UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INF - 2014/1

FUNDAMENTOS DE COMPUTAÇÃO GRÁFICA
RELATÓRIO TRABALHO FINAL

Nome: Adolfo Henrique Schneider Cartão: 208154

1. Introdução

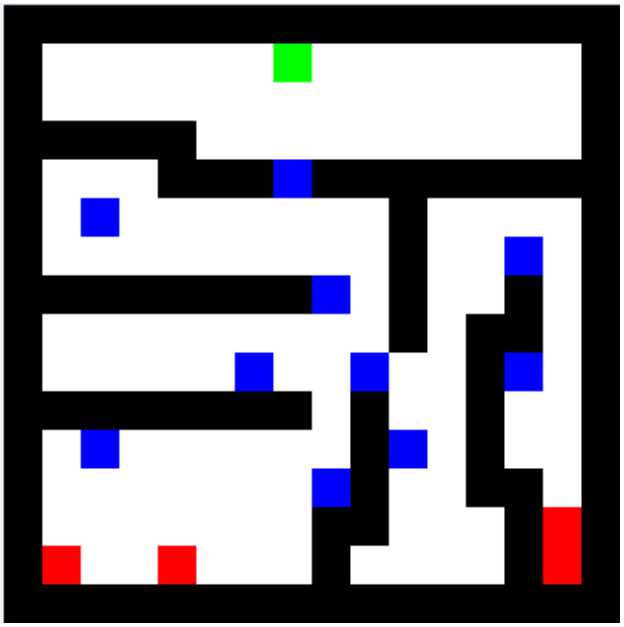
A proposta desse trabalho é desenvolver um jogo em 3D usando OpenGL puro, sem qualquer tipo de auxílio através de frameworks de desenvolvimento. Para isso, é utilizado a GLUT, um toolkit para OpenGL.

Dessa forma, temos as ferramentas necessárias para botar em prática a criação da viewport principal, efetuar a modelagem/carregamento da cena, aplicar texturas e iluminação, trabalhar a lógica do jogo (condições de término, etc.) e detectar colisões dos personagens entre si e com os objetos e limites do mundo.

2. Desenvolvimento

2.1 Modelagem/carregamento da cena

O mapa é montado através de um bitmap de 16x16 pixels onde pixels brancos representam espaços livres, pixels pretos representam as paredes (blocos indestrutíveis e fixos), pixels azuis representam blocos móveis, pixels vermelhos representam inimigos e o pixel verde representa o jogador. Nessa primeira inicialização são carregados os modelos e posteriormente as texturas.



A renderização da cena é separada em 3 métodos, o render do chão que printa um .bmp 2D, o render do Map que é encarregado de mostrar todos os modelos 3D e por último o render do minimap que implementa os 2 renders já citados porém em uma viewport redimensionada.

2.2 Texturas e Iluminação

A iluminação utilizada é simples. O método initLight() implementa uma iluminação ambiente de forma que esse é chamado na inicialização do jogo.

Para carregar as texturas cada objeto (MotionBlock, Player, Wall, etc) tem um método loadTexture quecarrega uma textura .bmp. Optei por uma maneira diferente de carregar as texturas devido alguns problemas que encontrei com os exemplos passados em aula. A referência para as texturas é guardada dentro do vetor textures[].

2.3 Lógica do Jogo

A lógica do jogo foi desenvolvida separando cada artefato do jogo em objetos que extendem uma classe Block. Os blocos móveis possuem uma flag isFixed que indica se ele pertence ao mapa ou foi gerado pelo jogador. Quando é criado um novo bloco, inicialmente é criado um objeto TempMotionBlock que será exibido por no máximo 2 secundos. Os blocos gerados pelo jogador são gerenciados em dois vetores, um só para os TempMotionBlocks e outro para todos os MotionBlocks criados.

Os itens são destribuidos randomicamente entre os MotionBlocks assim que o mapa é iniciado. Uma flag hasItem é setada e tipo do item é indicado em um valor inteiro atribuido ao bloco móvel. Assim que o bloco é destruído, o item do tipo respectivo é criado na posição onde este ocupava.

O jogo tem dois casos de término. Um onde todos os inimigos são destruídos (jogador venceu), ou seja, a lista de inimigos é vazia. Ou então, quando o jogador foi atingido por algum inimigo (jogador perdeu). Nesses casos a glut continua o seu looping principal e as porém as mensagens de término são apresentadas.

2.4 Detecção de Colisões

A detecção de colisão é implementada pela classe Block, portanto todos as outras classes que são blocks tem a detecção de colisão facilmente detectada através do método collidesWith, onde um outro objeto Block é passado como parâmetro. Tal método verifica se ocorreu colisão no eixo z ou x.

Assim que o jogador preciona a tecla para efetuar um movimento para frente ou para trás, o movimento é feito (logicamente falando) e é verificado se o player colidiu com alguma parede, bloco ou inimigo. Se a colisão for verdadeira, o movimento é desfeito. Caso contrário, a nova posição é consolidada e o jogador é desenhado na tela.

3. Imagens