

UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências - Bauru

Bacharelado em Ciência da Computação

Adham Gabriel Roque Benelli

Estudo e relato do processo de criação de um jogo
digital

UNESP

2014

Adham Gabriel Roque Benelli

Estudo e relato do processo de criação de um jogo
digital

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

Orientador: Prof. Dr. Eduardo Morgado

UNESP

2014

Benelli, Adham Gabriel Roque.

Estudo e relato do processo de criação de um jogo digital / Adham Gabriel Roque Benelli, 2014
43 f. : il.

Orientador: Eduardo Morgado

Monografia (Graduação)-Universidade Estadual Paulista. Faculdade de Ciências, Bauru, 2014

1. Jogos digitais 2. Processo de criação 3. Leitura - Meios auxiliares. I. Universidade Estadual Paulista. Faculdade de Ciências. II. Estudo e relato do processo de criação de um jogo digital.

Adham Gabriel Roque Benelli

Estudo e relato do processo de criação de um jogo digital

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

BANCA EXAMINADORA

Eduardo Morgado
Professor Doutor
DCo – FC – UNESP – Bauru
Orientador

Simone das Graças Domingues Prado
Professora Doutora
DCo – FC – UNESP – Bauru

João Pedro Albino
Professor Doutor
DCo – FC – UNESP – Bauru

Bauru, 17 de Junho de 2014.

Resumo

Com o avanço da indústria de jogos no mercado, mostrou-se evidente a necessidade de profissionais capacitados atuando na área. Este documento tenta estudar e relatar o processo de criação de um jogo visando profissionais e aspirantes desenvolvedores de jogos.

Para isso, ao longo dessa monografia será descrito como foi o processo de criação de um jogo digital desenvolvido por estudantes da Universidade Estadual Paulista – Unesp Bauru. O projeto criado tenta explorar mecânicas atuais juntamente com um enredo emocional.

Palavras-chave: Jogos digitais, Game Design, Processo de criação.

Abstract

With the advancement of the gaming industry in the market, became evident the need for trained professionals working in this area. This paper tries to study and report the process of creating a game for professional and aspirants game developers.

For this reason, throughout this monograph will describe how was the process of creating a digital game developed by students at the Universidade Estadual Paulista - UNESP Bauru. The project tries to explore current mechanical created along with an emotional storyline.

Keywords: Digital Games, Game Design, Creation Process.

LISTA DE FIGURAS

Figura 1 - Fluxograma.....	16
Figura 2 - Esboço de uma possível cena de Sillage em forma de storyboard.	16
Figura 3 - Atração ao centro.	18
Figura 4 - Atração aos cantos	18
Figura 5 - Esboço da primeira idéia de jogo proposta	21
Figura 6 - Esboço inicial de Sillage	21
Figura 7 - Tela principal do Asana	24
Figura 8 - Ambiente de Tiled.....	26
Figura 9 - map.xml com as marcações do arquivo tmx.	27
Figura 10 - Exemplo de visão em primeira pessoa (Counter Strike: Global Offensive).....	28
Figura 11 - Exemplo de visão em terceira pessoa (Mario Galaxy 2).	29
Figura 12 - Cubo visto de forma isométrica.....	30
Figura 13 - Exemplo de tiles.	31
Figura 14 - Mapa de tiles	32
Figura 15 - Exemplo de uma matriz de números representando o mapa.	32
Figura 16 - Tiles de borda.....	33
Figura 17 - map.xml (O mapa não processado está encapsulado em data).....	34
Figura 18 - Modificação na região de areia	35
Figura 19 - Player principal de Sillage.....	36

Sumário

1 Introdução	9
1.1 Mercado de jogos	9
1.2 Problema	10
1.3 Justificativa	10
1.4 Objetivos	11
1.4.1 Objetivo geral	11
1.4.2 Objetivos específicos	11
2 Referencial teórico	12
2.1 Processo de criação de um jogo digital	12
2.2 Profissionais envolvidos no processo	12
2.3 Tipos de jogos	13
2.4 História nos jogos	14
2.5 Roteiro	15
2.6 Character Design	16
2.7 Level Design	17
2.8 Sound Design	18
3 Projeto e desenvolvimento	20
3.1 Pré-Projeto	20
3.1.1 Idealização do jogo (Brainstorm)	20
3.1.2 Trabalhando a ideia	22
3.1.3 Trabalhando a história	22
3.1.4 Finalizações	23
3.1.5 Documento de Game Design	23
3.2 Ferramentas	23
3.2.1 Asana	24
3.2.2 Unity	25
3.2.3 Microsoft Visual Studio	25
3.2.4 Tiled	26
3.3 Desenvolvimento	27
3.3.1 Perspectiva de jogo	27
3.3.2 Visão isométrica	30
3.3.3 Tiles e suas aplicações	31
3.3.4 Mapa	33
3.3.5 Jogador e o que ele representa	35

3.3.6 Inteligência Artificial	36
3.4 Otimizações	37
3.4.1 Object Pool	37
3.4.2 Corotinas	38
3.4.3 Aplicações das otimizações	38
4 Conclusões	40
4.1 Problemas e soluções	40
4.2 Considerações finais	40
Referências bibliográficas	41
Bibliografia complementar	42

Capítulo 1

1 Introdução

1.1 Mercado de jogos

“No other sector has experienced the same explosive growth as the computer and video game industry. Our creative publishers and talented workforce continue to accelerate advancement and pioneer new products that push boundaries and unlock entertainment experiences. These innovations in turn drive enhanced player connectivity, fuel demand for products, and encourage the progression of an expanding and diversified consumer base.”
(GALLAGHER, Michael D., informação verbal na ESA 2013 Annual Meeting)

A produção de jogos digitais no Brasil e no mundo vem crescendo gradativamente nesses últimos anos e as previsões para o futuro são ainda melhores. De acordo com uma notícia da SEBRAE, o Brasil é o quarto maior mercado de jogos do mundo, movimentando R\$5.3 bilhões em 2012. Esse aumento da produção vem como resposta ao crescimento do número de jogadores ao redor do globo - 35 milhões só no Brasil - tanto para consoles quanto para computadores ou aparelhos móveis.

A indústria de jogos nunca vivenciou um crescimento de tamanha magnitude como atualmente, mas não são apenas as grandes empresas do ramo que estão aproveitando essa onda imensa de novos jogadores. Pequenas empresas vêm criando mais e mais jogos sem a ajuda de uma grande publicadora (ou jogos *indie*), abrindo caminho para praticamente qualquer entusiasta a se aventurar nessa empreitada de desenvolver um jogo digital.

Essa onda de jogos *indie* vem sido aceita pelo público jogador, mostrando a eles a possibilidade de novas fontes de entretenimento com custo mais baixos do que os jogos produzidos pelas grandes empresas (vide mercado de jogos *mobile*, onde em média os preços dos jogos são de R\$1,00 a R\$5,00). Essa diminuição de preço também pode ser atribuída a popularização da distribuição digital. Hoje em dia jogos não necessitam de uma caixa, com manual, cartucho ou CD-ROM, e também não precisam ser distribuídos fisicamente, diminuindo drasticamente o custo de se produzir um jogo.

Surgimento de novas tecnologias de desenvolvimento de jogos (*Unity*, *Construct 2D* e outros), combinado a plataformas de distribuição de conteúdo pela internet como *Steam*, *Google Play* ou *Splitplay* (primeiro *marketplace* de jogos *indie* da América Latina) auxiliaram a popularização e democratização do desenvolvimento de jogos.

Mas é claro, o caminho é longo, complexo e muito trabalhoso, existem milhares de jogos criados, mas apenas uma faixa muito pequena desses jogos cai na graça do público e conseguem ser reconhecidos mundialmente. Com essa competição acirradíssima desse universo dos *games*, um bom planejamento e desenvolvimento de todas as etapas de um processo de criação de jogo são essenciais para a sobrevivência nesse mercado.

O intuito desse projeto de termino de conclusão de curso é analisar, relatar e aplicar o processo de criação de um jogo digital, cobrindo etapas de concepção, criação e desenvolvimento do mesmo.

1.2 Problema

O processo de criação de um jogo não obedece a uma regra ou uma série de passos que obrigatoriamente funciona para todos os casos. A criação de um jogo de *puzzle* é diferente da criação de um jogo de aventura. De acordo com Jessel Schell, a maior preocupação da pessoa que cria um jogo deve ser na experiência que o jogador vai ter ao jogá-lo. Levando isso em conta, cada jogo exige um processo de criação diferente, já que suas experiências, não são similares.

Mas apesar desse processo ser variado para cada jogo, ele pode ser desmembrado para um melhor controle sobre a criação além de melhorar a distribuição de tarefas. Sendo assim, é possível dividir o desenvolvimento de um jogo nas seguintes etapas:

- Criação do documento de *Game Design*;
- Produção da arte e som;
- Produção da *engine*;
- Acoplamento da arte com a *engine*;

Essas etapas englobam dentro dela muito trabalho e planejamento, e está aí a diferença na produção de um jogo para outro. Essa experiência única que cada *game designer* busca é onde está localizada a maior dificuldade em criar um jogo de qualidade, e também, onde existem poucos documentos discutindo sobre. O porquê de cada decisão, qual é o propósito de se fazer o som dessa forma? Que tipo de sentimento está buscando ao usar essas cores na arte? Por que usar essa mecânica? Estas são perguntas que geralmente surgem ao longo do processo, mas não existem muitas discussões sobre esse tipo de tomada de decisão.

1.3 Justificativa

O documento final desse projeto irá conter todo o processo de criação de um jogo digital feito por alunos da Unesp Bauru, dois deles cursando Design Gráfico, e dois cursando Ciência da Computação, sendo um destes o autor deste documento.

Com esse processo todo detalhado e analisado, acredita-se que será de grande ajuda aos futuros *game designers* a conhecerem mais sobre as etapas envolvidas na criação de um jogo, os problemas frequentemente encontrados, e toda a trajetória de um grupo relativamente pequeno, as dificuldades, esforços realizados durante todo esse processo.

1.4 Objetivos

1.4.1 Objetivo geral

Desenvolver e analisar todo o processo de criação de um jogo digital, detalhando todas as decisões tomadas, tanto em parte de execução quanto de criação de arte, som e roteiro. Criar também um documento detalhado de toda a trajetória da criação, desde sua concepção até um *demo* do jogo.

