**Grupo:**

Alan Vitor Costa de Oliveira

Canoi Gomes de Aguiar

**Projeto:**

Jogo 2D de plataforma no estilo runner com ranking de maior pontuação entre os jogadores.

**Sumario Executivo:**

O objetivo do jogo é permitir que o jogador controle um personagem que estará correndo em uma fase “infinita”. Quanto mais longe o jogador chegar na fase, mais pontos ele vai ter. No final, o jogador pode enviar sua pontuação para o ranking, e compará-la a dos outros usuários. O jogo será tematizado na era medieval, onde um guerreiro foi salvar a princesa, e quando alcançou o objetivo, acabou despertando o dragão. Incapaz de lutar, ele se viu obrigado a fugir com a princesa em seus braços, enquanto o dragão os perseguia.

Aqui usuário e o cliente serão a mesma pessoa, no caso o jogador que for adquirir o jogo para a sua própria diversão.

As principais contribuições do jogo é gerar entretenimento ao usuário e estimular seu espirito competitivo de maneira saudável e equilibrada. Além da competição queremos que o usuário ultrapasse seus próprios limites, aumentando sua concentração, que deve estar sempre focado para não colidir com os obstáculos e morrer, e também desviar dos ataques do dragão para garantir uma boa pontuação.

**Requisitos funcionais:**

Acessar ao menu de controle:

Mostra ao jogador o menu de controles. Onde ele poderá visualizar os comandos usados para controlar o personagem.

Acessar ao menu do ranking:

Leva o jogador a uma tela mostrando o ranking organizado com o nome dos jogadores seguido de suas pontuações, na ordem da maior pontuação para a menor. O nome dos jogadores são limitados a terem apenas três letras, assim como nos fliperamas.

Acessar o jogo:

Inicia a partida, a fase é gerada e permite ao jogador controlar o personagem.

Enviar nick:

Caso o jogador morra, entrará em uma tela e ele dever informar seu nick para que seja levado ao ranking.

Buscar no ranking:

Essa opção permitia ao jogador buscar por nomes de jogadores na lista do ranking, revelando a pontuação do jogador pesquisado ou revelando uma mensagem de erro caso não exista jogador com o nome procurado. Mas resolvemos retirar esta opção pois decidimos que o ranking irá possuir apenas a pontuação de 10 jogadores que serão postadas todas de uma vez na tela de ranking.

Movimentação:

O jogador pode apertar a tecla direcional para cima para pular.

Coletar itens:

Decidimos retirar esta opção pois resolvemos que a pontuação seria dada pela distância percorrida pelo jogador e por isso não existe mais utilidade para a coleta de itens.

Loop gravitacional:

Caso o jogador entre em um buraco na fase, ele ira atravessar o cenário no eixo y.

**Requisitos não funcionais:**

Allegro5:

Biblioteca usada para criação de jogos.

C++:

Linguagem de programação escolhida para o projeto.

Tileset:

Uma imagem contendo partes do cenário que serão montados através de uma matriz.

Sprites:

Imagem com as animações do personagem.

Sistema:

O jogo será feito para Windows.

Movimentação:

O mapa se movimenta para trás constantemente dando a ilusão de que o personagem está correndo para frente.

Tela inicial:

Tela que mostra ao jogador o título do jogo, e os botões para acessar os menus.

Tela de ajuda:

Tela que mostra ao jogador os controles do jogo e os créditos.

Tela de ranking:

Tela que exibe o ranking atual ao jogador.

Tela de jogo:

Mostra o cenário, os obstáculos, o contador da pontuação e o personagem.

Tela de morte:

Mostra uma lapide na qual o jogador precisa escrever seu nick no jogo para ser enviado ao ranking.

Tela de história:

Mostra uma animação contando a história do jogo.

**Estruturas estudadas em sala de aula:**

1. Busca sequencial.
2. QuickSort
3. Fila.

a) Busca Sequencial: Antes iria ser usado no ranking, para buscar o nome do jogador e atualizar a pontuação dele, mas nós decidimos deixar nomes repetidos, como nos fliperamas. Agora nós usamos a busca na hora de testar se está havendo colisão entre o personagem e algum bloco de chão que está dentro de uma matriz de objetos Blocos.

Complexidade: A complexidade será O(n²) no pior caso já que ele sempre tentar percorrer toda a matriz, porém se ele achar a colisão entre o personagem e o bloco, ele quebra o laço.

Corretude:

b) QuickSort: Escolhemos utilizar o QuickSort, pois iremos ordenar a lista do ranking de acordo com as pontuações dos jogadores. Quando um novo jogador enviar a pontuação, nós iremos comparar se a pontuação dele é maior que pontuação do ultimo jogador do ranking, caso for, colocamos a pontuação dele na ultima posição, e podemos usar ele como pivô no Quick para ordenar o vetor,e acharmos a posição a qual ele pertence.

Complexidade: No melhor caso será (n.logn) e no pior caso será (n²).

Obs: Porém nós nunca teremos pior caso, já que o vetor sempre estará ordenado quando for lido, então só precisamos jogar a ordenação, para o último elemento do ranking(Caso ele tenha entrado agora) encontrar a sua posição.

Corretude:

Inicialização:

c) Fila: Para criar um mapa “infinito” usaremos uma fila de matrizes e desenharemos o mapa nessas matrizes através de um Tileset, guardaremos alguns padrões de matrizes e iremos adiciona-las na lista de forma aleatória para que os mapas raramente se repitam.

Complexidade:

Inserção: O(1)

Remoção: O(1)

Corretude: