Relatório - Logo EComp

Everson Toledo - 76884

Centro de Ciências Computacionais – C3 – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.201-900 – Rio Grande – RS – Brasil

everson.toledo@hotmail.com

1. Introdução

Este relatório tem por objetivo explicar técnica e detalhadamente o desenvolvimento de um logo da Engenharia de Computação da FURG que conta com uma engrenagem externa e no centro traz um símbolo de somatório sobreposto por um cabo USB, com os dizeres "ECOMP" acima do somatório e "FURG" abaixo (utilizado na confecção de um moletom do curso no ano de 2014), com alguns ajustes de posicionamento, feito para a disciplina de Sistemas Gráficos com a utilização da linguagem C++ e da ferramenta OpenGL.

O relatório será dividido de acordo com os critérios a serem avaliados no trabalho, geometria, transformações, visibilidade, rasterização e iluminação. Posteriormente se encontrará a conclusão.

Imagem 1
Logo original

2. Geometria

Foram utilizadas funções desenvolvidas juntamente com funcionalidades da própria biblioteca OpenGL para montar a geometria do projeto. A mesma conta com uma parte em 3D feita com sólidos e o desenvolvimento de desenhos em uma das faces.

2.1. Engrenagem

A engrenagem base está em 3D, sendo uma sobreposição de quatro cilindros em sua parte central, o externo de coloração branca, o mais interno de coloração azul marinho, rgb (0,0,0.15) na própria ferramenta, o próximo branco e por fim o da face, na mesma coloração azul do anterior. Uma sobreposição de dois quadrados forma cada dente da engrenagem, sendo o externo branco e o interno no já referido azul marinho.

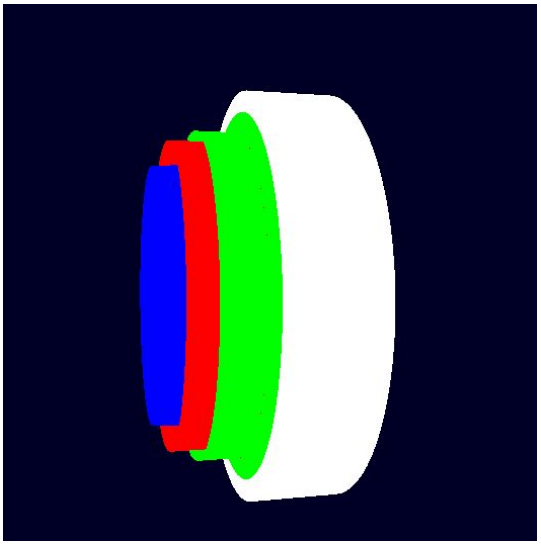

Imagem 2
Cilindros que formam o centro da imagem com profundidade aumentada e cores diferentes para que fiquem claros.

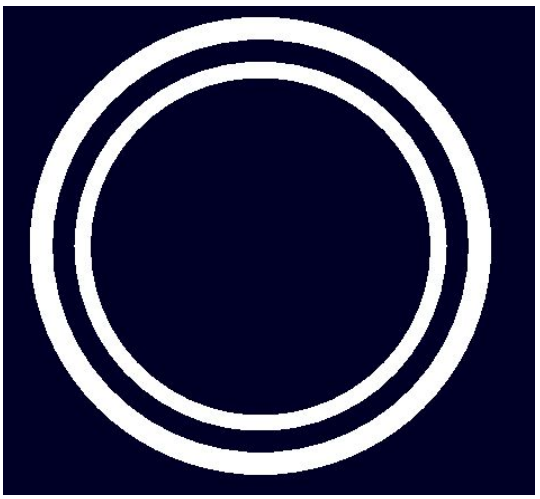

Imagem 3
Vista frontal dos cilindros já com suas cores e posições finais.

