Virus Project
Manual do desenvolvedor

Antonio Roberto de campos Júnior
Renato Scaroni
Stefano Tommasini

20 de dezembro de 2011

1 Sobre as fases do projeto

O processo de produção do jogo foi feito em quatro fases: primeiras idéias, concept, primeiras implementações e ajustes.

2 Primeiras idéias ou fase 0

2.1 O que foi pedido?

Foi pedido que fosse feito um planejamento preliminar do jogo, ou seja, trazer idéias básicas de gameplay bem como um blog do projeto e algumas telas que mostrassem algo próximo do que poderia ser jogo em sua fase final.

2.2 O que foi feito?

Foi entregue tudo o que foi pedido, através do blog VirusProject. Nele estão algumas das primeiras telas desenhadas para o jogo, das quais algumas foram mantidas e outras foram modificadas em outras fases, mas tratemos disso nos tópicos seguintes.

3 Concept ou fase 1

3.1 O que foi pedido?

Foi pedido que fosse feito uma prova de conceito das idéias que foram pedidas na fase zero.

3.2 O que foi feito?

Foi entregue um aprimoramento das idéias preliminares e um programinha muito simples feito Java e awt apenas para se ter idéia de como ficaria o design do jogo. Nada mais pode ser feito não havia sido definido qual engine deveria ser usada, logo não havia conhecomento para se desenvolver nada.

4 Primeiras implementações ou fase 2

4.1 O que foi pedido?

Foi pedido que fosse feito toda uma implementação do jogo da idéia principal do projeto.

4.2 O que foi feito?

Foi entregue um demo do jogo no qual várias idéias foram descartadas, a começar pelo tabuleiro que foi mudado de um design em perspectiva para um design chapado. Foi nessa fase que foram decididas as EDs do jogo bem como a divisão de classes e a escolha da Slick como a engine a ser utilizada. Foram feitas quatro classes que constituem os elementos do jogo: casinha, vírus, anticorpo e célula, as três últimas herdando da primeira. Além dessas foram criadas mais cinco classes referentes aos estados de jogo.

4.3 Algumas considerações sobre as EDs utilizadas

No programa ultilizamos uma matriz como tabuleiro, cujas posições consistiam em objetos da classe casinha (detalhes das classes mais adiante). Além dessas existem as classes virus e anticorpo que extendem casinha de forma que cada quadradinho do tabuleiro pode ser um virus ou um anticorpo. Como essas entradas da

matriz recebem apenas referencias das casinhas foram implementados dois arrays que servem como pilhas e gurdam os virus ainda vivos e outro faz a mesma função mas para os anticorpos.

4.4 Algumas considerações sobre a engine utilizada

A engine escolhida para o desenvolvimento do projeto foi a Slick, uma engine para jogos 2D em linguagem Java. Essa engine trabalha com estados do programa, sendo assim foram criados 5 estados: GameState, InfoState, InstructionsState, MenuState e State Manager. Todas essas classes tem estrutura básica em comum, elas contém três funções principais:

getID retorna um inteiro que é o ID do estado de jogo

init Inicializa as variáveis do estado

Render Desenha os elementos gráficos na tela

Update Recebe entradas e corresponde ao loop principal do estado

5 Ajustes ou fase 3

5.1 O que foi pedido?

Relatórios juntamente com a versão final do jogo

5.2 O que foi feito?

Foi entregue o jogo, esse manual e o de usuário.

6 Implementação

Agora passaremos a expor rapidamente as principais classes implementadas e seus principais atributos e métodos.

6.1 casinha

Entradas do tabuleiro.

6.1.1 Atributos

int tipo tipo do objeto guardado

int localizacao sua localização no array de virus ou de anticorpo, dependendo do tipo guardado

int ultimo_turno Ultimo turno em que foi usado

6.2 virus

Unidade que o jogador controla.

6.2.1 Atributos

int vida auto-explicativo

int ataque auto-explicativo

6.2.2 Métodos

public void atacar(anticorpo a, casinha M[[], int selx, int sely, int ax, int ay)] auto-explicativo

6.3 Anticorpo

Unidade inimiga.

6.3.1 Atributos

int vida auto-explicativo

int ataque auto-explicativo

boolean flag indica se o anticorpo esta agindo ou não

6.3.2 Métodos

public void atacar(anticorpo a, casinha M[[], int selx, int sely, int ax, int ay)] auto-explicativo

public void atacar_base (celula C) auto-explicativo

public void interage (casinha M[[], int x, int y, virus V[], int Cx, int Cy, celula C)] IA

private int abs(int i) Retorna o valor absoluto de um inteiro

6.4 GameState

Unidade inimiga.

6.4.1 Atributos Principai

static int n = 10; numero de linhas

static int m = 15; numero de colunas

int selx, sely; quadrado selecionado

int ax, ay; bixinho selecionado

public casinha[[] M = new casinha[30][30];]matriz do tabuleiro

public virus[V = new virus[1000]; int topoV;]array de virus

public anticorpo[A = new anticorpo[1000]; int topoA;]array de anticorpos

public celula C = new celula(); celula

Image land = null;

Image sair = null;

Image virusmg = null;

Image anticorpomg = null;

Image hemaciamg = null;

```
Image bordamg = null;
Image ganhoumg = null;
Image perdeumg = null;
Image botoes = null;
Image selecionar = null;
Image atacar = null;
Image mover = null;
Image encerrar = null;
Image voltar = null;
public int tempo;
float largura = 42*1.2F+0.25F;
float inix = 135;
float iniy = 95;
float altura = 51.38F;
boolean player = true;
boolean flag = false;
boolean atacando = false;
boolean acabou;
boolean ganhou;
boolean perdeu;
boolean exit;
boolean menu_sair;
boolean virusCausouDano;
```

boolean anticorpoCausouDano;

int option;

int cont;

int ultimo;

int PAUSA;

int Cx, Cy; localizacao da celula

int stateID = 0;

casinha aux;

java.util.Random rand = new java.util.Random **();**

Sound musica = null;

6.4.2 Métodos

private int achaMouse(Input input) Capta posição do mouse

private void turnoPC() auto-explicativo

private void turnoJogador (GameContainer gc, StateBasedGame sb, int delta)
Auto-explicativo

private int abs(int i) Retorna o valor absoluto de um inteiro