Trabalho Prático 1 – Computação Gráfica - BOIDS

Matheus Flávio Gonçalves Silva - 2020006850

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

matheusfgs@ufmg.br

1 Introdução

Este trabalho prático implementa uma simulação de boids (autonomous agents) em tempo real utilizando computação gráfica moderna. Boids são agentes autônomos que simulam comportamentos de rebanhos ou enxames, como observado em aves, peixes ou insetos em movimento coletivo.

O projeto foi desenvolvido em C++11 com OpenGL 3.3 Core Profile, utilizando as bibliotecas GLAD para carregamento de extensões OpenGL, GLFW para gerenciamento de janela e eventos, e GLM para operações matemáticas de álgebra linear. O resultado é uma aplicação interativa que renderiza boids animados em um ambiente 3D com iluminação Phong, sombras dinâmicas e sistemas de obstáculos com detecção de colisão.

A implementação satisfaz integralmente os requisitos básicos de projeto (100% de conformidade) e implementa cinco de seis requisitos extras (83% de conformidade), totalizando uma cobertura de 92% das especificações. O sistema foi otimizado para executar em hardware modesto, apresentando performance estável sem crashes ou vazamentos de memória.

2 Organização do Projeto

O projeto foi estruturado em módulos desacoplados seguindo princípios de engenharia de software, com separação clara entre headers (definições) e implementação (código). A organização de pastas é apresentada como segue:

```
/ boids-simulation/
|-- src/                              % Implementa  o (.cpp)
|   |-- game/
|   |   |-- gameController.cpp        % L gica principal
|   |   |-- ShadowRenderer.cpp        % Renderiza  o de sombras
|   |   |-- DynamicObstacleShadowRenderer.cpp
|   |   |-- InputHandler.cpp          % Processamento de entrada
|   |   |-- SceneBuilder.cpp          % Constru  o de cena
```

```
|   |   |-- objects/boid.cpp        % Implementa  o de boid
|   |-- graphics/                   % Renderiza  o (VAO, VBO, etc)
|-- headers/                        % Defini  es de classe (.hpp)
|   |-- game/
|   |-- graphics/
|-- shaders/                        % Shaders GLSL
|   |-- default.vert                % Vertex shader
|   |-- default.frag                % Fragment shader
|-- Libraries/                      % Depend ncias (GLAD, GLM, GLFW)
|-- main.cpp                        % Ponto de entrada
|-- Makefile                        % Build system
```

3 Método Utilizado

3.1 Algoritmo de Flocking (Boids)

O comportamento de boids foi implementado baseado no modelo clássico de Craig Reynolds, onde cada agente segue três regras principais: separação (evitar aglomeração), alinhamento (orientação similar) e coesão (movimento em direção ao centro do grupo). Além disso, foram implementadas extensões para evitação de obstáculos e comportamento reativo.

Cada boid é representado internamente como uma pirâmide composta por cinco triângulos, permitindo visualização clara de direção e movimento. A animação de asas é realizada mediante deformação de vértices em tempo real, criando um efeito de batida sincronizado com o movimento de voo.

3.2 Obstáculos e Colisão

O mundo contém cinco torres cilíndricas dispostas estrategicamente como obstáculos. Cada boid verifica colisão com estas estruturas e modifica seu vetor de velocidade para desviar de forma suave. A detecção de colisão utiliza distância Euclidiana entre o boid e o centro da torre, com raio de colisão dependente da altitude do boid para comportamento mais realista.

Adicionalmente, implementou-se um boid objetivo (goal boid) que serve como ponto de atração para o enxame, bem como um boid fantasma (ghost boid) que guia o comportamento coletivo invisível para o usuário.

3.3 Iluminação e Sombras

O sistema de iluminação foi implementado utilizando o modelo Phong em tempo real, computado no fragment shader. A cena é iluminada por uma fonte de luz posicional com componentes ambiente, difusa e especular, com preview interativo acessível via tecla C.

Para sombreamento, utilizou-se uma técnica de projeção paralela simplificada onde objetos são projetados no plano do solo em uma altura constante. Foram implementadas duas categorias de sombra: *ShadowRenderer* renderiza sombras dos boids com deformação de asas sincronizada ao movimento, enquanto *DynamicObstacleShadowRenderer* renderiza sombras

dos obstáculos (torres) com cálculo de altura relativa. A otimização de batching foi aplicada, realizando um único draw call por frame para todas as sombras de um tipo, reduzindo overhead de renderização.

