# IF RENDER-EX: UM SOFTWARE PARA ENSINO DE COMPUTAÇÃO GRÁFICA

João Felipe Abrahan, Rodrigo Sanches Devigo, Karina Kristiane Vicelli

Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul – Dourados-MS

joaofelipeabrahan@hotmail.com, rodrigo.devigo@ifms.edu.br, karina.vicelli@ifms.edu.br

## Resumo

Este projeto teve como objetivo, o desenvolvimento de um software que auxilie os estudantes do Instituto Federal de Mato Grosso do Sul a compreender e colocar em prática de forma mais fácil os conceitos abordados nas disciplinas de Computação Gráfica baseados em uma API OpenGL. Para tal, o software desenvolvido utiliza as pipelines de render do OpenGl, e a API OpenTK cujo abstrai alguns requisitos para criação de janela Windows e loop render, facilitando assim o desenvolvimento e a utilização para o usuário final. O software permite ao usuário final, carregar um modelo 3D e os materiais com suas respectivas texturas (Diffuse, Normal, Specular) e utilizar o GLSL para aplicar shaders sobre um modelo 3D, assim como também permite a adição de pontos de luz e câmera.

**Palavras-chave:** Processamento Gráfico, Engenharia de Software, Linguagens de Programação.

## Introdução

A computação gráfica teve seu início em 1951 com Jay Forrester e Robert Everett por meio do Whirlwind, um computador capaz de processar e projetar imagens tridimensionais em monitores, a computação gráfica tem se expandido ao longo dos anos e é uma das principais influenciadoras no sucesso de grandes corporações como Pixar, Disney, Adobe, Nvidia, Autodesk entre outras.

A partir do início do século XXI, a demanda por desenvolvimento e aperfeiçoamento das áreas interligadas à Computação Gráfica torna-se cada vez mais perceptível aos desenvolvedores. O crescente mercado de desenvolvimento de jogos como demonstrado no 2o Censo da Indústria Brasileira de Jogos Digitais (2018), assim como a demanda por estudos e desenvolvimentos de novas técnicas para efetivar o realismo no processo de renderização em tempo real como abordado por Schneider (2015, p. 9, tradução nossa) “Nuvens CG realistas não são um osso duro de roer. Portanto, antes de tentarmos resolver todo o problema de criar um céu cheio deles, pensamos que seria bom explorar diferentes maneiras de criar e iluminar ativos de nuvem individuais.”, ou mesmo estudos relacionados ao cinema como mencionado por 3D (2017, p. 33, tradução nossa) “Nosso objetivo é preencher a lacuna entre os gráficos de qualidade cinematográfica e o fator de imersão fornecido pela visualização de uma cena 3D com precisão de qualquer ponto de vista.”.

A renderização é uma das técnicas mais utilizadas, e se encontra principalmente em áreas que demandam de arte visual como cinema e jogos digitais, podendo também ser encontradas nos setores da construção civil e simulação. O termo Renderização (Rendering) refere-se ao processo que permite obter uma imagem digital denominada render, resultante de cálculos que podem envolver iluminação, sombra, projeção, reflexo, sobreposição, entre muitos outros componentes. Existem dois tipos principais de renderização onde a diferença entre elas está relacionada a velocidade a qual as imagens são produzidas. A renderização em tempo real (Real-Time) é o tipo de renderização utilizado principalmente em jogos digitais e aplicações gráficas interativas, já que ambas necessitam de velocidade na produção do render, por outro lado existe a renderização pré-processada (Offline) utilizada em aplicações que não necessitam de alta velocidade de processamento na qual é gerado um arquivo de saída predefinida como um vídeo, a renderização Offline é muito utilizada nas indústrias cinematográficas e de engenharia civil.

