MAC420/5744 - Introdução à Computação Gráfica
Prof. Marcel Parolin Jackowski
Departamento de Ciência da Computação
IME/USP - Segundo Semestre de 2019
Primeiro Exercício-Programa
Data de entrega: 30/09/2019

Mundo Esférico

Em computação gráfica, objetos tridimensionais são criados através de modelos geométricos. Um modelo geométrico é definido por um conjunto de vértices e a conectividade entre esses vértices. Estas informações devem criar uma malha triangular que então é processada pela GPU por um conjunto de programas chamados de *shaders*. Ao final, a unidade gráfica produz uma imagem *renderizada* dos modelos presentes em uma cena.

Neste EP (Exercício Programa), vamos utilizar o *pipeline* moderno do OpenGL e a a biblioteca **mac420** para gerar malhas triangulares para renderizar esferas. Exercitaremos duas abordagens diferentes. Na primeira abordagem, iremos amostrar vértices utilizando coordenadas (polares) esféricas e na segunda abordagem, utilizaremos *Tessellation Shaders* para aproximar a geometria de uma esfera através da subdivisão sucessiva de um icosaedro. Em ambas abordagens, novos atores (subclasses de *Actor*) deverão ser criados.

1. Geração de Esfera utilizando Coordenadas Polares

As *coordenadas polares* representam um sistema de coordenadas bidimensionais que representam um ponto (x, y) no espaço bi-dimensional por valores (r, ϕ), tal que r é a distância do ponto em relação à origem e ϕ é o ângulo relativo ao eixo x. É possível utilizar coordenadas polares para gerar esferas utilizando valores (r, ϕ, θ), onde r é a distância até a origem, e os ângulos ϕ e θ representam rotações nos eixos XY e XZ, respectivamente (Fig. 1). Essas coordenadas também são chamadas de coordenadas polares esféricas porque descrevem pontos na superfície de uma esfera de raio r.

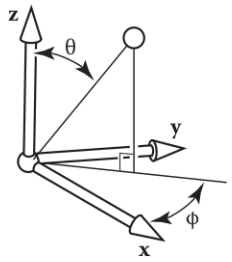

Figura 1: Exemplo de coordenadas esféricas (Figura retirada de [1], capítulo 2, página 42).

Neste primeiro exercício, você deve implementar um novo ator na biblioteca **mac420** que modele uma esfera de raio r (parâmetro). Essa esfera deve ser gerada com v amostras igualmente espaçadas do ângulo θ e h amostras igualmente espaçadas do ângulo ϕ. Seu modelo deve aceitar no construtor os parâmetros r, v e h, o modelo também deve conter as normais e as coordenadas (u, v) de textura para cada vértice. A Fig. 2 mostra um exemplo de malha utilizando amostras das coordenadas esféricas para a geração de seus vértices.

2. Geração de Esfera por Subdivisão de Icosaedro no Tessellation Shader

O *Tessellation Shader* é um estágio do *pipeline* gráfico do OpenGL que tem como objetivo a geração de novos vértices a partir de uma malha poligonal base. Assim, ele permite que a aplicação controle o nível de detalhes do modelo de forma dinâmica, através da geração de novos vértices em tempo real. A fase de tesselação ocorre após a execução do *Vertex Shader* e antes do *Geometry* e *Fragment shaders*. O *shader* de tesselação possui três etapas. Na primeira, chamada de *Tessellation Control Shader* (TCS), o *shader* recebe um *patch* e define fatores de tesselação para o mesmo. Os fatores de tesselação são utilizados pelo *Tesselation Engine*

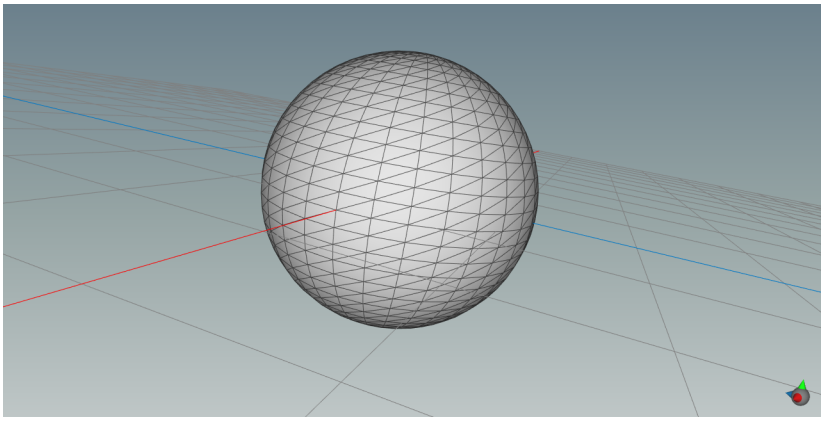

Figura 2: Exemplo de malha de uma esfera gerada utilizando coordenadas polares. Essa esfera foi gerada com $r = 0.5$, $v = 30$ e $h = 30$.

