UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

Disciplina: [**1107190] Introdução à Computação Gráfica - Turma 01**.
Professor: **Christian Azambuja Pagot (email: christian@ci.ufpb.br)**.
Data: **28/09/2016**.

Trabalho Individual II

Objetivo

O objetivo deste trabalho é familiarizar os alunos com a estrutura e o funcionamento do *pipeline* gráfico através da implementação de um *pipeline* completo, capaz de transformar vértices descritos no espaço do objeto em primitivas rasterizadas no espaço de tela.

Atividade

O sistema de desenvolvido no trabalho individual I (TI1) implementa um dos últimos estágios do pipeline gráfico, responsável pela rasterização das primitivas no espaço de tela. Neste sistema, a descrição dos modelos é feita através de coordenadas inteiras definidas sobre o espaço bidimensional discretizado da tela.

Esta forma de descrever primitivas, entretanto, não é prática pois, em muitos casos, os modelos geométricos são tridimensionais e se extendem para além dos limites definidos pela tela. De forma a facilitar o processo de descrição de modelos, opta-se por fazê-lo em um espaço mais adequado, tanto em termos de extensão quanto de dimensões, chamado de *espaço do objeto*. Após a descrição no espaço do objeto, o modelo é submetido a uma série de transformações que resultam em seu mapeamento final para o espaço da tela. A imagem final então é obtida através da rasterização das primitivas projetadas na tela.

Esta segunda atividade individual consiste na implementação das transformações que levarão os vértices descritos originalmente no espaço do objeto para o espaço de tela. A rasterização das primitivas será feita pelo código previamente desenvolvido no trabalho TI1.

Desenvolvimento

Os alunos deverão implementar todas as transformações do *pipeline* em forma de matriz e utilizando coordenadas homogêneas. Abaixo são enumeradas as etapas normalmente encontradas ao longo de um *pipeline* gráfico. Cada etapa é acompanhada de uma breve descrição de suas características, dados de entrada e de saída. Ao lado de cada etapa consta a indicação se deve ser implementada.

1. **Transformação: Espaço do Objeto → Espaço do Universo (Implementar)**
   Esta etapa do *pipeline* é responsável por transformar vértices, originalmente descritos no espaço do objeto, para o espaço do universo. Isto é feito através da multiplicação destes vértices por uma matriz denominada *matriz de modelagem (*ou *model matrix)*. A matriz de modelagem é composta por uma sequência de transformações geométricas que posicionam o modelo no universo. As transformações que podem compor a matriz de modelagem são a rotação, translação, escala e o cisalhamento (*shear*).

2. **Transformação: Espaço do Universo → Espaço da Câmera (Implementar)**
   Esta etapa do *pipeline* é responsável por transformar vértices do espaço do universo para o espaço da câmera. Isto é feito através da multiplicação dos vértices por uma matriz denominada *matriz de visualização (*ou *view matrix)*. A matriz de visualização é composta por uma translação e uma rotação. Esta translação e esta rotação são definidas a partir das informações da câmera (posição, direção e "up").

3. **Transformação: Espaço da Câmera → Espaço Projetivo ou de Recorte (Implementar)**
   Esta etapa do *pipeline* é responsável por transformar vértices do espaço da câmera para o espaço de recorte (ou projetivo). Isto é feito através da multiplicação dos vértices por uma matriz denominada de matriz de projeção (ou *projection matrix*). Após esta transformação, a coordenada homogênea dos vértices pode (e provavelmnente irá) apresentar valores diferentes de 1.

4. **Transformação: Espaço de Recorte → Espaço "Canônico" (Implementar)**
   Esta etapa do *pipeline* é responsável por transformar pontos do espaço de recorte para o espaço canônico (ou NDC – *Normalized Device Coordinates*). Isto é feito em duas etapas. Primeiro, dividem-se as coordenadas dos vértices no espaço de recorte pela sua coordenada homogênea. Esta transformação gera uma alteração da geometria da cena, tornando menores os objetos que estiverem mais distantes da câmera. Adicionalmente, esta transformação mapeia o volume de visualização (*view frustum*), com formato piramidal, em um hexahedro. A seguir, os vértices são multiplicados por uma matriz adicional que, ao aplicar escalas e translações, transforma estes vértices para o espaço canônico, transformando o hexahedro anterior em um cubo de coordenadas unitárias e centrado na origem.

5. **Transformação: Espaço "Canônico" → Espaço de Tela (Implementar)**
   Esta etapa do *pipeline* é responsável por transformar pontos do espaço canônico para o espaço de tela. Isto é feito através da multiplicação dos vértices por uma matriz específica que envolve escalas e translações.

6. **Rasterização (Implementada no TI1!)**
   Esta etapa gera a rasterização dos modelos no espaço de tela.

Após o término da implementação, os alunos deverão comparar os resultados obtidos com os resultados gerados por uma aplicação equivalente escrita em OpenGL. Os dois sistemas devem gerar os mesmos resultados.

