

Universidade Federal de Santa Maria  
Disciplina: Computação Gráfica  
Primeiro Semestre de 2024  
Prof. Cesar Tadeu Pozzer  
Data: 12/04/2024

## Trabalho 3- Transformações Geométricas

### Ferramentas:

Linguagem C++, utilizando a API Canvas2D (disponível no site da disciplina) e IDE Code::Blocks, compilando com MinGW 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.

A Canvas2D pode ser customizada, porém não é permitido o uso de funcionalidades OpenGL que não estão presentes na Canvas2D (ex: texturas, shaders). Pode-se criar novas sobrecargas de funções, novos métodos, classes, enums, templates, etc.

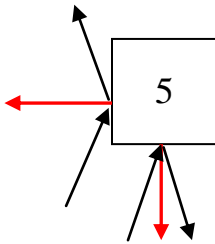
### Descrição:



Link:

[https://play.google.com/store/apps/details?id=ballz.brick.breaker.ballz.bounce.free&hl=en\\_US&pli=1](https://play.google.com/store/apps/details?id=ballz.brick.breaker.ballz.bounce.free&hl=en_US&pli=1)

O jogo Balls Bounce (e vários outros similares) consiste em disparar um “projétil” com um canhão contra blocos quadrados (pra simplificar), que pode ser representado como 4 segmentos de reta. Quando o projétil colide com um bloco (ou linha), ele deve refletir com o mesmo ângulo em relação ao vetor normal (ou em relação a superfície), como mostrado na figura abaixo. Ele pode colidir com os 4 lados do quadrado.



Quando o projétil colidir com a linha de baixo do jogo, ele é descartado. O projétil pode colidir com vários quadrados até sair do jogo. A cada colisão, o valor do quadrado é reduzido em 1. Quando chegar a zero, o quadrado desaparece (ou explode de uma forma bem bonita). A cada jogada do usuário, uma nova fileira de quadrados é introduzida na parte de cima. A cada novo disparo, aumenta-se em 1 o número de projéteis, ou seja, no primeiro disparo 1 projétil, no segundo 2, e assim por diante. A posição do canhão pode ser o local onde o último disparo saiu da tela.

A direção de disparo deve ser controlada com o mouse.

Para algoritmos de colisão com linhas, veja o demo `gl_4_intersection` do material de computação gráfica avançada, ou elabore outra solução.

Procure baixar e jogar o jogo para ver a dinâmica de funcionamento. Após, desinstalar e somente jogar a versão de vocês.

#### **Critérios que serão avaliados (básicos):**

- Classes em C++ para definição de vetores
- Sistema do controle do canhão
- Sistema de movimentação do projétil (ou rajadas).
- Colisão do projétil com os alvos
- Proibido colocar anúncios entre cada fase do jogo
- Controle de FPS (controle real da velocidade da animação)

#### **Critérios avançados de interação (Nota acima de 9.0):**

- Explosões com partículas (até 2 pontos)
- Salvamento do estado atual do jogo, com tela de pause (até 1 ponto). Salvar o estado no jogo na pasta `saves`, no mesmo nível de hierarquia da pasta `images`.

- Powerups (até 1 ponto)
- Menu inicial (até 0,5)
- Criação de várias fases do jogo (até 1 ponto). Muda-se de fase quando todos os quadrados são removidos.
- Outros tipos de primitivas, além de quadrados (0.5 pontos).
- Formas de interação dos projéteis com os cantos dos quadrados.
- 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.

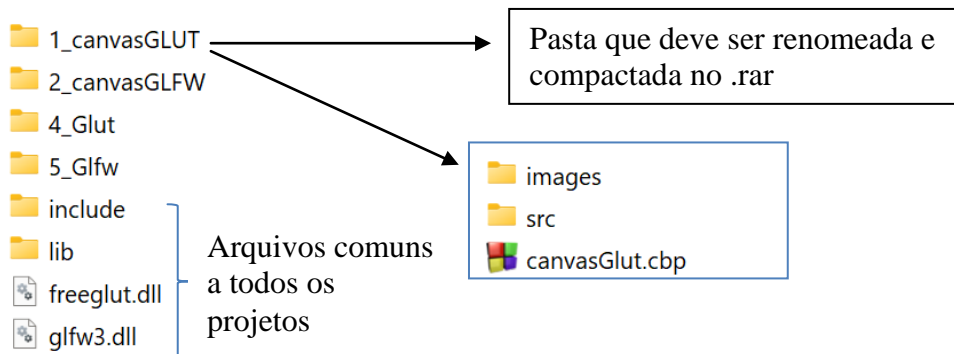
O trabalho deve ser individual. Dúvidas entrem em contato com o monitor ou comigo. A avaliação também poderá ser oral (a critério do professor).

### Passos:


1. Tenha certeza que entendeu bem vetores antes de iniciar. Para isso, faça todos os exercícios passados nos lab 2 e 3.
2. Programa para testar ângulo entre vetores, em todas as configurações possíveis
3. Programa para calcular o vetor de reflexão
4. Programa para testar colisão com as linhas.
5. Após tudo isso, elabore o jogo e bom divertimento.

### Formato de Entrega:

- O trabalho deve ser entregue pelo Google Classroom.
- Deve-se utilizar como base o projeto 1\_canvasGlut disponível nos demos da disciplina, como ilustrado na seguinte figura.



- A pasta 1\_canvasGlut tem todos os códigos fonte e recursos (images, src e projeto). Esta pasta **deve ser renomeada** com o nome do aluno. Ex: Trab1Maria, Trab2Paulo, Trab3Pedro, Trab4JoaoPedro, etc. Esta estrutura vai facilitar a execução e correção dos trabalhos. 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. **Os caminhos relativos para as pastas include, lib e para as dlls (que são os arquivos comuns) devem ser mantidos, e no padrão Windows.**
- 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.
- O trabalho será compilado em Windows 

### **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. Utilizar diferentes arquivos para diferentes classes.
- 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).
- Você pode discutir estratégias e ajudar o colega na implementação, porém evite passar código fonte. Programas semelhantes terão a nota 0 (zero).
- Compilação: programas que não compilarem/linkarem perderão muita nota.