Relatório do desenvolvimentos dos algoritmos propostos
Aluna: Júlia Silva Castro

Em todos os meus exercícios estão presentes as funções:

- Função tecla(key,x,y) que faz da letra 'q' do teclado a responsável pelo encerramento da janela.

```
void tecla(unsigned char key, int x, int y){
    if (key == 'q')
        exit(0);
}
```

Figura 1. Exibição da função tecla().

- Função inicializa() que possui alguns métodos importantes como:
  - glMatrixMode(GL_PROJECTION) que mostra o tipo de matriz de projeção utilizada;
  - glLoadIdentity() que uma vez criada uma matriz, eu preciso dessa função pra criar uma matriz identidade que recebe os valores;
  - gluOrtho2D(0,200,0,200) que define o espaço do ambiente a ser trabalhado, nesse caso eu começo do x inicial com valor 0 e vai até 200 e o mesmo se repete com o y.

```
void inicializa(void){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,200,0,200);

}
```

Figura 2. Exibição da função inicializa().

- Função main(int argc, char** argv) que possui alguns métodos importantes como:
  - glutInit(&argc, argv) usada para iniciar a biblioteca GLUT;
  - glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB) que seta nossa janela inicial, nesse caso em apenas uma janela utilizando o sistema de cor RGB;
  - glutInitWindowsSize(800,500) que define o tamanho da exibição da janela de saída;

- glutInitWindowsPosition(200,10) que define a localização que a janela de saída vai aparecer na minha tela;
- glutCreateWindow("introdução a opengl") que define o título da janela de saída;
- glutDisplayFunc(desenha) que é o responsável por colocar os meus desenhos criados na tela de saída;
- inicializa() responsável por executar função inicializa()
- glutKeyBoardFunc(tecla) que estabelece a função *callback* que é chamada pela GLUT cada vez que uma tecla é pressionada;
- glutMainLoop() função muito importante para fazer com que o programa realmente funcione e só pode ser chamada no máximo uma vez.

```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(800,500);
    glutInitWindowPosition(200,10);
    glutCreateWindow("Introdução a OPENGL");
    glutDisplayFunc(desenha);
    inicializa();
    glutKeyboardFunc(tecla);
    glutMainLoop();
    return 0;
}
```

Figura 3. Exibição da função principal, o main().

- Função desenha() que é responsável pelo código das figuras criadas, essa é alterada a cada exercício proposto. Em meu relatório, a medida que explicarei o que foi feito nos exercícios só exibirei prints dessa função.

Exercício 1 - Foi pedido para que criássemos uma sequência de 20 pontos em ordem crescente.

```
void desenha(){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    int x=0;int y=30;int tam=3;

    for(int i=1; i<21;i++){
        glPointSize(i);
        glBegin(GL_POINTS);
        glColor3f(1, 0, 0);
        glVertex2f(x,y);
        x=x+i;
        glEnd();
        glFlush();
    }
}
```

Figura 4. Código ex1.

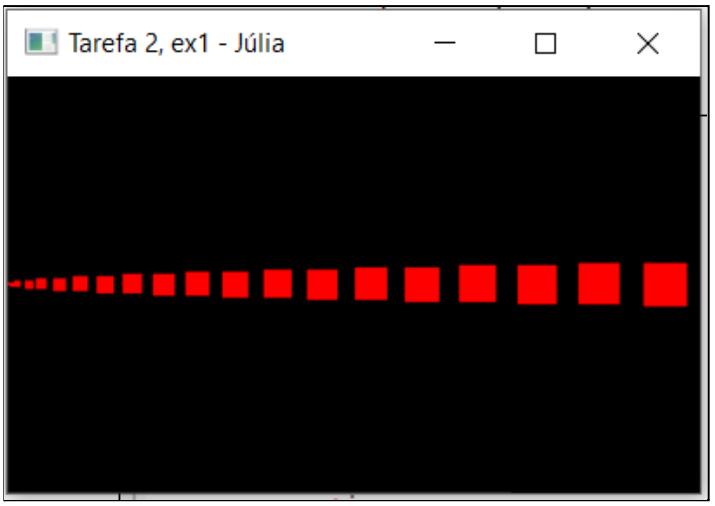


Figura 5. Saída do ex1.

Para criar uma sequência de 20 pontos eu usei um laço. Dentro desse laço eu chamo a função glPointSize que recebe por parâmetro o contador, ou seja, o tamanho do meu ponto aumenta à medida que o laço é percorrido. Também chamo a função glBegin(GL_POINTS) responsável por criar meus pontos.

A função flColor3f() recebe por parâmetro a quantidade de cor vermelha, verde e azul que meu ponto vai receber, no caso (1,0,0) é a cor vermelha. A função glVertex2f() define as posições x,y do meu ponto, no caso o y sempre vai ser 30 e meu X aumenta ao decorrer da execução do laço, devido ao valor dele ser somado com o valor do contador. Assim meus pontos vão sendo exibidos sempre linearmente em relação ao eixo y e variam sua posição x para direita.

Exercício 2 - Foi pedido para que criássemos uma linha metade vermelha e metade verde e descobríssemos o valor do pixel do meio.

Figura 6. Código ex2.

```
void desenha(void){
    glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT);

    glLineWidth(60);
    glBegin(GL_LINES);
        glColor3f(1, 0, 0);
        glVertex2f(20,150);
        glColor3f(0, 1, 0);
        glVertex2f(80,150);
     glEnd();
     glFlush();

    glPointSize(5);
    glBegin(GL_POINTS);
        glColor3f(0.5,0.5,0);
        glVertex2f(50,160);

    glEnd();
    glFlush();
}
```

Figura 7. Saída ex2.

