UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL
CIÊNCIA DA COMPUTAÇÃO
TÓPICOS EM COMPUTAÇÃO

LEANDRO SOUZA DA SILVA
NIELSON FERNANDES SILVA

REMAKE SUPER BOMBERMAN 5

DOURADOS, MS

2014

UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL

LEANDRO SOUZA DA SILVA

NIELSON FERNANDES SILVA

REMAKE SUPER BOMBERMAN 5

Trabalho de semestre apresentado para avaliação do Curso de Ciência da computação, Nível Superior, da Universidade Estadual de Mato Grosso do Sul, Dourados - MS

DOURADOS, MS

2014

SUMÁRIO

RESUMO

Este resumo trata de um *remake* do jogo *Super Bomberman 5* da Nintendo para o console super snes produzido pela Hudson Soft em 1997.

INTRODUÇÃO

Um *remake* de um jogo tem como objetivo principal ser uma cópia de um jogo original, com finalidades das mais diversas, nos atemos aqui a entender como se pode construir um jogo 2D com algumas técnicas e ferramentas de alto nível.

Planejamento do jogo

O planejamento do remake foi feito a partir do levantamento de requisitos que posteriormente foi modelado em um diagrama de classes, isso deu a base para que o jogo fosse implementado. Nessa etapa também fez-se a escolha das ferramentas de trabalho que deu a base para construção do jogo, são elas: PyGame e Gimp. Essa tarefa teve um custo de 4 semanas.

Metodologia usada na implementação

Após o planejamento e a modelagem do sistema, o *remake* passa para a fase de implementação. O jogo está composto da seguinte forma:

- O arquivo Setup_Bomber_Windows cria um executável e as blibliotecas necessárias para execução no windows.

- O arquivo Cenario_1 cria um cenário e seta algumas opções como nome de personagens, lista de comandos, nome dos sprites e etc.

- O arquivo Objetos.py é composto por diversas classes que dão base ao jogo, a classe Objeto é a principal, ObjetoAnimado herda Objeto, as demais classes herdam de Objeto ou de ObjetoAnimado, são elas: Tijolos, Explosão, Poderes, Bomba, Persoangem. As classes desse arquivo implementam suas ações e animações, que dão vida ao jogo.

- O arquivo Gestor_Inicializacao.py contém a classe ProcessadorInicialDeJogo, essa classe da suporte para uma inicialização do jogo, baseando-se na entrada de um arquivo de texto pré-configurado. Ela contém métodos que retornam uma matriz de listas contendo todos os objetos do jogo.

- O arquivo Gestor_Entrada.py contém a classe GestorEntrada, que espera eventos e os captura, mantendo-os numa lista, essa lista pode ser retornada por um de seus métodos, que posteriormente pode conter comandos referentes a um dos personagens do jogo.

- O arquivo Gestor_Colisao.py contém a classe GestorColisao que possui métodos para verificar se um dado objeto colidiu com outro objeto, parede, bomba, e etc.

- O arquivo Gestor_Regras.py contém a classe GestorRegras que é responsável por estabelecer as regras do jogo, gerindo todos os objetos que há no cenário e ativando suas animações quando uma regra os contempla.

- O arquivo Renderizador.py contém a classe ProcessadorGrafico que é responsável por toda a parte gráfica do jogo, a parte que lida com a API PyGame, ela possui métodos para controlar o timeClock do jogo, para renderizar uma dada imagem que lhe é passada, para atualizar a tela e outros.

- O arquivo Menu.py contém apenas métodos, esse arquivo contém a definição da main, onde todos os objetos necessários para o jogo funcionar são criados, contém também o loop principal do jogo, onde tudo acontece.

- Este projeto conta ainda com um conjunto de *sprites,* que são imagens que ajudam na animação dos objetos animados e inanimados e na construção do cenário como um todo.

A tarefa de implementação custou 4 semanas.

REFERÊNCIAS

http://www.pygame.org/docs/