Trabalho de Computação Gráfica: Logo do 7-Eleven

João Kawazoe, *Estudante, Ciência da Computação,* and João Mendes, *Estudante, Ciência da Computação,*

***Abstract*—This work resume our efforts on making 7-eleven's logo.**

***Index Terms*—Computação Gráfica, OpenGL, Transformações Gráficas, logo, 7-eleven.**

I. Introdução

ESTE é o relatório proposto sobre o trabalho de criação de logos, no qual foi utilizada a linguagem Java com a biblioteca OpenGL. Este, tem o objetivo de reproduzir a logo do *7-eleven* por meio de código Java com OpenGL.

A. Relatório de desenvolvimento da reprodução da logo do 7-Eleven

Para a solução do problema, o qual era sintetizar a logo do *7-Eleven*, a solução era resumir todas as formas mais complexas como as letras e o numero **7** da figura para figuras geométricas mais simples, como retângulos, trapézios e triângulos, e para as curvaturas presentes na imagem original simplesmente aplicamos uma função de curvatura nas arestas de algumas dessas formas geométricas. Para solucionar o problema da genuinidade das cores, abrimos a logo original no programa da Microsoft **Paint** e pegamos uma amostra de cada cor e verificamos qual eram seus respectivos valores **RGB** (*Red Green Blue*), porem foi feito uma adaptação de valores pois a função usada para estabelecer as cores dos polígonos trabalha com porcentagem de valores **RGB** e não com faixas de valores que vão de 0 à 255, então para cada amostra de cor calculou-se a porcentagem de *red* (vermelho), *green* (verde) e *blue* (azul).

1) Background: Primeiramente, foi determinado que o background (plano de fundo) seria a parte mais exterior a imagem e não sobrepunha outras figuras, a conclusão foi que a a parte mais "inferior" seria o "quadrado" verde na logo.

Agradecimento ao Stack Overflow.
João Kawazoe e João Mendes cursam na UEL.
Manuscript received September 30, 2019

Fig. 1. Background em destaque.

```
//********* BACKGROUND  ********
gl.glOrtho(0, resolutionWidth, 0, resolutionHeight, -1, 1);
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f(10,22);
    gl.glVertex2f(resolutionWidth-10,22);
    gl.glVertex2f(resolutionWidth-10,resolutionHeight-10);
    gl.glVertex2f(10,resolutionHeight-10);
}
gl.glEnd();
```

Fig. 2. Trecho do codigo do background.

A Figura 2 mostra o codigo responsável por desenhar o *background* da Logo *7-Eleven*. Na A função **glOrtho** serve para que a "régua" do *frame* vá de 0 ao tamanho da resolução da imagem, essa alteração foi feita para que o posicionamento dos polígonos fosse mais preciso, pois o padrão da régua do *frame* OpenGl vai de 0 a 1. A função **glColor3f** serve para predeterminar a cor do polígono a ser desenhado no *frame*, sendo os parâmetros *red*, *green* e *blue* na respectiva ordem e os valores de 0 a 1 correspondendo a sua respectiva porcentagem, logo o verde mostrado na Figura 1 é 50% Verde e 38% azul, sendo vermelho nulo. Em seguida **glBegin** e *glEnd* determinam o começo e o fim do polígono a ser desenhado respectivamente. O parâmetro **GL2.GL_POLYGON** passado para **glBegin**, serve para indicar que será determinado por vértices um polígono qualquer, já a função **glVertex2f** é a função que determina os vértices dos polígonos em duas

dimensões, e na Figura 2 as 4 chamadas de **glVertex2f** são responsáveis por desenhar um quadrado verde no frame.

2) Canvas branco: Foi a segunda figura a ser desenhada, pois estava mais interior ao *background* mas existiam figuras que as sobrepunham. Antes de tudo, o cavas foi simplificado como uam trapézio invertido, e depois foram adicionados dois vértices a mais em cada canto para que fosse possível desenhar os cantos arredondados da figura original