Como mencionado anteriormente, é notável a importância da computação gráfica no mundo da computação, uma vez que se configura como uma das áreas em que a conexão entre a humanidade e a computação é estreita e tênue, assim como uma das áreas mais versáteis e com abrangência em diferentes setores. Por outro lado, uma das maiores dificuldades nesta área refere-se à renderização Real-Time, pois isso demanda de um alto grau de processamento computacional, assim como técnicas ou algoritmos que simulam o realismo presente na renderização Offline.

Visto a perspectiva crescente do mercado e a busca constante por estudos voltados ao aprimoramento e pesquisa neste setor, este projeto visa desenvolver um programa que possa servir como fundamentação introdutória de forma teórica e prática, para que os acadêmicos do curso Superior de Tecnologia em Jogos Digitais do IFMS *Campus* Dourados possam ter um melhor entendimento desta área assim como abrir oportunidades para futuras pesquisas dentro da área da computação gráfica.

## Metodologia

O desenvolvimento do projeto foi organizado em 4 etapas: Pesquisa bibliográfica, levantamento dos requisitos e análise do projeto da ferramenta, desenvolvimento da ferramenta e desenvolvimento do material didático de apoio para a ferramenta.

**Pesquisa bibliográfica:**

Nesta primeira etapa foi realizado uma ampla procura por materiais que pudessem ser úteis para auxiliar no entendimento dos conceitos iniciais de todo o fluxo de desenvolvimento de um software baseado em DirectX 11 junto ao Windows Forms. O primeiro passo proposto para pesquisa foi uma análise das ferramentas do Windows Forms e sua utilização junto a linguagem de programação C#, a princípio bons resultados foram obtidos, visto que o autor do projeto conseguiu desenvolver uma interface gráfica inicial.

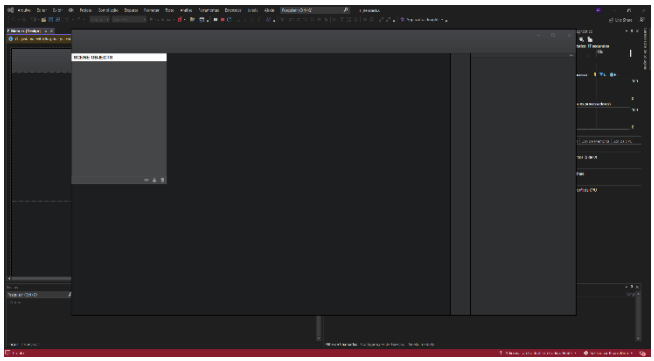

**Figura 1.** Interface gráfica desenvolvida inicialmente utilizando o Windows Forms.

Por mais que durante o primeiro momento de desenvolvimento, o conceito apresentasse bons resultados, ao tentarmos implementar o DirectX descobrimos um grande problema, a integração junto ao DirectX. A integração do DirectX era feita inteiramente em C++ o que impossibilitava a implementação do mesmo de forma fácil e rápida dificultando sua integração ao Windows Forms e aumentando em muito os requisitos de conhecimento para implementá-lo.

Como primeira opção para corrigirmos o problema tentamos averiguar maneiras de converter os comandos do C++ para o C# e vise versa, criando um wrapper utilizando uma linguagem C++ baseada no .NET denominada C++/CLI, embora a utilização do C++/CLI resolvesse o problema em partes, se tornava um trabalho muito complexo e que fugiria do objetivo do projeto de ser algo fácil de apresentar aos estudantes, como tal buscamos algum wrapper já existente, a qual nos levou ao SharpDX.