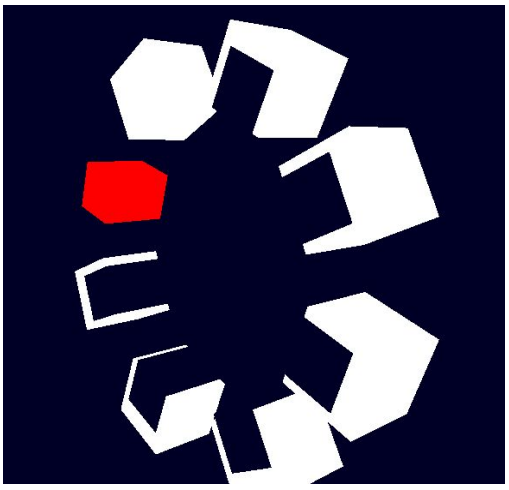

Imagem 4

Quadrados que formam os dentes da engrenagem sendo um apenas com o externo (branco), outro apenas com o interno (vermelho) e os demais já sobrepostos e com suas cores finais.

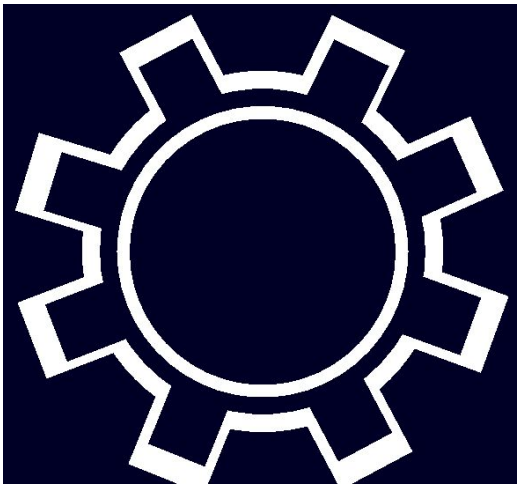

Imagem 5

Vista frontal final da engrenagem.

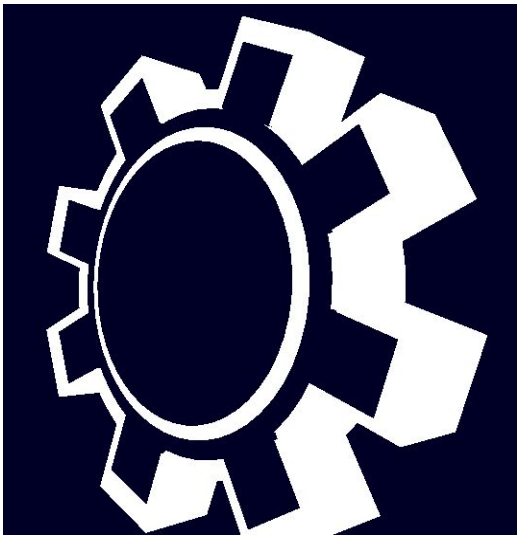

Imagem 6

Vista lateral final da engrenagem.

2.1.2 Face

A face do logo está sobre o cilindro central da engrenagem, conta com o símbolo de somatório e os escritos “ECOMP” na parte superior e “FURG” na parte inferior em branco e o cabo USB em preto, tudo feito manualmente com a função **glBegin()** utilizando como parâmetros as seguintes opções já oferecidas pela OpenGL: GL_QUADS, GL_TRIANGLE_FAN e GL_POLYGON. Dentro da função foi utilizada a **glVertex3f()** preenchida manualmente com cada vértice de cada polígono que forma cada figura. A finalização se deu através da função **glEnd().**

Para a montagem do somatório foram utilizados seis polígonos, sendo o primeiro para a parte superior, o segundo para a parte inferior, o terceiro para a diagonal entre a parte superior e o centro, o quarto para a diagonal entre a parte inferior e o centro e os dois últimos na direita das partes superior e inferior formando as “pontas”.

A montagem das letras foi uma das partes mais complexas do projeto, pois cada letra é formada por uma série de polígonos e o mapeamento dos pontos foi inicialmente feito através do software Geogebra, depois passado para o OpenGL, onde foram definidas a escala e o posicionamento de uma barra base, que serviu para o cálculo de todas as letras seguindo a proporção dada pelo Geogebra.

Para finalizar o logo, foi sobreposto o cabo USB ao somatório, feito com dois círculos formados por triângulos através da opção GL_TRIANGLE_FAN da função **glBegin()**, um posicionado na base do cabo central e outro no cabo lateral à esquerda, um quadrado feito com GL_QUADS posicionado no cabo lateral direito e um triângulo feito com a GL_POLYGON posicionado no topo. Tudo foi ligado por uma série de polígonos verticais e diagonais formando a figura final presente no projeto.

