**Relatório do Trabalho da Disciplina Fundamentos de Computação Gráfica**

Nícolas Casagrande Duranti - 287679

Nicolau Pereira Alff - 243667

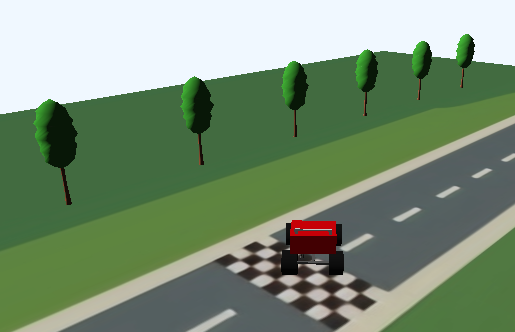
**Desenvolvimento do Jogo:**

Este trabalho foi desenvolvido utilizando-se a linguagem C++, com uso de OpenGL. O código produzido foi baseado nos códigos disponibilizados pelo professor e complementado por nós nas aulas de laboratório. O jogo desenvolvido é um jogo de corrida, no qual o usuário consegue controlar os movimentos do carrinho em tempo real, com o objetivo de completar o percurso no menor tempo possível.

**Funcionamento do jogo:**

O jogo é formado por duas partes: a primeira, em que o jogador controla o veículo pela pista e, a segunda, na qual mostra-se o pódio e troca-se para a câmera look at. Para alterar da parte um para a parte dois, o jogador tem de completar o percurso e “entrar” no troféu. Se o jogador tentar ir ao troféu antes de passar pela linha de chegada, irá apenas colidir com o objeto e parar.

**Uso de conceitos de Computação Gráfica:**

No desenvolvimento deste programa aplicou-se diversos conceitos vistos em aula. Para realizar o desenho dos objetos na tela, por exemplo, utilizamos as matrizes model, view e projection aprendidas na cadeira. Todos os objetos virtuais utilizados são carregados de arquivos no formato .obj, sendo que o objeto da árvore, do pneu, do cone e, na parte final, do troféu são desenhados com mais de uma instância, modificando-se a matriz model utilizada.

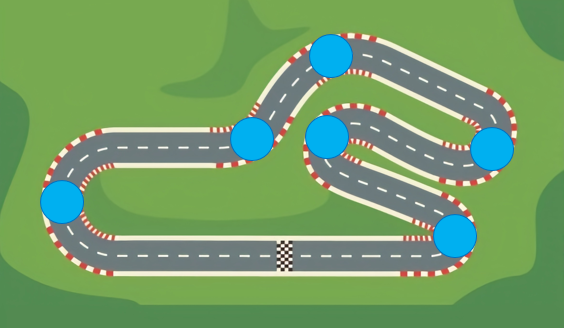
Utilização de várias instâncias do objeto da árvore.

A aplicação desenvolvida apresenta os dois tipos de câmera vistos em aula. A câmera livre é utilizada no momento da corrida, podendo ser movimentada para frente, para trás e rotacionada para ambos os lados através do teclado. Para que o movimento do carro e da câmera fossem sincronizados e para que o veículo estivesse sempre na imagem obtida pela câmera, transladou-se repetidamente a posição do carrinho para a posição da câmera. Ou seja, utilizamos a localização da câmera para controlar os movimentos do carro.

Entretanto, a fim de obtermos uma visão em terceira pessoa, foi necessário alterar as informações passadas como parâmetro para a construção da matriz view. A posição da câmera foi recuada, deslocando-a no sentido contrário do vetor view, e elevada, somando-se duas unidades na sua coordenada y. Além disso, o vetor view foi somado à um vetor **v** = (0, -2, 0), fazendo com que a câmera “olhasse” em direção ao carro.

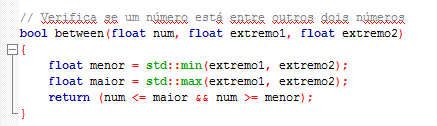
Já a câmera look at é utilizada após o término da corrida, quando mostramos o pódio. A visão da câmera fica fixa em um ponto logo abaixo do veículo, e é possível movimenta-la ao redor dele com o mouse.

Os testes de intersecção utilizados foram três: ponto-esfera, cubo-plano e cubo-cubo. Este primeiro tem grande importância na dinâmica do jogo, pois possibilita controlar se o carro passou por uma determinada região da pista ou se cortou caminho. Considerou-se uma esfera em cada curva do percurso e, para que o jogador o complete, é preciso que o veículo passe, em ordem, por todas as esferas. O teste realizado verifica se o centro da bounding box do carro está contido na próxima esfera que ele deve intersectar.