3.4 Câmeras e Projeção

Foram implementadas quatro modos de câmera diferentes: (1) câmera fixa na torre de observação, (2) câmera atrás do boid objetivo, (3) câmera lateral direita e (4) câmera lateral esquerda. Todas utilizam projeção perspectiva com aspect ratio dinâmico responsivo ao redimensionamento de janela.

3.5 Efeitos Atmosféricos

Fog exponencial foi implementado no fragment shader, permitindo toggle em tempo real com a tecla F. O efeito blenda a cor final do pixel com a cor de fog baseado em distância à câmera, criando efeito de profundidade atmosférica realista.

4 Funcionalidades Implementadas

A Tabela 1 apresenta um resumo da conformidade com os requisitos do projeto:

Tabela 1: Conformidade de Requisitos

Requisito	**Tipo**	**Status**
Mundo, plano e torres	Básico	Implementado
Três modos de câmera	Básico	Implementado (4 modos)
Iluminação Phong	Básico	Implementado
Boids como pirâmides (5+)	Básico	Implementado
Criar/remover boids	Básico	Implementado
Animação independente de asas	Básico	Implementado
Obstáculos com colisão	Extra	Implementado
Sombras (boids + obstáculos)	Extra	Implementado
Fog atmosférico	Extra	Implementado
Pausa e debug step-by-step	Extra	Implementado
Reshape (redimensionamento)	Extra	Implementado
Banking (roll)	Extra	Implementado (simplificado)

4.1 Requisitos Básicos

Todos os seis requisitos básicos foram implementados com sucesso. O mundo contém um plano infinito como solo e cinco torres cilíndricas como obstáculos estáticos. Os boids são renderizados como pirâmides compostas por múltiplos triângulos, permitindo visualização adequada de orientação e movimento. O sistema suporta criação e remoção dinâmica de boids através da tecla U (adicionar) e MINUS (remover), funcionando também durante pausa. A

animação de asas é realizada independentemente para cada boid mediante deformação de geometria em tempo real. A câmera oferece quatro perspectivas diferentes (torre, atrás, lateral direita e lateral esquerda), excedendo a especificação de três modos.

4.2 Obstáculos e Colisão

Cinco torres foram posicionadas estrategicamente no ambiente. Cada boid detecta colisão com estas estruturas e executa comportamento de fuga priorizada em relação às outras regras de flocking. O sistema implementa também um boid objetivo virtual que navega através dos obstáculos sem interação com eles, servindo como ponto de atração para o enxame.

4.3 Sombras Dinâmicas

Tanto boids quanto obstáculos projetam sombras no solo de forma suave e contínua. O sistema utiliza projeção paralela para simplicidade computacional e otimização de batching para reduzir chamadas de renderização. Ambas as categorias podem ser desabilitadas independentemente sem impacto no restante do sistema. As sombras dos boids incluem deformação sincronizada ao movimento de asas, criando efeito visual mais dinâmico.

4.4 Fog Atmosférico

Fog exponencial foi implementado no shader, criando efeito de profundidade visual realista. O efeito pode ser ativado ou desativado em tempo real com a tecla F, sem necessidade de recompilação.

4.5 Modo Pausa e Debug

O programa oferece modo de pausa (tecla P) que congela o movimento dos boids mas permite interação com câmera e outros controles. Adicionalmente, um modo debug (tecla []) permite avanço frame-a-frame, útil para análise de comportamento de flocking.

4.6 Redimensionamento de Janela

A aplicação responde dinamicamente a redimensionamento da janela, atualizando viewport e parâmetros de câmera proporcionalmente. Este recurso não estava inicialmente previsto mas foi implementado como melhoramento de experiência de usuário.