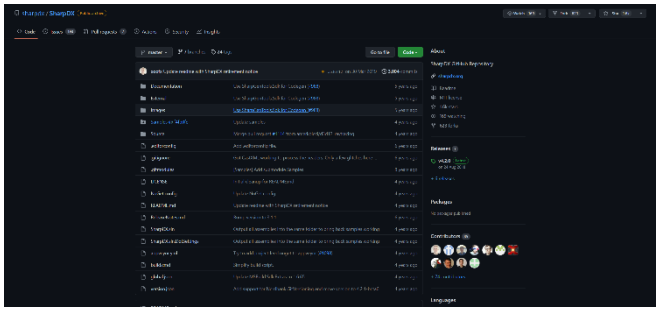

**Figura 1.** Repositório da API SharpDX.

O SharpDX apresentou bons resultados, sendo um wrapper já bem desenvolvido e fácil de implementar convertendo quase que de forma idêntica aos métodos de implementação padrões do DirectX em C++. Mesmo sendo uma solução que aparentava ter grande progresso para implementação possuía o problema de ter sido descontinuado em 29 de março de 2019, dessa forma não recebia mais suporte ou manutenção por parte dos desenvolvedores. Como tal a equipe decidiu por não utilizar pois poderia apresentar problemas ao decorrer de implementação por falta de manutenção no código fonte do SharpDX.

Por fim, devido aos diversos problemas encontrados para a integração da API DirectX em um software com core C#, alteramos nossa proposta passando para uma busca por formas de implementar a API OpenGL a fim de sancionar o problema da falta de compatibilidade. Ao realizarmos a busca encontramos uma API chamada OpenTK que além de possibilitar a integração com o C# também eliminava a necessidade de gerar a janela via Windows Forms, a partir dessa descoberta começamos a basear todo o sistema e recursos necessários utilizando esta ferramenta como base.

**Levantamento dos Requisitos e Análise do Projeto da Ferramenta:**

Após a pesquisa bibliográfica ter nos apresentado uma solução a partir da utilização da API OpenTK, começamos a estruturar o desenvolvimento do software utilizando os recursos da API estabelecida, partindo pela leitura da documentação do OpenTK e da pipeline de implementação do OpenGL, assim como a estruturação da implementação dos recursos a serem utilizados.

Durante as aulas de Computação Gráfica foi identificada a dificuldade dos estudantes entenderem o pipeline do OpenGL e os diversos tipos de shader (Vertex, Fragment, entre outros). Portanto o objetivo traçado para a ferramenta seria da possibilidade de o estudante descrever uma cena por meio de um arquivo do formato JSON informando a posição da câmera, a iluminação da cena, quais modelos de objetos 3D (formato .obj) serão desenhados e como serão desenhados (os shaders). A ferramenta fornecerá um conjunto de shaders padrões que utiliza o modelo de iluminação Phong e um

conjunto de variáveis padrões para facilitar o entendimento por parte do estudante.

Após o objetivo traçado a próxima atividade foi definir a ordem de montagem dos componentes para a etapa de desenvolvimento da ferramenta, sendo definido a seguinte ordem: câmera, Iluminação, objeto 3D e shader padrão para a cena.

**Desenvolvimento da ferramenta:**

Com o planejamento de desenvolvimento do software já concluído, partimos para o desenvolvimento em si da ferramenta, utilizamos a IDE Visual Studio Community 2019 e o .NET Framework 4.8.

Começamos a produção configurando a integração do OpenTK, seguindo a pipeline por eles proposto para a integração e utilização do OpenGL, após isso, partimos para a construção das classes responsáveis pela Viewport (Câmera), nesta etapa definimos o tamanho da área de render, e um controle de movimentação básico para a câmera.

O próximo objetivo a ser construído foi a classe responsável pela iluminação que seguiu os conceitos abordados no manual do OpenTK. A configuração da iluminação seguiu os princípios de luz do tipo Global, onde existe um objeto representando uma luz que erradia por toda a cena, similar ao Sol.

Após o desenvolvimento tanto da Viewport, quanto da luz primária da cena, começamos a montar as classes de importação do modelo 3D, a classe contém um leitor de vértices do objeto, a qual após pegar um modelo 3D no formato .OBJ integra ele dentro da cena.

Por fim tem a construção da classe responsável pela configuração do shader, nela o usuário determina as texturas de entrada, assim como valores relacionados ao especular do material.