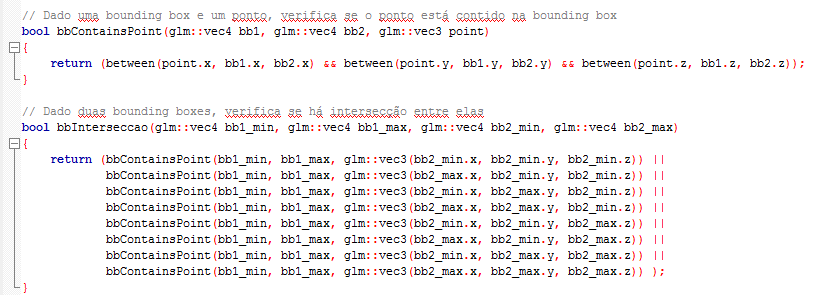


Localização das esferas utilizadas como checkpoints na pista.

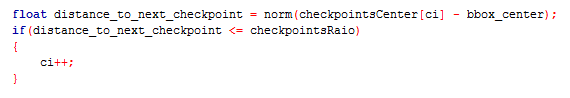
O teste cubo-plano foi utilizado para verificar se o jogador cruzou a linha de chegada e, então, parar o tempo e habilitar a possibilidade de mudar para o pódio. Além disso, com este teste, garante-se que o carro não ultrapasse as bordas da pista. Já o teste cubo-cubo permite analisar se o veículo colidiu com os objetos estáticos (árvores, pneus e troféu) ou com os obstáculos móveis (cones).



Código usado para os testes cubo-plano e como função auxiliar para o teste cubo-cubo.



Código usado para os testes cubo- cubo.

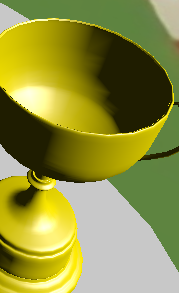


Código usado para os testes ponto-esfera.

O modelo de iluminação aplicado foi a combinação de Lambert e **Blinn-Phong. Além disso, foi implementado os modelos de interpolação para iluminação de Gouraud e de Phong. O modelo de Gouraud foi aplicado no segundo troféu, mostrado juntamente com o pódio. Ainda,** foram utilizadas texturas para determinar a cor de alguns objetos. Este recurso foi aplicado para colorir o plano (pista), o carro e as árvores.

Troféu à esquerda: **Blinn-Phong e Phong shading.**

**Troféu à direita: Blinn-Phong e Gouraud shading.**



A movimentação dos obstáculos (cones) foi implementada com Curvas de Bezier. Com isso, os cones se locomovem em arcos suaves sobre a pista. Abaixo, é possível visualizar a posição de um obstáculo em três momentos diferentes:

