Relatório Trabalho 3.1 - Computação Gráfica

Matheus Baron Lauritzen[1], **Gustavo Baron Lauritzen**[1], **Gabriel Bósio**[1], **Eduardo da Rocha Weber**[1]

[1]Escola Politécnica – Ciência da Computação – Universidade do Vale do Itajai (UNIVALI)
Itajaí – Santa Catarina – SC – Brasil

Resumo. *Este trabalho consiste em um relatório e uma implementação prática sobre efeitos de iluminação realistas em Realidade Aumentada (RA), inspirado no artigo de Moura et al., "Efeitos de Iluminação Realistas em Realidade Aumentada Utilizando Imagens HDR". O relatório aborda os autores, a área de implementação e os principais algoritmos do artigo, como imagens High Dynamic Range (HDR) e renderização em GPU. Detalhamos efeitos como Bloom, Dazzling Reflection, Exposure Control e Glare, que visam integrar objetos virtuais em cenas reais de forma convincente. Como parte prática, implementaremos o efeito Bloom, que simula o ofuscamento da luz aplicando filtros gaussianos sucessivos em pontos luminosos da cena e os compondo com a imagem original. O objetivo é demonstrar como a iluminação realista aprimora a integração visual de objetos virtuais em aplicações gráficas interativas.*

Nome do Artigo

Efeitos de Iluminação Realistas em Realidade Aumentada Utilizando Imagens HDR

Autores do Artigo

Moura, G. S., Teixeira, J. M. X. N., Teichrieb, V., Kelner, J.

Área de Implementação do Artigo

O artigo se insere na área da **Realidade Aumentada (RA)**, focando na **computação gráfica** e **renderização em tempo real** para alcançar **iluminação foto-realista utilizando imagens HDR (High Dynamic Range)**. O objetivo principal é a integração visualmente aceitável de objetos virtuais em cenários reais, buscando a fusão convincente para aumentar a imersão e interatividade do usuário.

Algoritmos e Técnicas Utilizadas no Artigo

O artigo explora algoritmos e técnicas avançadas para efeitos de iluminação realistas em RA, fundamentados em:

- **Imagens HDR (High Dynamic Range):** Capturam um amplo espectro de luz para simular fenômenos complexos, utilizando mapas de radiância e mapeamento esférico de texturas.
- **Bloom:** Simula o espalhamento da luz de fontes intensas através da aplicação de filtros gaussianos sucessivos.
- **Dazzling Reflection:** Efeito de ofuscamento gerado por reflexões de fontes de luz intensas em objetos virtuais, preservando a informação HDR.
- **Exposure Control (Tone Mapping):** Mapeia intensidades de luz HDR para uma faixa visível (LDR), simulando a adaptação do olho humano.
- **Glare:** Simula arestas radiais de fontes de luz intensas, utilizando Transformada Rápida de Fourier (FFT) otimizada para GPU para processamento em tempo real.

Conclusões e Observações sobre o Artigo

O artigo valida a incorporação de efeitos de iluminação realistas em Realidade Aumentada utilizando imagens HDR. A principal contribuição é a demonstração da viabilidade de execução desses efeitos *em tempo real* por meio do uso intensivo de *shaders* na GPU. Os resultados mostram que é possível gerar efeitos visuais convincentes e interativos com as tecnologias da época (2007), como Bloom e Dazzling Reflections. As observações detalhadas incluem:

- A capacidade de aplicação dos efeitos em tempo real é confirmada, destacando a eficiência do processamento paralelo das placas gráficas, fundamental para manter a interatividade do usuário.
- A escolha dos **filtros gaussianos** para a simulação do Bloom é ressaltada como uma solução computacionalmente eficiente. Sua natureza separável (aplicação horizontal e vertical) permite otimizações significativas no desempenho.
- A abordagem arquitetural do programa é notavelmente **modular**. O sistema é estruturado em vários passos de pós-processamento, onde cada efeito ou parte dele é implementado por um ou mais desses passos. Embora os passos de um mesmo efeito possam ter dependência sequencial, efeitos distintos geralmente operam de forma independente, conferindo grande flexibilidade e capacidade de combinação.
- É reconhecido pelos próprios autores que a implementação inicial do Bloom, apesar de funcional e demonstrativa do conceito, não alcança o grau de precisão e sutileza visuais desejados. Sugere-se que a composição de resultados intermediários do desfoque poderia aprimorar a qualidade final do efeito, indicando um caminho para futuras melhorias.
- Uma limitação técnica apontada para o mapeamento esférico é a **perda de informação nas bordas do mapa**, o que pode resultar em distorções perceptíveis em certas áreas do mundo virtual. Isso sugere a necessidade de técnicas mais avançadas para a captura e representação de ambientes HDR.
- No momento da escrita do artigo, a implementação de outros efeitos de iluminação, como Glare e Exposure Control, havia sido iniciada mas ainda não apresentava resultados visíveis ou totalmente concluídos, indicando que eram áreas em desenvolvimento.
- Como trabalhos futuros, o artigo aponta a necessidade de integrar a iluminação com dados reais do ambiente para uma **iluminação adaptativa** mais robusta, o que seria um avanço significativo para a imersão em RA. Além disso, a realização de um **estudo minucioso sobre o desempenho** do shader desenvolvido é considerada essencial para validar a escalabilidade e a otimização da solução proposta.

