UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

PEDRO ARTHUR BOHN
PEDRO HENRIQUE RAMAZZINI FERREIRA
SÉRGIO FERNANDO STRAZZABOSCO NETO

PROJETO FINAL DE ALGORITMOS DE PROGRAMAÇÃO

Prof. Renan Maffei
Prof. Eder Scheid

Porto Alegre
2024

O relatório a seguir se trata do projeto final da cadeira de Algoritmos de Programação, lecionada pelo professor Renan Maffei e pelo professor Eder Scheid.

Nosso projeto consiste de um jogo eletrônico no estilo Tower Defense, programado em C, na IDE do programa CodeBlocks. Juntamente com o especificado, também foi utilizada a biblioteca gráfica "*Raylib.h*" para melhores resultados visuais.

A modalidade Tower Defense consiste em um jogo em que o jogador pode andar livremente por um mapa em que um número de inimigos andam por um caminho determinado e seu alvo é a base do jogador. Para contra-atacar, o jogador pode coletar recursos espalhados pelo mapa e posicioná-los no caminho que os inimigos percorrem. Quando atingem um recurso, os inimigos perdem uma vida e, essencialmente, morrem. Se atingirem a base, porém, morrem e reduzem a vida da base. A partida é perdida se a base perder todas suas vidas ou o jogador perder as suas. A partida é ganha se os inimigos restantes na tela forem eliminados.

O programa foi separado em algumas partes como:

- Leitura do mapa, jogador, inimigos, recursos e buracos;
- Movimentação do jogador e dos inimigos incluindo colisões;
- Coleta e posicionamento de recursos;
- Uso de buracos;
- Vidas do jogador, base e etc;
- Aplicação de salvamento;
- Textura, sons e textos.

Em seguida será apresentada uma suma de partes das funções.

A leitura do mapa foi desenvolvida com base na leitura de um arquivo de texto 30X60. O programa irá ler caractere por caractere e, dependendo de qual é, irá ou desenhar uma parede no mapa, ou irá salvar posições do jogador, inimigos ou recursos etc.

Quando o programa encontra o jogador, algum inimigo ou recurso no mapa, ele irá salvar os dados de posição no mapa dentro de uma struct específica do tipo "JOGADOR" ou "INIMIGO".

A movimentação do jogador é baseada puramente na interação com o usuário. Dependendo da tecla pressionada, um multiplicador de deslocamento "dx" ou "dy" irá somar ou subtrair do valor da posição atual do jogador. Quando encontra uma parede, o jogador não consegue atravessá-la por causa de um condicional adicionado nas condições necessárias para "dx" e "dy" assumirem um valor diferente de 0. Já os inimigos necessitam de um algoritmo que inicia seu deslocamento e, dependendo de onde houver paredes, o deslocamento é alterado. Por exemplo, se o inimigo estiver se movendo para a esquerda e encontrar uma parede, ele para de se mover

horizontalmente e verifica se possui paredes em cima e embaixo de si. Caso não haja, o inimigo assume tal deslocamento.

A mecânica de buracos funciona da seguinte maneira. Primeiro, o programa confere se a posição atual do jogador é a mesma de um buraco. Se isto for verdadeiro, o programa procura um buraco no mesmo eixo do deslocamento do jogador e altera a posição do jogador para tal buraco.

Dentro das estruturas de jogador e inimigo há a variável de vidas. Dependendo da colisão com inimigos, recursos ou bases, as vidas diminuem e, se chegarem a 0, o inimigo deixa de existir.

O salvamento vai sempre criar um arquivo do tipo binário chamado “TOWDEFSAVEX.bin”. Dentro deste, temos especificações do tipo qual o mapa jogado, struct de informações atuais do jogador e structs de informações atuais dos inimigos. Quando o usuário abre este save, o programa lê todas as informações e abre uma espécie de “novo jogo” mas agora com informações pré existentes.

As texturas foram aplicadas utilizando a “*Raylib.h*” que conta com inúmeras funções de apresentação gráfica, sonora e até de IHM como teclado e mouse. Também, as texturas são obras originais dos DEV’s feitas em pixel art. Os textos também utilizam a biblioteca e são apresentados através de funções desta.