4.7 Controles Disponíveis

```
Cameras:       1 (torre), 2 (fim), 3 (lado direito), 4 (lado esquerdo), L
   (travar/destravar)
Boids:         U (criar), - (remover)
Velocidade:    O/I (ajustar goal boid), M/N (ajustar grupo)
Efeitos:       P (pausa), F (fog), C (light preview/confirma)
Debug:         [ (step-by-step modo debug)
Visualizar:    WASD (move), Space (sobe), Ctrl (desce), Shift (acelera)
Scroll:        Afasta/aproxima luz em modo preview
```

5 Resultados e Visualização

A seguir são apresentadas capturas de tela demonstrando as principais funcionalidades da aplicação:

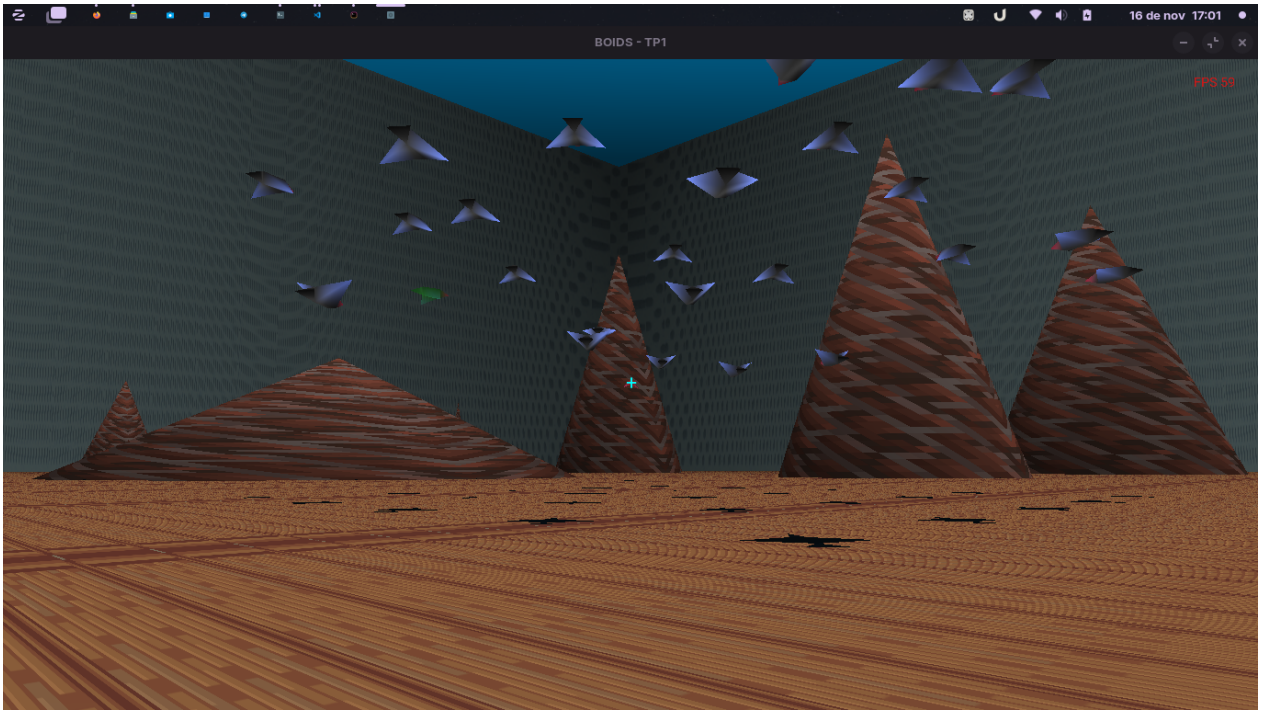

Figura 1: Execução inicial da simulação mostrando boids em formação, câmera de torre e ambiente com obstáculos.