A função glLineWidth é responsável por definir a espessura da minha linha, então eu defini o valor 60 para ficar fácil a visualização dos pixels. Após definir a espessura, eu criei minha linha através da função glBegin(GL_LINES); com o primeiro vértice nas posições x=20, y=150 através da função glVertex2f(20,150) recebendo pixel de valor vermelho através da função glColor2f(1,0,0); depois eu criei meu segundo vértice nas posições x=80, y=150 através da função glVertex2f(80,150) recebendo pixel de valor verde através da função flColor(0,1,0).

E então eu criei meu ponto através da função glBegin(GL_POINTS) de tamanho 5, valor definido pela função glPointsize, que recebia o vértice na posição x=50, y=150 recebendo metade do pixel vermelho e metade do pixel verde.

A interseção de cores da linha é meio amarelada, é um amarelo fraco pois as cores verdes e vermelhas fortes estão somente na extremidade, no meio é somado metade de pixel de um e metade do pixel de outra.

Exercício 3 - Foi pedido para que criasse a figura de uma casinha.

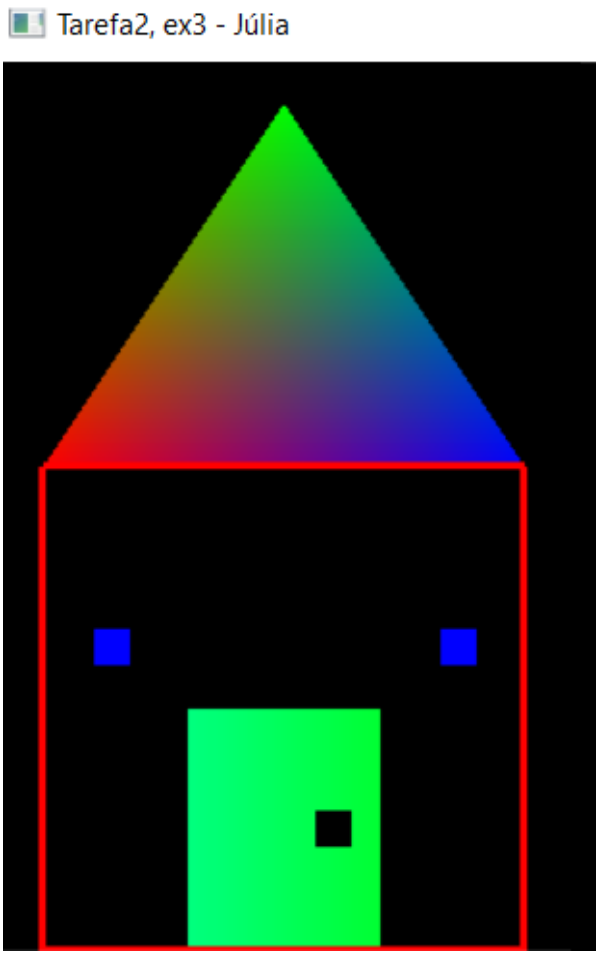
Figura 8. Código ex3.

```
void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    glLineWidth(8);
    glBegin(GL_TRIANGLES);
        glColor3f(1, 0, 0);
        glVertex2d(50,80);
        glColor3f(0,0 , 1);
        glVertex2d(100,80);
        glColor3f(0, 1, 0);
        glVertex2d(75,140);
     glEnd();
     glFlush();

    glBegin(GL_QUAD_STRIP);
        glColor3f (0.0, 1.0, 0.5);
        glVertex2d(65,0);
        glVertex2d(65,40);
        glColor3f (0.0, 1, 0.2);
        glVertex2d(85,0);
        glVertex2d(85,40);
     glEnd();
     glFlush();
```

```
glLineWidth(3);
glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
glBegin(GL_QUADS);
    glColor3f(1, 0, 0);
    glVertex2d(50,0);
    glVertex2d(50,80);
    glVertex2d(100,80);
    glVertex2d(100,0);
 glEnd();
 glFlush();

 glPointSize(15);
 glBegin(GL_POINTS);
    glColor3f(0, 0, 1);
    glVertex2f(57,50);
    glVertex2f(93,50);
    glColor3f(0, 0, 0);
    glVertex2f(80,20);
 glEnd();
 glFlush();
```

Figura 9. Saída ex3.

Figura 9 - Saída ex3.



Na construção dessa figura eu sei 4 funções de formas diferentes, vamos começar a explicar pela construção do telhado.

Para o telhado eu usei a função glBegin(GL_TRIANGLES) que é a responsável por criar um triângulo, os vértices dessa forma é definido pelas funções glVertex2d(x,y) e entre cada vértice eu defini uma cor pela função glColor3f(r,g,b).

Em seguida eu defini a porta, colorida de verde. Para criar uma porta eu usei a função GL_QUAD_STRIP, responsável por criar quadrados. Nessa forma, eu defini os dois vértices da direita da porta com uma cor misturada em verde mais forte com azul médio e depois defini os outros dois vértices da parte esquerda da porta com uma cor misturada de verde forte e um azul mais claro, o que deu esse cor gradiente na porta.

Após ter o telhado e a porta prontas, eu construí um quadrado para representar a casa, criado pela função glBegin(GL_QUADS). Nessa forma geométrica eu precisei usar o glPolygonMode() para não colorir o quadrado da casa. O quadrado é criado pela cor vermelha, definido na função glColor() e seus vértices definidos pela função glVertex2d().

Para finalizar a arte, eu construí 3 simples pontos para representar as janelas e a maçaneta da porta. Para a construção de pontos usei a função glBegin(GL_POINTS) com espessura 15 definida pela função glPointSize(). Criei dois pontos azuis, definidos pela função glColor() e suas posições pela função glVertex(). Depois defini a cor preta para maçaneta da porta, pela função glColor3f() sua localização.