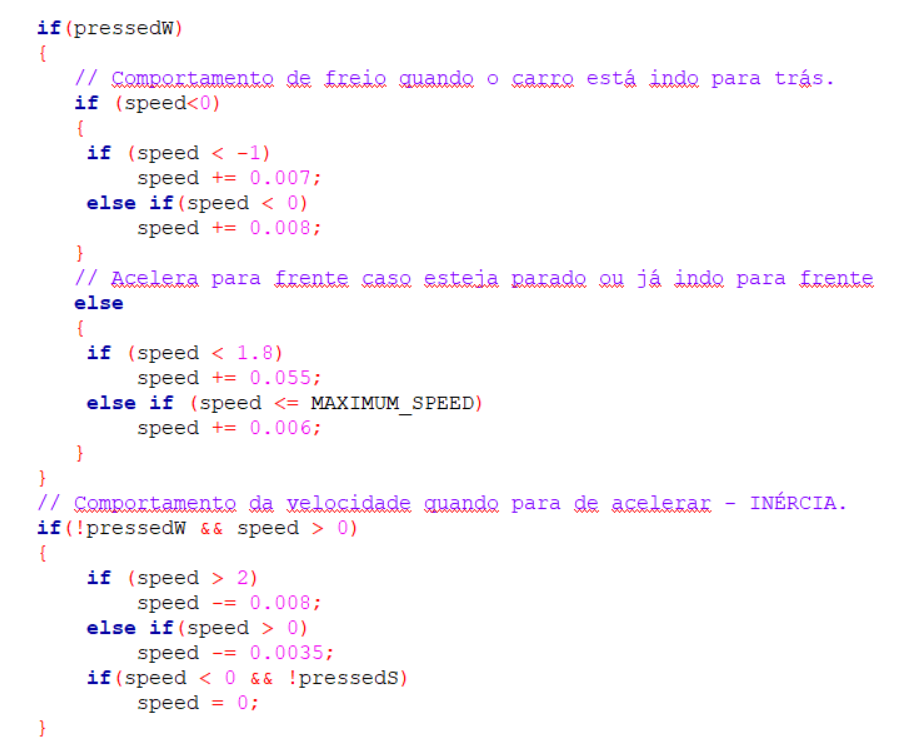




Por fim, todas as movimentações de objetos foram computadas baseadas no tempo decorrido. Desse modo, tanto o carrinho/câmera quanto os obstáculos se movimentam sempre na mesma velocidade, independentemente da CPU em que o jogo está sendo executado.

**Aspectos adicionais:**

Para uma melhor noção de veracidade do jogo e uma melhor adaptação física da realidade para o contexto de corrida, elaboramos uma forma de inserir aceleração dentro do jogo. Simula-se aceleração e frenagem nos dois sentidos, frente e trás, além de adicionar inércia ao movimento, de forma que, mesmo não freando, ao se parar de acelerar, o carro vai aos poucos perdendo velocidade, até parar. Para uma melhor jogabilidade, foram necessários diversos testes, como os demostrados a seguir:



Código mostrando o comportamento para a tecla W. Para a tecla S, o código é muito parecido, porém com sinais invertidos, por se tratar da ré.

Ainda, adicionamos a funcionalidade de áudio no jogo: no momento em que o jogador completa o percurso, toca-se uma música. Para isso, utilizamos a biblioteca irrKlang, que permite trabalhar com arquivos de áudio no formato MP3.

**Instruções:**

O controle da movimentação do carro é feito através das seguintes teclas:

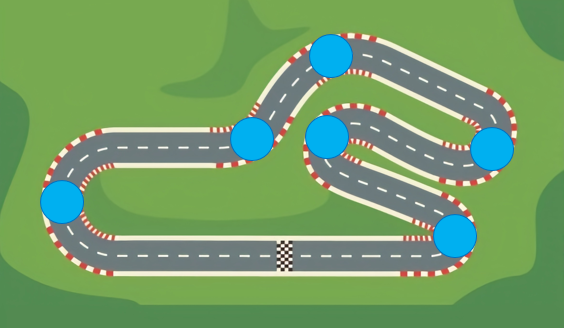
• W - acelerador

• S - freio e ré

• A - virar para a esquerda

• D - virar para a direita

O objetivo do jogador é finalizar o percurso no menor tempo possível, passando por todas as curvas e cruzando a linha de chegada. O controle do caminho executado pelo carrinho é feito através de checkpoints, localizados em cada curva. Os checkpoints devem ser cruzados em ordem, ou seja, se o jogador não passar pelo checkpoint um, mesmo que passe pelo checkpoint dois, este último não será contabilizado. É possível acompanhar o checkpoint atual no canto inferior esquerdo da tela.



1

2

3

4

5

6

Localização e ordem dos checkpoints**.**

O tempo começa a contar assim que o jogador apertar a tecla W pela primeira fez. Após cruzar a linha de chegada, se os seis checkpoints foram alcançados corretamente, o tempo para de aumentar. Um relógio cronometrando o tempo é mostrado na parte superior central da tela do jogo.

Após terminado o percurso, é habilitada a possibilidade de ir para o pódio. Para isso, o jogador precisa apenas encostar o veículo no troféu. Ao fazer isso, o jogador poderá usar o mouse para controlar a câmera ao redor do carrinho e do pódio.

**Compilação e Execução:**

Uma vez que a linguagem utilizada foi C++, é possível compilar e executar o código através do Codeblocks, ou compilar diretamente pela linha de comando utilizando compiladores como o gcc. Além disso, para o correto funcionamento do áudio, é necessário realizar a linkagem da biblioteca irrKlang antes de realizar a compilação. Para fazer isto, no codeblocks, basta ir em settings, compiler, Linker settings, add, e adicionar o caminho “/lib/libirrKlang.a”.