OBS 1: A câmera perspectiva do OpenGL apresenta alguns parâmetros extras que a câmera sintética a ser implementada não possui. Estes parâmetros são o *aspecto* e o ângulo de visão (FOV – *Field of View*). Abaixo seguem dicas de como lidar com estas limitações da câmera a ser implementada:

1. **Aspecto:** Aspecto é a razão entre a largura e a altura da imagem final (ou da *viewport*). Como estamos utilizando sempre *viewports* quadradas (mesma largura e altura), o aspecto é sempre 1. Assim, quando for criada a câmera em OpenGL, deve-se setar o aspecto para 1.

2. **FOV:** A câmera sintética a ser implementada não possui o parâmetro FOV. Entretanto, é possível que se aproxime o FOV da câmera ajustando-se as posições do *near* e do *far* planes. O aluno deve olhar os slides da aula, as referências bibliográficas, e determinar esta aproximação.

Este trabalho vem acompanhado de uma biblioteca para carga de modelos simples descritos em formato OBJ (*Wavefront .obj*). O objetivo é que os alunos façam as comparações entre o seu sistema e o OpenGL através do rendering (*wireframe*) de modelos mais complexos. O próprio arquivo da biblioteca já vem acompanhado de um modelo (*monkey_head2.obj*) que pode ser utilizado nos testes de comparação. Incentiva-se que o aluno também faça testes com outros modelos.

O Apêndice I deste trabalho contém uma breve descrição de como exportar arquivos OBJ a partir do Blender.

OBS 1: Todos os sistemas de coordenadas, com exceção do espaço de tela, seguem a convenção RHS.

OBS 2: Todo o código necessário à realização desta atividade deve ser escrito em C/C++. Não é permitido o uso de nenhuma biblioteca externa, com a exceção de:

- glut ou SDL
- OpenGL
- *loader* de arquivos OBJ fornecido junto com o trabalho, ou algum outro loader (como o Assimp - http://assimp.sourceforge.net)
- uma biblioteca para operação de matrizes e vetores (como a Glm - http://glm.g-truc.net/0.9.7/index.html).

Dica

A execução do trabalho pode ser facilitado se algumas medidas forem adotadas:

- Para a carga do modelo OBJ, pode-se criar uma estrututura (como um *array ou o std::vector<T>* da STL) que armazene listas de linhas ou triângulos. Isso também facilita a aplicação de transformações sobre um conjunto maior de primitivas (através do uso de *loops*).
- Melhor comodidade pode ser obtida com o uso de programação orientada a objetos, onde cada estrutura pode ser representada por uma classe e as operações entre elas pode ser feita através da sobrecarga de operadores.

Entrega

Prazo: Os trabalhos devem ser entregues até o dia **19/10/2015**, impreterivelmente até as **23 horas e 55 minutos**.O trabalho será considerado entregue assim que estiver acessível no *blog* criado pelo aluno para a disciplina.

No *blog* deverão constar, pelo menos, *printscreens* (acompanhados de todos os parâmetros utilizados na sua geração) e textos explicativos. O aluno deve incluir também pelo menos um vídeo que ilustre o modelo tridimensional sendo rotacionado. Abaixo segue um detalhamento de cada um dos componentes que devem aparecer no *post*.

- ***Printscreens***: Devem demonstrar a evolução do processo de implementação, eventuais problemas encontrados, e resultados gerados pela versão final da implementação.

- **Texto**:
  - Descrição da implementação, onde fique claro como o aluno resolveu os problemas que se apresentaram.
  - Discussão sobre os resultados gerados.
  - Possíveis melhoras.
  - Dificuldades encontradas.
  - Referência bibliográficas.
  - A inclusão de trechos de código no corpo do blog deve se limitar ao mínimo indispensável para o correto entendimento do texto.

Avaliação

A avaliação dos trabalhos levará em consideração os seguintes critérios:

- Aderência ao tema proposto.
- Clareza, organização, corretude e legibilidade do texto.
- Capacidade de síntese.
- Relevância dos elementos apresentados (imagens, trechos de código, etc.) no contexto das discussões.
- Embasamento técnico dos argumentos.
- Eficiência e eficácia das soluções apresentadas.
- Capacidade de análise dos resultados e autocrítica.
- Completude do trabalho.

Importante:

- Não serão aceitos trabalhos atrasados ou que apresentem indícios de plágio.
- Não serão aceitos trabalhos enviados por qualquer outro meio que não o *blog* informado previamente pelo aluno.

Apêndice I

O software de modelagem 3D *Blender* é capaz de salvar seus modelos em diversos formatos, incluindo o *Wavefront .obj*. Uma vez que o modelo esteja finalizado, para que se possa exportá-lo, deve- se seguir os seguintes passos:

1. Selecionar somente o objeto que se deseja exportar.
2. Selecionar no menu principal *File/Export/Wavefront (.obj)* (→ **Figura 1**).
3. No painel de opções que surgirá à esquerda (→ **Figura 2**) **marcar apenas** os ítens: *Selection Only*, *Triangulate Faces*, *Objects as OBJ Objects*.
4. Para que o objeto não seja exportado em um sistema de coordenadas diferente do utilizado na aplicação (RHS), certifique-se de que abaixo deste painel a direção seja *-Z Forward*, e que o Up seja *Y Up*.