1.4.2 Objetivos específicos

- Estudar e aplicar técnicas imersivas para melhoria da experiência do jogador
- Criação de um documento de *game design* do jogo, mostrando todo seu conteúdo.
- Aprender e utilizar as tecnologias utilizadas para a criação
- Escrita do documento de toda a trajetória do jogo
- Desenvolvimento de um *demo* do jogo

Capítulo 2

2 Referencial teórico

2.1 Processo de criação de um jogo digital

Antes de discutir sobre a criação de um jogo, se faz necessário parar um tempo para analisar o que é um jogo. Jogo é um termo do latim “jocus” que significa gracejo, brincadeira, divertimento. Do alemão *Spiel*, jogo ou *game*, é qualquer atividade praticada apenas pelo prazer e sem propósito consciente. Por essa definição, qualquer atividade que traga prazer pode ser considerada um jogo, tais como: dançar, tocar instrumentos musicais, atuar, brincar.

Tais definições mostram quão vaga e sutil a simples resposta da pergunta “o que é um jogo?” pode ser. Eric Zimmerman e Katie Salen dizem em seu livro: “A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome” (ZIMMERMAN, Eric; SALEN, Katie, A regra do jogo). Essa é uma das definições mais aceitas, e é a que será utilizada para todos os fins nesse documento.

2.2 Profissionais envolvidos no processo

Existem várias funções no desenvolvimento de um *game*, e pode haver vários profissionais específicos, ou dependendo do caso, poucos profissionais exercendo duas funções ou mais. Abaixo estão listados as mais recorrentes:

- **Game Designer:** que pode ou não ter outras funções na produção. Preocupa-se em desenvolver todas as experiências no jogo.
- **Level Designer:** Vai preocupar-se com o ambiente do jogo, todo o percurso que o jogador irá percorrer, toda interação do player com o mundo artificial.
- **Content Designer:** Preocupam-se com os desafios, missões, puzzles dentro do jogo.
- **Concept Artist:** vai desenvolver toda a estética, os conceitos visuais, que irão guiar os outros profissionais durante as outras etapas do desenvolvimento.
- **Desenvolvedor: ou Programador,** irá desenvolver todo o código do jogo, programando a motor do jogo.
- **3D Artist:** Caso haja a necessidade de modelos 3D no jogo. Ele irá modelar, texturizar, animar os objetos, personagens, geralmente seguindo o conceito elaborado pelo *Concept Artist*.
- **Sound Designer:** Pode ou não ser o compositor. Irá criar todos os sons, efeitos sonoros, feedback sonoros, dependendo, até a trilha sonora.

- **User Interface Designer.** Vai desenvolver toda a interface e menu do jogo, sendo ela evidente ou não.

2.3 Tipos de jogos

A definição do que é um jogo mostra-se vaga e intuitiva ao mesmo tempo, e essa dificuldade também ocorre na classificação de um jogo. Não é uma tarefa fácil escolher uma única definição. Normalmente os jogos são uma mistura de elementos, abrangendo mecânicas e características de duas ou mais definições.

Abaixo há uma lista de definições de alguns tipos de jogos mais conhecidos:

Ação: Como o próprio nome diz, são jogos em que o jogador possui uma maior interatividade com o personagem, seja em uma luta ou conflito estratégico. Normalmente possuem conflitos com força física violenta, como combate com tiros ou de espadas. Tudo acontece em tempo real, considerando assim muito a capacidade de raciocínio rápido e os reflexos do jogador.

Adventure (aventura): Proveniente do gênero "Adventure", tradicional nos computadores da década de 80, é um modo de jogo sem desafios de reflexos ou ação, salvo alguns minijogos. Normalmente, exige que o jogador resolva vários enigmas ou quebra-cabeças, interagindo com os personagens ou o ambiente, na maioria das vezes de uma forma onde não há confronto.

Plataforma: São os jogos em que, como o próprio nome diz, é necessário pular de uma plataforma para outra. Grandes exemplos desse estilo são "Mario" e "Sonic", mas pode-se citar também "Prince of Persia" e o atual "Little Big Planet". Em todos estes jogos se enfrenta alguns inimigos, seja pulando sobre eles ou dando golpes.

Musical: São jogos em que o jogador, seguindo o ritmo de uma música, deve fazer pontuações acionando algum comando. Normalmente, todo o cenário e o visual desses jogos são influenciados pela música. Exemplos de jogos do gênero são "Guitar Hero" e "Pump It Up!".

Quebra cabeça (Puzzle): São jogos em que o objetivo é resolver um problema através do raciocínio lógico, forçando o jogador a pensar nas maneiras possíveis de se cumprir o objetivo. Diferentemente do jogo de aventura, os jogos de quebra-cabeça normalmente possuem pouca ou nenhuma história, e costumam ser mais casuais. Um exemplo típico deste tipo de jogo é Tetris, no qual se deve, através de peças compostas por quatro quadrados que caem de cima da tela, montar linhas inteiras horizontais.

Corrida: Têm o objetivo de fazer com que o jogador tente completar um percurso, seja linear ou cíclico, no menor tempo possível, ou à frente de seus adversários. Os meios de transporte que podem ser usados podem variar. Entre os jogos de corrida, é possível separar em vários subgêneros:

- **Simulação:** este tipo tenta reproduzir uma corrida real da forma mais real possível, tendo um forte foco na física do jogo como a série “Fórmula 1”.
- **Arcade:** possuem foco na velocidade e na corrida em si, porém sem dar tanta importância à física como em uma simulação. É o caso dos jogos da série “Need for Speed”.
- **Combate:** são corridas que envolvem, de alguma forma, o combate dos corredores. Desta forma, um jogador pode interferir na corrida de outro “atacando-o”, desta forma, prejudicando o outro competidor. Exemplos que podem ser citados são “Burnout” e “Mario Kart”.

Esportes: Os jogos de esportes simulam esportes reais, como futebol e basquete. Uma das formas mais comuns seria de controlar um atleta, porém existem outros estilos que focam mais na estratégia de lidar com o esporte, ou seja, de gerenciar. Neste último caso, o jogador é um gerente de uma equipe e terá que, ao mesmo tempo em que monta um bom time, administrar o dinheiro e pensar em estratégias.

RPG: Proveniente do termo em inglês “Role Playing Game”, e originado dos RPG’s de mesa, no qual os jogadores interpretam personagens e a partir disto criam histórias. Nos RPG’s de jogos digitais, são utilizados os aspectos mais importantes de RPG’s de mesa, conforme a possibilidade de adaptação, como atributos dos jogadores e pontos de vida, ou até mesmo um sistema de níveis, no qual se consegue melhorar suas habilidades conforme derrotam mais inimigos. Também possuem histórias que normalmente são muito bem trabalhadas. Existem três tipos de combates mais comuns dentro do estilo de RPG, que são:

- **Action:** aqueles que ocorrem em tempo real com ações características de jogos do gênero de ação, anteriormente citados.
- **Turn Based:** as ações de cada personagem são intercaladas em turnos cíclicos.
- **Tactical:** possuem o conceito de turnos como o de cima, mas a movimentação e o combate são como em um tabuleiro de xadrez.

2.4 História nos jogos

Direto a sua definição, o que é uma história? De forma simplificada, uma história nada mais é do que uma explicação – explica como um personagem ou protagonista tentou superar um desafio, obtendo sucesso ou não. É claro que esse protagonista não é necessariamente um herói ou uma pessoa.

História e jogos estão de certa forma, entrelaçados. Os dois possuem protagonistas, conflitos e culminam num desfecho desse conflito. Então é possível dizer que *games* são histórias em uma mídia diferente? Absolutamente não.

Jogos possui uma ferramenta poderosíssima o que faz deles uma das mídias mais imersivas e exploradas nesse século: a interação. Numa história, lhe é explicado como um personagem desempenhou alguma ação para resolver algum

problema. Já no jogo, você é capaz de controlar tal personagem e desempenhar tal ação, sendo responsável pela solução do conflito.

Uma história bem formulada é capaz de imergir e criar experiências quase que reais ao leitor, então um jogo integrado a uma história bem trabalhada é capaz de potencializar essas características. Mas criar uma história para um jogo não é o mesmo que criar uma história para outra mídia. Jesse Schell (2008, p. 263) diz:

A masterful storyteller knows how to create this desire within a listener's mind, and then knows exactly how and when (and when not) to fulfill it. This skill translates well into interactive media, although it is made more difficult because the storyteller must predict, account for, respond to, and smoothly integrate the actions of the participant into the experience.

2.5 Roteiro

Uma definição simples de roteiro seria: a forma escrita de qualquer projeto audiovisual. Syd Field, um grande roteirista estadunidense, define roteiro como uma “história contada em imagens, diálogos e descrição, dentro do contexto da estrutura dramática”.

No contexto de *game design*, roteiro é a descrição de toda a história do jogo. No roteiro é apresentado o enredo do jogo junto com todos os diálogos e informações que o jogador poderá encontrar durante a história. Apesar de que possa parecer desnecessário o tempo e trabalho empregado para desenvolver um roteiro bem estruturado para um jogo, ele certamente valerá a pena no futuro.

Desenvolver um roteiro de boa qualidade é uma tarefa difícil, é necessário muito tempo de estudo para identificar aspectos positivos da história do jogo e tentar explorar ao máximo a imersão e identificação do jogador. Apesar disso, Joseph Campbell, um antropólogo nascido no começo dos anos de 1900, criou a Jornada do Herói (ou Monomito). Esse Monomito é uma estrutura de roteiro genérica que capta a essência de várias histórias épicas. Até os dias de hoje é utilizada a jornada do herói na elaboração de não somente jogos (até porque essa não foi preocupação de Campbell quando criou essa estrutura), mas também de livros, filmes e seriados.

Outra estrutura muito utilizada é a narrativa em três atos. Essa narrativa é a forma mais natural e intuitiva de se contar uma história, dividindo-a em três partes, introdução, desenvolvimento e conclusão. É fácil imaginar histórias que se encaixem nessas divisões, já que, normalmente, todas as histórias possuem um início, meio e fim. Thiago Fogaça, professor de roteiro formado na FAAP, explica estrutura de três atos como:

A Estrutura de Três Atos é um espelho da mente humana. Quando contamos algo para um amigo, imaginamos uma situação, fantasiamos com o futuro ou lembramos de algo, isto vem perfeitamente estruturado com Apresentação, Desenvolvimento e Conclusão. Uma criança em conhecimento algum de dramaturgia contará uma história desse jeito. Três Atos não é uma fórmula, é como percebemos histórias desde o início dos tempos.

