GameTest - Jogo não-nomeado até segunda ordem

Grupo	
Nome Completo	Matrícula
Raphael Nogueira Rezende Laroca Pinto	202135014
Antônio Marcos da Silva Júnior	202135002
João Vítor de Castro Martins Ferreira Nogueira	202065560
Adriano Eiterer Oliveira	201865118

Sumário

1 Introdução

1.1 Propósito

O jogo busca ensinar aos alunos, os conceitos de gerência de projetos ensinados nas Aulas de Engenharia de Software de uma maneira divertida e utilizando uma área familiar aos alunos.

1.2 Escopo

Como o tempo da disciplina é curto, muitas das funcionalidades de que se esperam em um jogo completo não serão implementadas, mesmo assim, esperamos aprender bastante sobre a gerência de projetos e os conceitos de Engenharia de Software. Para a gerência de projetos, será utilizado quadros TaskBoard feitos do nosso jeito. Os programadores serão divididos em 2 equipes, A e B, a equipe A será responsável por tratar das funcionalidades críticas do jogo (tais como renderização, tratamento de input etc), já a equipe B tratará das funcionalidades periféricas (como o carregamento de níveis e arquivos).

Com isso, poderemos trabalhar em paralelo e desenvolver mais rápido. Usaremos um paradigma misturado com Clássico e Espiral. Uma vez que as funcionalidades são construídas e testadas e corrigidas, porém não voltaremos atrás a não ser que seja encontrado um erro muito grave ou seja necessário uma modificação muito grande nos módulos prontos.

Para a equipe A, será o responsável por ela Raphael Nogueira R L Pinto, assim como ele cabe a gerência do projeto inteiro. O segundo integrante da equipe A será Antônio Marcos. Para a equipe B, o líder será João Victor, com o segundo integrante sendo o Adriano.

As datas de inicio e término das respectivas atividades se encontram no documento anexo TaskBoard.ods, assim como o estado das mesmas.

1.3 Definições e Abreviações

- **GameTest:** nome dado ao Jogo por enquanto até ser definido seu nome oficial;
- **RF**: requisito funcional;
- **RNF**: requisito não funcional;
- **Player**: Jogador em inglês, o objeto que representa o jogador;
- **NPC**: Abreviação para *Non Playable Character*, persoangens que não são controlados pelo jogador;
- **RPGMaker**: Uma Engine para Jogos que foi tão difundida e única, que hoje em dia jogos são feitos copiando as características chave da Engine, tais características são mas não se limitam à, 2D, Arte em PixelArt,Baseado em Tiles, Visão 2D, Etc ;
- **DRM**: Abreviação de *Digital Rights Manager*, hoje em dia é usado como termo geral para sistemas de proteção a pirataria e copia ilegal ;

1.4 Visão Geral do Documento

- **Seção 2 - Descrição Geral:** apresenta uma visão geral do sistema, especificando a perspectiva do produto e detalhamento do escopo do sistema através da discretização das funções do produto. Além disso, são explicitadas as características gerais dos usuários do produto e as restrições que poderão limitar as possibilidades de desenvolvimento.
- **Seção 3 - Descrição dos Requisitos Funcionais (RF):** apresentação de todos os requisitos funcionais do sistema. Descreve as principais ações do produto, considerando a aceitação e processamento das entradas e o processamento e geração das saídas.
- **Seção 4 - Requisitos Não Funcionais:** apresentação de todos os requisitos não funcionais do sistema. Descreve todos os aspectos qualitativos do sistema, explicitando os detalhes de facilidade de uso, manutenibilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.

2 Descrição Geral

2.1 Perpectivas do Produtos

Para aplicarmos os conceitos de Engenharia de Software, será desenvolvido um jogo, com o intuito de tornar o ensino mais lúdico e agradável, portanto, dizemos que os objetivos no geral do nosso produto, é divertir e ensinar. Por limitações de tempo, não poderemos concluir o jogo a ponto dele poder ser jogado pelo público geral, mas somente por nós, que estamos desenvolvendo..

