Desenvolvimento de um Motor Gráfico de Apoio a Ambientes Lúdicos Educacionais

João Gabriel Lima Moraes[1];Victor Travassos Sarinho[2]

1. Bolsista PIBIC/CNPq, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: joaofeirense@gmail.com

2. Orientador, Departamento de Ciências Exatas, Universidade Estadual de Feira de Santana, e-mail: vsarinho@gmail.com

PALAVRAS-CHAVE: Jogos Digitais; Motor de Jogo; Motor Gráfico, Jogos Educacionais.

INTRODUÇÃO

A interatividade permitida por meio de jogos digitais, e o ambiente lúdico proporcionado pelos mesmos, colabora para a assimilação de conteúdos didáticos em geral. Jogos eletrônicos de sucesso proporcionam uma experiência imersiva de níveis variados, permitindo ao jogador sentir-se parte do universo fictício proposto pelo mesmo.

Interatividade e imersão são características importantes que podem ser bem exploradas em jogos educativos, tanto à nível de abstração quanto no ensino e na aprendizagem de assuntos diversos. Como resultado, tem-se atualmente no uso de jogos eletrônicos uma ferramenta viável e inovadora para a educação escolar em si (Lewis e Jacobson, 2002).

Considerando a produção de jogos digitais, um motor de jogos é uma ferramenta que permite ao desenvolvedor agregar funções básicas para o funcionamento de um jogo, o que varia desde o gerenciamento de periféricos de entrada e saída até o som e a renderização de modelos gráficos (Ward, 2008). Motores de jogo são amplamente difundidos na indústria de jogos eletrônicos, apesar dos mesmos limitarem as possibilidades de desenvolvimento e portabilidade do jogo em si. Questões de desempenho também devem ser consideradas, uma vez que motores de jogos 3D costumam exigir um bom hardware do usuário para que se possa ter uma jogabilidade “agradável” durante uma partida.

Motores de jogo podem ser divididos em duas partes: motor físico e motor gráfico. O motor gráfico é o colar de diamantes dos jogos eletrônicos. Ele toma como base conceitos matemáticos baseados em vetores e transformações geométricas, no intuito de representar e de permitir a interação de diferentes formas complexas de um jogo em diferentes pontos e dimensões na tela. Em outras palavras, ele incorpora o complexo código necessário para identificar e renderizar a perspectiva do jogador de um modelo de ambiente. O motor gráfico é uma caixa preta, e não pode ser aberta para quaisquer modificações do usuário (Lewis e Jacobson, 2008).

O projeto propôs a criação de um motor gráfico baseado na Interface de Programação de Aplicações(API) OpenGL(Bensted, 2009), tomando como ponto de partida o estudo de métodos gráficos para renderização e rasterização de imagens e modelos 3D, bem como o manuseio dessas estruturas com o intuito de montar um jogo da categoria proposta.

METODOLOGIA

Iniciando os estudos mencionados, foi necessário a obtenção do material relativo a openGL para produção do motor gráfico. Opengl é uma API que é capaz de reproduzir na tela imagens descritas através da abstração de um espaço vetorial. À princípio foi um desafio realizar as configurações necessárias para a manipulação do sistema. Uma série de estudos feitos a partir de tutoriais e materiais licenciados demonstrou a limitação na documentação da API, que dispõe de poucas ferramentas de aprendizado. Uma vez configurado, fez-se necessário o estudo aprofundado dos componentes.