Exercício 4 - Foi pedido para que criássemos os eixos x e y e figuras nos quatros eixos cartesianos.

```
void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    glLineWidth(5);
    glBegin(GL_LINES);
        glColor3f(1, 1, 1); //eixo x e eixo
        glVertex2f(100,0);
        glVertex2f(100,300);
        glVertex2f(0,100);
        glVertex2f(300,100);
        glColor3f(0, 1, 1); //letra J
        glVertex2f(10,170);
        glVertex2f(40,170);
        glVertex2f(10,130);
        glVertex2f(25,130);
        glVertex2f(25,130);
        glVertex2f(25,170);
        glColor3f(1, 1, 0); //letra U
        glVertex2f(50,170);
        glVertex2f(50,130);
        glVertex2f(70,170);
        glVertex2f(70,130);
        glVertex2f(50,130);
        glVertex2f(70,130);
        glVertex2f(60,170); //acento U
        glVertex2f(70,190);
        glColor3f(1, 0, 0); //Letra J
        glVertex2f(130,170);
        glVertex2f(130,130);
        glVertex2f(130,130);
        glVertex2f(150,130);
        glColor3f(0, 1, 0); //letra I
        glVertex2f(45,80);
        glVertex2f(45,30);
        glColor3f(0.2, 0.5, 1); //letra A
        glVertex2f(130,30);
        glVertex2f(150,80);
        glVertex2f(150,80);
        glVertex2f(170,30);
        glVertex2f(139,50);
        glVertex2f(162,50);
    glEnd();
    glFlush();
```

```
glPointSize(5);
glBegin(GL_POINTS);
    glColor3f(1.0, 0.0, 1.0);
    glVertex2f(45,85);

 glEnd();
 glFlush();
```

Figura 10 - Código ex4.

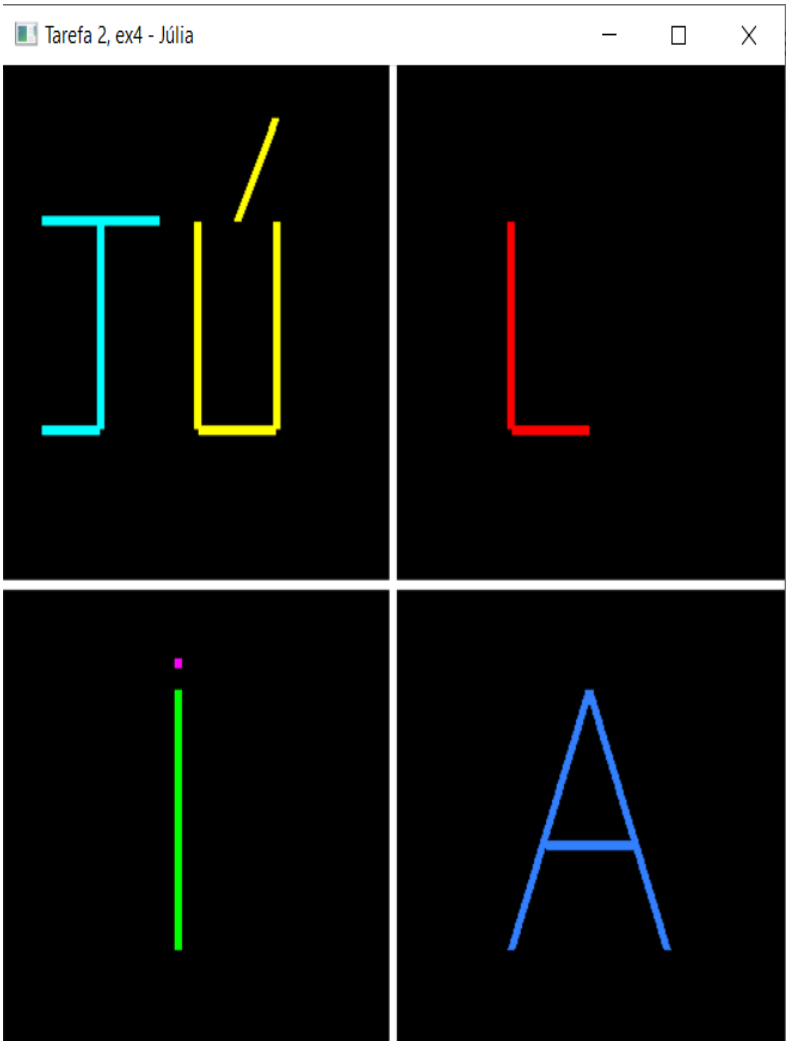


Figura 11 - Saída ex4.

Nesse exercício eu dividi as letras do meu nome entre os quatros quadrantes, eu usei somente linhas,para formar os eixos x e y e também as letras e o ponto para colocar acento na letra i.

Foi uma atividade tranquila pois foi como todas as outras, usei as funções glBegin(GL_LINES) na criação das linhas e depois defini as cores pelo método glColor3f(r,g,b) e seus vértices pelo método glVertex2f(x,y) e repeti o mesmo processo na criação do ponto, trocando apenas o método glBegin que, neste caso, recebe GL_POINTS como parâmetro.