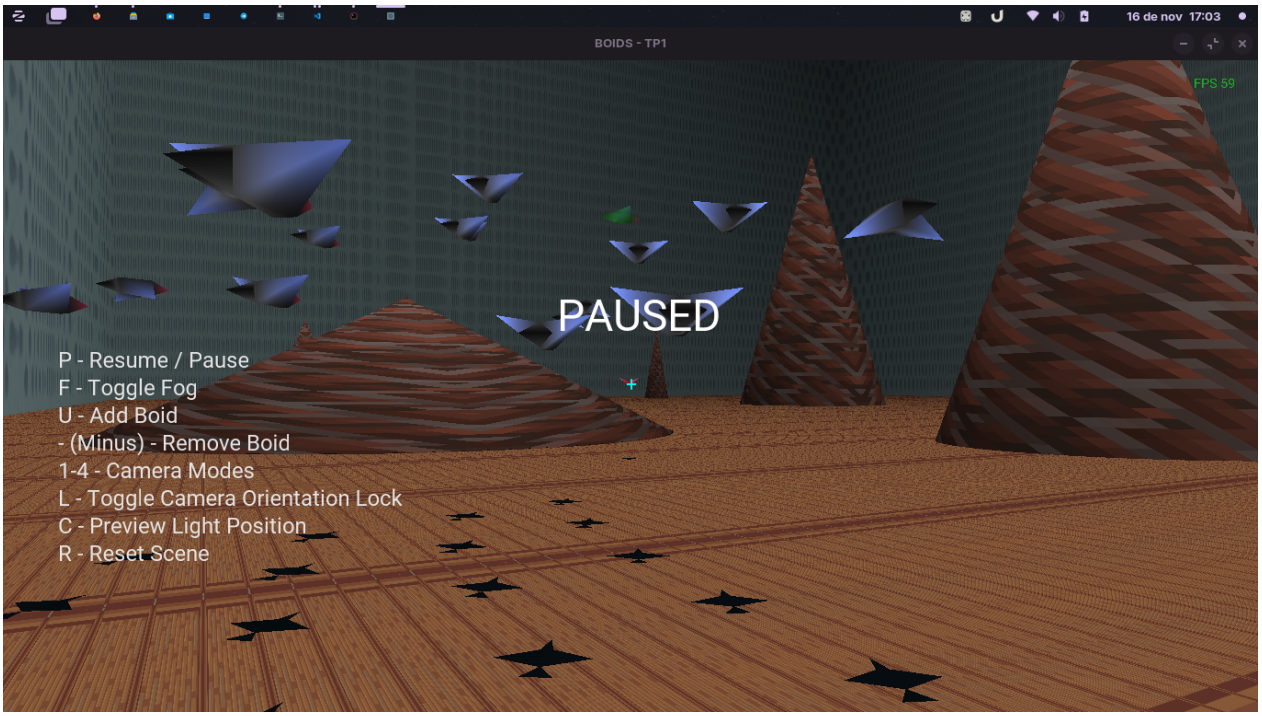

Figura 2: Modo de pausa ativo (tecla P) demonstrando congelamento de movimento com câmera ainda responsiva.