Estudo do pipeline do opengl e recursos da API:

Todos os objetos a serem desenhados em uma cena (*frame*) devem ser descritos na entrada do pipeline através de dados geométricos (os vértices que compõem a imagem), sobre os vértices inseridos são feitas as modificações geométricas necessárias, como a especificação espacial ponto a ponto, as transformações geométricas (rotação, translação e escala), cortes, caso a imagem vá além da área de impressão especificada no *frame*, divisão de perspectiva que define qual o tamanho relativo a visão do espectador para cada objeto impresso em tela e a transformação de câmera, que define o ângulo sob o qual os objetos são observados. Depois que todas essas operações são efetuadas, é a hora de pintar os pixels descritos na imagem, processo chamado de rasterização, além disso, é aplicado um algoritmo que define qual a prioridade de impressão dos pixels, dando a ideia de profundidade da imagem e por fim a aplicação de *alpha* que define a transparência de um pixel. Tudo isso é escrito no espaço de memória da placa gráfica denominado *FrameBuffer* (Ushaw, 2014).

Dando seguimento ao estudo realizado, era necessário dominar como eram feitas as descrições geométricas de maneira automatizada por meio de software, bem como eram feitas as transformações necessárias, com o intuito de utilizá-las livremente, dessa forma, é possível fazer uso dessas funcionalidades para descrever o movimento de um corpo sólido, por exemplo, objeto este que pode ser utilizado em um jogo digital.

OpenGL faz uso de diversas bibliotecas que auxiliam na produção de aplicações que envolvem a transformação de dados geométricos em imagem (renderização), bem como outras funções essenciais para um motor de jogo, como gerenciamento de periféricos, gerar um ambiente de impressão em tela de acordo com as especificações de monitor/sistema operacional, e o próprio interfaceamento com a placa gráfica do computador.

Para tal, foi necessário estudar as bibliotecas supracitadas no intuito de averiguar o seu comportamento quando incorporadas simultaneamente ao projeto.

Uma vez que openGL não possui uma plataforma de aprendizagem que cubra a produção de game engines e afins, foi necessário averiguar diversos materiais de estudo com o objetivo de compilar informações suficientes sobre a API e como desenvolver jogos fazendo uso de suas ferramentas. A princípio descobriu-se que nas versões anteriores de opengl (3.0 e inferior) o gerenciamento de dados geométricos e cores era feito em software, o que tornava o processo lento. Nas novas versões de opengl (3.1 e superior), a programação de

dados geométricos é mais centralizada em shaders e linguagem *glsl* (OpenGL Shading Language). Dessa forma a abordagem adotada foi iterativa, seguindo a seguinte ordem:
1-Criação de Contexto. 2-Manipulação de Triângulos em OpenGL. 3-Manipulação de quadrados. 4-Manipulação de Pontos. 5-Linhas. 6-Composição de Linhas, e geração de Polígonos quaisquer. 7-Composição de polígonos. 8-Desenhando formas circulares usando composição de polígonos.8-Periféricos de Entrada e saída.9-Utilizando Perspectiva.10-Cores.

Após o estudo acerca destas funções, foi possível ir adiante fazendo os estudos acerca das versões mais modernas de openGL, aprofundando-se em glsl e shaders, como listado a seguir:
1-Arquivos shader:*Vertex Shader/Fragment Shader*. 2-Texturas. 3-Transformações geométricas. 4-Projeção.

O uso de shaders transferiu o boa parte do processamento de imagem para a GPU, o que otimizou o processo de criação de frames, permitindo a síntese de jogos mais complexos e uma qualidade de imagem superior pela indústria.

O gerenciamento de periféricos não difere das versões legado para as atuais, o que exige que o desenvolvedor domine os conteúdos de ambas as gerações de openGL no intuito de criar qualquer tipo de aplicação relacionada a jogos. A importância da abordagem incremental consiste no fato que opengl constrói imagens complexas a partir de um algoritmo que abstrai qualquer imagem a ser processada em pontos, esse pontos são computados como pequenos polígonos, que são usados pelos shaders para aplicar a cor. OpenGL possui bibliotecas como a GLM(Raccio,2002), projetadas com o propósito de fazer uma grande quantidade de cálculos por milisegundo, uma imagem 3D pode ser abstraída em milhões de polígonos para apresentar um aspecto realista. Por isso a necessidade de aprender tão profundamente a manipulação de imagens primitivas, como triângulos e polígonos em geral.