Exercícío 4 - Foi pedido para que criássemos a mesma figura nos quatros quadrantes de forma simétrica

```
void desenha(void){
    glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT);
    //blue,green,red
    glLineWidth(8);
      glBegin(GL_QUAD_STRIP);
        glColor3f (0, 1, 0); //primeiro quadrante
        glVertex2d(120,140);
        glColor3f (0, 0, 1);
        glVertex2d(120,200);
        glColor3f (1, 0, 0);
        glVertex2d(160,140);
        glColor3f (0, 0.8, 0);
        glVertex2d(160,200);
     glEnd();
     glFlush();
     glBegin(GL_QUAD_STRIP);
        glColor3f (0, 1, 0); //segundo quadrante
        glVertex2d(100,140);
        glColor3f (0, 0, 1);
        glVertex2d(100,200);
        glColor3f (1, 0, 0);
        glVertex2d(60,140);
        glColor3f (0, 0.8, 0);
        glVertex2d(60,200);
     glEnd();
     glFlush();
     glBegin(GL_QUAD_STRIP);
        glColor3f (0, 0.8, 0); //terceiro quadrante
        glVertex2d(60,40);
        glColor3f (1, 0, 0);
        glVertex2d(60,100);
        glColor3f (0, 0, 1);
        glVertex2d(100,40);
        glColor3f (0, 1, 0);
        glVertex2d(100,100);
     glEnd();
     glFlush();
     glBegin(GL_QUAD_STRIP);
        glColor3f (0, 1, 0); //quarto quadrante
        glVertex2d(120,100);
        glColor3f (0, 0, 1);
        glVertex2d(120,40);
        glColor3f (1, 0, 0);
        glVertex2d(160,100);
        glColor3f (0, 0.4, 0);
        glVertex2d(160,40);
      glEnd();
      glFlush();

}
```

Figura 12 - Código ex4.

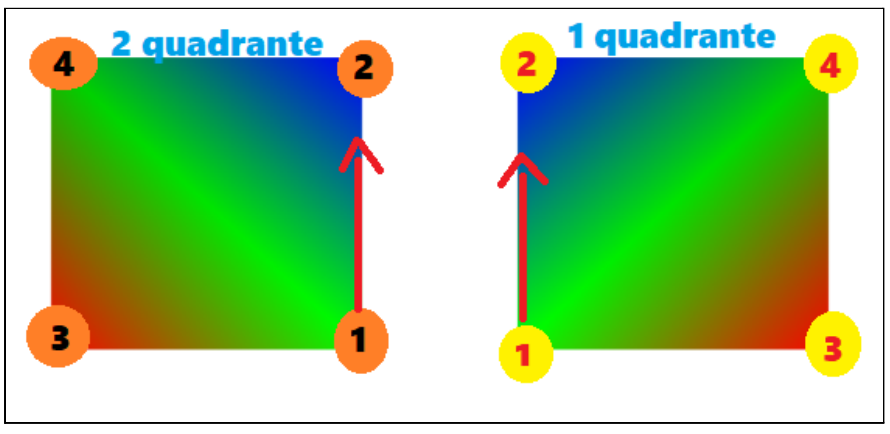
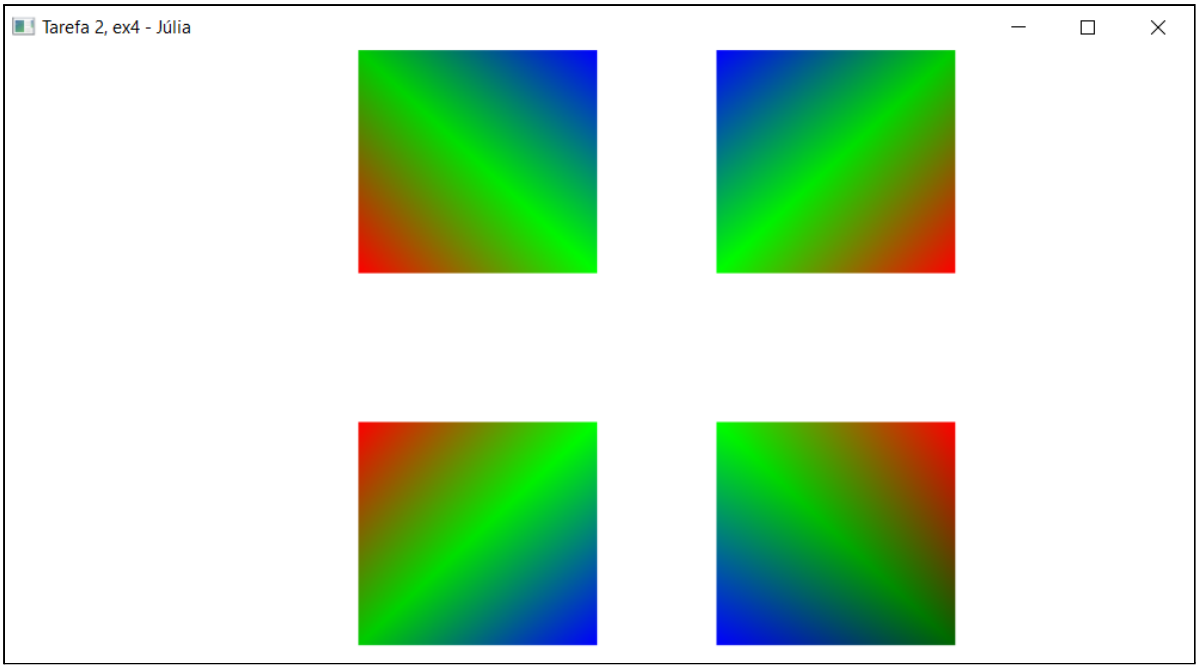


Figura 13. Explicação da diferença na ordem da definição dos vértices

Figura 14 - Saída ex4.



Nesse exercício eu usei somente a forma GL_QUAD_STRIP para que os quadrados ficassem coloridos dessa forma. Chamei 4 funções dessas para formar cada quadrado.

Como sempre, usei os métodos glVertex2d(x,y) para definir as posições de cada vértice (no caso, 1 quadrado tem 4 vértices) e o método glColor3f (r,g,b) entre cada vértice para variação das cores.

A curiosidade desse exercício é a ordem que você define os vértices, se seu objetivo é a simetria. Como detalhado na Figura 13, nós criamos os vértices na sequência de crescimento que você quer a "linha" verde.

Exercício 6 - Neste exercício foi pedido para reproduzir duas figuras, uma só com as linhas e a outra toda colorida, usando a forma GL_TRIANGLE_STRIP.