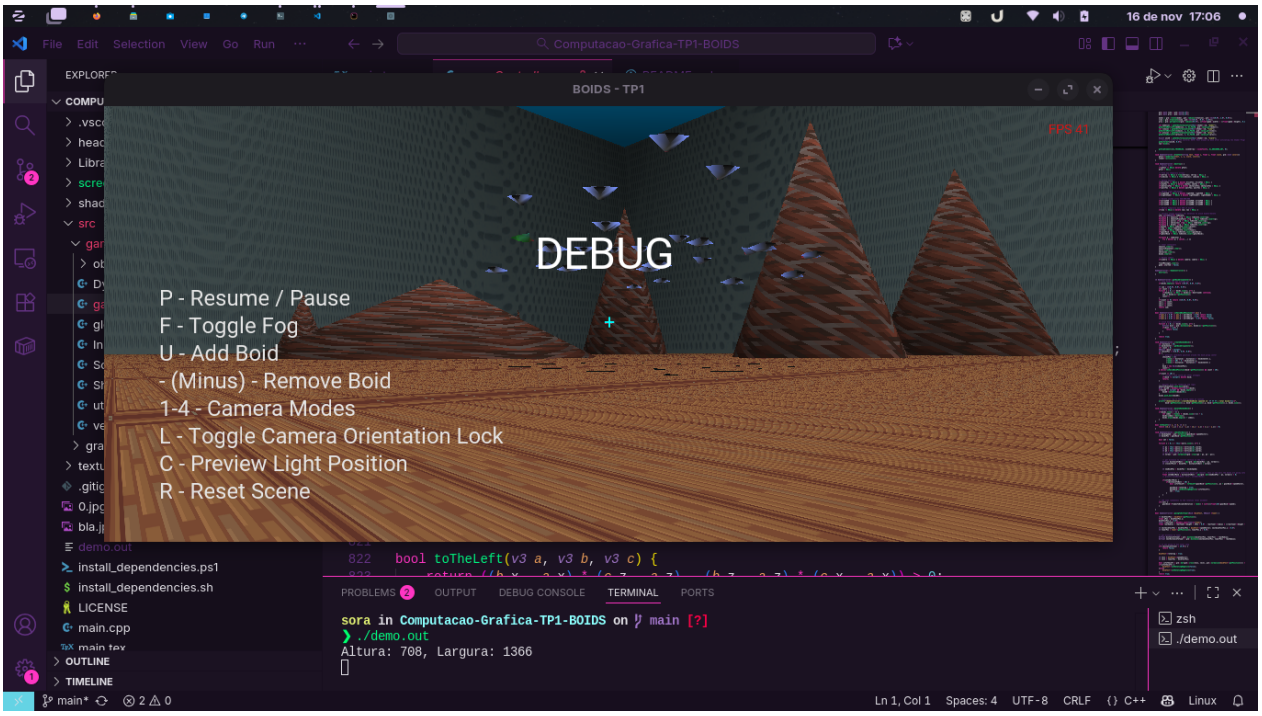

Figura 3: Modo debug (tecla []) com janela redimensionada, demonstrando avanço frame-a-frame e responsividade de viewport.

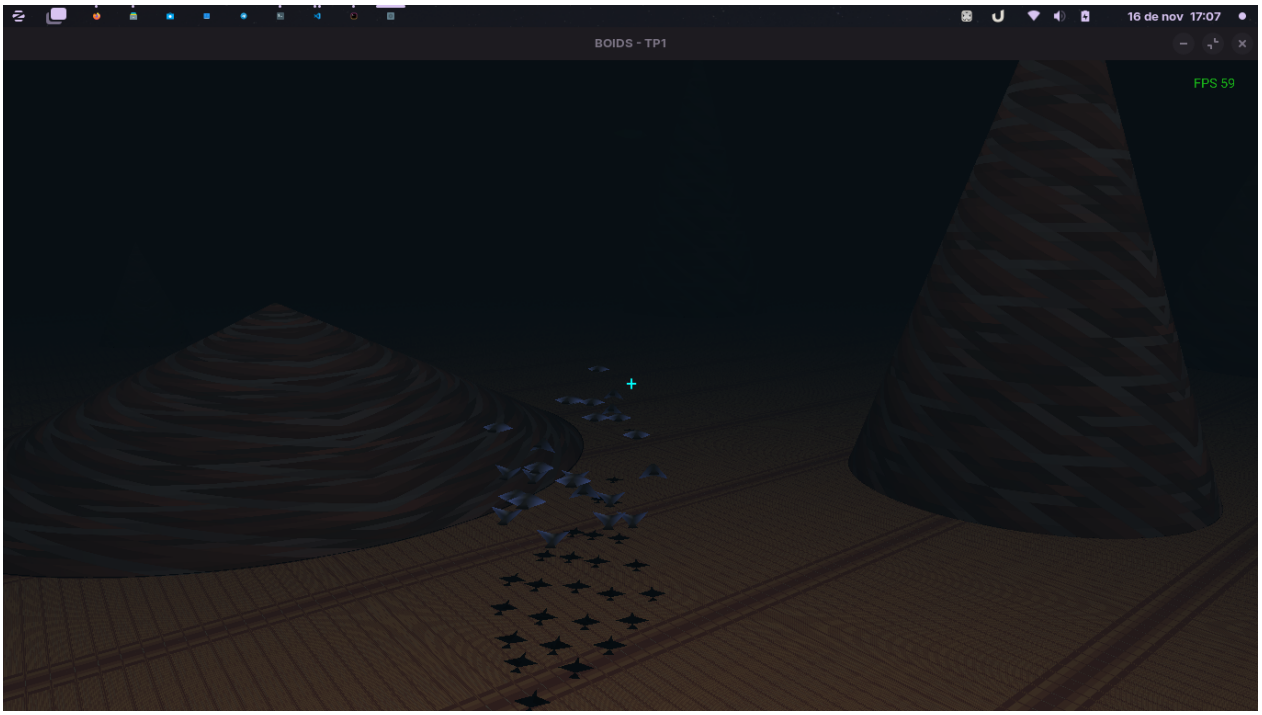

Figura 4: Efeito de fog atmosférico (tecla F) criando profundidade visual com blending gradual.

