EC212 - Computação Gráfica

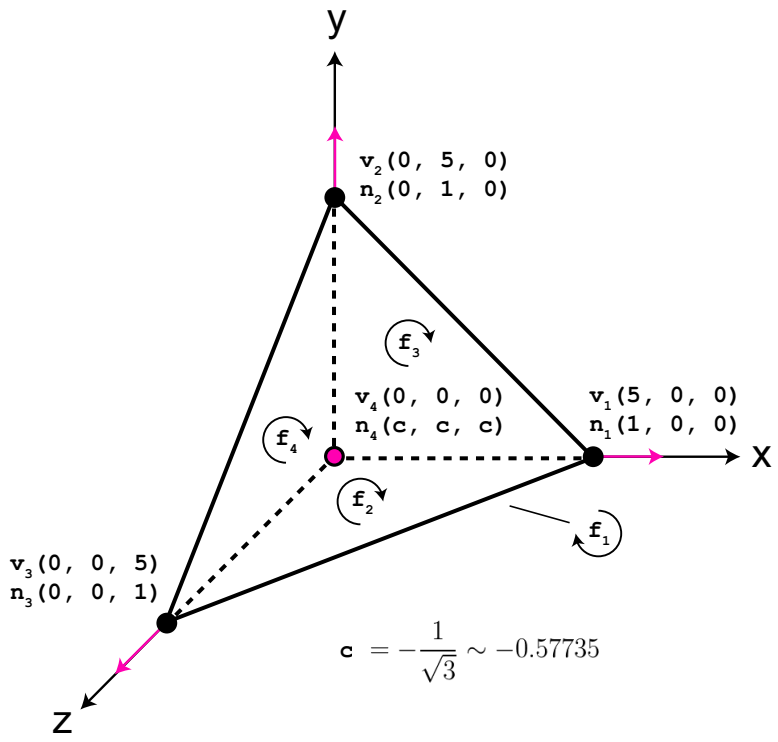
Danilo Peixoto Ferreira
(danilopeixoto@gec.inatel.br)
Instituto Nacional de Telecomunicações
22 de agosto de 2019

Orientações

Envie o projeto para o endereço de e-mail danilopeixoto@gec.inatel.br com o assunto `[EC212] Teste 1`. O projeto deve estar compactado em uma pasta ZIP nomeada de acordo com o padrão: `NOME_MATRICULA.zip`. Para reduzir o tamanho do arquivo, remova a pasta `external` da estrutura do projeto.

Questões

1. (15 pontos) Crie um arquivo Wavefront OBJ que represente o tetraedro abaixo:



Observações:

a) Utilize um editor de texto ASCII.

b) Represente as facetas no sentido anti-horário.

c) Adicione o arquivo na estrutura do projeto (ex: `res/meshes/tetrahedron.obj`).

2. (5 pontos) Configure a cor do plano de fundo da janela como branco por padrão.

3. (10 pontos) Carregue a geometria do arquivo Wavefront OBJ (Questão 1) para o OpenGL. As duas abordagens abaixo são válidas:

   - utilizando as funções `readTriangleMesh` e `loadTriangleMesh`, ou
   - descrevendo o tetraedro diretamente por código fonte C++.

4. (5 pontos) Modifique o programa de *shader* de vértices:

   a) Utilize um editor de texto ASCII.

   b) Aplique a expressão abaixo ao atributo normal a nível de vértice e exporte para o *shader* de fragmentos:

   $N_{out} = normalize(transpose(inverse(M)) * N)$

   Onde:

   ```
   M - matriz de transformação de modelo
   N - atributo normal do vértice
   N_out - atributo normal de saída
   ```

   c) Atente-se a dimensão dos vetores e matrizes.

5. (5 pontos) Modifique o programa de *shader* de fragmentos:

   a) Utilize um editor de texto ASCII.

   b) Aplique a expressão abaixo ao atributo normal exportado pelo *shader* de vértices e configure para ser exibido como cor do fragmento:

   $C = \frac{N_{in}+1}{2}$

   Onde:

   ```
   N_in - atributo normal de entrada
   C - cor do fragmento
   ```

   c) Atente-se a dimensão dos vetores.

6. (10 pontos) Crie uma função para o evento da posição do cursor do *mouse* e aplique uma transformação de rotação em torno do eixo y a matriz de modelo.

   Como a função do evento retorna a posição atual na tela, crie uma variável global para armazenar a última posição do *mouse* em x. O ângulo de rotação será:

   $a = \frac{(x_i - x_{i-1})}{1000}$

   Onde:

   ```
   a = ângulo de rotação
   x_i - posição atual
   x_(i-1) - posição anterior
   ```

Anexo

Projeto:

Execute o arquivo `post_build.bat` na estrutura do projeto para configurar as dependências.

Casting de vetores no GLSL:

```
// vetor 3D para 4D
vec4 y(vec3(0.0f), 1.0f); // pontos
vec4 y(vec3(0.0f), 0.0f); // vetores

// vetor 4D para 3D
vec3 y(vec4(0.0f).xyz);
```

Imagem final: