Trabalho Prático 1
Breakout

Arthur Souto Lima

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

arthursl@ufmg.br

1. Instruções

O programa foi implementado em C++, compilado pelo compilador G++ da GNU Compiler Collection. Juntamente com os códigos-fonte, está incluso também um arquivo *makefile* que é utilizado para compilar o programa através do comando *make*. O projeto utilizou OpenGL através da biblioteca *freeglut*, bem como da biblioteca *GLU*. Assim, a compilação no makefile é feita com as flags de linker para essas três bibliotecas.

Além disso, convém ressantar que o programa foi implementado num ambiente Linux e está adaptado para tal, sobretudo no que tange à questões de compilação e de bibliotecas. Seu funcionamento correto no Windows não é garantido.

1.1. Breve demonstração

Uma breve demonstração do projeto pode ser vista no Youtube através do link:

`https://youtu.be/KLxIGEYJc9A`

2. Implementação

2.1. Concepção

Como foi requisitado na implementação, o projeto deveria ser um jogo semelhante ao clássico Breakout, utilizando como pacote gráfico o OpenGL. Assim, para auxiliar no desenvolvimento, foi concebida uma ”mini-engine” para manejar os objetos da cena, bem como suas interações tanto entre si como com o jogador. Temos basicamente um único estado, que é o estado de jogo com a fase carregada. Por isso, a classe *Level* é quem controla e gerencia todo o jogo, basicamente.

Considerando que a função de display seria chamada a cada frame para redesenho de cada quadro, ela basicamente chamava dois procedimentos do *Level*: um de desenho (*draw*) e um de atualização (*update*). O primeiro abarcava basicamente a renderização dos objetos na tela e o segundo englobava alguns métodos de movimentação, bem como gerenciava as colisões e avaliava o estado do jogo (se, por exemplo, ocorreu game over ou a bolinha saiu dos limites).

Os objetos que vemos na cena advêm de uma mesma interface *GameObject*, que possui uma posição (coordenadas), uma largura e um comprimento, além de um método que detecta colisão com um outro objeto fornecido e um outro que contém as funções de desenho deste objeto. Assim, no método *draw* de *Level*, deveríamos chamar as funções *draw* dos objetos que desejássemos renderizar na tela.

O sistema de colisão implementado é a colisão AABB simples, que verifica contato com base nas coordenadas e no tamanho dos objetos. Desse modo, esse método possui algumas limitações, sobretudo quando a colisão é perfeitamente diagonal. Apesar disso, a maior parte das colisões no jogo durante a execução se mostram bem definidas.

2.2. Funcionalidades Básicas

As funcionalidades básicas são aquelas ditas obrigatórias na especificação. O cursor do mouse controla a velocidade do paddle: mouse no meio indica velocidade zero, mouse na direita ou na esquerda indica velocidade positiva ou negativa, respectivamente. Para facilitar deixar o paddle parado numa posição específica, ou seja, deixá-lo com velocidade zero, há uma pequena tolerância no centro da tela (configurável por constantes).

O botão direito do mouse pausa o jogo a qualquer momento e é, inclusive, usado para começar o jogo e passar de fase, quando todos os blocos são quebrados. O botão esquerdo, com o jogo em curso, mostra os dados básicos dos principais elementos na tela e pausa o jogo. Nesse estado, o botão direito funciona como um ”passo”. Apertando o botão esquerdo novamente o jogo sai da pausa, mas mantém as informações. Apertando-o mais uma vez, as informações desaparecem.

No teclado, a tecla Q finaliza o programa, enquanto a tecla R cria um novo level.