(estágio fixo do *pipeline*) para quebrar cada *patch* em pequenas primitivas como triângulos, pontos e linhas. As primitivas geradas pelo *Tessellation Engine* são enviadas para o próximo *shader*, chamado de *Tessellation Evaluation Shader* (TES), nessa etapa, o *shader* deve especificar a posição final de cada vértice na malha. No TES, é possível especificar se os pontos gerados são para primitivas do tipo *quads* ou *triangles*. Quando *triangles*, a variável *gl_TessCoord* corresponde a coordenadas baricêntricas do ponto. Quando *quads*, a variável *gl_TessCoord* corresponde a coordenadas *uv* de retângulo. A variável *gl_TessCoord* é global e preenchida para o TES (Fig. 3). Geralmente, o TES também é responsavel por calcular os vetores normais e as coordenadas de textura dos novos vértices gerados (embora seja possível calcular esses valores no *Geometry Shader*).

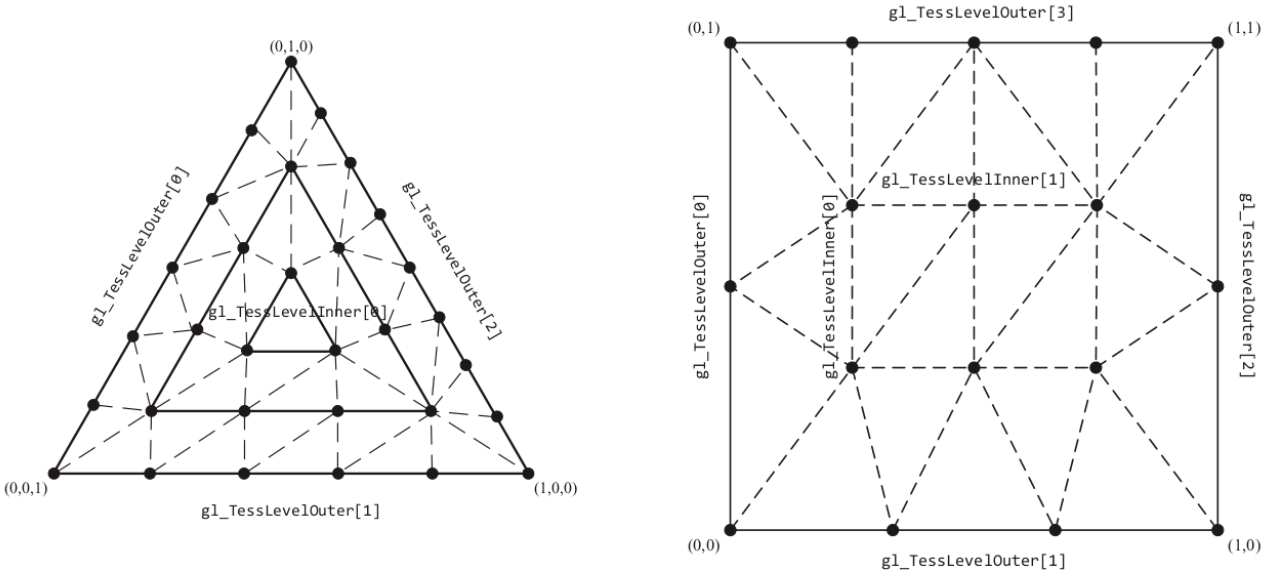

Figura 3: Tesselação da malha feita por TCS e TES. No TCS as variáveis `gl_TessLevelOuter` e `gl_TessLevelInner` devem ser preenchidas para indicar o número de divisões do *patch*. O TES pode calcular a nova malha utilizando os vértices originais e a variável `gl_TessCoord`. Os valores das variáveis `gl_TessLevelInner`, `gl_TessLevelOuter` e `gl_TessCoord` dependem da primitiva a ser gerada *triangles* ou *quads* (Figuras retiradas de [2], capitulo 9, páginas 492 e 495).

