com

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

Nome: Empresa:

1. Que tipos de jogo (gêneros) já foram desenvolvidos em sua instituição? Em quais teve participação direta?

Jogos da Instituição:

☐ First Person Shooter
 ☐ Estratégia
 ☐ Plataforma
 ☐ Corrida
 ☐

Adventure

Participação Direta:

☐ First Person Shooter
 ☐ Estratégia
 ☐ Plataforma
 ☐ Corrida
 ☐

Adventure

2. Quais Game Engines (GEs) você já teve oportunidade de utilizar pessoalmente? Deseja mantê-los, ou pretende trocar quando aparecerem novas versões ou alternativas?

Game Engines:

☐ Unreal Engine
 ☐ Torque
 ☐ Ogre
 ☐ Renderware
 ☐ Game Studio
 ☐

Pretende continuar:

☐ Manter
 ☐ Trocar

3. Quais são as tarefas que você costuma executar através de um GE? Quais outras tarefas um GE deveria auxiliar?

4. O que é necessário saber, que informações é necessário dominar para ser um bom desenvolvedor de jogos?

5. Existe um FAQ, Manual, site da web, que seja leitura essencial para quem desenvolve com os GEs que você utiliza?

6. Em sua opinião, qual o melhor GE disponível atualmente? Que características este GE possui que o tornam especial? O que podia melhorar?

Melhor Game Engine:

☐ Unreal Engine
 ☐ Torque
 ☐ Ogre
 ☐ Renderware
 ☐ Game Studio
 ☐

Características Especiais:

Melhorias:

Game Engines: Questionário - Programador 5/2/2007 14:54

7. Em que características os GEs disponíveis atualmente (comerciais e gratuitos) deixam a desejar? Como estes aspectos poderiam melhorar?

Resposta 7

8. Em que proporção de suas tarefas você utiliza os recursos do Game Engine (GE)?

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

9. Em que proporção você desenvolve pessoalmente alguma funcionalidade nova para seus jogos?

☐ 0% ☐ 25% ☐ 50% ☐ 75% ☐ 100%

Ao pressionar ENVIAR, as respostas serão encaminhadas através de seu programa de email. Caso este não seja seu computador pessoal, é melhor copiar as questões e enviá-las diretamente para eduardo.jacober@gmail.com

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

Game Engines: Questionário - Desenvolvedor de Engines 5/2/2007 15:14

Nome: Empresa:

1. Para que plataforma (PC/Linux/Mac, console, mobile, etc) você desenvolve seus jogos? ☐ PC ☐ Linux ☐ Mac ☐ Console ☐ Mobile

2. Qual a infraestrutura de software que você utiliza (Linguagens de Programação, APIs, Game engines, Scripts, Modelagem, etc) ?

3. Quais métodos de desenvolvimento, e quais artefatos de projeto você recomenda?

4. Qual método é utilizado atualmente para conseguir informações dos usuários de seu Game Engine?

5. Descreva a importância da usabilidade dentro de seu projeto.

Ao pressionar ENVIAR, as respostas serão encaminhadas através de seu programa de email. Caso este não seja seu computador pessoal, é melhor copiar as questões e enviá-las diretamente para eduardo.jacober@gmail.com

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

Game Engines: Questionário - Jogador 5/2/2007 15:17

Nome: Seu Nome Empresa: Sua empresa/Instituição

1. Que jogo você está jogando atualmente? Resposta 1
2. Qual o melhor jogo de todos os tempos? (Vale jogos eletrônicos, console, de tabuleiro, cartas, etc.) Resposta 2
3. Quais características fazem este jogo ser tão especial?
4. Você sabe se foi utilizado um Game Engine para construção deste jogo?
☐ Sim, foi. ☐ Não, não foi. ☐ Não sei.
5. Para você, o Game Engine de um jogo faz alguma diferença? Influí na decisão de qual jogo comprar? ☐ Sim ☐ Não

Ao pressionar ENVIAR, as respostas serão encaminhadas através de seu programa de email. Caso este não seja seu computador pessoal, é melhor copiar as questões e enviá-las diretamente para eduardo.jacober@gmail.com

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

Game Engines: Questionário - Comprador 5/2/2007 15:19

Nome: Seu Nome Empresa: Sua empresa/Instituição

1. Qual seu envolvimento no processo de escolha/compra de um Game Engine (GE)? Resposta 1
2. Este processo envolve quantas pessoas em sua instituição? Resposta 2
3. Você já teve oportunidade de utilizar algum GE pessoalmente? Com que frequência? De que forma? Resposta 3
4. Quais são os objetivos que decidem a escolha de um GE para sua instituição? Resposta 4
5. Onde os produtos disponíveis atualmente (comerciais ou gratuitos) deixam a desejar? Resposta 5
6. Qual sua participação na instalação, manutenção e gerenciamento do GE em sua instituição? Resposta 6
7. Existe algum informativo, catálogo, glossário etc. que ajuda um iniciante a entender melhor sobre as funcionalidades dos GEs e poder diferenciá-los? Resposta 7

Obrigado pelas informações! Suas respostas não serão divulgadas isoladamente.