Fig. 3. Canvas em destaque.

```
//********** WHITE CANVAS  ********
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin( GL2.GL_POLYGON);{
    for(x=20f; x> 0f; x--){
        y = (pow(x,2)/26);
        gl.glVertex2f(125-x, 70+y);
    }
    //gl.glVertx2f(120,60);
    for(x=0f; x< 20f; x++){
        y = (pow(x,2)/33);
        gl.glVertex2f((resolutionWidth-125)+x, 70+y);
    }
    //gl.glVertex2f(resolutionWidth-120,60);
    for(x=0f; x< 25f; x++){
        y = (-pow(x,2)/23)+(x*2.1f);
        gl.glVertex2f((resolutionWidth-60) - x, (resolutionHeight-66) + y);
    }
    //gl.glVertex2f(resolutionWidth-60,resolutionHeight-40);
    for(x=0f; x< 25f; x++){
        y = (pow(x,2)/21);
        gl.glVertex2f(((60+25) - x), (resolutionHeight-41) - y);
    }
    //gl.glVertex2f(60,resolutionHeight-40);
}
gl.glEnd();
```

Fig. 4. Trecho do codigo do canvas branco.

A cor do Canvas na logo original é branca, logo na função **glColor3f** os valores para vermelho, verde e azul são todos 1 (100%). Já os laços **for** dentro de **glBegin** são responsáveis por determinar cada ponto da curva entre um ponto e o outro nos cantos do trapézio, para descobrir qual função se adequava mais a curva da logo original foi usado a técnica da tentativa e erro em relação aos valores, tendo em mente que o divisor na variável de segundo grau diminuía a intensidade da curva e o multiplicador na variável de primeiro grau alterava a posição em relação ao eixo **x**.

3) Número 7: O número **7** foi divido em 3 partes principais, a parte do topo, a parte lateral superior e a parte inferior.

A parte superior foi a terceira a ser desenhada, pois se encontrava interior ao canvas e era sobreposta pela sua lateral, também era a figura mais simples, um retângulo laranja.

Fig. 5. Parte superior do número 7 em destaque.

```
//**********  SEVEN  ********
//**********OrangeTop*********
gl.glColor3f(0.96f, 0.5f, 0.12f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f(145,resolutionHeight-219);
    gl.glVertex2f(145,resolutionHeight-112);
    gl.glVertex2f(resolutionWidth-145,resolutionHeight-112);
    gl.glVertex2f(resolutionWidth-145,resolutionHeight-219);
}
gl.glEnd();
```

Fig. 6. Trecho do código da parte superior do número 7.

A Figura 6 mostra o código usado para desenha-lo no frame. A função **glColor3f** nesse caso irá receber como parâmetro 96% vermelho, 50% verde e 12% azul. Por sua vez as 4 chamadas de **glVertex2f** determinam sua posição na imagem.

A parte lateral foi desenhada em quarto, pois sobrepõe a parte superior laranja do 7 e é constituída por dois polígonos de quatro lados, um branco “por baixo” e um vermelho “por cima”, ambos tem a mesma forma, porém o branco era levemente maior para criar um certo tipo de borda, conforme a Figura 7.

Fig. 7. Parte lateral do número 7 em destaque.

```
//**********whiteBorder_upper**********
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f(358,205);
    gl.glVertex2f(228,205);
    for(x=0; x< 212; x++){
        y = (-pow(x,2)/218) + (x*1.95f);
        gl.glVertex2f(x+248, y+286);
    }
    gl.glVertex2f(resolutionWidth-140,520);
    gl.glVertex2f(resolutionWidth-140,360);
}
gl.glEnd();
```

Fig. 8. Trecho do código da parte lateral branca do número 7.

Na Figura 8 o laço **for** é responsável por dar a lateral branca uma curvatura, usando a mesma técnica mencionada anteriormente, e como a figura é branca, as cores vermelha, verde e azul são todas 100%.

```
//**********redSide_upper**********
gl.glColor3f( v: 0.92f, v1: 0.1f, v2: 0.17f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f( v: 376, v1: 286);
    gl.glVertex2f( v: 263, v1: 286);
    for(x=0; x< 200; x++){
        // quanto menor o divisor mais rapido a curva.
        // quanto maior maior o multiplicador mais ingrime a reta.
        y = (-pow(x, v1: 2)/230) + (x*1.9f);
        gl.glVertex2f( v: x+263, v1: y+286);
    }
    gl.glVertex2f( v: resolutionWidth-138, v1: resolutionHeight-220);
}
gl.glEnd();
```

Fig. 9. Trecho do código da parte lateral vermelha do número 7.