Apesar da forma tradicional de se criar um roteiro seja em forma de um texto, roteiros podem existir na forma de *flowchart* (gráfico de fluxo) ou como

storyboard. Utilizando o *flowchart*, o roteirista dá mais ênfase nas decisões e as consequências dessas decisões quando cria o roteiro, criando assim um grafo de fluxo (Figura 1). Esse tipo de roteiro é utilizado para mapear de forma mais explícita todas as escolhas possíveis que o jogador pode fazer durante o jogo.

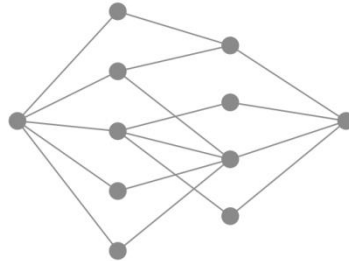


Figura 1 – Exemplo básico de fluxograma

Já o *storyboard* é utilizado quando além da história ou enredo, também se quer passar a fotografia ou a visão daquela história. *Storyboard* é utilizado para pré-visualizar a história contada pelo jogo como uma história em quadrinhos (Figura 2). Dessa maneira, é possível saber se a história e as animações estão se encaixando e se harmonizando.

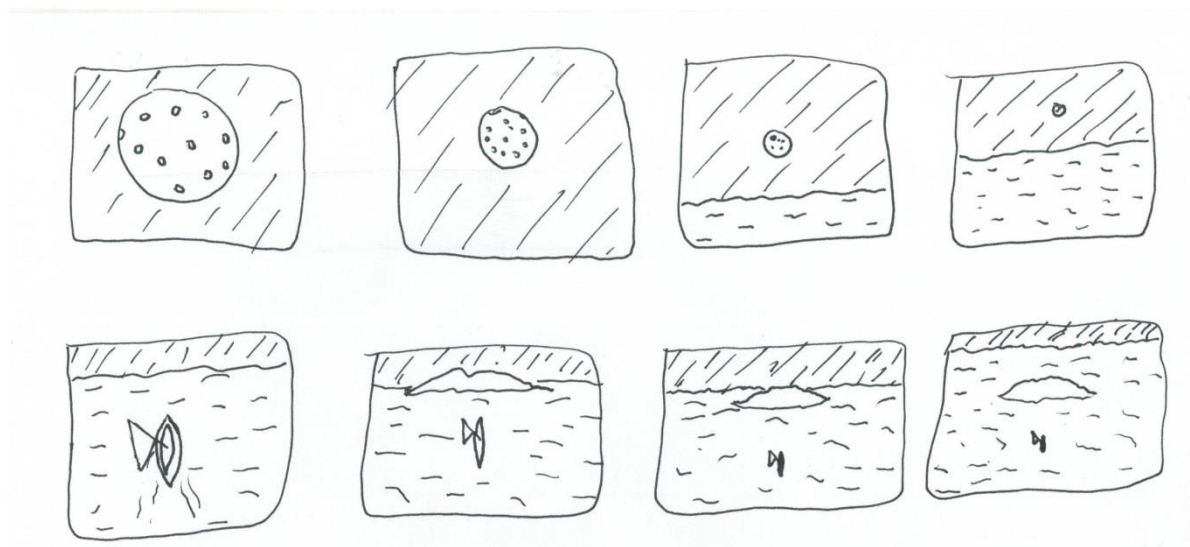


Figura 2 - Esboço de uma possível cena de *Sillage* em forma de *storyboard*.

É possível perceber que o roteiro traz consigo muitas informações a respeito do jogo, por isso um estudo mais aprofundado e um carinho especial é necessário nesse estágio de desenvolvimento para a criação de um jogo.

2.6 Character Design

Character Design é a idealização, planejamento e criação de um personagem, desde altura, peso e roupas usadas até sua personalidade, com todos

os traços psicológicos específicos. Um personagem bem produzido deve possuir todos os seus traços e personalidades bem estabelecidos harmonizando com seu ambiente criando assim uma imersão maior do jogador. Quanto melhor produzido esses personagens, é criada uma maior identificação do jogador com o jogo, melhorando sua experiência.

O problema é que criar um personagem com essas características não é uma tarefa trivial. Quando se pensa em todas as variáveis possíveis de um personagem, é fácil se perder em um mar de atributos e criando personagens que não se encaixam, fazendo com que ele destoe do resto do jogo.

Por isso é bom se certificar de que certas características básicas do personagem são supridas, para isso aconselha-se criar uma lista de objetivos que o personagem deve cumprir durante o jogo. Segue uma lista genérica de perguntas que devem ser respondidas para a criação de um personagem.

- Qual é a importância desse personagem? Esse personagem é principal? É um ajudante? Qual a relevância desse personagem na história?
- Qual a personalidade desse personagem? Ele é agressivo, espontâneo, triste, alegre, quieto, agitado, depressivo? Como essas características vão ser expressas no jogo?
- Esse personagem é controlado pelo jogador (*non-player character* ou NPC) ou não? O que ele faz no jogo? Como ele participa do jogo?

Essas são características principais que devem estar bem respondidas para criação de um personagem. É claro que nada no mundo da criação de jogos é uma receita de bolo. Tudo depende do estilo e das características do jogo. Em alguns casos serão necessários outros tipos de perguntas para obter a noção completa do personagem.

2.7 Level Design

O *level design* é o planejamento de todo ambiente que o jogador interage e explora. É o exercício de posicionar todos os objetos, estruturas para enfatizar a experiência do jogo. Esse é um processo tanto artístico quanto técnico, exigindo conceitos de design e lógica. Esse processo começa com a ideia do mapa/fase, que pode incluir esboços, modelos virtuais e até modelos físicos do mesmo.

Um estudo sobre *design*, mais precisamente, um estudo sobre o comportamento das pessoas quando sujeitos a um *layout*, é imprescindível, já que caso o *level designer* consiga atrair a atenção do jogador para partes estratégicas do mapa, pode drasticamente, aumentar a experiência de seu jogo.

Para tentar exemplificar esse estudo, olhe a Figura 3. Da forma em que foram dispostas as formas geométricas, gera uma atenção especial ao centro da imagem. É muito provável que um jogador, ao encontrar uma sala com esse padrão, examine o centro da sala, não dando tanta importância aos cantos dela.

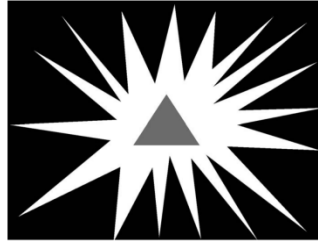


Figura 3 - Atração ao centro.

Em contrapartida, na Figura 4, a atenção é voltada para os cantos extremos do mapa, assim, um jogador nessa situação, tenderia a pensar que o ambiente é maior do que ele enxerga, assim, tentará explorar ambientes que não estão dispostas na cena inicial.

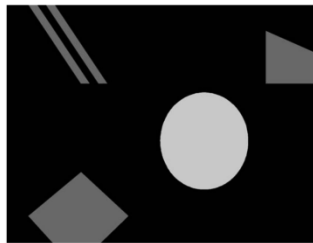


Figura 4 - Atração aos cantos

Um bom *designer* de ambientes deve aproveitar dessas técnicas para conseguir o foco do jogador, assim guiando o mesmo pelo ambiente, dando uma falsa sensação de escolha do jogador.

2.8 Sound Design

O estudo de *sound design* refere-se ao planejamento e produção de qualquer música, efeito sonoro e/ou voz usado em jogos. Cada uma dessas partes, quando presentes (nem todo jogo terá voz, por exemplo), estão conectadas entre si, e devem ser pensadas não separadamente, mas como irão dialogar entre si no decorrer de um jogo. Ajustes simples, como o volume de cada uma dessas categorias em certo momento do jogo, pode fazer toda a diferença.

Quando um *game designer* pensa em adicionar sons ao jogo, normalmente apenas o usa para ambientar o jogador, utilizando os sons apenas como uma forma a mais de garantir a imersão do *game*. Porém, os sons, e juntamente com um planejamento desses sons podem ter um papel muito maior.

Jesse schell, mostra como é utilizado os sons de maneira eficiente em um restaurante, ele diz (2008, p. 292):

“Restaurants use this method all the time. Fast music makes people eat faster, so during a lunch rush, many restaurants play high energy dance music, because faster eating means more profits. And of course, during a slow period, like three in the afternoon, they do the opposite. An empty restaurant

often is a sign of a bad restaurant, so to make diners linger, they play slow music, which slows down the eating and makes customers consider ordering an extra cup of coffee or a dessert. Of course, the patrons don't realize this is happening — they think they have total freedom over their actions.”

Se essa técnica é utilizada em restaurantes, é possível aplicá-las a uma mídia interativa para conseguir um padrão de comportamento do jogador. Conseguindo fazer com que o jogador responda de forma correta a esses estímulos através dos sons, é possível guiar sua experiência, sem ele perceber.

Fica então clara a importância de se ter um bom *sound design* na hora de conceber um novo jogo. A boa execução pode mudar completamente o impacto que o *game* irá causar e até mesmo a maneira como ele irá ser jogado.

Capítulo 3

3 Projeto e desenvolvimento

Para a produção desse projeto, o mesmo foi dividido em duas etapas: pré-projeto e desenvolvimento. No pré-projeto foi feita a concepção da ideia do jogo, juntamente com suas mecânicas e história do jogo. Foram feitas reuniões com o grupo de desenvolvimento para decidir ideias de jogos. Com essa ideia pronta foi criado o documento de Game Design do jogo e definidas tecnologias que serão utilizadas para o desenvolvimento do projeto.

Já na etapa de desenvolvimento, como o próprio nome diz, foi desenvolvido o projeto usando ferramentas já escolhidas na fase de pré-projeto. Com essas ferramentas foram criadas mecânicas básicas necessárias para o jogo (*engine* do game) visando sempre a experiência descrita no documento de *Game Design*.