ANEXO A – Heurísticas de Usabilidade

Dez Heurísticas de Usabilidade de Jakob Nielsen

1. Visibilidade do estado do sistema

O sistema deve sempre manter os usuários informados sobre o que está acontecendo, através de retorno apropriado dentro de um tempo razoável.

2. Concordância entre o sistema e o mundo real

O sistema deve falar a língua do usuário, com palavras, frases e conceitos familiares ao usuário, ao invés de termos orientados ao sistema. Seguir convenções do mundo real, fazendo a informação aparecer em uma ordem natural e lógica.

3. Controle e liberdade do usuário

Usuários normalmente escolhem funções do sistema por engano e precisarão de uma “saída de emergência” bem demarcada, para sair do estado indesejado sem ter que passar por um diálogo extenso. Suporte a *undo* e *redo*.

4. Consistência e padrões

Usuários não devem precisar se preocupar se palavras, situações ou ações diferentes significam a mesma coisa. Seguir as convenções da plataforma.

5. Prevenção de erros

Melhor que boas mensagens de erro é um projeto cuidadoso que previne um problema de ocorrer antes de mais nada.

6. Reconhecimento ao invés de lembrança

Fazer objetos, ações e opções visíveis. O usuário não deve ter de lembrar informação de uma parte do diálogo para outra. Instruções de uso do sistema devem estar visíveis ou facilmente resgatadas sempre que apropriado.

7. Flexibilidade e eficiência de uso

Aceleradores – invisíveis para o usuário novato – podem acelerar a interação para o usuário avançado de forma que o sistema possa atender usuários com ou sem experiência. Permitir aos usuários acelerar ações frequentes.

8. Estética e design minimalista

Diálogos não devem conter informação que é irrelevante ou raramente necessária. Toda unidade de informação extra em um diálogo compete com as unidades de informação relevantes e diminui sua visibilidade relativa.

9. Ajuda para os usuários compreenderem, diagnosticarem e se recuperarem de erros

Mensagens de erro devem ser expressas em linguagem clara (sem códigos), precisamente indicando o problema e construtivamente sugerindo uma solução.

10. Ajuda e documentação

Apesar de ser melhor se o sistema puder ser utilizado sem documentação, pode ser necessário prover ajuda e documentação. Qualquer informação deste tipo deve permitir busca facilmente, ser focada na tarefa do usuário, listar passos concretos a serem seguidos, e não ser muito extensa.

Página original: http://www.useit.com/papers/heuristic/heuristic_list.html

GLOSSÁRIO

DEFINIÇÕES

Ambiente Virtual: Espaço sintético (criado através de parâmetros e fórmulas matemáticas) simulado por computador.

Avatar: Personagem ou objeto do ambiente virtual que representa o jogador.

Brinquedo: Objeto sobre o qual aplicou-se a transgressão de diversão.

Componentização: Definição de componentes de *software*.

Condição de fim: Teste realizado periodicamente durante um jogo, identificando se o jogo deve terminar.

Diversão ou **estado de diversão:** uma transgressão sobre a realidade, libertando um determinado objeto, pessoa ou lugar das suas regras convencionais por determinado período tempo.

Estratégia: Escolha de ações durante um jogo com a intenção de aumentar as chances de vitória de um jogador.

Interação Humano-Computador: Comunicação bidirecional (ou seja, as entidades alternam-se nos papéis de emissor e receptor) entre duas entidades distintas: um ser humano e um sistema digital.

Jogo abstrato: Produto-jogo que possui poucos outros elementos além dos elementos de jogo.

Jogo eletrônico: Produto-jogo baseado em um sistema digital.

Jogo ou **estado de jogo:** uma transgressão sobre a diversão, acrescentando-se a essa, elementos não-convencionais de estrutura e regras.

Mapa: Ver *Game Level*.

Mod-Game: Jogo eletrônico criado a partir de produto-jogo comercial completo, utilizando-se de ferramentas fornecidas pelo estúdio do jogo.

Persona: Modelo de usuários utilizado como artefato de projeto de software.

Produto-jogo: Produto comercial ou atividade que possui elementos de jogo.

Projeto de Interação: Método para identificar e modelar os usuários de um sistema, produzindo uma especificação para um sistema digital centrado nesses usuários.

Simulação: Aplicação de um sistema digital que responde interativamente conforme um modelo matemático programado. É comum, porém não necessário que o tempo de resposta desse sistema seja tal que permita sensação de controle ao usuário.

