Efeito Bloom

Alunos: Matheus Baron Lauritzen, Gustavo Baron Lauritzen, Gabriel Bósio e Eduardo da Rocha Weber

Artigo

Nome: Efeitos de Iluminação Realistas em Realidade Aumentada Utilizando Imagens HDR
Autores: Moura, G. S., Teixeira, J. M. X. N., Teichrieb, V., Kelner, J.

Link:
https://www.gprt.ufpe.br/grvm/wp-content/uploads/Publication/ShortPapers/2007/_WRVA2007_Mouraetal.pdf

EFEITOS DE ILUMINAÇÃO REALISTAS EM REALIDADE AUMENTADA UTILIZANDO IMAGENS HDR

Guilherme de S. Moura, João Marcelo X. N. Teixeira, Veronica Teichrieb, Judith Kelner
Universidade Federal de Pernambuco, Centro de Informática
Grupo de Pesquisa em Realidade Virtual e Multimídia
Av. Professor Moraes Rego S/N, Prédio da Positiva, 1º Andar
Cidade Universitária, Recife - Pernambuco
CEP 50670-901
e-mail: {gsm, jmxnt, vt, jk}@cin.ufpe.br

Área de Implementação do Artigo

O artigo se insere na área da Realidade Aumentada (RA), com foco em Computação gráfica e renderização em tempo real para propor uma aplicação capaz de renderizar efeitos de iluminação realísticos. O intuito do trabalho é utilizar estes efeitos em Realidade Aumentada, de forma a auxiliar a inserção de objetos virtuais em cenários reais de forma visualmente suave. O documento descreve a implementação de alguns destes efeitos.

Algoritmos e Técnicas Utilizadas no Artigo

- Bloom;
- Dazzling Reflection;
- Exposure Control (Tone Mapping);
- Glare;

Dazzling Reflection

Exposure Control (Tone Mapping)

Glare

Implementação Proposta: Efeito Bloom

O efeito Bloom simula o ofuscamento da luz, onde ela parece ultrapassar os limites dos objetos, fenômeno que ocorre devido ao espalhamento do raio luminoso no olho humano. Para aplicações em tempo real, sua simulação envolve a aplicação de filtros gaussianos sucessivos nas áreas mais luminosas da cena, desfoque que é então combinado com a imagem original. Em Realidade Aumentada, o Bloom é crucial para conferir realismo e evitar que objetos virtuais pareçam artificiais, integrando-os de forma convincente com a iluminação do ambiente.

Configurações do OpenGL e Janela;

Declaração dos Shaders;

Configuração de Framebuffers;

Configuração Inicial dos Shaders;

Variáveis de Controle do Bloom;

```
glfwInit();
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "Efeito Bloom com Toro", NULL, NULL);
if (window == NULL)
{
    std::cout << "Falha ao criar a janela GLFW" << std::endl;
    glfwTerminate();
    return -1;
}
glfwMakeContextCurrent(window);

if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
{
    std::cout << "Falha ao inicializar GLAD" << std::endl;
    return -1;
}

glEnable(GL_DEPTH_TEST);
glEnable(GL_CULL_FACE);

// --- Declaração dos Shaders ---
Shader sceneShader("scene.vert", "scene.frag");
Shader brightPassShader("quad.vert", "bright_pass.frag");
Shader blurShader("quad.vert", "blur.frag");
Shader compositeShader("quad.vert", "composite.frag");

// --- Configuração dos Framebuffers ---
setupFramebuffers();


brightPassShader.use();
brightPassShader.setInt("sceneTexture", 0);

blurShader.use();
blurShader.setInt("image", 0);

compositeShader.use();
compositeShader.setInt("sceneTexture", 0);
compositeShader.setInt("bloomTexture", 1);

// Variáveis para controle do Bloom
bool enableBloom = true;
float bloomExposure = 8.0f;
float bloomThreshold = 0.8f;
```

Renderiza a Cena em HDR;

Extrai o Brilho (Bright Pass)