Já na Figura 9 é mostrado o código responsável por desenhar a figura vermelha em cima da figura branca. Ambas são semelhantes, porém esta é ligeiramente menor que a branca e possui 92% de vermelho, 10% de verde e 17% de azul. Além disso, para não deixar a lateral direita do polígono direito (Figura 7) foi colocado um triângulo branco na sua lateral direita inferior, como uma curvatura semelhante no lado direito, como mostra a Figura 10.

```
//Triangulo Branco
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f(462,286);
    for(x=0; x< 87; x++){
        y = (-pow(x,2)/110) + (x*1.9f);
        gl.glVertex2f(x+376, y+286);
    }
}
gl.glEnd();
```

Fig. 10. Trecho do código do triângulo branco do número 7.

Finalmente a parte inferior do número 7, segue a mesma logica da lateral, porem esta não será necessário a aplicação de nenhuma curvatura, como é mostrado nas figuras 11,12,13.

Fig. 11. Parte inferior no número 7 em destaque

```
//**********whiteBorder_bottom**********
gl.glColor3f( v: 1.0f, v1: 1.0f, v2: 1.0f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f( v: 220, v1: 40);
    gl.glVertex2f( v: 366, v1: 40);
    gl.glVertex2f( v: 370, v1: 200);
    gl.glVertex2f( v: 220, v1: 200);
}
gl.glEnd();
```

Fig. 12. Trecho do código da parte inferior branca do número 7.

```
//**********redSide_bottom**********
gl.glColor3f(0.92f, 0.1f, 0.17f);
gl.glBegin( GL2.GL_POLYGON);{
    gl.glVertex2f(230,48);
    gl.glVertex2f(356,48);
    gl.glVertex2f(363,181);
    gl.glVertex2f(237,181);
}
gl.glEnd();
```

Fig. 13. Trecho do código da parte inferior vermelha do número 7.

4) Letra E: A letra **E** foi a quinta a ser desenha já que ela é simples. A idéia para reproduzir a letra E é considera-lo como um retângulo e depois adicionar mais dois retângulos brancos sobrepondo parte do seu lado direito como mostra as Figuras 14 e 15 . Suas cores são idênticas ao plano de fundo.

Fig. 14. Letra E em destaque.

```
//*********  E  *******
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(125,185);
    gl.glVertex2f(125,280);
    gl.glVertex2f(170,280);
    gl.glVertex2f(170,185);
}
gl.glEnd();
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(150,240);
    gl.glVertex2f(150,260);
    gl.glVertex2f(170,260);
    gl.glVertex2f(170,240);
}
gl.glEnd();
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(150,205);
    gl.glVertex2f(150,225);
    gl.glVertex2f(170,225);
    gl.glVertex2f(170,205);
}
gl.glEnd();
```

Fig. 15. Trecho do código da letra E.

5) Letra L: A letra **L** foi a sexta figura a ser desenhada, também muito simples usado a mesma técnica da letra **E**, como mostra as Figuras 16 e 17. Um retângulo maior verde e por cima outro retângulo menor o sobrepondo no canto superior direito.

Fig. 16. Letra L em destaque.

```
//**********  L  ********
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(180,185);
    gl.glVertex2f(180,280);
    gl.glVertex2f(230,280);
    gl.glVertex2f(230,185);
}
gl.glEnd();
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(205,205);
    gl.glVertex2f(205,280);
    gl.glVertex2f(230,280);
    gl.glVertex2f(230,205);
}
gl.glEnd();
```

Fig. 17. Trecho do código da letra L.

6) Letra V: A letra **V** foia sétima figura a ser desenhada. É composta por dois retângulos angulados sobrepostos formando um angulo agudo em seu interior, como mostra as Figuras 18 e 19.

Fig. 18. Letra V em destaque.

```
//**********  V  ********
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(295,280);
    gl.glVertex2f(320,280);
    gl.glVertex2f(330,185);
    gl.glVertex2f(310,185);
}
gl.glEnd();
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(335,280);
    gl.glVertex2f(355,280);
    gl.glVertex2f(340,185);
    gl.glVertex2f(320,185);
}
gl.glEnd();
```

Fig. 19. Trecho do código da letra V.