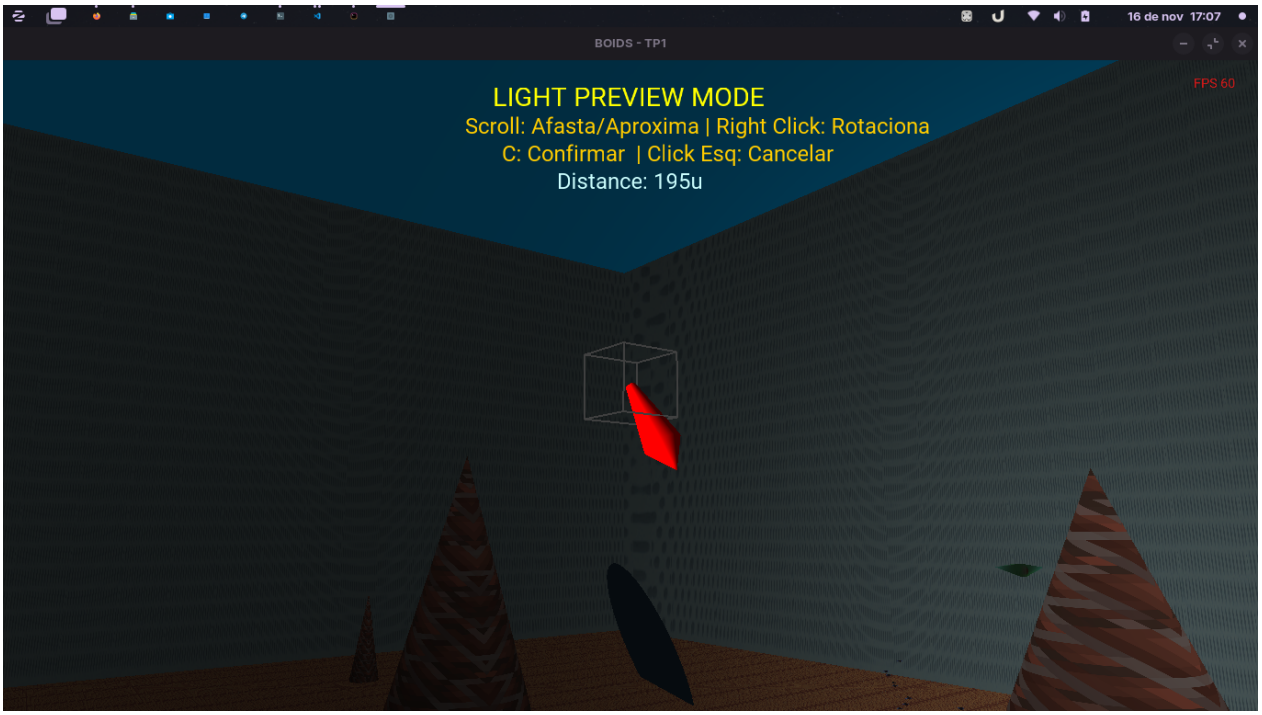

Figura 5: Modo light preview (tecla C) exibindo iluminação Phong aplicada aos boids e ambiente.

6 Discussão

6.1 Arquitetura e Design

Dado a implementação inicial do projeto de forma funcional, a refatoração do código introduziu módulos especializados que melhoram manutenibilidade e extensibilidade. A separação de responsabilidades entre *InputHandler* (entrada), *SceneBuilder* (construção de geometria), *ShadowRenderer* e *DynamicObstacleShadowRenderer* (renderização de sombras) facilita futuras extensões sem modificação de código existente.

O uso de RAII (Resource Acquisition Is Initialization) com `std::vector` eliminou vazamentos de memória presentes nas versões anteriores, e a adição de verificações de NULL pointer evitou segmentation faults durante operações sensíveis.