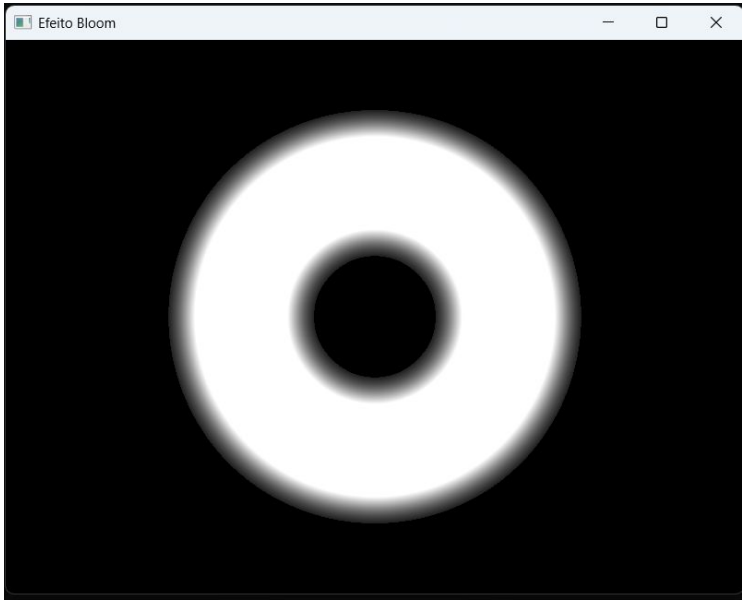
```
// --- Loop de Renderização Principal ---
while (!glfwWindowShouldClose(window))
{
    // --- 1. Renderiza a cena para o FBO principal (hdrFBO) ---
    glBindFramebuffer(GL_FRAMEBUFFER, hdrFBO);
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glm::mat4 projection = glm::perspective(glm::radians(45.0f), (float)SCR_WIDTH / (float)SCR_HEIGHT, 0.1f, 100.0f);
    glm::mat4 view = glm::lookAt(glm::vec3(0.0f, 0.0f, 5.0f), glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f));

    sceneShader.use();
    sceneShader.setMat4("projection", projection);
    sceneShader.setMat4("view", view);

    // --- Matriz de transformação global para o objeto ---
    glm::mat4 objectModel = glm::rotate(glm::mat4(1.0f), glm::radians(180.0f), glm::vec3(1.0f, 0.0f, 0.0f));

    // --- Renderiza o objeto ---

    // Cor do objeto (brilhante para o bloom)
    sceneShader.setVec3("objectColor", glm::vec3(25.0f, 25.0f, 25.0f));

    sceneShader.setMat4("model", objectModel);
    glCullFace(GL_BACK);
    renderSphere();

    glBindFramebuffer(GL_FRAMEBUFFER, 0); // Volta para o framebuffer padrão após renderizar a cena


    // --- 2. Extrai o brilho da cena ---
    brightPassShader.use();
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, colorBuffer);
    brightPassShader.setFloat("bloomThreshold", bloomThreshold);
```

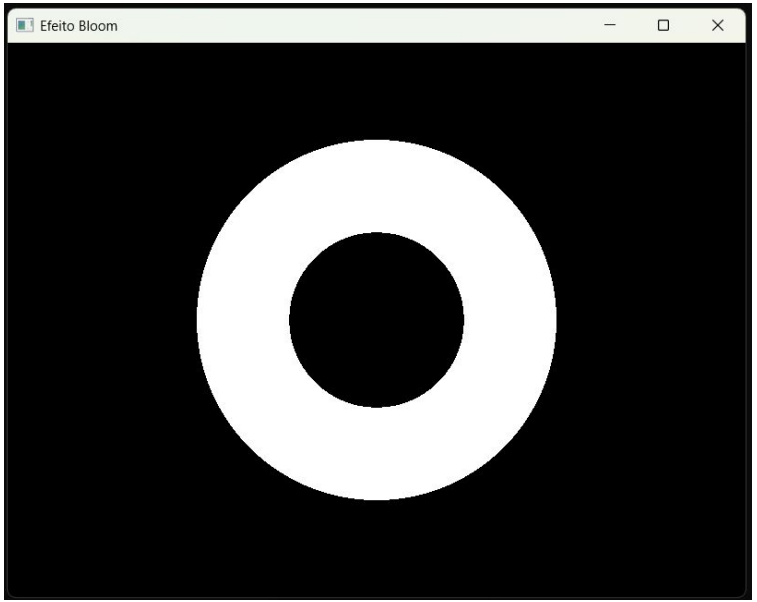
Aplica o Desfoque Gaussiano (Gaussian Blur);

Renderiza a Imagem Final na Tela;

```
        glBindFramebuffer(GL_FRAMEBUFFER, pingpongFBO[0]);
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        renderQuad();


        // --- 3. Aplica o Gaussian Blur ---
        bool horizontal = true;
        unsigned int amount = 40;
        blurShader.use();
        for (unsigned int i = 0; i < amount; i++)
        {
            glBindFramebuffer(GL_FRAMEBUFFER, pingpongFBO[horizontal]);
            blurShader.setBool("horizontal", horizontal);
            glActiveTexture(GL_TEXTURE0);
            glBindTexture(GL_TEXTURE_2D, i == 0 ? pingpongColorbuffers[0] : pingpongColorbuffers[!horizontal]);
            renderQuad();
            horizontal = !horizontal;
        }


        // --- 4. Renderiza a imagem final na tela ---
        glBindFramebuffer(GL_FRAMEBUFFER, 0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        compositeShader.use();
        glActiveTexture(GL_TEXTURE0);
        glBindTexture(GL_TEXTURE_2D, colorBuffer);
        glActiveTexture(GL_TEXTURE1);
        glBindTexture(GL_TEXTURE_2D, pingpongColorbuffers[!horizontal]);

        // Aplica o exposure com base na flag enableBloom
        compositeShader.setFloat("exposure", enableBloom ? bloomExposure : 0.0f);
        renderQuad();


        glfwSwapBuffers(window);
        glfwPollEvents();
    }

    glfwTerminate();
    return 0;
```

Resultado





Obrigado pela atenção!