FRAMEWORK PARA O ENSINO DE COMPUTAÇÃO GRÁFICA: ABSTRAÇÃO DA COMPLEXIDADE DE SOFTWARE NO CONTEXTO DO ENSINO SUPERIOR

LAMAS, C.H.S.[1]; OLIVEIRA, E.S.[2]; ROCHA, F.A.[2].

[1]Docente do IFNMG – *Campus* Salinas; [2]Discentes do Bacharelado em Sistemas de Informação do IFNMG – *Campus* Salinas.

Introdução

A humanidade, ao longo de sua história, sempre usou formas de grafismo para se expressar, desde pinturas rupestres até mapas, desenhos e outras formas de grafismo. Com o advento da computação, surgiram novas tecnologias que ampliaram as ferramentas disponíveis para a representação gráfica, potencializando a expressividade humana através dos meios digitais. Essas tecnologias se agrupam em torno do conceito de computação gráfica, um domínio que trata do uso de gráficos de computador. A computação gráfica é amplamente utilizada no desenvolvimento de software e em diversas indústrias, especialmente em aplicativos que servem como ferramentas de desenho auxiliado por computador (CAD), muito usados nas engenharias e desenvolvimento de produtos. Também é utilizada em áreas da cultura como cinema, videogames e indústria fonográfica. No entanto, o campo da computação gráfica enfrenta dificuldades na formação de mão de obra especializada (RODRIGUES et al., 2021), devido à exigência de múltiplas habilidades e conhecimentos para progredir nos estudos do campo, à complexidade das *APIs* (*Application Programming Interfaces*) de software atualmente disponíveis e à necessidade de conhecimento matemático, especialmente nos campos da álgebra linear e geometria analítica, física, entre outros (DODGSON, 2017).

Além disso, a principal dificuldade, para os estudantes, é a complexidade do software observada nas *APIs* atualmente disponíveis, como a *OpenGL*, *Vulkan*, *DirectX* ou *Metal* (DOPPIOSLASH, 2017). As rotinas de software dessas *APIs* foram concebidas para dar ao desenvolvedor acesso e controle direto aos recursos de Hardware das *GPUs* (*Graphical Processing Unities*) e suas funções de software não mapeiam adequadamente esses recursos de hardware aos conceitos mais fundamentais da teoria do campo da computação gráfica. Isso torna o estudo muito difícil, pois responsabiliza o estudante por realizar ele mesmo esse mapeamento enquanto estuda simultaneamente os conceitos do campo da computação gráfica e as funcionalidades das *APIs*.

O trabalho de pesquisa relatado tem como objetivo geral implementar uma *framework* que facilite e amplie as ferramentas para o ensino de computação gráfica no ensino superior, fornecendo uma camada de abstração sobre as *APIs* gráficas atualmente disponíveis. Para alcançar esse objetivo, alguns objetivos secundários precisarão ser observados. Isso inclui a investigação das *APIs* gráficas atualmente disponíveis para o desenvolvimento de software de computação gráfica, a concepção do desenho de uma estrutura que funcione sobre essas *APIs*, a implementação do código-fonte da estrutura que será expressa no formato de uma biblioteca de software para uso nos softwares clientes e a elaboração de documentação e material didático para acompanhar o uso da estrutura pelos estudantes. Por fim, com a pesquisa já em curso esperamos contribuir significativamente com o arcabouço formativo sobre computação gráfica, reduzindo as barreiras de entrada e promovendo um maior entendimento e domínio dessa área crucial nas ciências da computação e nas indústrias criativas.

Material e Métodos

A execução do trabalho proposto está sendo realizada em etapas distintas e demandará a triangulação de métodos. Inicialmente, foi conduzida uma investigação do estado da arte das *APIs* gráficas atualmente disponíveis para o desenvolvimento de software proposto. Nesta etapa, realizamos uma revisão sistemática da documentação disponível sobre essas *APIs*, permitindo uma comparação entre elas para decidirmos sobre quais *APIs* iniciamos nosso trabalho de implementação e quais serão descartadas do projeto, ao menos inicialmente. O corpus desta revisão é constituído pelas próprias documentações das *APIs* disponibilizadas por suas páginas mantenedoras na internet. Em seguida, concebemos a estrutura da nossa *framework* como um software que forneça uma camada de abstração, mapeando os construtos de código da API escolhida na etapa anterior e traduzindo-os em classes de objetos e chamadas de funções mais diretamente compreendidas e relacionadas aos conceitos tradicionalmente discutidos no ensino da computação gráfica. Finalmente, estamos implementando a *framework* concebida utilizando a linguagem de programação C#, para a criação do software (através de *bindings* das *APIs* para C#). Escolhemos essa linguagem dada a facilitade iniciarmos o trabalho de pesquisa utilizando a biblioteca OpenTK (SZABÓ, 2022) e posteriormente também a integração com a biblioteca Veldrid (NYLAND, 2020) que permitirá uma maior flexibilidade em termos de APIs. É importante salientar que o software está sendo desenvolvido no formato de uma biblioteca de software, ou seja, foi desenvolvido para ser carregado para uso por outros programas em desenvolvimento, que chamamos de softwares clientes. Durante todo esse processo de execução da nossa proposta de desenvolvimento, os eventos e fatos percebidos pelos pesquisadores estão sendo registrados em um diário do pesquisador, onde relatamos as atividades, dificuldades e descobertas durante o processo de execução da proposta. O objetivo dessa última ação é realizar um relatório da pesquisa que nos permita, por fim, avaliar nosso próprio fazer profissional como desenvolvedores e pesquisadores e contribuir para a bibliografia do campo.