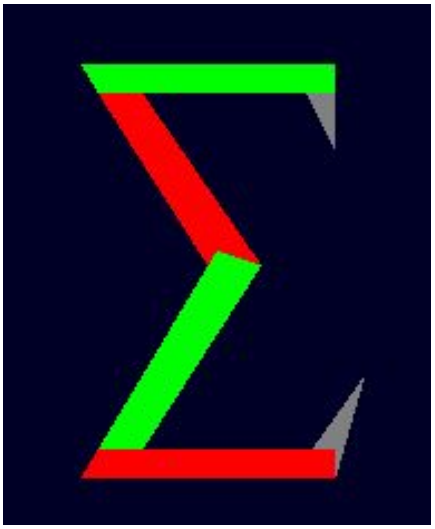

Imagem 7
Polígonos que formam o somatório.

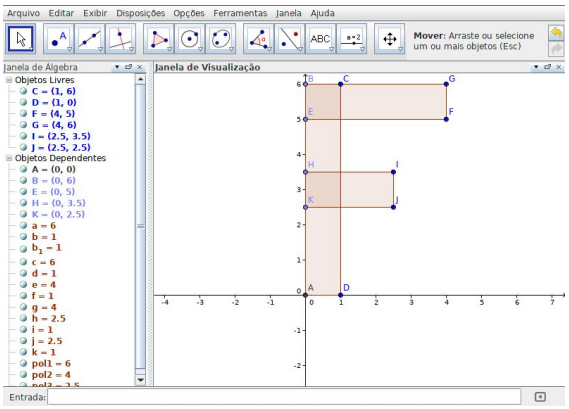

Imagem 8
Letra “F” no software Geogebra.

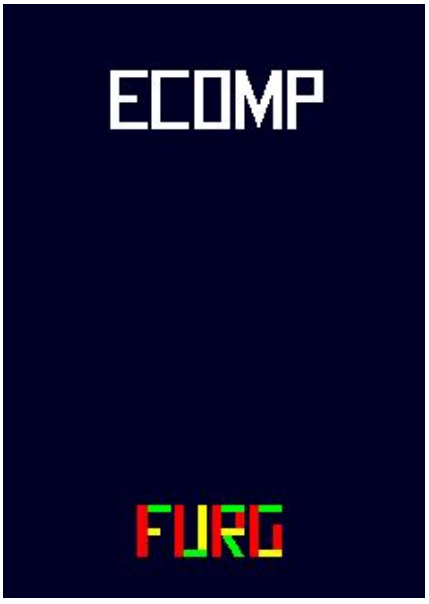

Imagem 9
Inscrições “ECOMP” em sua versão final e “FURG” com seus polígonos destacados.

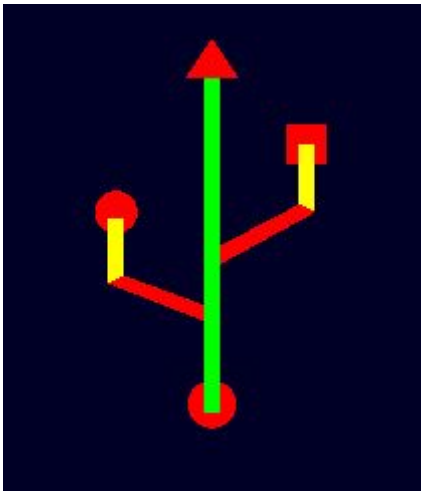

Imagem 10
USB com seus polígonos em destaque.

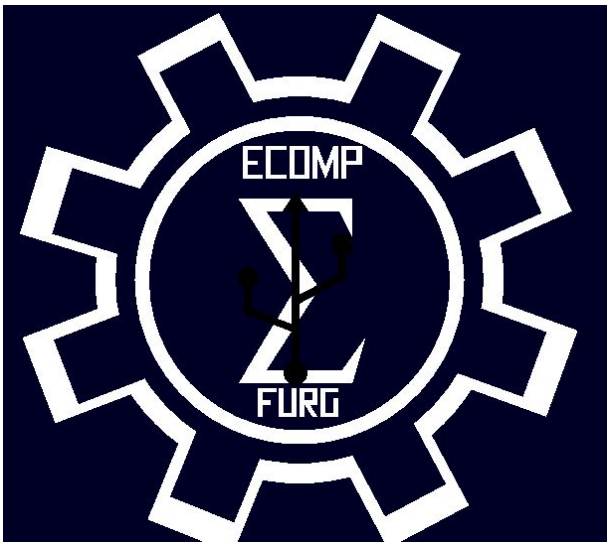

Imagem 11
Vista frontal final.