3.1 Pré-Projeto

3.1.1 Idealização do jogo

De acordo com Jessel Schell, um jogo é uma experiência que o jogador interage e todo bom criador de jogos deve se preocupar com essa experiência. Foi dessa forma que foi iniciada a sessão de *brainstorm* da equipe. Tudo no jogo deveria ser pensado, produzido e aplicado de forma que aumentasse essa experiência do jogador. Ao todo foram quatro reuniões de *brainstorm* para a concepção básica do jogo que fora produzido.

A primeira das experiências discutidas foi frenesi. Com essa experiência era o foco um jogo onde houvesse muita agitação do personagem, muitas cores na tela quase tangenciando a linha da poluição visual. Também era necessário que essa experiência fosse engraçada para o jogador, explorando esse exagero de informações na tela. Dado esse conceito, foram idealizadas algumas mecânicas de jogo que impulsionassem essa experiência como lutas rápidas em um ambiente fechado e mutável, até quatro jogadores jogando ao mesmo tempo, e muitos inimigos (figura 5).

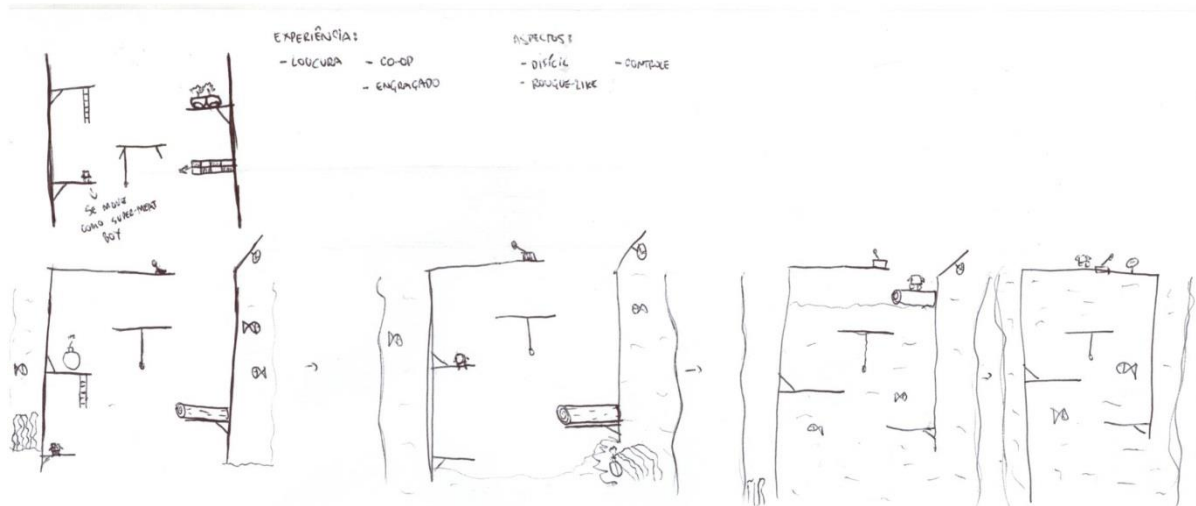


Figura 5 - Esboço da primeira idéia de jogo proposta

Apesar da equipe ter gostado dessa ideia de jogo, foi proposto trabalhar com algo mais lúdico e subjetivo, algo que tivesse carregado de emoções mais fortes. Então em outra reunião de *brainstorm*, surgiu a ideia de trabalhar na experiência “toda ação tem uma reação”. Essa foi aceita quase que imediatamente pela equipe e ideias de como trabalhar com ela começaram a surgir (Figura 6). Ao final foi proposto um enredo básico do jogo, que ainda não tinha um nome nessa etapa.

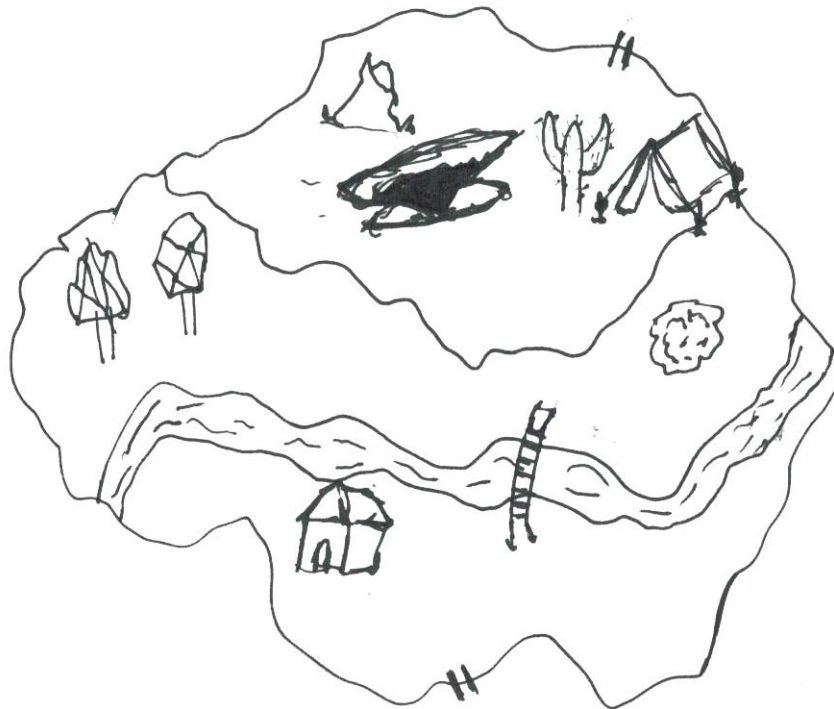


Figura 6 - Esboço inicial de Sillage

3.1.2 Trabalhando a ideia

Seguindo essa ideia, o jogo começaria apresentando o jogador perdido em uma ilha que nunca estivera antes. Nela o jogador é capaz de cortar árvores e adquirir madeira, minerar pedras preciosas, e conseguir carne e peles de animais selvagens. Nesta ilha também possuem cidades e civilizações primitivas, onde em algumas delas o jogador pode vender esses recursos e comprar equipamentos melhores, além de encontrar missões e informações da história do jogo. A ilha é dividida em três grandes regiões: floresta, deserto e ambiente de neve. Cada uma dessas regiões possuem civilizações, animais, e outros recursos específicos de cada região. O jogador deve sobreviver na ilha, investigando essas cidades e ambientes para tentar descobrir o que aconteceu.

Para atingir a experiência de ação e reação foram usadas todas essas interações que o jogador pode desempenhar para modificar o ambiente a sua volta. Toda vez que o jogador morrer e começar uma nova partida, a ilha estará modificada de acordo com sua última partida. Quando o jogador corta muitas árvores numa partida, na próxima vez que jogar a ilha estará sofrendo uma escassez de árvores. O mesmo vale para minerar pedras, ou caçar animais. Como o recurso dessa rodada estará raro, seu preço de venda nas cidades estará maior. A região onde esse recurso é mais encontrado (árvores em floresta, pedras em deserto ou animais de qualquer região) sofre uma diminuição em tamanho proporcional a ilha.

Dessa forma, toda vez que o jogador começar uma partida, ele estará imerso em um ambiente fabricado por escolhas passadas, tendo que se adaptar às mecânicas diferentes. Usando essas modificações do ambiente e em cidades acredita-se que o jogo seja capaz passar a sensação de que as escolhas e ações do jogador seja um fator mais impactante que o normal.

3.1.3 Trabalhando a história

Será apresentada ao jogador a história gradativamente durante as missões do jogo. A história central é sobre um tema subjetivo e emocional: a ilha é uma metáfora de um relacionamento acabado. A história será sobre uma pessoa comum tentando superar a quebra de um vínculo emocional muito forte. A cada nova partida, o jogador voltará à ilha através de uma memória desse relacionamento, começando o jogo com uma frase escrita na tela como “esse era seu perfume” ou “você adorava essa música”. Sempre uma frase que remete algo do passado para enfatizar o fato do relacionamento estar acabado.

Com esse tema sobre relacionamento acabado, um estudo psicológico sobre os estágios do luto foi utilizado para ajudar na concepção de novas missões e histórias do jogo. Elisabeth Kübler-Ross (psiquiatra que nasceu na Suíça) propôs o modelo Kübler-Ross em seu livro *On Death and Dying*, após anos de trabalho e

estudo com enfermos em fase terminal. Com esse modelo ela ajudou famílias, médicos, enfermeiros e próprios enfermos a lidarem melhor com a temática da morte. Esse modelo propõe cinco estágios em que pessoas que sofrem uma grande perda, tragédia ou luto devem percorrer para superar essa quebra de vínculo emocional. João Carlos Gama Martins de Macedo diz:

“A partir de mais de duzentas entrevistas efectuadas junto de doentes em fase terminal, Kübler-Ross julgou encontrar um padrão específico de reacções psicológicas que o ser humano percorre à medida que a morte se aproxima. Concluiu, assim, que a maioria dos doentes passaria pelas seguintes fases, de forma quase sequencial:

- negação e isolamento (denial and isolation);
- raiva (anger);
- negociação (bargaining);
- depressão (depression);
- aceitação (acceptance).”

A utilização de conhecimentos fora da esfera de desenvolvimento de *games* para criação de conteúdo em jogos é uma prática muito utilizada e aceita. Com essa abordagem é possível criar detalhes estratégicos capazes de aprimorar a experiência do jogador.

3.1.4 Finalizações

O projeto então foi concebido. Uma experiência única e temática emocional forte. Só faltava uma única coisa a ser definida: o nome. Foi então que pesquisando palavras de outras línguas, foi encontrada a palavra *sillage*, que no francês significa “o perfume que fica no ar quando alguém passa” ou “a onda formada quando um barco passa na água”. Essas definições se encaixam perfeitamente na história e experiência que foi idealizada e por essa razão se tornou o nome do jogo.

3.1.5 Documento de *Game Design*

Com a ideia do jogo já trabalhada, foi criado o documento de *game design* que serviria de guia para o desenvolvimento do mesmo. Nele consta características principais do jogo como resumo, história, experiência a se alcançar, jogabilidade e mecânicas que serão aplicadas. Esse documento é muito útil para uma equipe estar sempre se baseando nas mesmas informações para criar o jogo.