Resultados e Discussão

Neste capítulo, apresentamos os resultados parciais obtidos ao longo do últimos meses de desenvolvimento da nossa *framework* e discutimos as implicações desses resultados para o ensino de computação gráfica no ensino superior. Para melhor compreensão, começaremos por abordar as funcionalidades implementadas e, em seguida, discutiremos os exemplos de casos de uso. Nossa *framework* foi projetada para simplificar o ensino de computação gráfica, fornecendo uma camada de abstração sobre as *APIs* gráficas tradicionais, como *OpenGL* e *DirectX*. Abaixo, detalhamos as principais funcionalidades implementadas, e seu impacto no processo de ensino e aprendizagem:

1. *Camera* (Câmera): Implementamos uma classe *Camera* que permite aos estudantes criar e controlar facilmente a posição e orientação da câmera em uma cena gráfica. Isso proporciona uma experiência mais intuitiva e visual para os estudantes em comparação com as complexas manipulações de matrizes de visão. Acreditamos que Ao eliminar a necessidade de lidar diretamente com matrizes de transformação, os estudantes podem direcionar mais tempo e energia para entender os conceitos fundamentais da computação gráfica, como projeção, perspectiva, câmera virtual e sistemas de coordenadas, além disso a implementação da classe em questão serve também como exemplo de implementação, que pode ser estendida e alterada para outros contextos.

2. *Texture* (Textura): Facilitamos o carregamento e aplicação de texturas aos objetos, permitindo aos estudantes enriquecer suas cenas com imagens e padrões. Em muitos casos, a textura pode ser usada para representar conceitos abstratos, como superfícies rugosas, materiais reflexivos, ou até mesmo informações científicas complexas. Tradicionalmente, a manipulação de texturas pode ser complexa, envolvendo carregamento de imagens, mapeamento de coordenadas UV e interação com *shaders*. Nossa *framework* abstrai grande parte dessa complexidade, tornando a aplicação de texturas mais acessível, especialmente para estudantes iniciantes.
3. *Light* (Luz): Implementamos um sistema de iluminação que permite a adição de diferentes tipos de luzes, como luz ambiente, luz direcional e luz pontual, facilitando a exploração dos conceitos de iluminação na computação gráfica. A iluminação desempenha um papel crucial na aparência de materiais e superfícies e, com as funcionalidades já implementadas de iluminação, os estudantes podem explorar como diferentes materiais reagem à luz, incluindo reflexão, refração e sombreamento. Isso ajuda a compreender a relação entre iluminação e textura.
4. *Shader* (Sombreadores): Simplificamos a criação e aplicação de *shaders*, que são peças fundamentais no processo de renderização gráfica, responsáveis por calcular como a luz interage com os objetos na cena. Ao simplificar a criação de *shaders*, nossa *framework* encoraja os estudantes a experimentar e serem criativos. Eles podem criar *shaders* personalizados para alcançar efeitos visuais específicos, como sombreamento suave, reflexões ou texturas procedurais, promovendo o desenvolvimento de habilidades de design gráfico.
5. *Shadow Map* (Mapeamento de Sombras): Implementamos o mapeamento de sombras para que os estudantes possam explorar os efeitos de sombra em suas cenas gráficas, o que é fundamental para a criação de ambientes realistas. Sombras desempenham um papel crucial na criação de ambientes realistas em gráficos 3D e envolve técnicas avançadas para determinar quais áreas de uma cena estão em sombra e quais estão iluminadas. A nossa framework abstrai parte dessa complexidade, permitindo que os estudantes se concentrem na compreensão dos princípios de sombreamento em vez de se perderem em detalhes técnicos.
6. *Transform* (Transformação): A funcionalidade de transformações simplifica a aplicação de transformações geométricas, como rotação, escala e translação, aos objetos da cena, esses são conceitos fundamentais na computação gráfica, mas podem ser complexos de compreender a princípio pois se baseiam principalmente em multiplicações de matrizes, ao permitir que os estudantes experimentem e visualizem os efeitos das transformações, nossa *framework* busca ajudar a facilitar a aprendizagem baseada em resultados visuais.