3. Transformações

As transformações geométricas são uma das bases do trabalho, estão presentes em todos os passos e possuem variados parâmetros. O ponto de maior destaque nesse quesito são os quadrados sobrepostos que formam os dentes da engrenagem.

Foi feito um quadrado dimensionado através da função **glutSolidCube()** com escala definida pela função **glScalef()**, posteriormente esse quadrado foi replicado e apenas com o uso das transformações de rotação e translação com as funções **glRotatef()** e **glTranslatef()** esses quadrados foram posicionados se sobrepondo e tangenciando os cilindros que formam o centro da engrenagem formando assim os seus dentes. O resultado final desse trabalho pode ser visto na imagem 3 logo acima.

Outro ponto que vale ser destacado é a possibilidade de rotacionar o objeto em si, o código possui funções desenvolvidas de rotação sobre os eixos X (rotaciona para cima e para baixo) e Y (rotaciona para a esquerda e para a direita). Essas funções são habilitadas pela entrada de dados do teclado com uma função desenvolvida exclusivamente para isso e que utiliza os comandos **GLUT_KEY_UP, GLUT_KEY_DOWN, GLUT_KEY_RIGHT** e **GLUT_KEY_LEFT** da própria OpenGL para indicar a direção da rotação e o comando **GLUT_KEY_END** para recolocar a imagem em sua posição original. Para que seja possível rotacionar todos os objetos ao mesmo tempo e da mesma forma, todos possuem a função **glRotatef()** duas vezes além da que define o seu posicionamento no cenário, com os parâmetros: glRotatef(rotação_em_x,1,0,0) para rotacionar em X e glRotatef(Rotação_em_y,0,1,0) para rotacionar em Y, sendo **rotação_em_x** e **rotação_em_y** variáveis globais inicializadas em zero (imagem sem rotação) e com os valores alterados de acordo com a entrada de dados lida do teclado.

Toda a face principal da figura também se utiliza de diversas transformações geométricas, incluindo escalas variadas para o somatório, as letras, a rasterização, a

USB e o próprio cilindro base e também as já mencionadas funções para rotacionar junto com o restante do objeto.

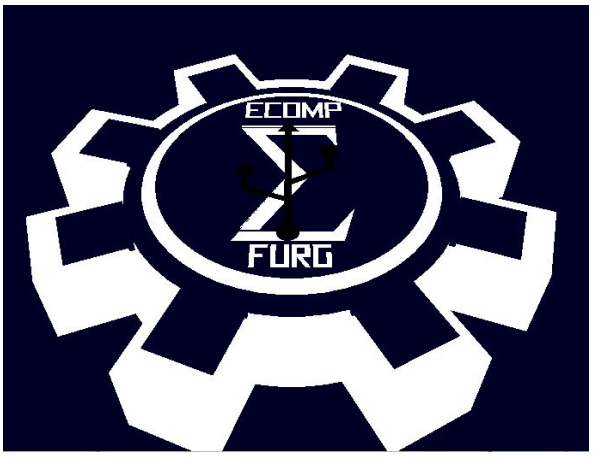

Imagem 12
Objeto rotacionado em torno do eixo X.

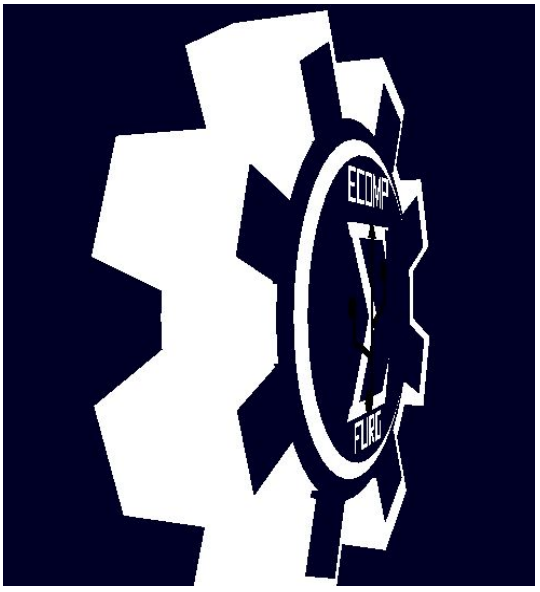

Imagem 13
Objeto rotacionado em torno do eixo Y.