A bolinha pode ser rebatida nas paredes laterais, bem como no teto. Se ela cair abaixo da altura do paddle, o jogador perde uma vida. As vidas indicam quantas bolas o jogador ainda tem disponível. Além disso, a bolinha pode ser rebatida pelo paddle no intento de quebrar os blocos. Cada bloco, ao ser quebrado, rebate a bola, além de adicionar 25 pontos no placar do jogador. Preferiu-se colocar uma contagem de pontos ao invés de simplesmente a quantidade de quantos blocos já haviam sido destruídos tanto para dar mais imersão ao jogo quanto para propiciar que seja possível, como uma proposta futura, adicionar blocos especiais dourados, por exemplo.

Tanto o placar quanto as vidas do jogador ficam no superior da tela, mas não interagem (colidem) com outros objetos. Há ainda um display que indica em que sentido está a aceleração do paddle, com base na posição do mouse.

2.3. Constantes

Como a maioria dos jogos, dentre os parâmetros que pudessem variar durante a partida há limites superiores e inferiores para tal, a fim de propiciar uma melhor experiência ao jogador e facilitar ao desenvolvedor alterar nuances do jogo. Tais restrições estão definidas como constantes num arquivo de cabeçalho. Dentre elas estão velocidades máximas e mínimas, tamanho de objetos e espaçamento entre os blocos, por exemplo.

2.4. Funcionalidades Adicionais

Como indicado pela especificação, dever-se-ia implementar funcionalidades além das básicas. Aqui estão listadas as funcionalidades escolhidas no projeto:

- **Power-Ups Coletáveis**: quando um bloco é quebrado há uma chance de spawnar um power-up próximo do local do bloco. Eles descem com uma velocidade incrementalmente maior e devem ser pegos pelo paddle para serem ativados. Se ele cair da tela, ele despawna.

- **Level Procedural**: a disposição dos blocos gerada a cada level é procedural. A quantidade de linhas e de colunas é aleatória dentro de limites mínimo e máximo definidos. As cores em cada linha podem ser uma única ou duas alternadas. Uma linha também pode ter o padrão com lacunas. Uma mesma linha pode ter linhas alternadas e lacunas.
- **Efeito Space Invaders**: em alguns levels (aleatoriamente) os blocos se moverão em conjunto, como no clássico jogo Space Invaders. Eles se movem horizontalmente até uma parede, quando descem um pouco e seguem na outra direção horizontal.
- **Rebatida com Efeito**: a posição e velocidade do paddle influenciam no efeito da bola, que pode ser acelerada ou desacelerada dependendo disso.
- **Velocidade Crescente**: a cada bloco quebrado, há uam chance da velocidade da bolinha aumentar levemente. Isso pode se acumular e deixar o jogo cada vez mais difícil quanto mais blocos o jogador conseguir destruir.

2.4.1. PoweUps

Como dito acima, alguns blocos ao serem quebrados podem spawnar um power-up. O power-up gerado é um dos descritos abaixo, escolhido aleatoriamente. Se já existe um power-up spawnado, não se pode spawnar outro. Contudo, nada impede que, se um power-up estiver ativo, isto é, o jogador pegou ele com o paddle, spawne outro. Não podemos ter dois spawnados descendo na tela concomitantemente. A seguir, listam-se os powerups implementados:

- **Vida Extra**: o jogador ganha uma vida a mais.
- **Paddle Maior**: o paddle do jogador fica maior, o que facilita evitar que a bola caia.
- **Paddle Menor**: analogamente ao anterior, o paddle fica menor, dificultando consideravelmente o jogo.
- **Duas Bolinhas**: spawna uma segunda bolinha nesse mesmo jogo, que começa a viajar imediatamente ao power-up ser pego. Como uma bolinha normal, se ela cair da tela, o jogador perde uma vida. Por outro lado, com duas bolinhas ele consegue quebrar os blocos mais rápido. O caos gerado aumenta consideravelmente a dificuldade, mas também a recompensa, desse power-up

No caso dos power-ups que alteram o tamanho do paddle, o efeito só passa no level seguinte ou então se pegarmos o power-up oposto. No caso do das duas bolinhas, quando a segunda bolinha cair ou o usuário passar de level o power-up é desativado.