Em suma, o artigo serve como um marco importante ao demonstrar a viabilidade da integração de efeitos de iluminação realistas baseados em HDR em ambientes de Realidade Aumentada em tempo real, utilizando a capacidade de processamento paralelo das GPUs através de shaders customizados. Embora algumas limitações e trabalhos futuros sejam reconhecidos, a pesquisa pavimenta o caminho para experiências de RA cada vez mais imersivas e visualmente convincentes.

Implementação Proposta: Efeito Bloom

A implementação do efeito Bloom foi desenvolvida em **C++ utilizando a API gráfica OpenGL**, com o auxílio das bibliotecas `GLAD` para carregamento das extensões OpenGL

e GLFW para gerenciamento da janela e contexto. Esta abordagem de baixo nível permitiu um controle preciso sobre o pipeline de renderização e a validação dos conceitos de pós-processamento, conforme descrito na literatura.

O pipeline de renderização do Bloom é estruturado em quatro etapas principais, cada uma delas fazendo uso intensivo de *Framebuffer Objects (FBOs)* e *shaders GLSL* personalizados:

- **Renderização da Cena em HDR:** Inicialmente, a cena principal é renderizada em um *Framebuffer Object* dedicado (hdrFBO). A saída de cores desta etapa é armazenada em uma textura de ponto flutuante (colorBuffer) com formato GL_RGBA16F, essencial para capturar uma ampla gama dinâmica de cores (HDR) e evitar o *clipping* de valores de brilho que excedem o intervalo de 0 a 1. O objeto principal demonstrado, uma forma de "donut" (*toro*), é gerado parametricamente dentro da função renderSphere(), permitindo controle sobre seus raios maior (R_TORUS) e menor (r_TORUS). O shader scene.vert e scene.frag são responsáveis por esta etapa.
- **Extração de Brilho (*Bright Pass*):** Após a renderização da cena, um *shader* específico (bright_pass.frag) é aplicado. Este *shader* processa a textura HDR gerada na etapa anterior e isola apenas os pixels cujos valores de brilho (luminância) excedem um *limiar* (bloomThreshold). O resultado desta filtragem, contendo as regiões de alta iluminação da cena, é armazenado em um dos *framebuffers* de "ping-pong" (pingpongFBO[0]). O quad.vert é utilizado como *vertex shader* para todas as etapas de pós-processamento, renderizando um quadrilátero que preenche a tela.
- **Desfoque Gaussiano (*Gaussian Blur*):** A textura contendo as regiões brilhantes é então submetida a um processo de desfoque gaussiano iterativo. Utilizando um par de *framebuffers* de "ping-pong" (pingpongFBO[0] e pingpongFBO[1]) e o *shader* blur.frag, o desfoque é aplicado alternadamente nas direções horizontal e vertical. Este método em múltiplos passes (amount de iterações, configurável) cria um efeito de desfoque suave e uniforme, evitando a necessidade de um kernel de desfoque muito grande e computacionalmente caro em um único passe.
- **Composição Final:** Na etapa final, o *shader* composite.frag é empregado para combinar a textura da cena original (colorBuffer) com a textura das regiões brilhantes desfocadas (pingpongColorbuffers[!horizontal]). A intensidade do efeito Bloom é controlada por um parâmetro de exposição (bloomExposure), que pode ser ajustado para otimizar o resultado visual. Uma *flag* booleana (enableBloom) também permite ativar ou desativar o efeito completamente, oferecendo flexibilidade para o usuário.

Os parâmetros como o limiar de brilho (bloomThreshold), a intensidade (bloomExposure) e o número de passes de blur (amount) são expostos como *uniforms* nos *shaders* correspondentes, permitindo ajustes em tempo real para personalizar o efeito visual.

Observação: A parte prática com o código-fonte C++ e GLSL completo será adicionada como anexo ao relatório assim que finalizada.

Referência

[1] Moura, G. S., Teixeira, J. M. X. N., Teichrieb, V. e Kelner, J. ”Efeitos de Iluminação Realistas em Realidade Aumentada Utilizando Imagens HDR”. In *Anais do Workshop de Realidade Virtual e Aumentada (WRVA)*, 2007.