4. Visibilidade

No quesito visibilidade, o espaço foi definido com a função **glMatrixMode()** utilizando como parâmetros as opções **GL_PROJECTION** e **GL_MODELVIEW** e a perspectiva de visualização foi definida com a função **gluPerspective()**. A profundidade se dá através da variação dos parâmetros do eixo Z nas figuras, os cilindros trazem essa espessura diretamente na função **gluCylinder()** que os forma, nos cubos ela se dá na escala, com a função **glScalef()** e na face os objetos são sobrepostos de acordo com a coordenada do eixo Z de seus polígonos, terceiro parâmetro da função **glVertex3f()**. Foi utilizado o algoritmo Z-Buffer para tratar o

objeto devido à sua eficiência e facilidade de uso proporcionado pela própria OpenGL, basta adicionar a função **glEnable()** com o parâmetro **GL_DEPTH_TEST** na inicialização que o algoritmo é ativado e a função **glClear()** com os parâmetros **GL_COLOR_BUFFER_BIT** e **GL_DEPTH_BUFFER_BIT** na própria função do display para manter o buffer atualizado.

Imagem 14
Vista frontal com alterações desproporcionais de perspectiva em X e Y.

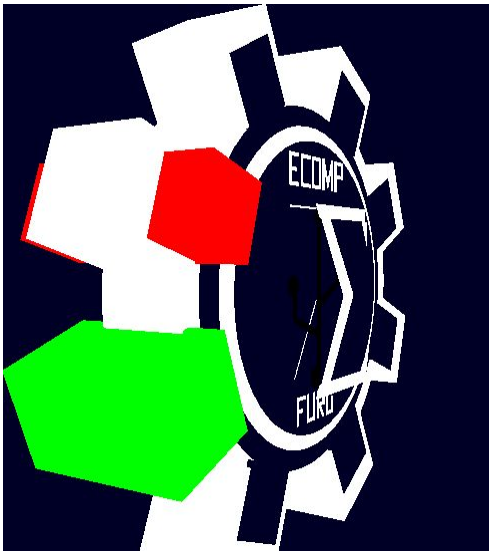

Imagem 15
Vista lateral com alterações na escala de um dos cubos externos (verde), um dos cubos internos (vermelho) e do somatório da face.

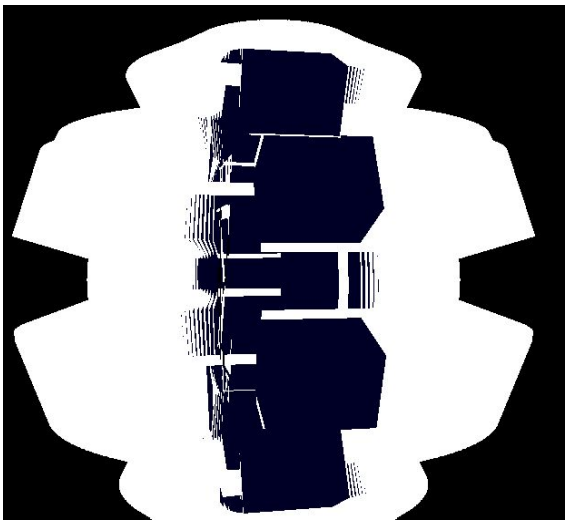

Imagem 16
Vista lateral do objeto sem o uso do algoritmo Z-Buffer.

5. Rasterização

Para rasterizar alguns objetos (linha superior e diagonal inferior esquerda do somatório da face) foi utilizado o algoritmo de Bresenham com duas variações implementadas, **RasterizaDxMaior()**, para tratar o caso com a variação em X maior e **RasterizaDyMaior()**, que trata o caso em que o maior intervalo se encontra no eixo Y. Ambas as funções recebem como parâmetro os pontos (x1,y1) e (x2,y2) iniciais e finais do segmento que deseja-se rasterizar. As escolhas desses objetos específicos foram justamente para aplicar as duas variações do algoritmo.

