Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Ciência da Computação

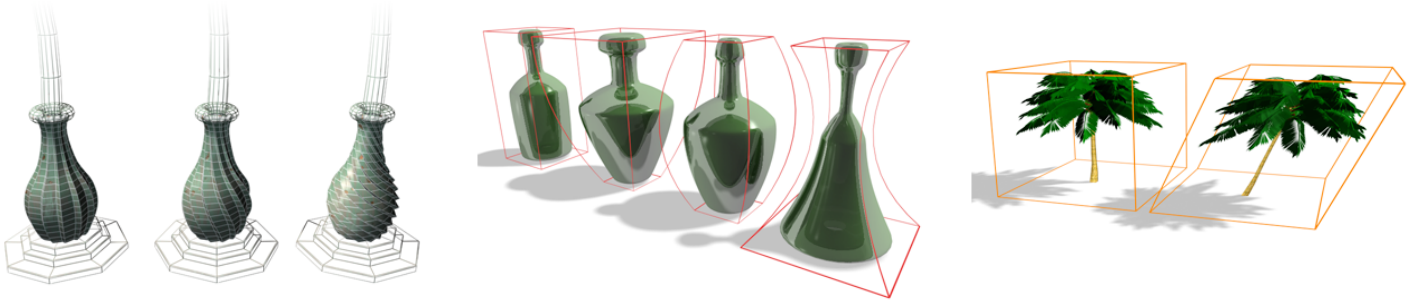
MATA65 - Computação Gráfica Período: 2018.1 Data: 26/04/2018.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Rafaela Alcantara

Atividade 2 - Transformações Geométricas

Motivação:

Transformações geométricas são a base matemática fundamental no processo de síntese de imagens em Computação Gráfica [1]. No primeiro estágios do *pipeline* gráfico seu papel é mudar as coordenadas dos vértices do objeto do seu sistema de coordenadas para o sistema global, a partir de mudanças em sua posição (translação), orientação (rotação) e dimensão (escala). Como podemos notar, as duas primeiras não modificam as posições relativas entre os vértices, portanto não são capazes de mudar sua forma original, são o que chamamos de **movimento rígido**. Já a escala é capaz de mudar a forma do objeto, ainda que de maneira uniforme em uma determinada direção.

As transformações geométrica, no entanto, não se limitam a movimentos rígidos e escalas, é possível construir qualquer transformação a partir de matrizes. Um exemplo são as **transformações de deformação** [2]. Três exemplos de deformações bastante utilizadas no contexto de modelagem são apresentadas na Figura 1.



(a) *(b)* *(c)*

Figura 1 - Operações de deformação em um objeto geométrico 3D. (a) Twist, (b) Tapering e (c) Shear.

Essas 3 transformações de deformação podem ser geradas a partir das transformações geométricas básicas de rotação (*twist*), escala (*tapering*) e translação (*Shear*), parametrizadas corretamente. Por exemplo, podemos perceber que na figura 1(a) a torção aplicada no objeto ao longo do seu eixo vertical aumenta conforme aumenta a sua altura.

Formalizando essa ideia, as 3 operações de deformação podem ser definidas como [3]:

$$T_0 = \begin{bmatrix} f(z) & 0 & 0 & 0 \\ 0 & f(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_1 = \begin{bmatrix} cos(f(z)) & -sin(f(z)) & 0 & 0 \\ sin(f(z)) & cos(f(z)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T_2 = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

onde as matrizes **T_0, T_1** e **T_2** correspondem, respectivamente, as operações de *taper*, *twist* e *shear*, na direção do eixo Z.

Especificações:

Nessa atividade sua tarefa é gerar uma aplicação em *Javascript/Three.js* capaz de carregar um modelo geométrico 3D qualquer e aplicar as deformações de *Taper*, *Twist* e *Shear*. Os valores e a direção da operação devem ser especificados pelo usuário.

Sugestão: comece por um modelo geométrico que voce tem o controle, como a forma da gota criada por voce na ultima atividade, ou uma das formas pedidas na atividade 0. Uma vez que voce domine o processo das deformações passe para modelos mais complexos e gerais, como os fornecidos no diretório de Assets da disciplina[1].

Desafio[2]: A forma mais direta de aplicar as transformações de deformação no modelo é fazer isso na aplicação *Javascript*. No entanto, a primeira etapa do *pipeline* gráfico, responsável pela aplicação das transformações nos vértices e executada pela GPU, pode ser programada através de *shaders* [1][4]. Pesquise sobre o que são *shaders* e como eles podem ser utilizados para modificar a geometria de um objeto diretamente na GPU/*pipeline* e tente aplicar as transformações de deformação diretamente nos vértices no *vertex shader*[3].

Entrega da atividade:

- O trabalho deverá ser submetido **somente** via **Moodle**, respeitando a data e hora limite para entrega. Em caso de qualquer problema de arquivos corrompidos ou similar o trabalho será considerado não entregue. Portanto, verifique bem o que for entregar!!

- A entrega no **Moodle** deve ser feita em **um único arquivo compactado (.tgz, .zip ou .rar) contendo um subdiretório com seu nome e dentro deste todos os arquivos necessários para a execução de seu código.**

  Na falta de algum arquivo (libs, scripts, modelos, texturas, etc), uso de caminhos absolutos, ou qualquer outra "falha" que necessite da edição de seu código fonte a atividade será desconsiderada!!

---

[1] Esse diretório encontra-se no Moodle, em formato compactado, na primeira aula de laboratório.

[2] Não faz parte da atividade.

[3] Na realidade é exatamente isso que o Three.JS faz, via WebGL, quando renderiza suas formas geométricas.

- A cooperação entre alunos é considerada salutar. No entanto, atividades com alto grau de similaridade serão tratadas como plágio, o que resultará em avaliação **zero** para **todos** os envolvidos.

- Da mesma forma, a internet está cheia de respostas. Use mas não abuse da consulta a códigos prontos. Ctl-C Ctl-V também é plágio!

- Qualquer dúvida, **não suponha** procure o professor[4] ou o estágiário[5] para esclarecimentos.

Referencias:

[1] Angel, Edward, and Dave Shreiner. **Interactive Computer Graphics with WebGL**. Addison-Wesley Professional, 2014.

[2] Barr, Alan H. **Global and local deformations of solid primitives**. In Readings in Computer Vision, pp. 661-670. 1987.

[3] Watt, Alan, Fabio Policarpo. **The computer image**. ACM SIGGRAPH Series, Addison-Wesley. 1998.

[4] Dirksen, Jos. **Learning Three.JS – the JavaScript 3D Library for WebGL**. Packt Publishing Ltd, 2015.

[4] antonio.apolinario@ufba.br

[5] rafa.alcantara23@gmail.com