Após o término dessa etapa de *brainstorm* e definições de projetos, foi iniciado a etapa de desenvolvimento e estudo da *engine* do jogo, assim como tecnologias para serem aplicadas ao projeto.

3.2 Ferramentas

Abaixo estão listadas as ferramentas importantes que foram utilizadas para realização do projeto Sillage. Também são apontadas superficialmente o funcionamento e descrição desses produtos além das principais características que levaram a escolha dessas ferramentas para o projeto.

3.2.1 Asana

Asana é um aplicativo *web* e *mobile* que visa a organização das tarefas de um projeto em grupo ou pessoal. Foi fundada oficialmente em novembro de 2011 por dois ex-membros da empresa Facebook, Dustin Moskovitz e Justin Rosenstein.

Sua meta principal é fazer com que uma pessoa ou um grupo gerencie e planeje um projeto de forma rápida, prática e sem utilizar *e-mails*. A licença do uso do aplicativo é grátis para equipes menores que quinze pessoas, acima desse número o preço varia de acordo com a quantidade de pessoas incluídas na equipe.

Esse aplicativo (Figura 7) cria um ambiente mais prático para gerenciar projetos. Cada grupo possui um ambiente próprio que contém projetos daquele grupo. E para cada projeto existem tarefas a serem cumpridas. Usuários criam tarefas e as atribuí a outros usuários, e nelas podem criar notas, comentários, marcações, anexar conteúdos e subtarefas.

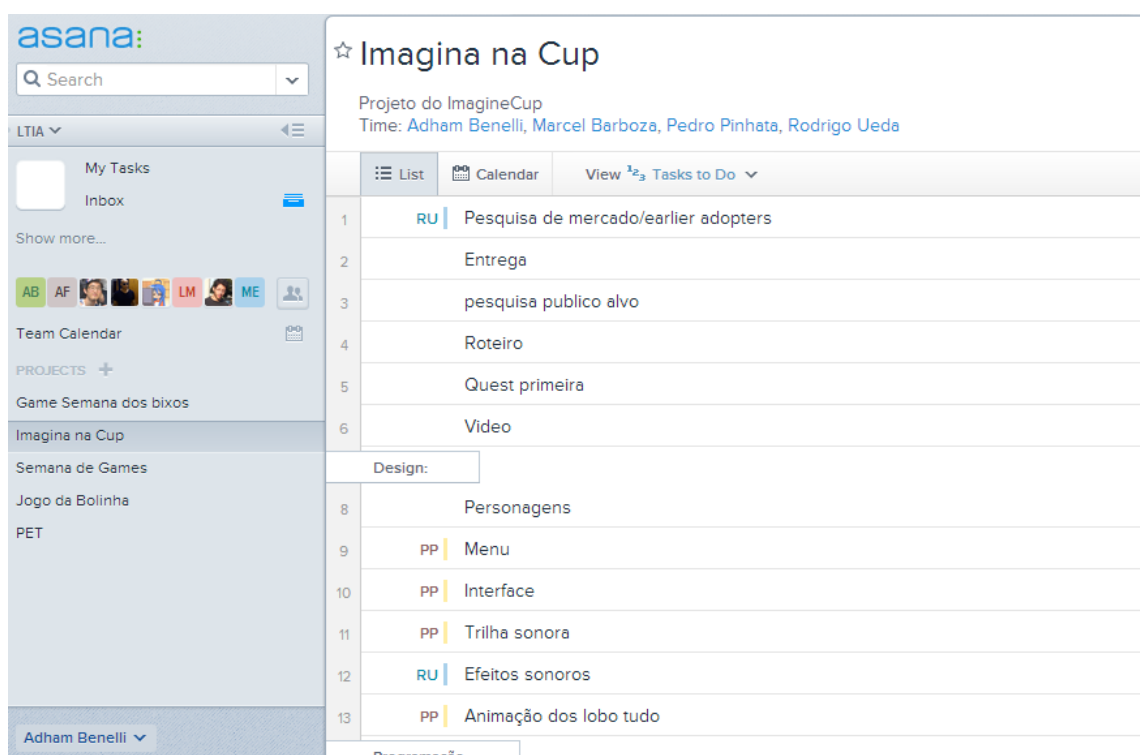


Figura 7 - Tela principal do Asana

No projeto, Asana foi utilizado para gerenciar as tarefas de desenvolvimento do jogo Sillage e manter rastreamento de seu desenvolvimento. Foi usado pela sua

facilidade de uso, atribuição de tarefas e sua rapidez de comunicação entre os membros, além das características já descritas acima.

3.2.2 Unity

Unity é umas das ferramentas de desenvolvimento de jogos mais completa do mercado, criada pela *Unity Technologies*. Com ela é possível criar jogos de maneira eficiente, organizada e em um único programa. Desde a produção de códigos até animações complexas podem ser feitas exclusivamente no Unity.

Essa ferramenta foi criada com o intuito de ser simples e que qualquer um curioso suficiente consiga criar jogos. Ela possui duas distribuições: a Unity Pro, que custa R\$ 1500,00 e a versão grátis que pode ser usada para fins lucrativos ou educacionais. Por essas características a ferramenta foi muito aceita por desenvolvedores indie e empresas de pequeno porte, ainda mais após a versão Unity 4.0 que possibilitou criar jogos multi-plataformas, até mesmo para a versão gratuita.

As principais características do Unity que levou a sua utilização nesse projeto, além das já descritas acima, foram:

- A implementação nativa de interações físicas no programa, como gravidade, atrito, colisão e outros.
- Compatibilidade com vários navegadores (utilizando o *plugin* Unity Web Player).
- Compatibilidade com ferramentas de modelagem 3D.
- Programação de *scripts* em C# (utilizando o Monodevelop ou Microsoft Visual Studio).

Outra característica muito marcante do Unity é sua comunidade muito ativa. No fórum oficial do Unity é possível encontrar muito material para referencias e estudos sobre o Unity e desenvolvimento de jogos em geral, além de encontrar soluções, *plug-ins* e bibliotecas.

Outro reservatório de recursos da Unity muito utilizado é o *Unity Asset Store*, onde existem milhares de soluções prontas para Unity, desde códigos, *shaders* até modelos 3D ou animações inteiras, sendo alguns desses recursos são pagos, outros são gratuitos.

3.2.3 Microsoft Visual Studio

Microsoft visual studio é um ambiente de desenvolvimento integrado (IDE) feita pela Microsoft que voltada para desenvolvimento da framework .NET e para algumas linguagens como C, C++, C#, *Visual Basic* e J#. A versão mais atual é *Visual Studio 2012* que utiliza o .NET framework 4.5

Esse ambiente é altamente robusto e poderoso. Com a utilização da plataforma de desenvolvimento .NET, é possível criar aplicativos para *Windows*,

Windows Phone, *Windows Server* e *Microsoft Azure*. O ambiente também possui um recurso chamado *IntelliSense* que ajuda o desenvolvedor a aprender mais sobre o código que está usando, manter o controle dos parâmetros que está digitando, e adiciona chamadas de métodos e propriedades de forma mais rápida e simples.

Foi escolhido esse ambiente para o desenvolvimento do projeto pela possibilidade de utilização conjunta com o *Unity*. Quando o ambiente é aberto pelo *Unity*, é aberto uma página com toda a documentação de classes e métodos do próprio *Unity*, além da ferramenta *IntelliSense* também acessar todas as estruturas exclusivas do *Unity*.

3.2.4 Tiled

Tiled é um poderoso editor de mapa de *tiles* (definições sobre tiles se encontram na página 31) capaz de criar grandes mapas de *tiles* de forma rápida e fácil. Esse editor também tem um formato único de salvar seus mapas criados chamado *tmx*, mas também aceita outros formatos.

A forma de se criar um mapa no *Tiled* é muito simples e intuitiva. Utilizando as ferramentas carimbo e ferramenta de preenchimento já é possível criar mapas inteiros apenas pintando os *tiles*. Os *tiles* ficam armazenados em uma imagem chamada *tile set*. Nela ficam gravadas todos os *tiles* do mapa. É apenas necessário selecionar o *tile* no *tile set* e “pintar” o mapa. Depois que o mapa está pronto, é possível salvar em extensões que podem ser utilizados por outros editores. Figura 8 mostra o ambiente do editor.

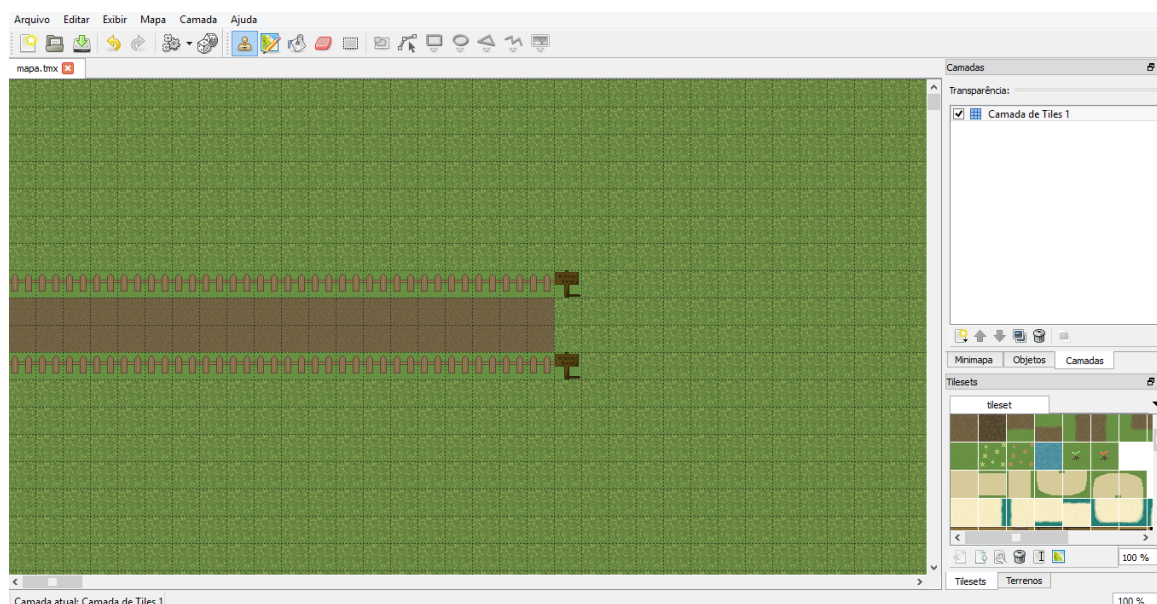


Figura 8 - Ambiente de Tiled

O arquivo *tmx* (Figura 9) é um documento de marcações (*tags*) com algumas já definidas. Na marcação *map*, estão informações sobre o mapa como tamanho do