2.2 Funções do Produto

Por se tratar de um jogo, listar as funções é complexo e muito extenso, portanto, trataremos aqui de uma maneira geral, e entraremos em maior detalhe na descrição dos requisitos funcionais sobre cada uma das coisas que o jogo deverá ter como função. O jogo, será no estilo RPGMaker, com visão 2D em perspectiva Pseudo-Isométrica. O jogador deve realizar o Input via teclado, com movimentação quadridirecional.Tudo isso, deve ser animado e com feedback tanto em UI quanto em modificações e colisões no mundo do jogo, assim como o Player deve ser capaz de utilizar itens e equipamentos para a sua vantagem no jogo. Deve possuir um sistema de áudio simples e não-direcional. Deve possuir sistema de interação com NPC's (Dialogo, Combate, Trocas Etc). Possuir ciclos de Dia e Noite e passagem do tempo, além de, afim de incrementar os gráficos, suporte para Shaders.

2.3 Restrições

Temos que, o sistema deve ser capaz ser rodado numa maquina de baixo custo, será usada como base, a maquina das seguintes especificações: I3 de Quarta Geração,8Gb de RAM e uma GPU NVIDIA GTX 750Ti. Além do mais, utilizaremos apenas maquinas com SO's Linux para agilizar no processo de programação e depuração

3 Descrição dos Requisitos Funcionais (RF)

RF01 – Sistema de Renderização

O jogo deve conter um sistema de renderização capaz projetar na tela *sprites* 2D, contendo operações de rotação, redimensão, movimentação e mudança de cor.

RF02 – Sistema de Layers de Renderização

O próprio sistema de renderização deve ser capaz de realizar a Pipeline de renderização levando em conta *Layers* (camadas) a fim de gerar uma ilusão de perspectiva Isométrica.

RF03 – Sistema de *Input* Via Teclado

O jogo deve lidar com o *input* via teclado, deve conter um sistema para lidar com todos os eventos vindos deste (KeyPressed, KeyReleased, etc).

RF04 – Jogador *Offline* e único

O jogo deve ser exclusivamente *offline* e deve possuir apenas um jogador simultâneo. Ou seja, não é necessário a programação de nenhum sistema de conectividade ou autenticação.

RF05 – Sistema de Movimento Quad-direcional

O jogador deve poder se movimentar em apenas 4 direções no mundo, exclusivas (Norte, Sul, Leste e Oeste).

RF06 – Sistema de Animação Baseado em *SpriteSheets*

O jogo deve possuir um sistema de animação funcionando a base de *SpriteSheets*, onde a janela de textura desliza em um atlas para dar a impressão de movimento.

RF07 – Sistema de Suporte Para *Shaders* GLSL

O jogo deve ter suporte para *Shaders* de Fragmento GLSL em pelo menos versão 1.2, a fim de incrementar a qualidade gráfica e se possível, um sistema de iluminação simples.

RF08 – Sistema para o Carregamento de Níveis Via Arquivo

O jogo deve poder carregar os níveis através de arquivos externos que serão armazenados as informações sobre o nível.

RF09 – Objetos Níveis Baseados em *Tiles*

Similar a construção de jogos estilo RPGMaker, o jogo deve ter toda a sua construção de mundo baseado em Blocos (*Tiles*)

RF10 – Sistema de Colisão

O jogo deve possuir um sistema de colisão para o jogador e inimigos, a fim que esses não atravessem objetos que possuem colisão ativa nem paredes.

RF11 – Sistema de UI de Jogo

O jogo deve possuir uma interface durante o jogo contendo informação vital para a jogabilidade. Sendo as informações: Vida, Hora, Local, prioridades em relação as outras.

RF12 – Sistema de Dialogo com leitura de Arquivos Externos

O jogo deve conter um sistema simples de dialogo sem escolhas que faz o uso de arquivos externos para guardar os dados do dialogo (Tal como linhas, Quem está falando, etc.).

RF13 – Sistema de Combate em Turnos

O jogo deve conter um sistema de combate em turnos similar a de RPG's antigos tal como Pokemon ou Final Fantasy.

