Trabalho Prático 2
Boids

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, para tratar dos quaternions bem como de operações com álgebra vetorial, usou-se uma simples biblioteca de álgebra linear chamada GMath [1].

Além disso, convém ressaltar 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/cmkpH2gEZV8`

2. Implementação

2.1. Concepção

O objetivo do projeto é implementar um bando de pássaros, ditos boids, que possuem tanto um comportamento local de alinhamento, coesão e separação com relação aos boids ao redor dele, bem como um comportamento global em que o bando deve seguir o boid líder e desviar de obstáculos. As implementações utilizadas foram baseadas principalmente em [Parker 2007] e [Eater 2020].

Tal como no Trabalho Prático anterior, foi concebida uma simplificada engine gráfica para coordenar todos os objetos da cena, ou seja, os boids, o boid líder controlado pelo usuário e os obstáculos. Essa engine está centrada na classe *World*.

Valendo-se do callback de visualização do OpenGL para desenhar cada frame, faz-se duas chamadas. Uma é para atualizar os objetos, a qual muda as posições com base nas velocidades, computa colisões e calcula as variações de velocidades dos boids do bando. A outra é para desenhar os objetos em suas respectivas posições.

[1] `https://github.com/YclepticStudios/gmath`

Sobretudo para a renderização dos boids, foi vantajoso o uso de quaternions para armazenar a pose corrente, bem como fazer as rotações seja com base na alteração de um vetor (caso dos boids comuns) ou com base em um eixo (caso do boid líder). Essa pose também era útil para acumular as rotações e rotacionar o objeto a ser desenhado, que era renderizado, inicialmente, na origem.

Os objetos que vemos na cena advêm de uma mesma interface *GameObject*, que possui basicamente uma posição (coordenadas), além de um método que contém as funções de desenho e outro com as funções de *update*. Assim, no método *draw* de *World*, deveríamos chamar os métodos *draw* dos objetos que desejássemos renderizar na tela. Um processo análogo ocorre no método *update* de *World*.

2.2. Funcionalidades Básicas

As funcionalidades básicas são aquelas ditas obrigatórias na especificação.

2.2.1. Controles do Usuário

Para alternar entre as três câmeras descritas, usa-se os números 1, 2 e 3, para, respectivamente, a câmera do alto da torre (inicial por padrão), a câmera atrás do líder e a câmera lateral do líder.

Para o controle do líder, Q e E controlam o Yaw, enquanto W e S controlam o Pitch. As teclas A e D controlam o roll deste boid.

Para alterar a quantidade de boids, aperta-se o ”+” para adicionar um boid numa posição próxima ao bando. Além disso, a tecla ”-” retira aleatoriamente um boid, sendo que sempre há pelo menos um boid comum além do líder.

2.2.2. Bando

Para fazer a simulação do bando, seguindo as diretivas de alinhamento, de separação e de coesão, uma classe Bando, que gerencia os boids, a cada quadro, calcula uma pequena alteração de velocidade para cada boid e soma essa variação à velocidade do boid (que tem um módulo limite, naturalmente), com base em cada um dos critérios. A ideia foi bastante inspirada nos pseudocódigos de [Parker 2007].

2.3. Constantes

Dentre os parâmetros que pudessem variar durante a simulação há limites superiores e inferiores para tal, a fim de propiciar uma melhor experiência ao usuário e facilitar ao desenvolvedor alterar algumas nuances. Tais restrições estão definidas como constantes num arquivo de cabeçalho. Dentre elas estão, por exemplo, o módulo máximo de velocidade e os fatores de cada regra do bando na alteração da velocidade do boid.

2.4. Funcionalidades Adicionais

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

- **Bater das Asas**: os boids tem uma animação de ”bater de asas” que é independente para cada boid.
- **Fog**: a tecla F ativa/desativa uma névoa no mundo
- **Obstáculos**: no mundo, há alguns obstáculos os quais os boids do bando procuram evitar, mesmo que isso cause desrespeito às outras regras de comportamento do bando. Esses obstáculos são tipicamente, esferas e torres (cilindro com um cone no topo).
- **Reshape**: a janela do programa pode ser redimensionada pelo usuário sem que se altere o *aspect ratio*. Essa funcionalidade foi inspirada na implementação de [Manssour 2003].
- **Modo Debug**: ativado pela tecla P, mostra alguns vetores importantes e possibilita alterações no comportamento do bando em tempo de execução.
- **Zoom Câmera**: todas as três câmeras podem ter um aumento/redução de zoom, utilizando os botões do mouse, para que seja possível observar mais detalhes de cada ponto de vista. Também inspirado no disponível em [Manssour 2003].

2.4.1. Modo Debug

Há uma pequena implementação de um modo depuração, ativado/desativado com a tecla P. Quando ativo, ele mostra os vetores velocidade de cada um dos boids, além do sistema de coordenadas do boid líder.

Nesse modo debug, além disso, no terminal, são apresentados os valores atuais do campo de visão dos boids, fator de velocidade local e fator de separação, que podem ser alterados com as teclas T/G, Y/H e U/J, respectivamente, sendo que a primeira aumenta e a segunda diminui. Naturalmente há limites para esses fatores para que haja uma mínima coerência do comportamento do bando. Isso permite modificar o comportamento dos boids durante a execução.

References

Eater, B. (2020). Boids algorithm demonstration. [2]. Acesso em: 15 out 2020.

Manssour, I. H. (2003). Introdução à opengl: 14.programando em 3d. [3]. Acesso em: 15 out 2020.

Parker, C. (2007). Boids pseudocode. [4]. Acesso em: 15 out 2020.

[2] https://eater.net/boids

[3] https://www.inf.pucrs.br/~manssour/OpenGL/Programando3D.html

[4] http://www.kfish.org/boids/pseudocode.html