mapa e tamanho dos *tiles*. Em *layer*, está o tamanho do mapa naquela região (com *Tiled* é possível criar um arquivo *tmx* que contenha o *tile map* de mais de um ambiente e cada um destes está em uma camada, sendo possível acessar cada um separadamente em um único arquivo). O mapa de *tiles* de cada mapa fica encapsulado na marcação *data* e é formado por um número (que representa o *tile* daquela posição) e uma vírgula para separar cada *tile*. Esse arquivo *tmx* é facilmente transformado para um arquivo *xml* apenas trocando sua extensão.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <map version="1.0" orientation="orthogonal" width="500" height="500" tilewidth="128" tileheight="128">
3   <tileset firstgid="1" name="puros" tilewidth="128" tileheight="128">
4     <image source="TodosTiles/puros.png" width="256" height="256"/>
5   </tileset>
6   <layer name="Camada de Tiles 1" width="500" height="500">
7     <data encoding="csv">...</data>
509 </layer>
510 </map>
511

```

Figura 9 - map.xml com as marcações do arquivo *tmx*.

3.3 Desenvolvimento

3.3.1 Perspectiva de jogo

Existem várias formas de enxergar o mundo e de apresentar o mundo. E essa afirmação também pode ser aplicada para mundos artificiais. Quando são discutidos jogos digitais e como eles apresentam o ambiente para o jogador, é perceptível uma grande quantidade de possibilidades diferentes.

A escolha de como o jogador irá visualizar os elementos na tela é um trabalho importante na fase de criação. A forma de se ver e interagir com esse mundo deve ser projetada para enfatizar a experiência e jogabilidade do *game*. Uma visão bem aplicada é capaz de aumentar à imersão do jogador, auxiliar a experiência e contribuir para a visual do jogo.

Dois tipos de perspectiva muito exploradas são a visão em primeira pessoa e visão em terceira pessoa, onde cada uma dela traz características diferentes ao jogo. Essas peculiaridades devem ser levadas em conta durante o processo de criação.

- Primeira pessoa: Este tipo de visão (Figura 10) é mais imersiva para o jogador, já que ele visualiza o mundo pelos olhos do personagem dentro do jogo. Traz também mais realismo ao jogo, já que é dessa forma que vemos nosso mundo real. Porém, essa perspectiva limita o campo de visão do jogador, que só enxerga o que está em sua frente.



Figura 10 - Exemplo de visão em primeira pessoa (*Counter Strike: Global Offensive*¹)

- Terceira pessoa: Já essa perspectiva (Figura 11), auxilia na noção espacial do jogador, possibilitando uma percepção do que está acontecendo em volta do personagem, dando a ele um movimento mais livre e preciso pelo ambiente. Mas há uma perda de realismo e imersão do jogador ao meio comparado com visão em primeira pessoa.

¹ Endereço eletrônico da imagem: <http://pcmedia.ign.com/pc/image/article/121/1217843/counter-strike-global-offensive-20120202043855411.jpg>



Figura 11 - Exemplo de visão em terceira pessoa (*Mario Galaxy 2*²).

Imaginem, por exemplo, um jogo onde há vários inimigos que surgem de todos os lados e o jogador deve destruí-los e ao mesmo tempo desviar dos tiros dos inimigos usando algum tipo de movimentação especial como agachar e rolar. Há também vários itens e lugares onde o jogador pode conseguir cobertura para se proteger dos ataques. Uma visão em terceira pessoa traz uma melhor percepção do ambiente, ajudando o jogador tomar decisões de onde se esconder e melhora sua movimentação pelo jogo. Já uma visão em primeira pessoa pode criar um realismo maior e imergir o jogador em uma batalha real, também melhorando o controle da mira do jogador. Todas essas características devem ser levedas em conta para a escolha de uma perspectiva para seu jogo.

Porém, é bom lembrar que o processo de criação não funciona como uma receita para todos os casos, escolhas como o tipo de perspectiva devem ser feitas para enfatizar a experiência do jogo e não ser vistas simplesmente pelas suas características. Há jogos em que uma menor área de visão seja benéfica à experiência do jogo (exemplos seriam jogos de terror, que com uma visão

² Endereço eletrônico da imagem: http://cdn.sonigate.com/product_images/1008801_Jogo-Nintendo-Wii-Super-Mario-Galaxy-2_4.jpeg

comprometida pode causar uma certa claustrofobia e complementar a experiência de medo do jogador).

3.3.2 Visão isométrica

Há vários tipos de perspectivas usadas na engenharia, na arquitetura e nos desenhos industriais, e cada uma delas é usada para explorar de forma diferente a cena projetada, mudando a forma com que o observador passa a perceber e interagir com a cena.

No passado, quando a computação gráfica e a tecnologia dos computadores em geral eram bem inferiores a de atualmente, desenvolvedores de jogos precisavam utilizar formas de apresentar o mundo ao jogador que enriquecesse o jogo tanto em questão de mobilidade como de realismo do jogo utilizando muito menos recursos que o de atualmente, e uma solução para isso foi a implementação da visão isométrica.

Visão isométrica é uma maneira de visualizar alguma cena. Na Figura 12 é possível visualizar três fases de um cubo, onde suas arestas centrais formam um ângulo de 120° entre si. Esse tipo de perspectiva é muito utilizado porque com ela é possível atingir uma sensação de profundidade, além de sua criação ser relativamente simples. Para atingir esse efeito, o observador precisa estar situado num ponto do infinito (perspectiva ortográfica), a 30° do eixo horizontal e a 45° do eixo vertical.

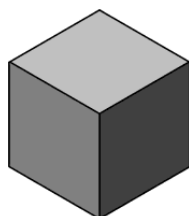


Figura 12 - Cubo visto de forma isométrica³.

Em *Sillage*, foi usada a visão isométrica (perspectiva em terceira pessoa) com o intuito de se produzir um mundo onde o jogador conseguisse visualizar melhor todos os recursos e inimigos a sua volta, dando uma maior liberdade de movimento para batalhar ou fugir de inimigos. Essa visão também enfatiza o visual artístico do jogo, possibilitando uma melhor visualização das texturas e animações incluídas na cena.

Para isso, foi necessário mudar a posição e rotação da câmera de visualização no *Unity*. Foi criado um *Empty GameObject* chamado “mapa” com a angulação necessária para conseguir o efeito isométrico e cada objeto que aparecesse na tela, seria filho de “mapa”, assim herdando sua rotação. Dessa

³ Endereço eletrônico da imagem: <http://img.ibxk.com.br/materias/IsometricCubeGray.png>

forma, o posicionamento e rotação dos objetos na tela ficam mais dinâmicos e automatizados.

3.3.3 Tiles e suas aplicações

Tiles têm sido usados em jogos há muito tempo. No tempo em que os computadores eram muito menos potentes que os de hoje em dia, desenvolvedores de jogos tinham que criar formas de conceber jogos que fossem mais graficamente bonitos (imagens mais detalhadas e mais pesadas) e que economizassem memória e processamento do computador.

Antes de aprofundar sobre “tile-based games” (ou jogos baseados em *tiles*), se faz necessário definir o que são tiles e mapas de tiles. Tiles (Figura 13) são pequenos azulejos (do inglês, tile significa azulejo) que quando posicionados de forma correta geram uma figura maior. Funciona como peças de um quebra-cabeça, que quando montado, forma uma imagem.

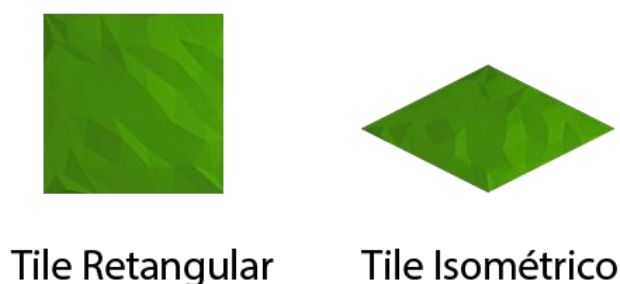


Figura 13 - Exemplo de tiles.

Um mapa de *tiles* (Figura 14) é uma figura formada por vários *tiles* juntos. Seguindo o mesmo exemplo, mapa de *tiles* é a figura que o quebra-cabeça apresenta depois de montado.

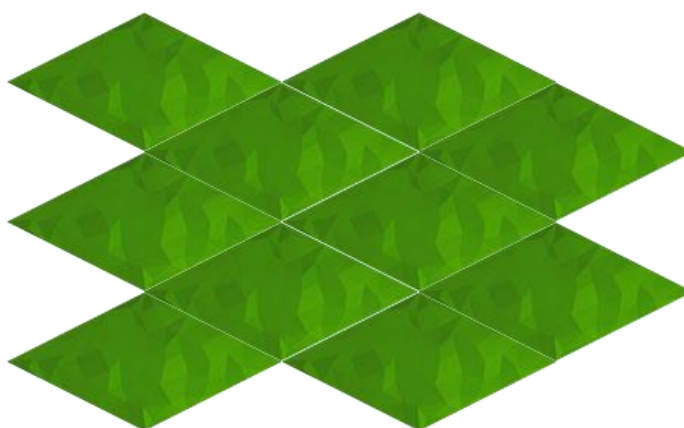


Figura 14 - Mapa de tiles

Essa técnica é muito utilizada quando é necessário criar uma região ou área de imagem muito grande sem consumir muita memória carregando essa figura inteira. Desta forma, apenas carregando pequenos *tiles*, é possível criar mapas grandes e dinâmicos, facilitando qualquer tipo de modificações do mapa, seja em tempo real, quanto na criação do *level design* do jogo.

Um mapa de *tiles* nada mais é que uma matriz, onde cada elemento da matriz representa um *tile*. O *tile* pode ser uma imagem, um número (Figura 15) que representa a imagem ou uma estrutura mais complexa, que podem armazenar qualidades de cada *tile* (se o *player* pode passar por esse *tile*, por exemplo).

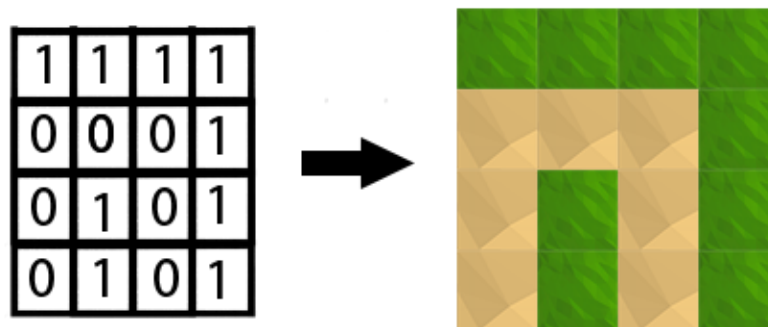


Figura 15 - Exemplo de uma matriz de números representando o mapa.

Apesar de ser relativamente simples a implementação, o resultado não é graficamente tão bonito quanto um mapa inteiramente desenhado sem auxílio de *tiles*. É possível perceber os *tiles*, já que não há nenhum tipo de suavização entre *tiles* diferentes - como no exemplo acima, há uma quebra entre o *tile* de areia e o *tile* de grama - e por isso não dá um resultado visualmente bonito.

O que pode ser feito para melhorar o visual do mapa de *tiles* é criar os chamados *tiles* de borda (Figura 16). Eles são *tiles* que fazem uma transição entre

dois *tiles* distintos. Esses *tiles* fazem com que o mapa fique mais graficamente bonito, retirando cortes fortes entre dois terrenos.

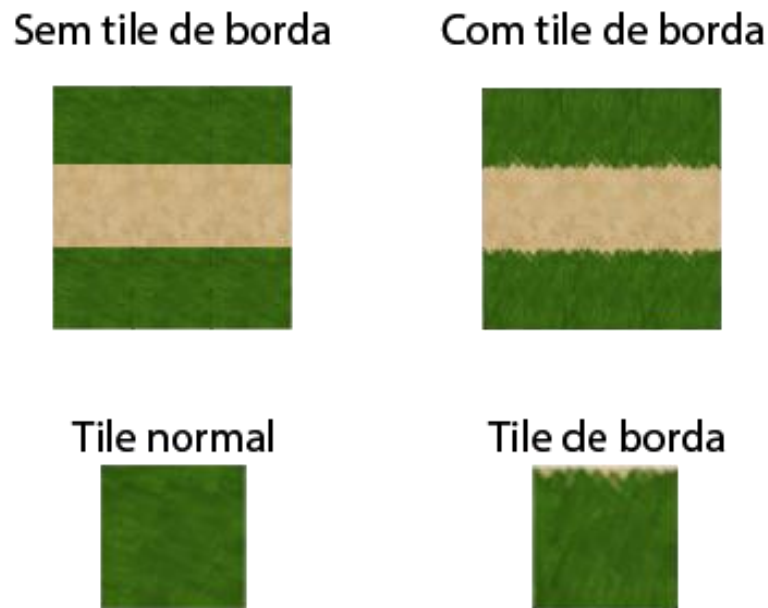


Figura 16 - Tiles de borda

Para utilizar *tiles* de borda, o desenvolvedor pode criar valores para ele diretamente na matriz, assim o algoritmo que posiciona os *tiles* na tela não precisaria ser modificado, esses tiles de borda seriam apenas outro tipo de *tile* para a lógica do algoritmo. E quando o mapa é gerado dinamicamente? Nesses casos, o posicionamento de tiles de bordas deve ser feito de forma dinâmica também.

No Sillage, como o mapa é dinâmico, foi necessário criar um algoritmo de posicionamento de bordas também dinâmico. Esse algoritmo funciona da seguinte forma: Para cada *tile* no mapa, é verificado os *tiles* vizinhos (as oito posições em volta do *tile*), caso algum desses vizinhos seja diferente que o *tile* central, foi encontrada uma borda. Então, com ajuda dos mesmos *tiles* vizinhos, é escolhido o *tile* que irá substituir o *tile* central. Assim todo o mapa modificado e *tiles* de bordas são aplicados.

3.3.4 Mapa

Jogos *tile-based* tem uma facilidade em criar um cenário dinâmico, já que modificar o mapa ou regiões do mapa, é necessário modificar os valores dos *tiles* no mapa de *tiles*, e não criar um novo mapa inteiro. Para isso é necessário um algoritmo de criação de mapa capaz de criar esse ambiente de forma dinâmica, desta forma é possível ter um mapa diferente para cada vez que jogar. Esses

algoritmos são os chamados: Geração procedural de conteúdo (*procedural content generation*).

No jogo produzido, o mapa de *tiles* é gerado a partir de um arquivo xml criado no Tiled chamado map.xml (Figura 17). Neste arquivo, dentro da marcação “data” existe um conjunto de números seguidos de uma vírgula. Este é o mapa de *tiles* não processado. Cada número presente nesse texto está associado a um tipo diferente de *tile* dentro do jogo. Quando o jogador morre ou fecha o jogo, o mapa do jogador volta a ser salvo nesse mesmo xml.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <map version="1.0" orientation="orthogonal" width="500" height="500" tilewidth="128" tileheight="128">
3   <tileset firstgid="1" name="puros" tilewidth="128" tileheight="128">
4     <image source="TodosTiles/puros.png" width="256" height="256"/>
5   </tileset>
6   <layer name="Camada de Tiles 1" width="500" height="500">
7     <data encoding="csv">...</data>
509   </layer>
510 </map>
511