Um icosaedro regular é constituido por 20 faces triangulares, 30 arestas e 12 vértices. Nós podemos utilizar a subdivisão sucessiva de um icosaedro, utilizando os pontos médios de suas arestas, para aproximar a geometria de uma esfera. Nesse exercício, vamos aproveitar o refinamento de um icosaedro gerado pelo *shader* de tesselação para obtermos uma malha que representa uma esfera. O exercício consiste em desenvolver um novo ator na biblioteca **mac420** que receba como parâmetros um raio r e uma quantidade de subdivisões s para gerar uma

malha esférica de raio r calculada a partir da tesselação de um icosaedro em s níveis (interno e externo). Note que diferentemente do ator *Icosahedron* disponibilizado na biblioteca, neste exercício, a malha inicial do icosaedro deve ser subdividida nos *shaders* de tesselação (TCS e TES). A Fig.4 exibe um exemplo de esfera utilizando essa técnica.

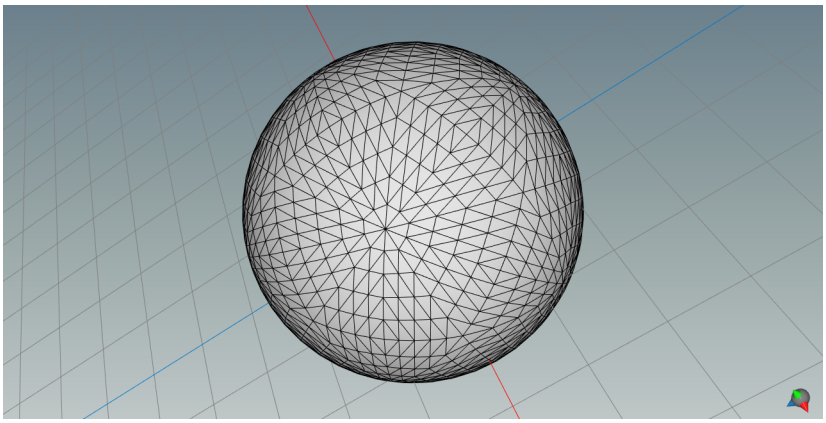

Figura 4: Exemplo de uma esfera gerada pela tesselação da malha de um icosaedro.

Lembre-se que além da malha, você também precisará definir os vetores normais e as coordenadas (u, v) de textura de seu modelo (embora não utilizadas neste EP).

Entrega

Este é um EP para ser feito em equipe de até 2 pessoas. Você deverá submeter um único arquivo no formato `.zip` contendo toda a sua solução. Não esqueça de providenciar um arquivo de `README` contendo os nomes dos participantes da equipe, e quaisquer outras informações que você considere importantes (e.g. o que foi feito e o que não foi, principais alterações no código, como funciona a interface da solução final).

Avaliação

Para cada abordagem você deverá gerar pelo menos 3 esferas com diferentes parâmetros (raio e número de amostras). A avaliação será modular (pontuação ganha por cada item). Cada item pode contribuir com zero (não atingiu satisfatoriamente), 50% (atingiu parcialmente) ou 100% (atingiu satisfatoriamente) de sua pontuação para o total. A tabela 1 contém os itens a serem analisados no processo de avaliação.

Tabela 1: Ficha de Avaliação

Item	**Descrição**	**Pontuação**
1	Modelo 1 corretamente implementado	2.5
2	Normais e coordenadas (u, v) de textura do Modelo 1 corretamente calculadas	1.5
3	Modelo 2 corretamente implementado	4.0
4	Normais e coordenadas (u, v) de textura do Modelo 2 corretamente calculadas	1.5
5	Arquivo de README com todas as informações relevantes da solução	0.5

Referências

[1] Steve Marschner and Peter Shirley. *Fundamentals of Computer Graphics, Fourth Edition*. A. K. Peters, Ltd., Natick, MA, USA, 4th edition, 2016.

[2] John Kessenich, Graham Sellers, and Dave Shreiner. *OpenGL®Programming Guide: The Official Guide to Learning OpenGL®, Version 4.5 with SPIR-V*. Addison-Wesley Professional, 9 edition, 2016.