7) Letra n: Finalmente a letra **n** foi desenvolvida por último, pelo fato de ser a única letra minúscula, possui formas um pouco mais complexas, mas ainda foi possível resumi-la em forma básicas como mostra a Figura 20. O **n** é composto pro 3 retângulos, um no lado esquerdo mais comprido, outro no lado direito um pouco mais baixo e mais largo, e outro na parte inferior central para fazer a divisão da letra.

Fig. 20. Forma básica da letra n.

Após desenhar a forma básica do **n** adicionamos uma curvatura na parte superior do retângulo direito e na parte superior do retângulo branco central como é mostrado nos laços do código da Figura 21. Assim o resultado é mostrado na Figura 22

```
//**********  n  ********
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(420,185);
    gl.glVertex2f(420,280);
    gl.glVertex2f(445,280);
    gl.glVertex2f(445,185);
}
gl.glEnd();
//--
gl.glColor3f(0.0f, 0.5f, 0.38f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(445,185);
    gl.glVertex2f(446,260);
    for(x=0.0f; x<41.0f; x++){
        y = -(pow(x,2)/65)+(x * (0.65f));
        gl.glVertex2f(x+445, y+273);
    }
    gl.glVertex2f(485,260);
    gl.glVertex2f(485,185);
}
gl.glEnd();
//--
gl.glColor3f(1.0f, 1.0f, 1.0f);
gl.glBegin(GL2.GL_POLYGON);{
    gl.glVertex2f(445,185);
    gl.glVertex2f(445,260);
    for(x=0.0f; x<16.0f; x++){
        y = -(pow(x,2)/25)+(x * (0.65f));
        gl.glVertex2f(x+445, y+253);
    }
    gl.glVertex2f(460,230);
    gl.glVertex2f(460,185);
}
gl.glEnd();
```

Fig. 21. Trecho de código da letra n.

Fig. 22. Letra n em destaque.

8) Transformação: A transformação da imagem é feita após um comando recebido através de um *KeyListener*, desta forma, com a inicialização correta das variáveis, e a implementação das funções dentro do *display* como ilustradas na Figuras 23, basta o usuário teclar de acordo com a seguinte instrução:

- Teclar ”d” para rotacionar no sentido horário;
- Teclar ”e” para rotacionar no sentido anti horário;
- Teclar ”s” para parar a rotação;
- Teclar ”g” para aumentar a escala da imagem;
- Teclar ”p” para diminuir a escala da imagem;
- Teclar as setas do teclado para trasladar de acordo com as respectivas direções da tecla.

```
private float rtri = 0;
public static int resolutionWidth =  600;
public static int resolutionHeight = 600;
private float rotacao = 0;
private float largura = 1;
private float altura = 1;
private float moverx = 0;
private float movery = 0;
@Override
public void display(GLAutoDrawable arg0){
    final GL2 gl = arg0.getGL().getGL2();
    gl.glClear ( GL2.GL_COLOR_BUFFER_BIT |  GL2.GL_DEPTH_BUFFER_BIT );
    gl.glLoadIdentity();
    //Transformacoes
    gl.glRotatef( rtri,  0.0f,  0.0f, rotacao );
    gl.glScalef( largura, altura, 1.0f);
    gl.glTranslatef(moverx, movery,  1.0f);
```

Fig. 23. Inicialização das variáveis e a chamada das funções.

II. Conclusão

Fazendo o uso da biblioteca Opengl, e simplificando as formas mais complexas em polígonos mais simples foi possível obter fidelidade na reprodução da logo do 7-Eleven, como pode ser visto na Figura 24, onde a logo original se encontra a esquerda e a reprodução por código a direita.

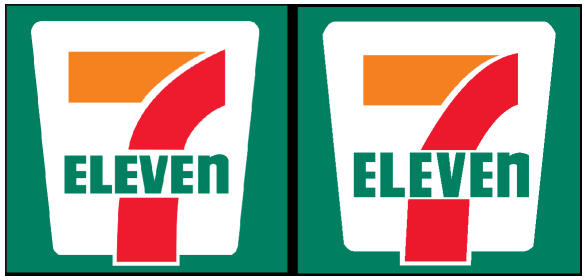

Fig. 24. Comparação entra a logo original (esquerda) e os resultados obtidos pelo código (direita).

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.