```

Figura 17 - map.xml (O mapa não processado está encapsulado em data).

É importante perceber que mesmo utilizando *tiles* para montar o mapa, desenhando ele todo de uma vez, não haveria melhoria significativa de desempenho se comparado a uma imagem única que contivesse o mapa desenhado. Para haver uma otimização nesse quesito, é necessário fazer com que o jogo apresente somente a parte do mapa (e seus objetos como árvores, animais, itens, etc) próxima ao jogador, dessa forma diminuindo processos desnecessários.

Nesta etapa do desenvolvimento foi necessário que outros tipos de otimizações fossem aplicadas ao jogo para uma melhora de desempenho. Essa melhorias foram feitas através da implementação de *object pool* e utilização de corotinas e vão ser mais discutidas no tópico Otimizações neste mesmo documento.

No Sillage foi usada a técnica de modificação do mapa baseado na última vez em que o jogador jogou o *game*, desta forma a questão de aleatoriedade do mapa, passou a ser sujeita as escolhas feitas pelo jogador nesta última rodada. O tipo de modificação implementada foi o aumento e diminuição de uma determinada região. Toda vez que uma região sofre uma modificação, o raio da região é diminuído ou aumentado em um *tile* (Figura 18).

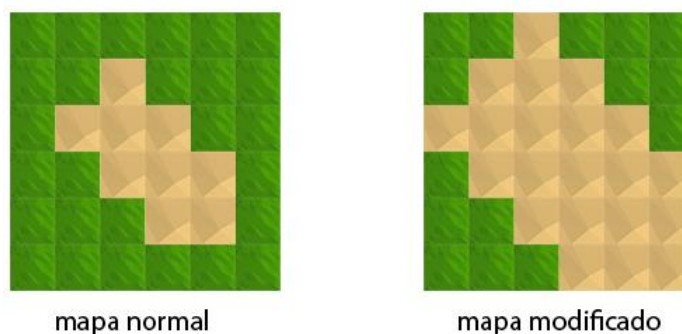


Figura 18 - Modificação na região de areia

3.3.5 Jogador e o que ele representa

Jogos são mídias interativas, e como toda mídia, a história tem um papel fundamental, porém não deve ser o foco total de um jogo. Um game normalmente possui um personagem, que o jogador pode controlar. E nesses casos, é importante fazer com que o personagem seja o foco do jogo para aproveitar a iteratividade que só *games* podem trazer a uma mídia. Ken Rolston, *game designer* americano, dá um relato no livro de Paul Schuytma *Design de Games: uma abordagem prática*: (2008, p. 175)

“Um design de game é uma lista dos objetos no mundo e as maneiras como o jogador pode interagir com eles. O jogador pode passar por eles, movê-los ou derrotá-los e pegar outros objetos. Alguns objetos ele pode pegar, outros pode usar, e alguns, só pode olhar. Mas tudo diz a respeito a fazer algo com um objeto.

Então, não pense no que está acontecendo na tela. Isso seria um filme. Imagine o que o jogador está pensando em fazer e em como ele vai fazer, com o quê e por quê.[...].

O game não trata do que acontece com o jogador, mas, sim, do que o jogador faz.”

No projeto *Sillage*, o personagem (Figura 19) é capaz de se movimentar livremente pelo mapa, trocar de equipamentos, cortar árvores, minerar pedras e atacar animais. Essas ações são todas desempenhadas usando o teclado e o mouse. Usa-se as setas do teclado para movimentar o player, o clique com o botão direito do mouse para ações como minerar e cortar árvore e um clique com o botão esquerdo para atacar. Essas ações são realizadas apenas se o personagem estiver equipado com o item apropriado (espada para atacar, machado para cortar árvores e picareta para minerar).



Figura 19 - Personagem principal de Sillage.

Existem duas classes para controlar todas as ações e informações do jogador que são: “Player” e “ControlePlayer”.

- “Player” é usado para armazenar todos os dados e informações do jogador como: quantidade de pontos de vida, pontos de ataque e defesa, itens, armas e armaduras.
- “ControlePlayer” é encarregado de todas as interações que o jogador pode fazer; Sua movimentação e ações que o jogador desempenha é tratado nessa classe.

Além dessas classes, há também outras que funcionam como estrutura de dados, armazenando informações e não controlando comportamentos. Essas classes são: Armas e Armaduras. Nelas, são declaradas informações de todos os tipos de armas e armaduras do jogo que o *player* irá utilizar como dano e alcance para armas e defesa para armadura. Dessas classes são criadas as instancias que serão utilizadas e armazenadas no *player*.