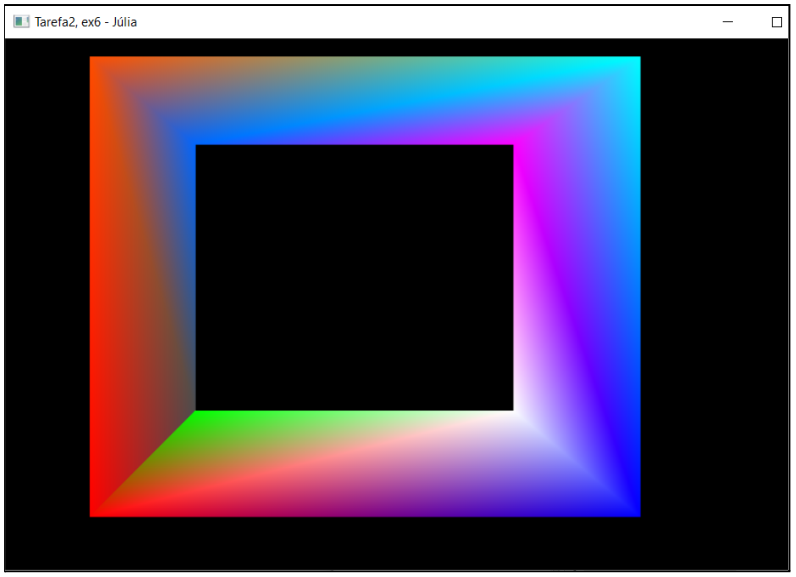


Figura 15 - Saída da figura colorida.

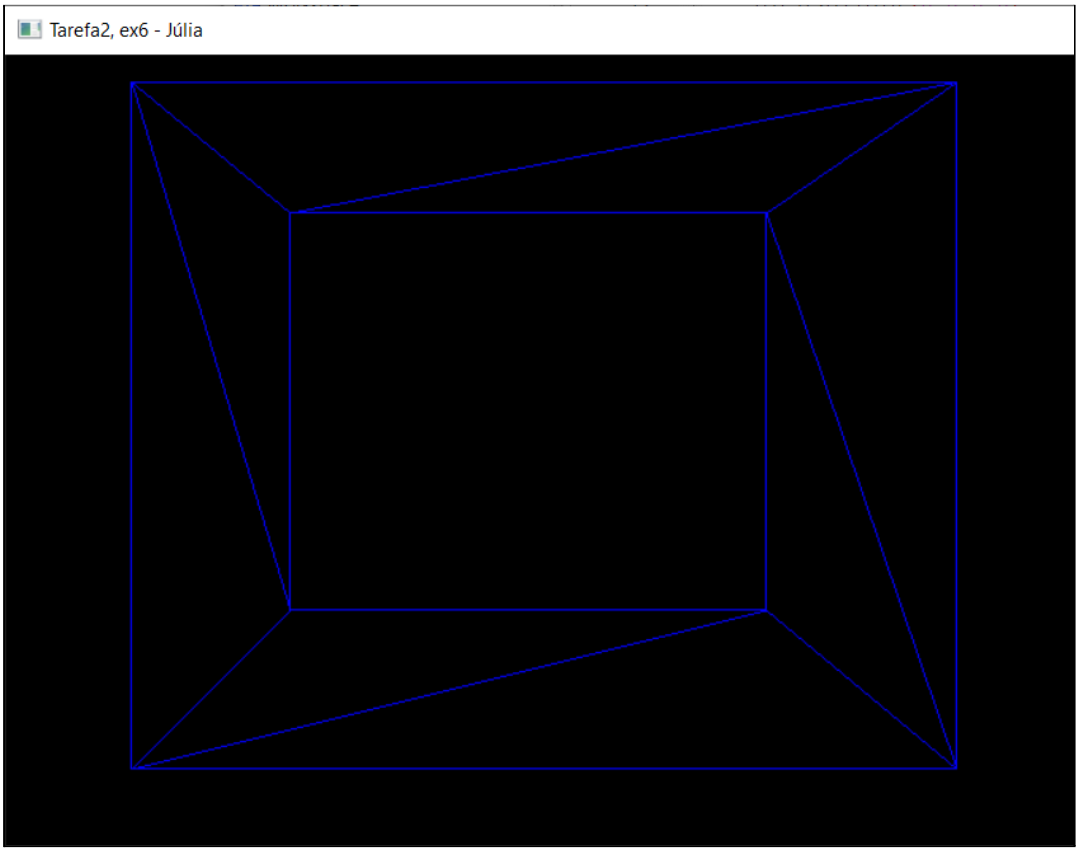
```
void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    //red, green, blue
    glLineWidth(1);
    //glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
    glBegin(GL_TRIANGLE_STRIP);

        glColor3f (0, 1, 1);
        glVertex3f(20,180,0);
        glColor3f (1, 0, 0);
        glVertex3f(20,20,0);
        glColor3f (0.3, 0.3, 0.3);
        glVertex3f(45,60,0);
        glColor3f (1, 0.3, 0);
        glVertex3f(20,193,0);
        glColor3f (0, 0.4, 1);
        glVertex3f(45,160,0);
        glColor3f (0, 1, 1);
        glVertex3f(150,193,0);
        glColor3f (1, 0, 1);
        glVertex3f(120,160,0);
        glColor3f (0, 0, 1);
        glVertex3f(150,20,0);
        glColor3f (1, 1, 1);
        glVertex3f(120,60,0);
        glColor3f (1, 0, 0);
        glVertex3f(20,20,0);
        glColor3f (0, 1, 0);
        glVertex3f(45,60,0);
        glColor3f (0, 0, 1);
    glEnd();
    glFlush();

}
```

Figura 16. Código da figura colorida.

```
void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    //red, green, blue
    glLineWidth(1);
    glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
    glBegin(GL_TRIANGLE_STRIP);
        glColor3f (0, 0, 1);
        glVertex3f(20,193,0);
        glVertex3f(20,20,0);
        glVertex3f(45,60,0);
        glVertex3f(20,193,0);
        glVertex3f(45,160,0);
        glVertex3f(150,193,0);
        glVertex3f(120,160,0);
        glVertex3f(150,20,0);
        glVertex3f(120,60,0);
        glVertex3f(20,20,0);
        glVertex3f(45,60,0);
    glEnd();
    glFlush();

}
```

Figura 17. Código da figura só com linhas.