Sistema de Jogo: Conjunto de objetos que permitem a criação de diversos jogos. Exemplos comuns são o tabuleiro 8x8, o baralho de 52 cartas, dados de 6 faces e o dominó duplo-seis.

Usabilidade: Medida da facilidade com que a interface de um produto (geralmente orientada a *software*) permite atender às necessidades de seus usuários.

ANGLICISMOS

Arcade (plataforma): Fliperama. Plataforma para jogos eletrônicos baseada em hardware especializado e para uso em estabelecimentos comerciais, normalmente com funcionamento à base de fichas.

Blog: Espécie de diário pessoal (*journal*) divulgado pela Internet.

Brainstorm: Atividade de projeto em grupo voltada ao processo criativo de listar idéias.

Framework: Estrutura que serve de alicerce para a atividade de programação.

Game Designer: Projetista de Jogo.

Game Engine: Motor de Jogo, Máquina de Jogo.

Game Level: Mapa, Cenário de uma etapa do produto-jogo. Possui dinâmica em comum; mas tempo de jogo, regras específicas, objetivos e condição de fim próprios.

Guidelines: Diretrizes, Boas-Práticas.

Hardware: Configuração específica (arquitetura) de um produto eletrônico.

Joystick: Dispositivo de interface física.

Ludology: Estudo dos aspectos de jogo.

Mouse: Dispositivo de interface física.

Participatory Design: Projeto participativo, onde as opiniões dos usuários são levadas em conta durante o processo de desenvolvimento.

Placeholder: Recurso temporário utilizado durante o desenvolvimento de jogos, à espera de que o recurso final esteja pronto;

Pixel: Contração de *Picture Element*, unidade de apresentação visual de uma imagem digital.

Sampling: Processo de amostragem de sinais.

Script, script programming language (linguagem computacional): Linguagem de programação, normalmente interpretada e na forma de trechos de código de suporte a uma estrutura computacional mais complexa. No âmbito do desenvolvimento de jogos, costuma ser uma forma de passar informações ao motor de jogo e de configurar parâmetros da “inteligência artificial”.

Serious Games: Prática de utilização da infra-estrutura dos jogos eletrônicos para outros fins além do entretenimento.

Site, subsite (Internet): Sítio. Coleção de páginas de hipertexto com diagramação e estrutura de navegação em comum.

Software: Conjunto de instruções passadas pelo programador a um *Hardware*.

Subject Matter Expert: Peritos no campo de aplicação.

Stakeholder: Quem tem interesse (financeiro ou político) em um desenvolvimento ou implantação de um produto.

Storytelling: Narração.

Tooltip: Widget que apresenta texto informativo.

User-friendly (interfaces): Interfaces Amigáveis.

Viewpoint (Computação Gráfica): Ponto de vista. A projeção de um ambiente tridimensional em uma janela plana do sistema.

Widget: Elemento de interface, com aparência e comportamento definidos.

REFERÊNCIAS

-
- ¹ enJine, versão 3.0 release11/Oct/2006: Engine for 3D Games in Java. São Paulo: Interlab, 2006. Biblioteca de programação. Disponível em <<http://enJine.incubadora.fapesp.br/portal>>.
- ² TORI, R.; NAKAMURA, R.; BERNARDES, J. Using Game Development and Java3D to Teach Computer Graphics In: **CMP Game Career Guide - 2006**. Disponível em: <http://www.gamecareerguide.com/features/286/features/286/features/286/using_game_development_and_java_3d_.php>. Acesso em: 25 de outubro de 2006.
- ³ HUIZINGA, J. **Homo Ludens**: o Jogo como Elemento da Cultura – 5ª. Ed. 1ª. reimpr. São Paulo: Perspectiva, 2004. 243p. (Coleção Estudos)
- ⁴ SALEN, K.; ZIMMERMAN, E. **Rules of Play**: Game Design Fundamentals, Cambridge: The MIT Press, 2004. 670 p.
- ⁵ WALTHER, B. K. Playing and Gaming: Reflections and clarifications In: **Game Studies**: the international journal of computer game research – v.3, n.1, May 2003. Disponível em: <<http://www.gamestudies.org>>. Acesso em: 28 Jun. 2004.
- ⁶ LUCAS, G. **Star Wars**: Episode IV – A New Hope. Abril Video [br], 1977. [1 disco, 121min., DVD]
- ⁷ CHION, M. **Audio-Vision: Sound on Screen**. New York: Columbia University Press, 1994.
- ⁸ JUUL, J. Games Telling Stories? In: RAESSENS, J.; GOLDSTEIN, J. **handbook of computer game studies**. MIT Press, 2005. p. 219-226

⁹ LÉVI-STRAUSS, C. **O Pensamento Selvagem**. São Paulo: Companhia Editora Nacional, 1966.