RF14 – IA de combate para NPC's

O sistema de combate deve ter uma IA para existir de fato o combate entre *Player* e NPC, a IA deve tomar decisões minimamente coerentes e racionais a frente das escolhas no turno dela.

RF15 – Permanência de Dados entre Seções do Jogo (*Sistema de Save*)

Comumente conhecido como Sistema de *Save*, o jogo deve manter o estado do *Player* e a sua progressão entre diferentes seções de jogo.

RF16 – Sistema de Inventário

O jogo deve possuir um inventário de itens no qual o jogador tem acesso e pode mexer em seus itens.

RF17 – Sistema de Equipamento/Armamento

O jogo deve possuir um sistema no qual o jogador pode mexer no seu equipamento e armamento, muito ligado ao inventário mas um sistema separado pois é necessária interação muito próxima com o combate.

RF18 – Sistema de Magia

O jogo deve possuir um sistema para o uso de magias, tanto em combate quanto fora dele. Magias devem usar um sistema de Uso e Descanso, similar ao RPG de mesa DD.

RF19 – Passagem do Tempo

O jogo deve simular a passagem do tempo em seu mundo. Isso inclui um ciclo Dia/Noite e mudanças no mundo

RF20 – IA De Comportamento para NPC's

O jogo deve possuir uma IA de comportamento, isso é, Os NPC's devem andar e interagir com o mundo.

RF21 – Sistema de Audio

O jogo deve possuir um sistema de áudio não-direcional com até 8 canais simultâneos para músicas, efeitos sonoros e barulhos.

Sistema de Interface de Mapa

O jogo deve possuir uma interface para o mapa da área que o jogador se encontra, sendo possível ver marcações em locais importantes no mundo.

RF23 – Sistema de Interface de Combate

O jogo deve possuir uma interface para o combate em turnos, contendo informações cruciais como a vida de ambos os combatentes, opções de ações Etc.

RF24 – Sistema de Comercio entre NPC e *Player*

O jogo deve possuir a capacidade para venda e compra de itens e equipamentos em NPC's especializados espalhados pelo mundo.

RF25 – Sistema de Consumíveis

O jogo deve possuir um sistema de consumíveis, Tais devem também ser considerados itens e pertencer ao inventário do jogador. Exemplo dos consumíveis são Poções, Alimentos e Pergaminhos Mágicos.

4 Descrição dos Requisitos Não Funcionais (RNF)

RNF01 – O jogo deve ser programado em C++.

O jogo deve ser programado em C++ para ter máxima performance.

RNF02 – O jogo deve ter suporte em múltiplos sistemas operacionais.

O jogo deve ter suporte em múltiplos sistemas operacionais para que todos possam jogar.

RNF03 – O jogo deve ser 2D e não utilizar *Engines* prontas.

O jogo deve ser 2D e não utilizar *Engines* prontas para evitar problemas de compatibilidade entre sistemas operacionais.

RNF04 – O jogo deve possuir apenas *input* via teclado.

O jogo deve possuir apenas *input* via teclado por ser muito complexo ajustar para ser possível a utilização de controles.

RNF05 – O jogo deve ser capaz de rodar na máquina descrita no documento.

O jogo deve ser capaz de rodar na máquina descrita no documento e superiores com performance aceitável.

RNF06 – A programação deve ser feita inteiramente em Linux

A programação deve ser feita inteiramente em Linux para agilizar no processo de depuração.

RNF07 – O jogo deve ter seu estilo em PixelArt

O jogo deve ter seu estilo em *PixelArt* para simplificar o trabalho dos artistas envolvidos.

RNF08 – O jogo deve estar em inglês.

O jogo deve estar em inglês para maior acessibilidade do publico geral.

RNF09 – O jogo deve acompanhar um manual escrito.

O jogo deve acompanhar um manual escrito para facilitar o entendimento do jogador em relação as mecânicas presentes no jogo.

RNF10 – O jogo não deve possuir DRM.

O jogo não deve possuir DRM por motivos do grupo ser a favor do software livre.