Figura 18. Saída da figura com linhas.



Nesse exercício eu iniciei a forma da figura por glBegin(GL_TRIANGLE_STRIP) e setei os vértices pelo método glVertex3f(x,y,z). Essa forma cria triângulos entre os pontos definidos.

Para gerar a imagem só com as linhas é necessário usar apenas um glColor3f (r,g,b) da cor escolhida e a função glPolygonMode(GL_FRONT_AND_BACK,GL_LINE). Enquanto para gerar a imagem colorida você define uma cor em cada vértice utilizando diversos glColor3f (r,g,b) e não usa o método glPolygonMode.

A parte mais difícil desse exercício foi definir os pontos no plano cartesiano para então passar para o código. Foi necessário me planejar antes.

Exercício 7 - Nesse exercício foi pedido a construção do gráfico de duas funções:

a) **sen(x), no intervalo -10 a 10.**

Nesse exercício houve a necessidade de alterar o método gluOrtho2D na função de inicializa() vista no começo do relatório. Os parâmetros do método agora são x inicial=-10, x final=10, y inicial=-10 e y=10.

Figura 19. Código do ex7.

```
void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);

    //eixos
    glColor3f(0,0,1.0);
    glLineWidth(3);
    glBegin(GL_LINES);
    glVertex2f(100,0);
    glVertex2f(-100,0);
    glVertex2f(0,100);
    glVertex2f(0,-100);
    glEnd();
    glFlush();

    //grafico sen
    int numPts=200;
    float x = -10;
    float incr=0.1;

    glColor3f(1.0,1.0,1.0);
    glBegin(GL_LINE_STRIP);

    for(int i=1;i<=numPts;i++){
        float y=sin(x);
        glVertex2f(x,y);
        x+=incr;
        }
    glEnd();
    glFlush();
}
```

No começo do desenvolvimento, chamei a função glBegin recebendo a forma de linha simples como parâmetro para representar os eixos x e y e setei os vértices.

E em seguida foi a parte mais complicada, de construir as curvas do gráfico da função do seno. Num primeiro momento estava em dúvida se usava linhas em loop ou em strip, mas a loop sempre conecta seu ponto final com o inicial, o que não daria certo pra esse exercício. Então usei a função GL_LINE_STRIP.

Também iniciei algumas variáveis como o “numPts”, responsável por quantas vezes o laço vai rodar, ou seja, define até onde vai a representação do gráfico do seno e outras como a variável x com valor -10 pois é o valor

de onde o gráfico está limitado, ou seja, x recebe seu valor inicial. E também, uma variável incremento que é somada com o x dentro do laço que serve para definir quantos pontos (x,y) terei. Essa variável incremento deve ser pequena para serem criados mais pontos e assim a linha se tornar uma curva.
Dentro do for eu crio a função y = sin(x) atráves da biblioteca #include <cmath> chamada no começo do código.

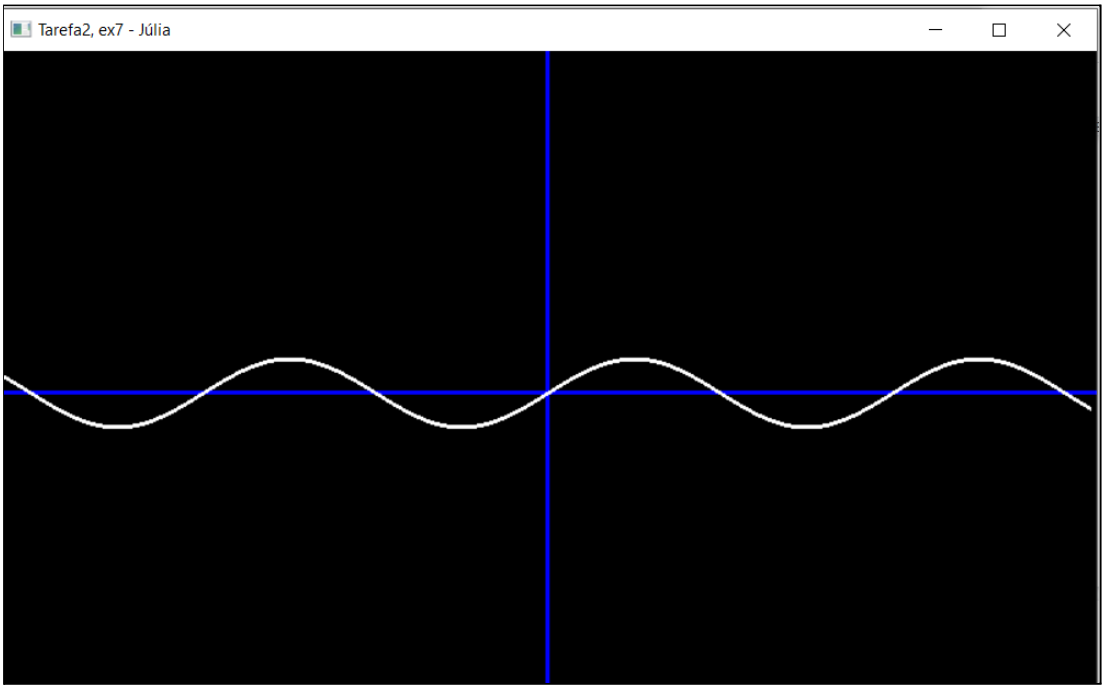


Figura 20. Saída do ex7 - a

b) Fazer o gráfico de $3x^2 + 5x - 6$, no intervalo -10 a 10.