Além disso, para ilustrar o uso eficaz da nossa *framework* no ensino de computação gráfica, construímos alguns exemplos de casos de uso organizados na forma de capítulos, estes são limitados por hora a uma implementação para *OpenGL* através da biblioteca *OpenTK,* através desses exemplos os estudantes podem acompanhar como são realizadas algumas das tarefas mais comuns de computação gráfica. De forma breve, entre os exemplos já implementados está a criação janelas gráficas para exibir suas cenas, eliminando a complexidade envolvida na configuração de janelas usando as *APIs* tradicionais, adicionalmente implementamos funcionalidades de troca de *buffer* é simplificada, permitindo aos estudantes atualizar a tela de maneira eficiente após a renderização de uma cena.

Um outro exemplo mostra como os estudantes podem criar e renderizar um triângulo com apenas algumas linhas de código, permitindo uma introdução suave aos conceitos de geometria e renderização. Demonstramos também em outros exemplos a passagem de dados uniformes para os *shaders*, tornando mais fácil para os estudantes personalizar a aparência dos objetos, assim como o envio de atributos para os *shaders*, permitindo a criação de efeitos visuais personalizados baseados em dados. Há também exemplos sobre como criar índices para geometria indexada de forma mais intuitiva, economizando tempo e minimizando erros, assim como funcionalidades de malha simplifica a criação de modelos 3D complexos, permitindo que os estudantes se concentrem nos aspectos conceituais da modelagem.

Por fim, demonstramos como as transformações geométricas podem ser aplicadas de forma interativa, permitindo que os estudantes visualizem e compreendam melhor os efeitos das transformações. Há também exemplos de transformações com câmeras. Além disso incluímos uma demonstração de como simular luzes e trabalhar com a aplicação de vetores normais, permitindo aos estudantes lidar com efeitos avançados de iluminação de maneira mais eficaz. Os resultados apresentados nesta seção demonstram o esforço que vemos realizando em nossa *framework* para simplificar a complexidade da computação gráfica, tornando-a mais acessível aos estudantes. Acreditamos que as abstrações que produzimos sobre as *APIs* gráficas tradicionais já proporcionam uma transição suave do aprendizado dos conceitos teóricos para a aplicação prática, sem sobrecarregar os estudantes com detalhes técnicos desnecessários em um primeiro momento.

Além disso, a nossa *framework* promove uma abordagem mais orientada a objetos no desenvolvimento gráfico, incentivando boas práticas de programação e a compreensão dos relacionamentos entre os elementos da cena gráfica. No entanto, é importante ressaltar que nossa *framework* não substitui a necessidade de compreender as *APIs* gráficas tradicionais, uma vez que esses conhecimentos ainda são valiosos no mercado de trabalho, em vez disso, nossa *framework* serve como uma ferramenta educacional para facilitar a aprendizagem inicial e a experimentação prática.

Considerações finais

Acreditamos que a implementação da *framework* proposta representa um passo significativo na facilitação do ensino de computação gráfica, ao fornecer uma camada de abstração sobre as *APIs* gráficas existentes, no entanto, ainda há trabalho a ser feito, a documentação completa e detalhada da *framework*, e dos exemplos, é essencial para garantir que os estudantes possam usá-la efetivamente. Além disso, mais exemplos podem ser adicionados ao longo do tempo para cobrir uma gama mais ampla de conceitos e técnicas de computação gráfica.

Referências

DODGSON, Neil A.; CHALMERS, Andrew. Designing a Computer Graphics Course for First Year Undergraduates. In: **Eurographics (Education Papers)**. 2017. p. 9-15.

DOPPIOSLASH, Claudia. The Graphics Pipeline. In: **Physically Based Shader Development for Unity 2017**. Apress, Berkeley, CA, 2018. p. 33-42.

NYLAND, Liv Anne; ØIE, **Jonathan. Joint Company IT Workspace**. 2020. Trabalho de Conclusão de Curso. NTNU.

RODRIGUES, Rui et al. Computer Graphics teaching challenges: Guidelines for balancing depth, complexity and mentoring in a confinement context. **Graphics and Visual Computing**, v. 4, p. 200021, 2021.

SZABÓ, Dávid; ILLÉS, Zoltán. Real-time rendering with OpenGL and Vulkan in C#. In: **Recent Innovations in Computing: Proceedings of ICRIC 2021, Volume 2**. Singapore: Springer Singapore, 2022. p. 599-611.