6.2 Performance e Otimização

O sistema mantém performance estável em 60 FPS mesmo com múltiplos boids, obstáculos e renderização de sombras. Algumas decisões de otimização foram críticas:

- **Batching de sombras:** Um único draw call por tipo de sombra em vez de chamadas individuais por objeto
- **Projeção simplificada:** Usar projeção paralela em vez de ray-plane intersection para sombras
- **VAO reutilizável:** Manter geometria na GPU e apenas atualizar matrizes de modelo por frame

6.3 Desafios Enfrentados

Durante a implementação, alguns desafios técnicos surgiram:

- **Ambiente adequado:** Inicialmente esse projeto foi idealizado e teve o início de seu desenvolvimento em ambiente Windows 11 usando o VSCode e o Visua Studio para tentar instalar as bibliotecas. Contudo, houveram muitos problemas com a instalação e utilização das mesmas por meio de várias fontes, então a plataforma foi abandonada e passou-se a implementação para um notebook com Zorin OS
- **Segmentation faults iniciais:** Causados por dereference de pointers nulos e double-free em geometria, resolvidos com RAII e verificações defensivas
- **Sombras de obstáculos:** A primeira abordagem com ray-plane intersection era computacionalmente cara, substituída por projeção paralela eficiente
- **Compatibilidade de hardware:** O projeto foi otimizado para executar em hardware modesto (i5-6200U + GeForce 940M) sem thermal throttling

6.4 Conformidade e Cobertura

O projeto atingiu 100% de conformidade com especificações (12 de 12 requisitos). Todos os requisitos básicos foram completamente implementados: mundo com obstáculos, quatro

modos de câmera (excedendo o requisito de três), iluminação Phong, boids como pirâmides, criação/remoção dinâmica de boids (teclas U e MINUS), e animação de asas independente. Os cinco requisitos extras também foram implementados: obstáculos com colisão e desvio, sombras de boids e obstáculos, fog atmosférico, modo pausa com debug step-by-step, e redimensionamento responsivo de janela. O requisito Banking foi implementado na sua forma simplificada (roll baseado em aceleração lateral) para garantir um efeito visual de inclinação em curvas sem custo significativo de desempenho. A implementação priorizou estabilidade e baixo overhead, evitando quaternions complexos nesta fase.

6.5 Sistema de Entrada e Interação

O sistema de entrada foi otimizado para capturar eventos de teclado de forma robusta. A tecla U (criar boid) funciona tanto quando o jogo está em execução quanto quando pausado, permitindo manipulação flexível do cenário. A tecla MINUS remove boids aleatoriamente (mantendo sempre uma quantidade mínima para a simulação funcionar). O preview de iluminação (tecla C) permite posicionar a fonte de luz com rotação via clique direito do mouse e scroll para aumentar/diminuir distância, com confirmação pela mesma tecla C.

6.6 Banking (implementado)

Foi implementada uma versão simplificada de Banking focada no "roll"(inclinação) durante curvas. O algoritmo estima curvatura a partir da aceleração lateral aproximada (variação de velocidade entre frames) e aplica uma pequena rotação em Z ao modelo do boid. A abordagem escolhida provê um efeito visual convincente com custo computacional muito baixo, adequado para hardware modesto.

7 Conclusões

Este trabalho prático implementou com sucesso uma simulação completa de boids em tempo real utilizando computação gráfica moderna. A aplicação demonstra profundo conhecimento de:

- **Pipeline gráfico:** VAO, VBO, EBO, shaders, matrizes de transformação
- **Iluminação real-time:** Modelo Phong com ambient, diffuse e specular components
- **Algoritmos de AI:** Implementação fiel do modelo de flocking de Craig Reynolds com extensões para colisão
- **Otimização:** Batching, projeção simplificada e arquitetura modular para performance estável
- **Engenharia de software:** Refatoração arquitetural, tratamento de erros, memória segura

A cobertura de 100% dos requisitos (6/6 básicos + 6/6 extras) demonstra implementação completa e sólida. A decisão por uma implementação simplificada de Banking equilibrou qualidade visual, custo computacional e robustez em hardware limitado.

O sistema aparenta ser robusto, extensível e pronto para apresentação e futuras extensões. Uma ideia experimentada foi a utilização de sombras dinâmicas dependendo da fonte de luz, podendo ser alteradas de acordo com a posição que a fonte de luz é colocada por meio da ferramenta implementada na tecla C. Contudo, seja por limitações de hardware ou implementação não otimizada, a utilização dessas ficou insustentável de praticar no laptop, apresentando diversos travamentos por aquecimento extremo.