Figura 21. Saída do ex7-b

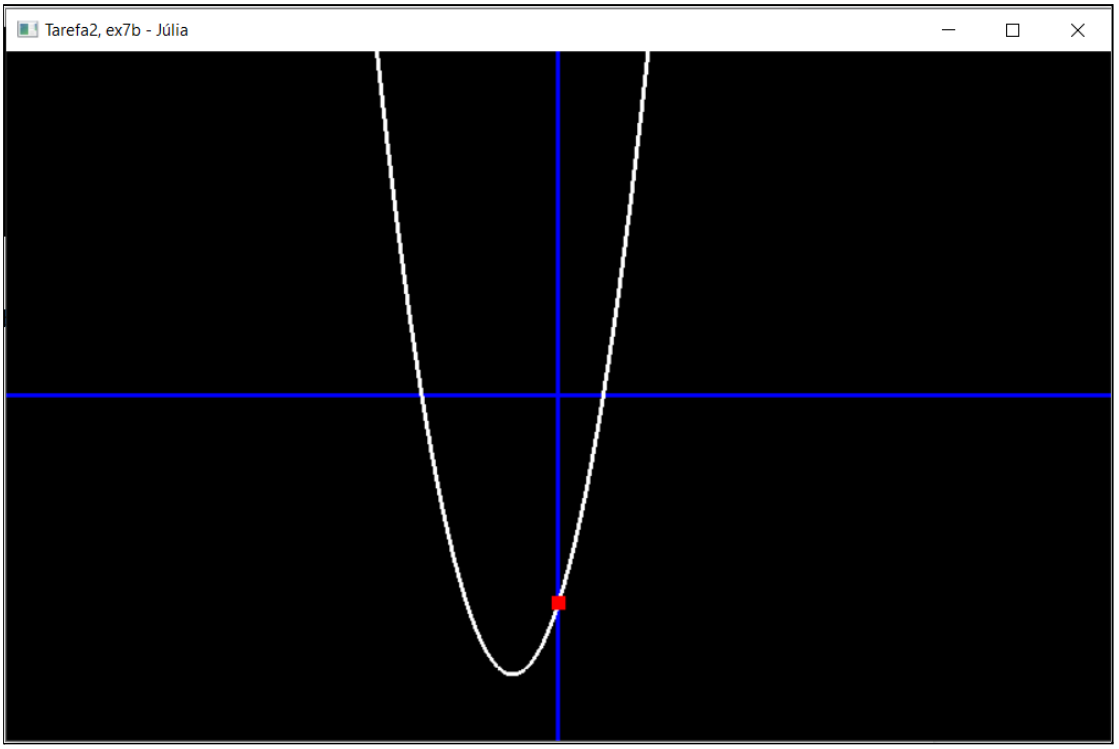


Figura 22. Código ex7-b

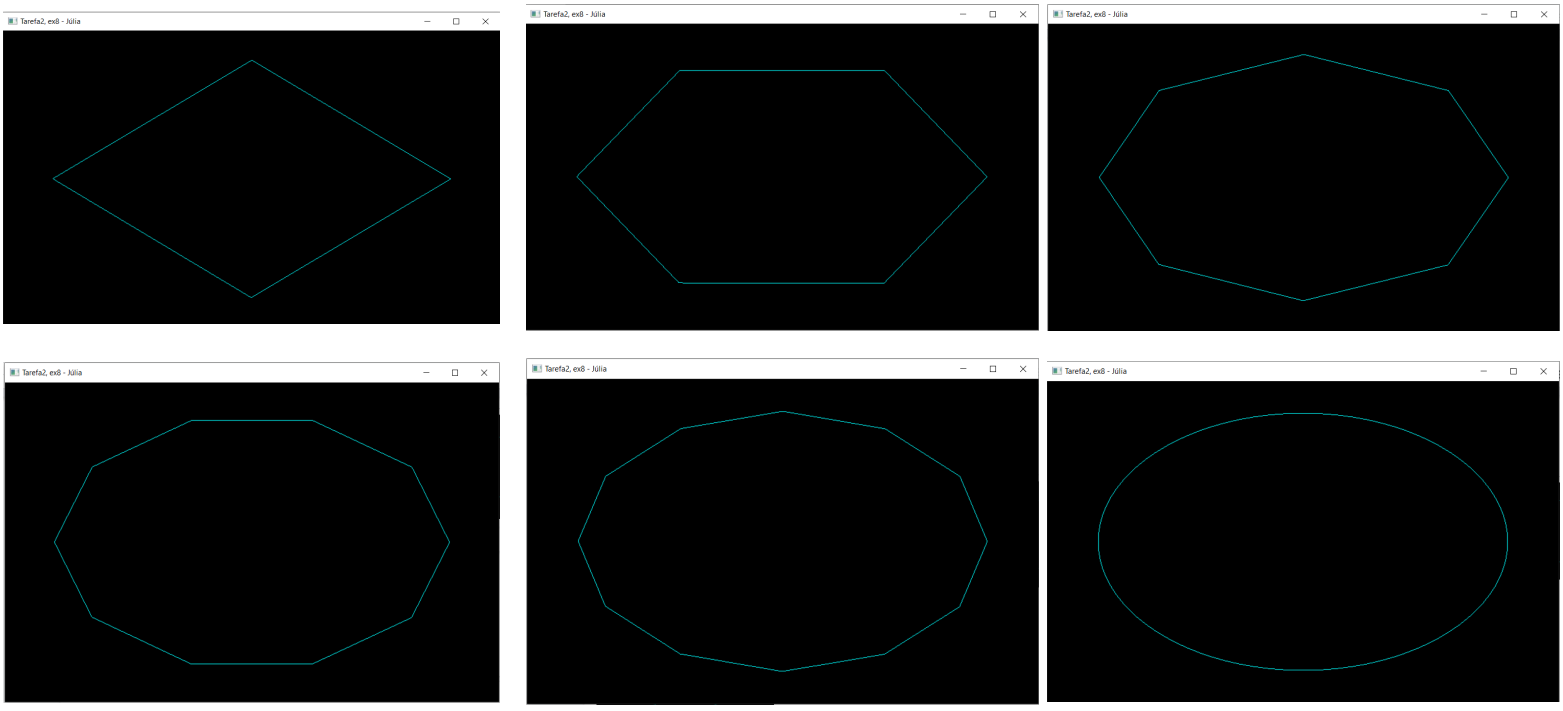
```
//grafico letra b
int numPts=200;
float x = -10;
float incr=0.1;

glColor3f(1.0,1.0,1.0);
glBegin(GL_LINE_STRIP);

for(int i=1;i<=numPts;i++){
    float y=((3*pow(x,2))+(5*x)-6);
    glVertex2f(x,y);
    x+=incr;
    }
glEnd();
glFlush();

glPointSize(10);
glBegin(GL_POINTS);
glColor3f(1, 0, 0);
glVertex2f(0,-6);
glEnd();
glFlush();
}
```