¹⁰ COLERIDGE, S. T. Biographia Literaria 1817 In: **Samuel Taylor Coleridge**, ed. by H.J. Jackson, Oxford, 1985. Disponível em: <<http://www.english.upenn.edu/~mgamer/Etexts/biographia.html>>. Acesso em: 28 Dez. 2006.

¹¹ BLOW, J. Game Development: Harder Than You Think. In: **ACM queue: tomorrow's computing today** – v.I, n.10, p.32-37, Feb. 2004.

¹² TORI, R.; BERNARDES, J.; BIANCHINI, R.; CUZZIOL, M.; JACOBBER, E.; NAKAMURA, R. Jogos Eletrônicos e Realidade Virtual In: TORI, R.; KIRNER, C. (Eds.) **Realidade Virtual: Conceitos e Tendências – livro do pré-simpósio - VII Symposium on Virtual Reality**, São Paulo: Mania de Livro, 2004. Cap.10, p.159-176.

¹³ NAKAMURA, R.; TORI, R.; BERNARDES, J.; BIANCHINI, R.; JACOBBER, E. A Practical Study on the Usage of a Commercial Game Engine for the Development of Educational Games In: **Anais do congresso WJogos 2003**, 2003a.

¹⁴ SHNEIDERMAN, B.; PLAISANT, C. **Designing the User Interface: Strategies for Effective Human-Computer Interaction** – 4th Ed. Addison-Wesley, 2005.

¹⁵ SHANNON, C.; WEAVER, W. **The Mathematical Theory of Communication**. University of Illinois Press, 1998.

¹⁶ KUBRICK, S.; CLARKE, A. **2001: Space Odissey**. Warner Home Video [br], 1968. [1 disco, 148 min., DVD)

¹⁷ LARSON, G. **Knight Rider**, Universal TV, 1982-1986. [Série de TV, 89 episódios de 60 min.]

-
- ¹⁸ ADAMS, D. **A vida, o universo e tudo mais**. Rio de Janeiro: Sextante, 2005. 221p.
- ¹⁹ REEVES, B.; NASS, C. **The Media Equation**: How People Treat Computers, Television and New Media Like Real People and Places, CSLI, 2003. 320p.
- ²⁰ COOPER, A.; REIMANN, R. **About Face 2.0**: The essentials of interaction design, Indianapolis: Wiley Publishing, 2003. 540p.
- ²¹ SHNEIDERMAN, B. **Leonardo's Laptop**: Human needs and the new technologies, Cambridge: The MIT Press, 2003. 269p.
- ²² LEWIS, C.; RIEMAN, J. **Task-Centered User Interface Design: A Practical Introduction**, 1994. Disponível em: <http://bmrc.berkeley.edu/courseware/cs160/fall99/Book/contents.v-1.html>. Acesso em: 28 de Junho de 2005.
- ²³ PREECE, J.; ROGERS, Y.; SHARP, H. **Interaction Design**: beyond human-computer interaction, IE-Wiley, 2002. 544p.
- ²⁴ MAYHEW, D. **The Usability Engineering Lifecycle**: a practitioner's handbook for user interface design, San Francisco: M. Kauffmann, 1999. 542p.
- ²⁵ NIELSEN, J. **Guerrilla HCI**: Using Discount Usability Engineering to Penetrate the Intimidation Barrier, 1994. Disponível em: http://www.useit.com/papers/guerrilla_hci.html. Acesso em: 25 de maio de 2006.
- ²⁶ SNYDER, C. **Paper Prototyping**: The Fast and Easy Way to Design and Refine User Interfaces, San Francisco: M. Kaufmann, 2003. 378p.
- ²⁷ BETHKE, E. **Game Development and Production**. Plano, Texas: Wordware Publishing, 2003. 412p.

²⁸ RIDING, R.; RAYNER, S. **Cognitive Styles and Learning Strategies:** Understanding style differences in learning and behavior. London: D. Fulton, 1998.

²⁹ PHILIPS, M.; HUNTLEY, C. **Dramatica:** A New Theory of Story - 3rd Ed., Screenplay Systems, 1996.

³⁰ NAKAMURA, R.; TORI, R.; JACOB, E.; BIANCHINI, R.; BERNARDES, J. Development of a Game Engine using Java In: **Anais do congresso WJogos 2003**, 2003b.

³¹ FOWLER, A. **A Swing Architecture Overview:** The Inside Story on JFC Component Design. Sun Developer Network, 1994. Disponível em: <<http://java.sun.com/products/jfc/tsc/articles/architecture/>>. Acesso em: 28 de Outubro de 2006.

³² JACOB, E.; TORI, R. Game Wrench: Design of a user-centered Interactive Game Engine In: **Anais do congresso SBGames 2006**.

³³ JACOB, E.; TORI, R. Game Wrench: um motor interativo centrado no usuário para criação de jogos como objeto de aprendizado In: **Anais do congresso ICECE 2007**.