Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Segundo Semestre de 2020
Prof. Cesar Tadeu Pozzer
Data: 11/12/2020

Trabalho 3 - Transformações Geométricas e Curvas

Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina) e IDE Code::Blocks, compilando com MinGW (disponível na versão 17.12 da IDE Code::Blocks). **Não podem ser utilizadas bibliotecas auxiliares. Não pode ser usada a API OpenGL.** A API cavnas2D pode ser customizada, bem como a classe Vector2.

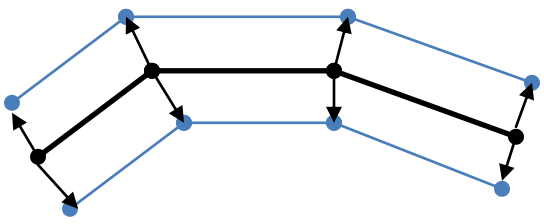
Objetivos:

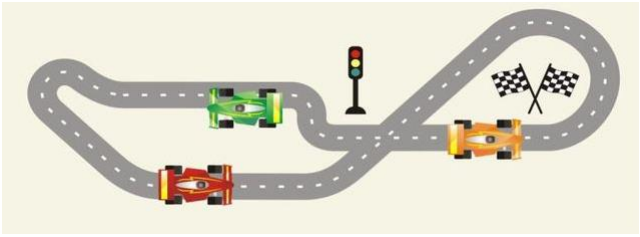
- Estudar curvas 2D
- Transformações geométricas
- Movimentação de objetos
- Manipulação de Vetores
- Criar e implementar uma API de widgets na Canvas2D
- Programação em C++

Implemente um veículo que se locomove em um espaço 2D sobre uma estrada definida pelo usuário, simulando um jogo de corrida de veículos. O trabalho tem duas etapas: criação da pista e simulação da corrida.

A estrada é composta por vários pontos de controle interpolados via alguma curva paramétrica (Bezier ou B-Spline). Os pontos de controle devem ser definidos pelo usuário, como o uso do mouse. À medida que o usuário insere os pontos de controle, deve-se ir desenhando o grafo de controle correspondente. Quando o usuário acabar a edição, a estrada curva deve ser gerada.

A estrada deve ter uma largura definida pelo usuário, e é gerada pela expansão do grafo de controle perpendicularmente em relação a sua direção, como mostrado na figura (A linha preta é o grafo de controle). A estrada deve formar um circuito fechado.





A pista como um todo, após ser gerada, deve permitir as seguintes transformações: rotação em relação ao centro da pista, translação e escala em relação ao centro da pista.

Durante a corrida, o veículo deve ter uma velocidade controlada, bem como direção de deslocamento. A geometria do veículo pode ser bem simples, como uma simples seta, que possa dar noção de sua direção.

O usuário pode interagir guiando o veículo com o uso das setas direcionais, e com outras duas para mudar a velocidade (e.g. "a" para acelerar e "s" para frear). Pode-se adicionar teclas para movimentos laterais. Pode-se utilizar o mouse também.

Deve-se controlar o ângulo máximo de rotação do veículo. O movimento do veículo deve ser suave e contínuo, independente do usuário estar ou não pressionando alguma tecla (Não deve ser usado o refresh do teclado – ver demo gl_3_FPS). Quanto maior a velocidade, menor deve ser o ângulo de giro.

A nota máxima sem bônus é 9.0.

Critérios avançados (Bônus):

- Recurso de edição em tempo real dos pontos de controle e automaticamente desenho da pista. (Até 1pt)
- Determinação se o veículo está andando em cima da pista e respectiva contagem de pontos. (Até 1pt)
- Vetores indicando as "forças" aplicadas ao veículo (como uma realidade aumentada), como aceleração, força centrípeta, posição em relação a pista, etc. (Até 1pt)
- Adicione objetos ao longo da pista que devem ser coletados para ganhar pontuação (semelhante o jogo Temple Run). (Até 1pt)
- Definir outros veículos autônomos andando na pista para simular uma competição de carros de corrida. (Até 3pt)
- Coloque recurso para disparo de projéteis que podem tirar os adversários da pista. Tratamento de colisão de tiros. Pode-se aproximar os alvos como círculos. (Até 2pt) (dependente do anterior)
- Placar do jogo (Até 1pt) (dependente dos 2 anteriores)
- Etc.

O trabalho deve apresentar uma lista de instruções, explicando de forma como o usuário deve interagir com o programa. Enumere no início do código fonte (arquivo main.cpp) os quesitos que foram implementados.

Data e Formato de Entrega:

- O arquivo deve ter o nome do aluno (ou uma abreviação nome+sobrenome, ou seja, algo que identifique bem o aluno). O arquivo deve ser enviado pelo Google Classroom.

- Dentro deste arquivo deve haver um diretório com o mesmo nome do arquivo e, dentro deste diretório, os arquivos do trabalho. Deve-se enviar somente: código fonte (.cpp, .h, .hpp) e o projeto (.cbp). Não devem ser enviadas libs, executáveis, DLLs.

- Antes do envio, certifique-se de que o projeto contido no seu .rar funciona em qualquer diretório que ele seja colocado e não dependa de outros arquivos não incluídos no envio do trabalho.

Critério de Avaliação:

- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e detalhes específicos de partes que mereçam uma explicação – não comente por exemplo o que faz b++.
- README.txt: incluir um arquivo "README.txt" contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras).
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Legibilidade: nome de variáveis, estruturação do código.
- Clareza: facilidade de compreensão – evite códigos complexos e desnecessários. Adote a solução mais simples possível.
- Funcionalidade: o programa deve satisfazer todos os requisitos. Programas que não compilarem ou que não atenderem nenhum requisito receberão nota 0 (zero).

Você pode discutir estratégias e ajudar o colega na implementação, porém evite passar código fonte. Programas semelhantes terão a nota 0 (zero).