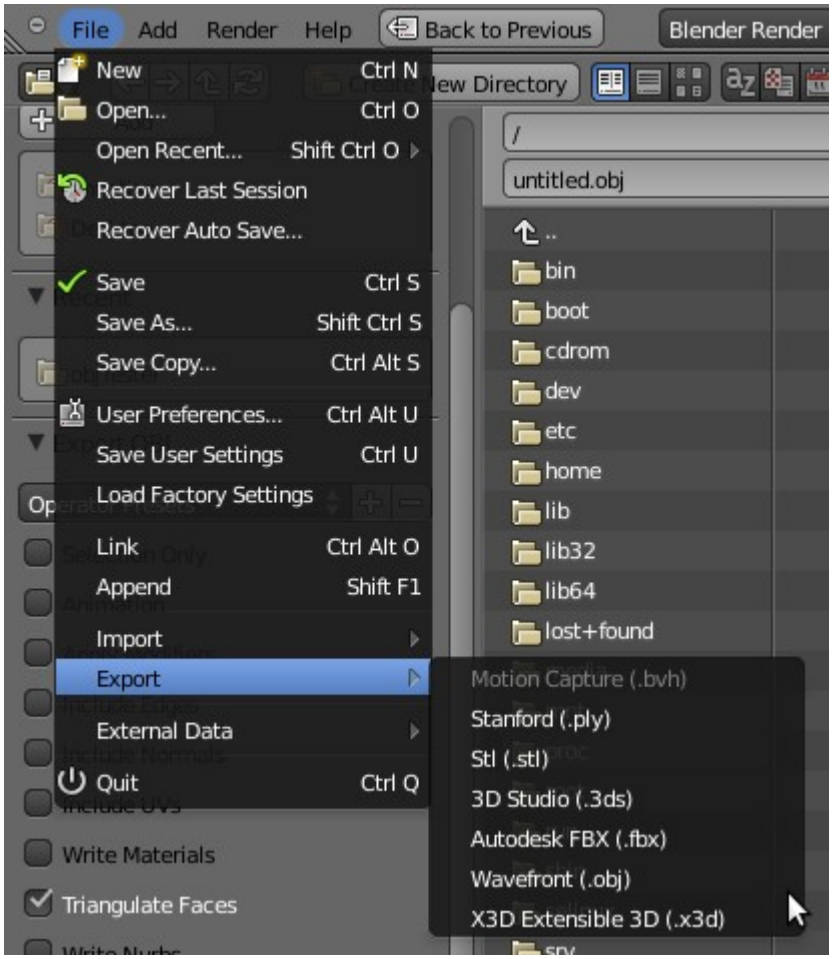

Figura 1 - Menu principal.

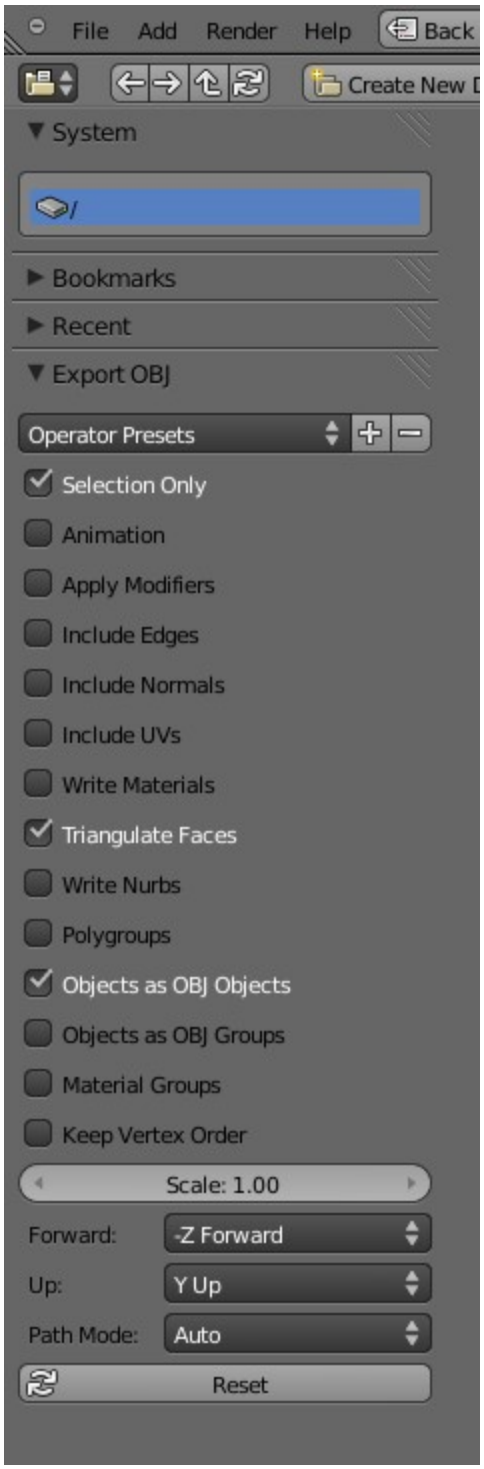

Figura 2 - Painel de opções para exportação de arquivo OBJ.