3.3.6 Inteligência Artificial

“IA é a parte da ciência da computação que se preocupa em desenvolver sistemas computacionais inteligentes, isto é, sistemas que exibem características, as quais nós associamos com a inteligência no comportamento humano - por exemplo, compreensão da linguagem, aprendizado, raciocínio, resolução de problemas, etc.”.

(BARR, Avron, 1981, p. 21)

Nos jogos, muitas vezes é necessário proporcionar ao jogador obstáculos que sejam mutáveis, que “entendam” o que está acontecendo a sua volta, e “raciocine” para aprimorar sua mecânica e criar um obstáculo mais desafiador para o jogador. Esse obstáculo, apesar de simular um comportamento inteligente, está longe de possuir realmente uma inteligência artificial completa.

Em suma, Inteligência artificial é quando o computador realiza ou simula funções que são associadas a seres inteligentes. Esse tipo de inteligência no mundo dos jogos é empregada de várias maneiras diferentes e quando usada de forma correta, auxilia na imersão e a experiência do jogo é enriquecida.

Paul Schuytema, em seu livro *Design de Games: Uma abordagem prática* escreve sobre a ilusão de inteligência artificial nos jogos: “O objetivo não é desenvolver uma inteligência de verdade, mas criar no jogador a sensação de que um desafio está reagindo a ele e que o que ele (jogador) faz realmente importa.” (SHUYTEMA, Paul, 2008, p. 347).

A inteligência artificial usada em *Sillage* é bem simples. Ela toda está encapsulada em uma classe chamada “NPCBehavior”. Nesta classe é criado o comportamento dos *Non-Player Characters* (NPC) sendo eles *villagers* (personagens humanoides da ilha) ou animais. Nesta IA, o NPC vaga aleatoriamente centrado em um ponto do mapa, nunca se afastando muito desse centro. Esta classe possui uma *flag* que mostra se o NPC é agressivo ou não. Se for agressivo, o NPC começará a seguir o jogador assim que o mesmo entrar na área de ameaça do NPC, atacando-o quando chegar próximo o suficiente. A perseguição acaba quando o jogador morre, o NPC morre ou quando o NPC não atacar e nem ser atacado durante 15 segundos.

Esse tipo de IA simples, onde o NPC perambula por uma área do mapa é muito comum nos jogos, principalmente em jogos do estilo RPG. Ela traz um dinamismo maior ao NPC e ao jogo, incrementando a experiência de aleatoriedade do mapa e melhorando sua rejogabilidade.

3.4 Otimizações

Em todos os tipos de sistemas computacionais, otimizações são bem vindas. Melhorias de desempenho em questão de memória, processamento ou armazenamento podem significar se seu *software* será de sucesso ou não. E é claro, esse fato também se emprega para jogos digitais.

Abaixo será citado duas otimizações que tiveram um papel muito importante durante o desenvolvimento do jogo *Sillage*, mas é claro que essas não foram as únicas melhorias implementadas. Melhorias de código vinculadas ao Unity, documentação do código e a própria Unity, que usa conceitos físicos já implementados com otimizações, auxiliaram na criação do jogo.

3.4.1 Object Pool

Object Pool é um tipo de padrão de desenvolvimento utilizado quando se precisa de muitos objetos sendo executados ao mesmo tempo e a instanciação desses objetos é uma tarefa muito custosa. Neste padrão, têm-se uma quantidade de objetos já instanciados prontos para o uso armazenados na memória (*pool*) e

quando se faz necessário o uso desse objeto, ele é retirado da *pool* e utilizado normalmente. Ao final do uso do objeto, este não é destruído, ele é colocado de volta a *pool* para ser utilizado novamente. Esse padrão é capaz de aperfeiçoar o desempenho de um sistema quando o custo de iniciação de um objeto é muito alto e a taxa de iniciação e destruição do mesmo é muito grande.

Pense em *Object pool* como uma biblioteca na vida real. Todos sabem que é mais barato emprestar um livro da biblioteca do que comprar um novo. Da mesma forma, é mais barato (em relação a memória e velocidade) para um processo emprestar um objeto do que criar (instanciar) um novo.

3.4.2 Corotinas

Outra técnica utilizada para aperfeiçoar o jogo foram as Corotinas. Elas são rotinas mais generalizadas que são capazes de suspender sua execução por uma quantidade arbitrável de chamadas de funções (isto é, corotinas são construções *stackfull*), dando controle para as outras rotinas, e podem ser invocadas a qualquer momento e ordem durante a execução (isto é, corotinas são valores de primeira classe).

No Unity, corotinas são capazes de pausar a execução de uma parte do código e retornar sua execução (com os valores salvos da última chamada) após um determinado tempo ou quantidade de *frames*. Essa técnica é muito útil para processos de sequência de eventos dependentes de tempo ou para evitar realizar cálculos complexos a cada *frame*, economizando processamento do jogo.

3.4.3 Aplicações das otimizações

No Sillage, são utilizadas as duas otimizações: *Object pool* conciliado com corotinas. Essas técnicas tiveram um papel muito importante durante o desenvolvimento do jogo e com elas foi possível uma melhoria muito significativa do desempenho. Um mapa construído a partir de um arquivo xml de 200 por 200 *tiles* era inalcançável (Unity deixava de responder pelo excesso de objetos sendo instanciados) antes da utilização dessas duas técnicas. Atualmente o algoritmo é capaz de gerar um mapa de 500 por 500 *tiles* com todos os seus objetos (árvores, pedras, NPCs e animais) em aproximadamente vinte segundos.

Foi feita uma classe de *Object pool*, que possui duas listas para cada tipo de objeto. Uma lista contendo objetos que podem ser usados a qualquer momento e a outra lista para objetos que estão sendo utilizados. A aplicação desses objetos na tela foram feitos através de uma corotina, economizando ainda mais processamento.

Caso aconteça da lista de objetos prontos para uso estar vazia (todos os objetos estão sendo utilizados) e for necessário mais um objeto, o algoritmo

instancia um novo. Essa é uma solução para não causar quebras no jogo em questão de quantidade de objetos na tela, porque na prática, lista de objetos prontos nunca deve ficar vazia (caso aconteça durante os testes é melhor aumentar o tamanho da lista do que depender de instanciar novos objetos).

Capítulo 4

4 Conclusões

4.1 Problemas e soluções

Pela própria ideia de jogo ser complexa com uma história grande e subjetiva não foi possível a criação de um *level design* aparente ao jogo. Foi criado um demo, que era a proposta inicial, mas sem a aplicação de todo o conteúdo idealizado para o jogo.

Devido a pouca experiência em produção de jogos 3D dos envolvidos, houve um esforço excessivo em aprender e se habituar ao desenvolvimento desses tipos de jogos. Problemas com carregamento de objetos do jogo foram solucionados através das otimizações já descritas nesse documento (*Object pool* e *Corotinas*).

4.2 Considerações finais

Os objetivos foram cumpridos, criando um documento (essa monografia) sobre o processo de criação de um jogo digital desenvolvido por alunos da Unesp – Bauru. Ao longo da trajetória foi discutido técnicas e conceitos sobre *Game design* e o desenvolvimento e aplicação desse conteúdo, gerando um *demo* chamado Sillage.

Para trabalhos futuros é possível a continuação do desenvolvimento do jogo Sillage, aplicando mais conteúdos ao jogo e possibilitando um relato sobre os processos de pós-criação de um jogo, como distribuição e *marketing* do mesmo.

Referências bibliográficas

GALLAGHER, Michael D.. **Essencial facts about the computer e vídeo games industry**. 2013. Disponível em <http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf>. Acesso em 30 mai. 2014.

SEBRAE. **Brasil tem o maior mercado de games do mundo em 2012**. 2013. Disponível em <<http://www.sebrae2014.com.br/Sebrae2014/Alertas/Brasil-tem-o-maior-mercado-de-games-no-mundo-em-2012>>. Acesso em 20 de maio de 2014.

ZIMMERMAN, Eric.; SALEN, Katie. **Rules of Play: Game Design Fundamentals**, The MIT Press. 2003.

SHELL, Jessel. **The Art of Game Design: A Book of Lenses**. CRC Press. 2008.

MACEDO, João Carlos Gama Martins de. **Elisabeth Kübler-Ross: a necessidade de uma educação para a morte**. Portugal. Universidade do Minho. 2004.

SCHUYTEMA, Paul. **Design de Games: uma abordagem prática**. São Paulo. Cengage Learning. 2011.

BARR, Avron. **The Handbook of artificial intelligence**. Stanford, Calif. 1981.

Bibliografia complementar

NOTÍCIASBR, Brasil tem o maior mercado de games no mundo em 2012. 2013 <<http://www.noticiasbr.com.br/brasil-tem-o-maior-mercado-de-games-no-mundo-em-2012-2-107493.html>>. Acesso em 30 mai. 2014.

FREEMAN, Tzvi. **Creating A Great Design Document.** 1997. <http://www.gamasutra.com/view/feature/3224/creating_a_great_design_document.php>. Acesso em 30 mai. 2014.

KRAMER, Wolfgang. **What is a Game.** 2000. <<http://www.thegamesjournal.com/articles/WhatIsaGame.shtml>>. Acesso em 30 mai. 2014.

WIKIHOW. **How to Design Your Own Game Character.** [S.l]. <<http://www.wikihow.com/Design-Your-Own-Game-Character>> Acesso em 30 mai. 2014.

NYSTROM, Bob. **Object pool. 2009.** <<http://gameprogrammingpatterns.com/object-pool.html>>. Acesso em 30 mai. 2014.

Unity, Documentation. **Coroutines.** [S.l]. <<http://docs.unity3d.com/Documentation/Manual/Coroutines.html>> Acesso em 30 mai. 2014.

BEST-PRACTICE SOFTWARE ENGINEERING. **Object pool patern.** 2013. <<http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/objectpool.html>> Acesso em 30 mai. 2014.

LEE, Vanessa. O papel da música nos jogos eletrônicos e a sua história. 2012. <<http://canaltech.com.br/materia/games/O-papel-da-musica-nos-jogos-eletronicos-e-a-sua-historia/>>. Acesso em 30 mai. 2014.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023: informação e documentação - referências - elaboração. Rio de Janeiro: ABNT, 2002.

_____. NBR 10520: informação e documentação – citações em documentos - apresentação. Rio de Janeiro: ABNT, 2002.