RESULTADOS

Cada conteúdo abordado nos materiais de OpenGL exigiu a produção de um trecho de código. Os conteúdos de OpenGL assimilados da versão legado(3.0 e inferior) foram compiladas em um único código fonte, capaz de gerar diferentes desenhos de acordo com os comandos do desenvolvedor. O projeto mais complexo gerado consiste em um cubo colorido, cuja rotação pode ser manipulada através do joystick, permitindo assim a manipulação do usuário diretamente.

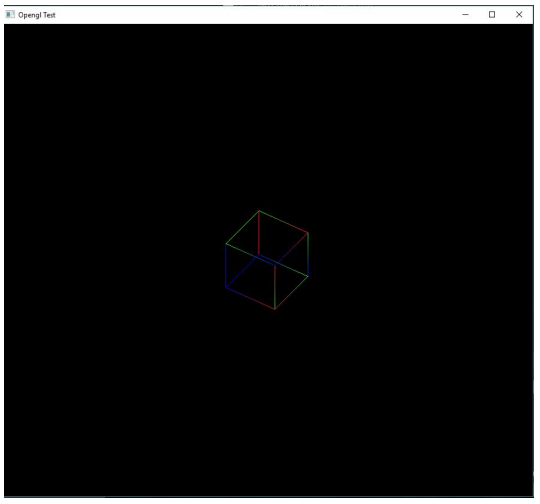

Figura 3: Cubo colorido, que pode ser movimentando usando o joystick.

Os conteúdos referentes à versão moderna também tiveram trechos de código compilados numa aplicação única, capaz de gerar diversos desenhos diferentes, é importante salientar que cada desenho que a versão moderna gera, necessita de um shader específico para tal. O estado atual da aplicação se encontra na fase de importação de texturas e transformações geométricas com os objetos gerados, podendo estas ser feitas através da interação com o usuário.

O jogo demonstrativo depende da aprendizagem de manipulação de luzes e câmera em diversos níveis e também na importação de modelos 3D, que podem ser gerados utilizando softwares de modelagem 3D, tais como o Blender, 3D Studio Max e o Autodesk Maya. A geração do jogo também depende da anexação do motor físico ao projeto, o que deve acrescentar uma movimentação fluida e realista aos objetos a serem adicionados no projeto.

Os códigos previstos estão em processo de conclusão.

CONSIDERAÇÕES FINAIS

O trabalho de obtenção dos conceitos necessários para a produção de um motor de jogo utilizando os conceitos de opengl foi um sucesso, considerando a falta de conteúdo acessível acerca da temática. Levando em conta o atraso, decorrente da chamada e do bolsista ter ocorrido somente cinco meses após o tempo previsto pelo cronograma previamente estipulado, os códigos obtidos apresentam grande avanço nos estudos da API e, com a adesão do motor físico, o jogo demonstrativo estará totalmente operacional.

O trabalho obtido já serve de ferramenta de aprendizagem de OpenGL para a produção de jogos, bem como a documentação sintetizada pode ser ativamente utilizada para a produção de jogos utilizando a API.

REFERÊNCIAS

LEWIS, M.; JACOBSON, J. 2002.Game Engines in Scientific Research http://publicvr.info/publications/Lewis2002.pdf -acessado em 29/03/2016

WARD, J. (2008). What is a Game Engine? - http://www.gamecareerguide.com/features/529/what_is_a_game_.php - acessado em 29/03/2016

BENSTED, L.(2009) Beginning OpenGL Game Programming. Second Edition. United States, Páginas 7-14.

USHAW D. G. (2014). Notas da disciplina graphics for games, csc3223.Newcastle University

RICCIO, C. Opengl mathematics - a c++ mathematics library for graphics programing. https://glm.g-truc.net/0.9.8/index.html, October 2002-2016. acessado em 01/07/2017.