

Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica
Primeiro Semestre de 2025
Prof. Cesar Tadeu Pozzer
Data: 24/04/2025

Trabalho 3 – Grand Theft Auto 2D



Ferramentas:

Linguagem C++, utilizando a API Canvas2D (disponível no [site da disciplina](#)) e IDE Code::Blocks, compilando com MinGW de 32 bits. **Não podem ser utilizadas bibliotecas auxiliares.** Desenvolva o trabalho sobre o demo 1_CanvasGlut ou 2_CanvasGlfw. Antes de enviar, retire todas as funções e arquivos não utilizados. Se alguém fizer em Linux, deve ajustar todos os paths para execução em Windows. Teste uma máquina Windows antes de enviar. O melhor neste caso é rodar o Windows em uma máquina virtual, como o Virtual Box ou Parallels.

Objetivos

- Explorar vetores, ângulos
- Explorar transformações geométricas
- Explorar algoritmos de interseção (ver demo gl_4_intersection da cadeira de CGA)
- Explorar curvas
- Explorar interação com teclado e mouse

Descrição

Implemente um jogo de tiro/corrida, onde o jogador é um tanque que deve acertar alvos em uma pista. O veículo do jogador (tanque de guerra) deve ser composto de duas partes:

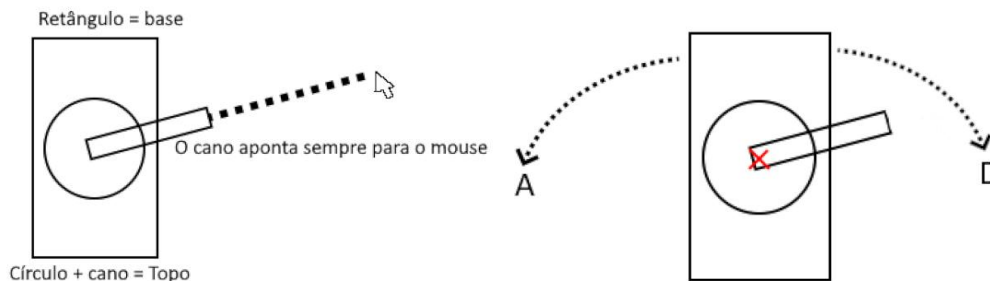
- Base: pode ser modelada como um retângulo, é onde ficam as esteiras que causam o movimento. A frente da base é para onde o retângulo aponta.
- Topo: parte onde fica o cano do tanque. Rotaciona independentemente da orientação da base.

O tanque sempre está em movimento para a frente. O jogador apenas aperta "A" e "D" para girar a base. O tanque deve continuar andando para a frente enquanto é rotacionado. Por exemplo:

se o jogador apenas segurar "A", o veículo vai fazer uma trajetória circular e acabar no mesmo lugar onde começou. A segunda imagem a seguir mostra como é a rotação, o pivô (X vermelho) é o centro do tanque, ou seja, o centro do círculo do topo. Não pode ser utilizado o refresh do teclado para a movimentação. Utilizar controle de FPS.

- Para modelar essa rotação, pode ser estudada a força **centrípeta** da física e um raio de rotação fixo. A força centrípeta é aplicada no vetor direção do tanque.

O topo do tanque é rotacionado independente da base. A sua orientação sempre aponta para onde o mouse está localizado. A primeira imagem a seguir ilustra como o topo deve se comportar.



O tanque vai estar em um cenário estilo uma pista, formada por curvas suaves (Bézier ou B-Spline).

- Pode ser utilizada uma curva central expandida ou duas curvas, uma para cada lado da pista. Se for utilizada a expandida, cuidar para a largura da pista ser constante em curvas.

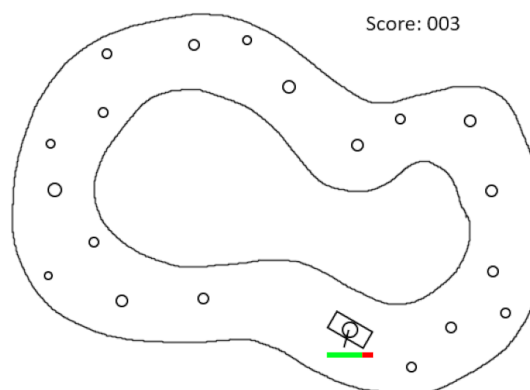
A pista deve formar um **loop** tal que se o jogador percorrer toda sua extensão, estará de volta ao início. A pista inteira deve aparecer na tela. **Deve ser utilizada a resolução 1280x720 ou 1920x1080.** Deve existir colisão do tanque com as bordas da pista, ou seja, o jogador não pode ir para fora. Deve haver um modo editor, onde o usuário posiciona pontos de controle para criar uma pista personalizada para o jogo.

O jogador ao clicar com o botão esquerdo do mouse, deve atirar um projétil **a partir da ponta do cano** do tanque. Utilizar vetores direção e velocidade. O projétil deve possuir sua própria colisão com o cenário (bordas da pista) e alvos. Não deve ir imediatamente (teletransportado) para a posição do mouse, ele deve ir suavemente com controle de FPS, assim como o resto do jogo. Deve haver um *delay* entre dois tiros do tanque, o usuário não pode clicar rapidamente e dar vários tiros seguidos.


Espalhados em pontos aleatórios da pista devem existir barris explosivos como alvos. Podem ser modelados como círculos. O projétil do jogador ao colidir com um alvo, deve acrescentar um ponto ao score do jogador e o alvo deletado.

Se o tanque colidir com as paredes ou diretamente com um alvo não explodido, ele deve perder vida. Uma barra de vida deve ser mostrada embaixo do tanque. A barra de vida deve acompanhar a posição do tanque na tela. Devem ser mostradas barras de vida em alvos que tenham mais vida/não são explodidos com um tiro só.