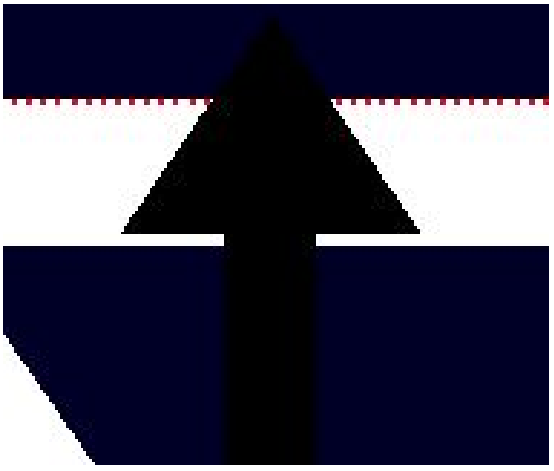

Imagem 17
Visão com perspectiva alterada da face com destaque para a linha superior do somatório rasterizada (linha pontilhada vermelha), também é visto o triângulo superior da USB em preto além do restante do polígono que forma a parte superior do somatório, em branco.

Imagem 18
Visão com perspectiva alterada da face com destaque para a linha diagonal do somatório rasterizada (linha pontilhada verde), também é visto o cabo central da USB em preto à direita além do restante do polígono que forma a diagonal inferior do somatório em branco e o cabo diagonal esquerdo da USB também em preto sobrepondo a linha rasterizada.

6. Iluminação

A iluminação do projeto conta com matrizes de luz ambiente, difusa, especular e posição da luz além de uma componente para definir o brilho do objeto.

O modelo de cor é o de Gouraud definido pela função **glShadeModel()** com o parâmetro **GL_SMOOTH** e são utilizadas as funções **glMaterialfv()** chamada algumas vezes com diferentes parâmetros tratando da especularidade, brilho e luz ambiente, **glLightfv()** que trata da posição da luz e da luz difusa, **glLightModelfv()** responsável pela luz ambiente e por último **glEnable()** que habilita os componentes do sistema de iluminação.

Embora com um vasto ramo de possibilidades implementado, a maioria dos componentes do sistema de iluminação está setada com parâmetros neutros ou bem semelhantes ao fundo para que o projeto seja fidedigno ao logo original, que conta com apenas duas cores e não se altera em função do ângulo em que é visualizado.

Imagem 19
Vista frontal com alteração na luz ambiente, utilização de uma tonalidade com a remoção da componente vermelha da luz final.

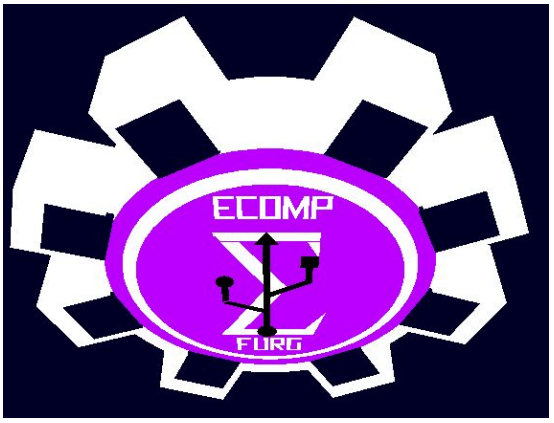

Imagem 20
Vista com rotação sobre o eixo X e alteração na especularidade.

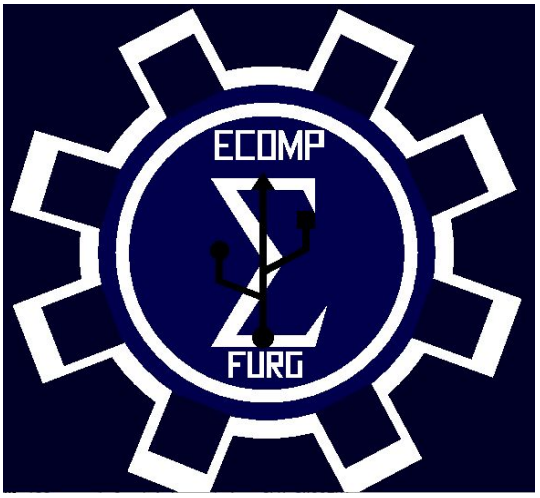

Imagem 21
Vista frontal com alteração na posição da luz.