Deste código em relação ao anterior só foi necessário alterar a função dentro do for, agora y recebe (3*pow(x,2))+(5*x)-6 que no caso, é a função que foi pedida.

Também sabemos que em função de segundo grau o ponto que intercepta o eixo y é (0,c) e nessa equação C=-6, então criamos um ponto em vermelho com essa localização de vértice para representar no gráfico.

Exercício 7 - Nesse exercício foi pedido a aproximação de objetos curvos.

Figura 23. Saída do ex7, saídas ao pressionar a tecla 'j'.



```
int lados = 4;

void desenha(void){
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);
    float incr = 360/lados;
    float angulo = 0;
    int raio = 40;
    int centro[2] = {0,0};
    glColor3f(0,1,1);
    glBegin(GL_LINE_LOOP);

    for (int i=0;i<lados;i++){
        float radianos = 3.14 * angulo/180;
        float coordX = centro[0]+raio*cos(radianos);
        float coordY = centro[1]+raio*sin(radianos);
        glVertex2f(coordX,coordY);
        angulo+=incr;
        }
    glEnd();
    glFlush();
}

void tecla(unsigned char key, int x, int y){
    if(key=='j') lados=lados+1;
    if(key=='a') lados=lados-1;
    if (key == 'q') exit(0);
    desenha();
}
```

Figura 24. Código do ex7.

Nesse programa, foi necessário criar uma variável global chamada 'lados' pois duas funções farão uso desta variável, sendo a função desenha() e a função tecla.

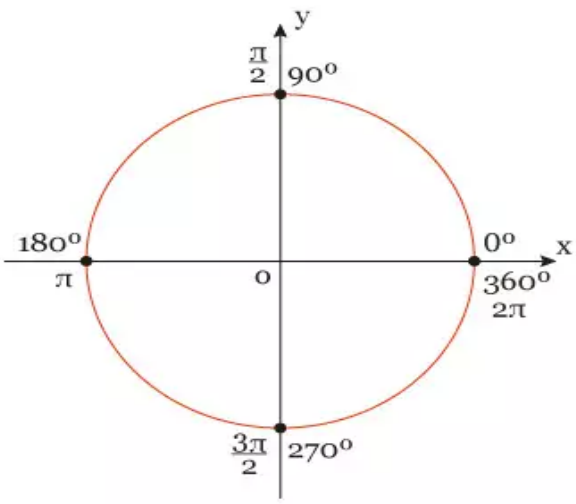
Na função tecla, eu acrescentei dois métodos, o primeiro método é o de aumentar os lados se você pressionar a tecla 'j', assim você vai colocando mais pontos na figura e aos poucos ela vai se tornando redonda. E o outro método, se caso você pressionar a letra 'a', você estará retirando os lados da figura e diminuindo-a.

Na função desenha, responsável por todo o cálculo dos pontos e das formas a serem desenhadas criei algumas variáveis como:

incr: responsável pelo cálculo dos meus futuros pontos. Como para montar uma circunferência é necessário infinitos pontos perto uns dos outros, meu valor de incrementação deverá ser muito baixo também, para que fiquem próximos uns dos outros dando a ideia de uma circunferência.

ângulo: são os ângulos dos pontos da minha figura. Iniciei com zero seguindo o plano cartesiano. Como mostra a Figura 25 ao lado.

raio: é a distância do centro da imagem até o ponto da circunferência, no meu caso eu escolhi 40 mas poderia ser qualquer outro valor.



centro: é o começo da minha figura. Eu iniciei ele no centro. Criei como um vetor mas isso funcionaria como localização no plano cartesiano (x,y) = [0,0].

Daí escolhi a cor da minha imagem, pelo método glColor e no caso é uma mistura de verde com vermelho e iniciei a forma da minha imagem, que no caso seria a LINE_LOOP, por ser uma linha que conecta seu ponto final com o ponto inicial, ideal para trabalhar com circunferências.

E por fim, os cálculos dos meus pontos que serão gerados, inicialmente criei um laço que será percorrido na mesma quantidade de lados que existir. Dentro do laço criei uma variável radianos que faz a conversão de ângulos para radianos. E as minhas variáveis de coordenadas x e y que recebem a fórmula de parametrização de uma circunferência.

Meus vértices da linha receberão essas coordenadas x e y, e meu ângulo vai aumentar de acordo com minha incrementação, lembrando que são valores mínimos com objetivo de serem criados múltiplos pontos próximos uns dos outros.

Por fim eu chamo o glEnd() e o glFlush().