**Figura 1.** Visão da Viewport do software.

**Desenvolvimento do material didático de apoio para a ferramenta:**

Devido ao tempo gasto com os problemas encontrados na etapa de Levantamento de Requisitos, essa etapa do processo não pôde ser cumprida. porém, o software final foi desenvolvido de forma que seja facilmente compreendido, assim como usa muitos dos conceitos já discutidos nas disciplinas de Computação Gráfica, dessa forma, pode ser facilmente utilizado por um professor ministrante da disciplina para apresentar de forma prática todo o conteúdo da ementa de aula.

## Resultados e Discussão

A ferramenta IF-RenderEx foi finalizada com algumas alterações do projeto inicial devido as complicações para a integração do C# com o DirectX, por mais que o resultado final tenha implementado uma API diferente daquela proposta inicialmente, sua finalidade de ser um software de fácil entendimento por parte dos estudantes se manteve.

Para a utilização do software o estudante deverá baixar a aplicação através do repositório do Github e abrir a solução utilizando a IDE Visual Studio 2019, em seguida o estudante poderá carregar um modelo 3D através da classe de input de modelo, a qual ele deverá passar o caminho do arquivo, assim como as texturas referentes ao modelo 3D, por fim o aluno poderá executar a aplicação a qual carregara o modelo e o apresentara em uma viewport.

Por fim, o aluno tem a possibilidade de alterar ou implementar seu próprio shader utilizando a linguagem GLSL, escrevendo um arquivo de formato .frag ou .vert e carregando-o através da classe responsável por carregar o material.

## Considerações Finais

O projeto passou por diversas adaptações ao decorrer da produção devido as limitações encontradas para atender a proposta inicial, por fim, a falta de materiais na área de desenvolvimento de aplicações DirectX e OpenGL aparentam ser um dos grandes motivos de falta de mão de obra na indústria de desenvolvimento de game engines.

## Agradecimentos

Venho por meio desta sessão do resumo, agradecer meu orientador o professor Rodrigo Sanches Devigo, pelo auxilio durante o processo de desenvolvimento deste projeto, a qual graças a sua colaboração e auxílio esse projeto pode ser finalizado.

## Referências

SAKUDA, Luiz Ojima; FORTIM, Ivelise (Orgs.). **II Censo da Indústria Brasileira de Jogos Digitais.** Ministério da Cultura: Brasília, 2018.

SCHNEIDER, Andrew; VOS, Nathan. **The Real-time Volumetric Cloudscapes of Horizon: Zero Dawn**, SIGGRAPH, 2015.

KONIARIS, Babis; KOSEK, Maggie; SINCLAIR, David; MITCHELL, Kenny. **Real-time Rendering with Compressed Animated Light Fields**, Graphics Interface, 2017.

HANSON, Chris. DirectXTutorial, 2022. Tutoriais de implementação do DirectX. Disponível em: http://www.directxtutorial.com. Acesso em: 25, setembro de 2022.

DocFX. OpenTK, 2021, Site da API OpenTK. Disponível em: https://opentk.net/. Acesso em: 25, setembro de 2022.

Github. SharpDx-Sample, 2018. Repositório da API SharpDX. Disponível em: https://github.com/sharpdx/SharpDX-Samples. Acesso em: 25, setembro de 2022.

## *IF Render-Ex: A Software for Teaching Computer Graphics*

***Abstract***

*This project aimed to develop software that helps students from the Federal Institute of Mato Grosso do Sul to understand and put into practice in an easier way the concepts covered in Computer Graphics disciplines based on an OpenGL API. To this end, the developed software uses the OpenGl render pipelines, and the OpenTK API, which abstracts some requirements for Windows window creation and render loop, thus facilitating the development and use for the end user. The software allows the end user to load a 3D model and the materials with their respective textures (Diffuse, Normal, Specular) and use GLSL to apply shaders over a 3D model, as well as adding points of light and camera.*

***Keywords:*** *Graphics Processing, Software Engineering, Programming Languages.*