🎬 ROTEIRO DE APRESENTAÇÃO

---

1. Introdução (30s - 1min)

Falar:

Olá, meu nome é Giovani Cancherini e vou apresentar meu projeto desenvolvido para o Trabalho 1 da disciplina de Computação Gráfica. O objetivo principal era criar uma aplicação 2D interativa utilizando OpenGL, com animação baseada em dados reais. Para isso, foi disponibilizado o "Cultural Dataset", onde utilizei os arquivos da subpasta Brazil. Este contém arquivos com trajetórias reais de pessoas extraídas de vídeos gravando o saguão do prédio 32 do alto.

- Objetivo: "Meu objetivo foi representar visualmente o movimento de pessoas detectadas em vídeo, animar suas trajetórias e permitir a interação com uma entidade controlável."

Mostrar:

- Mostrar o diretório de arquivos e o `Paths_D.txt` que foi utilizado.
- Explicar a estrutura dos dados brevemente: `[XXX]`, quantidade de frames, coordenadas (X,Y,F).

---

2. Explicando o Pipeline 2D (1 min)

Falar:

- "O pipeline gráfico 2D foi seguido nesta ordem:"
    - **SRO (Sistema de Referência do Objeto)**: entidades como quadrados e triângulo são definidas em coordenadas próprias.
    - **SRU (Sistema de Referência do Universo)**: aplicamos transformações para posicionar os objetos na cena.
    - **SRW (Sistema de Referência do Mundo)**: convertendo coordenadas baseadas no valor inicial `[XXX]` do arquivo.
    - **Recorte 2D**: o viewport limita a visualização da cena.
    - **SRV e SRD**: conversão para viewport (viewing) e dispositivo.

Mostrar:

- Explicar com o exemplo do triângulo ou dos quadrados animados.
- PIPELINE 2D ABAIXO

1. SRO → SRU: Aqui seria onde você transforma objetos para o mundo

DADOS = posição das entidades no arquivo

2. SRU/SRW: Você calcula os limites reais do mundo a partir das entidades

maiorValorX, menorValorX, etc. = define o mundo visível (SRW)

3. Clipping/SRV:

gluOrtho2D(left, right, bottom, top) = define a área que vai ser visível (SRV)

4. SRD: OpenGL faz isso automaticamente com base no tamanho da janela

---

3. Explicação da Estrutura de Código (1min)

Falar:

- "Reaproveitei métodos da `camera.py`, como: `main`, `Inicializa`, `Teclado`, `TeclasEspeciais`, `Animacao`, `Desenha` e `DesenhaQuadrilatero`."
- "Também criei as classes `Ponto`, `Quadrado` e `RGB` para facilitar a organização de coordenadas e cores."

Mostrar:

- Mostrar o cabeçalho do código com os `imports`, classes `Ponto`, `RGB`, `Quadrado`.
- Falar sobre o reaproveitamento como algo positivo (bom uso de código base da disciplina).

---

4. Visualização e Animação (1min)

Falar:

- "A animação representa pelo menos 6 entidades se movimentando com base nas coordenadas do dataset."
- "Cada quadrado representa uma pessoa detectada em vídeo, e sua posição é atualizada a cada frame."

Mostrar:

- A execução da animação.
- A função `desenhaQuadrilatero`, onde a posição e cor do quadrado são aplicadas.

---

5. Processamento: Detecção de Colisão (1min)

Falar:

- “Implementei um sistema de detecção de colisão entre o triângulo controlável e os quadrados.”
- “Utilizei o método `boundingBox()` para verificar interseção entre as caixas dos objetos.”

Mostrar:

- Mostrar a função `boundingBox()` e a `verificarColisoes()`.
- Demonstrar a colisão ao vivo e como ela é identificada (ex: mudança de cor, print no console, etc).

---

6. Interação com Teclado e Câmera (1min)

Falar:

- “Implementei um sistema para mover o triângulo com setas e adicionei suporte a movimento diagonal (2 teclas pressionadas).”
- “Usei IA para descobrir como detectar múltiplas teclas pressionadas com `glutGetModifiers()`.”
- “Também é possível mover a câmera pressionando `CTRL` junto com as setas.”

Mostrar:

- Código com o `teclas_pressionadas` e os ifs com diagonais.
- Mostrar o trecho da câmera:

```
python
CopiarEditar
if glutGetModifiers() and GLUT_ACTIVE_CTRL:
    ...
```

---

7. Leitura Aleatória dos Arquivos (30s)

Falar:

- “Implementei também uma função que aleatoriza o arquivo `Paths_D` usado, listando os arquivos do diretório com `os.listdir`.”

Mostrar:

- Mostrar o trecho:

```
python
CopiarEditar
for item in os.listdir(base_path):
    ...
```

---

8. Encerramento e Considerações Finais (30s)

Falar:

- “Concluindo, desenvolvi uma aplicação em Python com OpenGL que lê e anima dados reais, implementa interação com usuário e lógica de detecção de colisão.”
- “Utilizei conceitos de transformação 2D, reaproveitei estrutura fornecida e complementei com soluções via IA para teclas e movimentação.”

Mostrar:

- A execução final do programa em funcionamento.
- A movimentação da câmera e do triângulo.
- Exemplo de colisão.

🎤 DICAS EXTRAS:

- Grave sua tela e vá pausando/resumindo conforme cada item.
- Se possível, use um mouse highlighter ou zoom para destacar partes do código.
- Fale pausadamente e com clareza, explicando “o porquê” de cada parte, não só “o que” ela faz.