Exemplo de visual do programa final:



Requisitos do programa:

- A pista deve ser desenhada com curvas Bezier ou B-spline
- Autoria de pistas com o usuário definindo pontos de controle
- Movimentação correta do tanque
- Tanque dispara e os alvos são destruídos
- Deve ter um placar de pontuação
- Projétil tem colisão com bordas da pista e alvos
- A geometria do tanque e alvos deve ser bem simples
- Controle de FPS
- O trabalho será compilado em Windows 
- Barra de vida do tanque e danos na colisão
- Jogabilidade fluida, FPS baixo não deve impedir o jogo de ser jogado corretamente.

Extras (para nota acima de 9,0) – valores máximos:

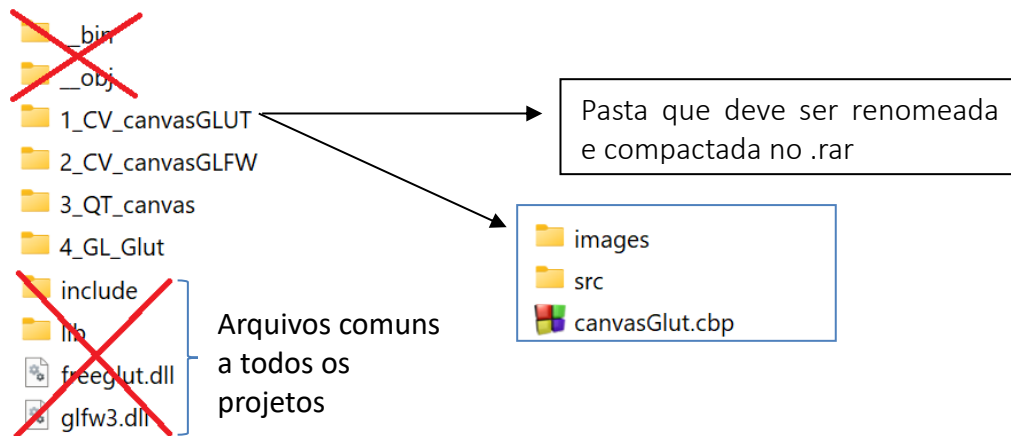
- Alvos com diferentes lógicas, atiram de volta, se movimentam etc. (até 1 pt)
- Alvos com diferentes visuais e colisão apropriada. (triângulo, estrela etc.) (até 2 pt)
 - Formas complexas podem ser modeladas como um conjunto de primitivas: colisão da estrela é um OR entre a colisão de triângulos que compõe a forma. Útil para formas côncavas.
- Power-ups para disparos seguidos ou escudo contra colisões. (até 1 pt)
- Efeitos de explosão bonitos. (até 2 pt)
- Preenchimento colorido da pista com detalhes (ex.: cinza para asfalto e linhas pontilhadas amarelas). (até 1 pt)
- Diferentes níveis de dificuldade ao explodir todos os alvos. (até 1 pt)
- Mais ideias que acrescentem ao conteúdo explorado neste trabalho também serão ser recompensadas.

Passos

- Primeiro terminem o Lab onde trabalhamos com vetores, ângulos, etc.
- Programinha para explorar o controle de FPS → velocidade de animação. Não é somente usar o delay(). Vejam os demos da disciplina.
- Explorar movimentação com uso do teclado
- Simular trajetória dos disparos
- Desenhar o tanque usando transformações de rotação/translação/escala (último lab). Entender bem onde é o pivô de rotação.
- Testar colisão linha/círculo ou linha/linha
- Juntar tudo.

Entrega:

- O trabalho deve ser entregue pelo Google Classroom. Veja no arquivo cg_1_apresentacao.pdf instruções de envio.
- Deve-se utilizar como base o projeto gl_1_canvasGlut disponível nos demos da disciplina, como ilustrado na seguinte figura a esquerda.



- A pasta gl_1_canvasGlut contém todos os códigos fonte e recursos. As pastas lib/include/bin/obj são comuns a todos os projetos, e por isso **não devem** ser enviadas. O mesmo vale para as dll.
- A pasta gl_1_canvasGlut **deve ser renomeada** com o nome do aluno. Ex: Trab1Maria, Trab2Paulo, Trab3Pedro, Trab4JoaoPedro, etc. Esta estrutura vai facilitar a execução dos trabalhos pelo aluno monitor. Todos os arquivos do trabalho devem estar dentro desta pasta, que deve ser a única pasta enviada, compactada em formato .rar, cujo nome deve ser o nome do aluno. Ex: FulanoSobrenome.rar.
- Dentro desta pasta (Trab1Maria, ...), como mostrado na figura a direita, existe:
 - o projeto code::blocks,
 - a pasta src que contém os códigos fonte (.c, .cpp, .h), e
 - a pasta imagens que pode conter arquivos BMP (opcional).
- Esta estrutura de pastas não pode ser modificada.
- **Não devem ser enviadas lib, exe, obj, DLL, pdf, doc.**
- **Retire todo código não utilizado** no trabalho (arquivos, métodos, variáveis, etc), bem como printiefis de depuração.

Critérios de avaliação:

- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e comentar o que cada método e classe faz.
- Clean code: estrutura do código e nomeação de métodos, classes e variáveis devem ser fáceis de ler e entender. Procurar manter o código o mais simples e organizado possível.
- README: incluir um arquivo "README.txt" contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras) e instruções de uso do programa caso o aluno julgue necessário ou caso tenha sido implementado uma funcionalidade extra que exija explicação.
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Funcionalidade: o programa deve satisfazer todos os requisitos. Programas que não compilarem ou que não atenderem nenhum requisito receberão nota 0 (zero).