Atividade 2: Programação Gráfica Interativa com WebGL

Discente: Nicolas Expedito Lana Mendes
Matrícula: 22.1.4028
Docente: Rafael Alves Bonfim de Queiroz
Universidade Federal de Ouro Preto
Departamento de Computação
BCC327 - Computação Gráfica

Fevereiro de 2025

1 Introdução

Este relatório descreve a implementação de uma aplicação gráfica interativa utilizando a API WebGL, conforme solicitado na Atividade 2 da disciplina BCC327 - Computação Gráfica. O objetivo foi criar um triângulo verde que pode ser movido com o mouse e redimensionado com a roda, explorando eventos, manipulação em tempo real e interatividade. A escolha do WebGL foi motivada por sua integração com eventos DOM e suporte a transformações gráficas em navegadores.

2 API Utilizada

A API empregada foi o **WebGL**, baseada em OpenGL ES, projetada para renderização 2D e 3D em navegadores. Ela suporta shaders programáveis e permite manipulação dinâmica de objetos gráficos por meio de uniforms e eventos JavaScript.

3 Trechos de Código

A seguir, são apresentados trechos relevantes do código implementado:

3.1 Definição dos Shaders

Os shaders incluem transformações de translação e escala:

```
const vertexShaderSource = `
    attribute vec2 a_position;
    uniform vec2 u_translation;
    uniform float u_scale;
    void main() {
        vec2 scaledPosition = a_position * u_scale;
```

```
        vec2 translatedPosition = scaledPosition + u_translation;
        gl_Position = vec4(translatedPosition, 0.0, 1.0);
    }
`;
const fragmentShaderSource = `
    precision mediump float;
    void main() {
        gl_FragColor = vec4(0.0, 0.0, 1.0, 1.0); // Cor verde
    }
`;
```

3.2 Interatividade com Eventos

Os eventos de mouse controlam a posição e escala:

```
canvas.addEventListener("mousedown", (event) => {
    isDragging = true;
    const [x, y] = getWebGLCoordinates(event.clientX, event.
       clientY);
    translation = [x, y];
    drawScene();
});

canvas.addEventListener("wheel", (event) => {
    event.preventDefault();
    scale += event.deltaY * -0.001;
    scale = Math.max(0.1, Math.min(scale, 2.0));
    drawScene();
});
```

4 Instruções de Execução

Para executar a aplicação:

1. Salve o código em um arquivo `.html`.
2. Abra o arquivo em um navegador web moderno (como Chrome ou Firefox) com suporte a WebGL.
3. Clique e arraste o triângulo para movê-lo; use a roda do mouse para ajustar seu tamanho.

5 Funcionalidade

A aplicação inicializa um contexto WebGL e renderiza um triângulo verde com fundo preto. O vertex shader aplica translação e escala via uniforms, atualizados em tempo real por eventos de mouse. O usuário pode arrastar o triângulo para qualquer posição na tela e alterar sua escala entre 0.1x e 2x usando a roda do mouse. Buffers otimizam a renderização, e a função `drawScene` é chamada para refletir as mudanças.

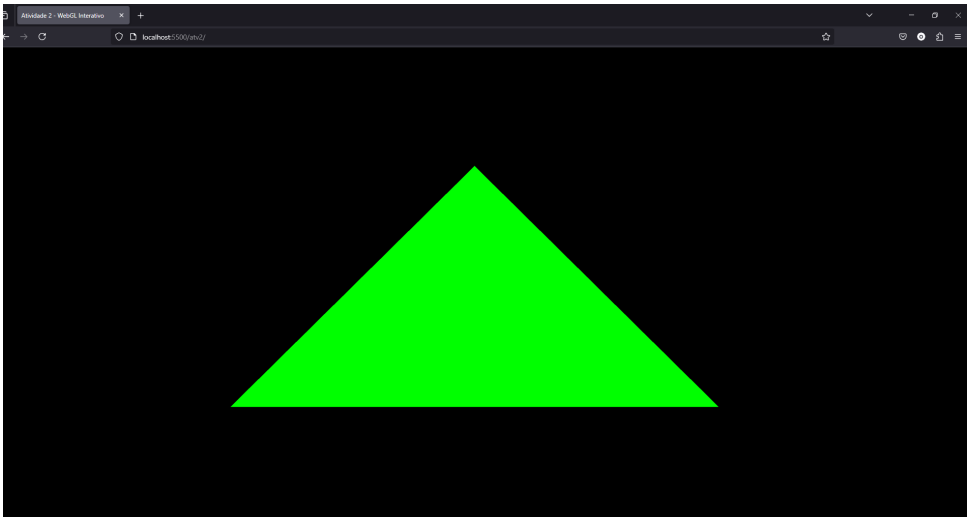

Figure 1: Triângulo verde interativo com fundo preto, posicionado no centro inicialmente.

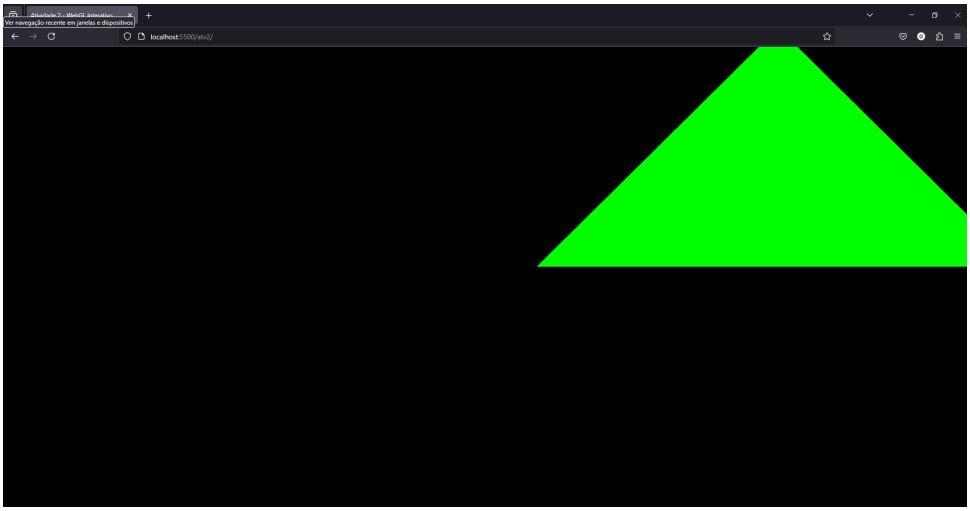

Figure 2: Triângulo verde interativo com fundo preto, posicionado no canto superior direito, após movimentação do cursor do mouse.

6 Conclusão

A atividade proporcionou uma exploração prática de programação gráfica interativa com WebGL, destacando o uso de eventos de mouse, transformações em tempo real e shaders dinâmicos. O resultado foi uma aplicação que permite ao usuário manipular um triângulo de forma intuitiva, atendendo aos requisitos de interatividade do enunciado. Este exercício reforçou a compreensão de como integrar entrada do usuário com renderização gráfica.