7. Conclusão

O objetivo do projeto foi atingido, pois um logo tridimensional remetente à FURG foi desenvolvido utilizando OpenGL e algoritmos vistos durante as aulas, explicitados ao longo deste relatório. Particularmente considero muito interessante o resultado final e a forma manual como o projeto foi executado, trazendo um grande aprendizado sobre a OpenGL. O principal problema enfrentado foi a falta de tempo devido à sobrecarga imposta pelo final do semestre e o trabalho com o algoritmo de rasterização, pois nessa fase surgiram alguns erros complexos que tiveram que ser tratados. Em resumo, um grande trabalho manual, porém com um grande conhecimento adquirido.

```
1309 //-----------------Função principal
1310
1311 int main(int argc, char **argv) {
1312     glutInit(&argc, argv);
1313     glutInitWindowSize(800,600);
1314     glutCreateWindow("Logo Moletom");
1315     glutSpecialFunc(teclado);
1316     glutDisplayFunc(display);
1317     init();
1318     glutReshapeFunc(reshape);
1319     glutMainLoop();
1320     return 0;
1321 }
```

Imagem 22
Função principal do programa com destaque para o tamanho do código (1321 linhas).

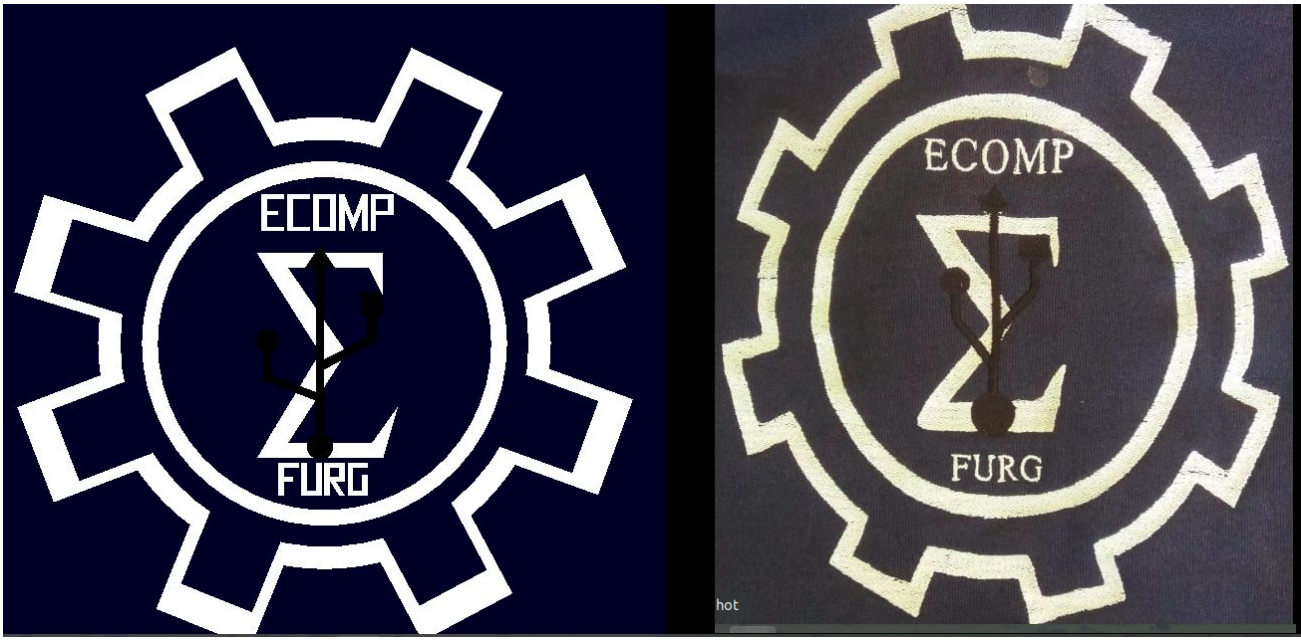

Imagem 23
Resultado final do projeto com o logo desenvolvido à esquerda e o logo original à direita.