OBJETIVO: Aprender e exercitar programação com geometria computacional usando o OpenGL.

QUESTÃO ÚNICA

OpenGL (*Open Graphics Library*) é uma especificação (API) para renderização de imagens tanto em 2D quanto em 3D. A maioria dos sistemas modernos (Linux, MacOS, Windows) já vêm com suporte a OpenGL, normalmente instalado pela placa de vídeo em uso (e.g, AMD, Nvidia). Entretanto, para criar (compilar) um programa usando OpenGL, é recomendado o uso de bibliotecas que facilitam o seu uso. Dentre estas bibliotecas, a mais conhecida é o GLUT (*OpenGL Utility Toolkit*), mais especificamente a sua implementação FreeGLUT.

Para quem usa laptop/PC próprio, será necessário instalar a biblioteca FreeGLUT (já instalado nas máquinas do IComp):

- No Linux: **`sudo apt install freeglut3 freeglut3-dev libglew-dev`**

Teste a instalação usando o Hello World disponível aqui: bit.ly/icomp-glut-hello . Para compilar o Hello World:

- No Linux: **`gcc hello_world.c -lGL -lglut -o hello_world`**

Para iniciar e abrir uma janela usando o GLUT, pode-se usar o seguinte código:

```
glutInit(&argc, argv);
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH );

// Cria uma janela de tamanho “largura” x “altura”
glutInitWindowSize(largura, altura);
glutCreateWindow ("Segmentos Aleatorios");
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, largura, 0.0, altura, -1.0, 1.0);

// Seta a cor do fundo da janela
glClearColor(1.0, 1.0, 1.0, 1.0);

// Seta a função “display” como sendo a função que irá pintar a janela (infinitamente)
glutDisplayFunc(display);
glutMainLoop();
```

Após isso, basta implementar a função “display”, que será executado infinitamente para pintar a janela com os formatos geométricos (linhas, cubos, etc). A interface desta função é a seguinte:

```
void display(void) {
       glClear(GL_COLOR_BUFFER_BIT);

       // Seu código aqui ...

       glFlush();
}
```

Como pode-se ver no código acima, esta função deve começar com o `glClear` e terminar com o `glFlush`.

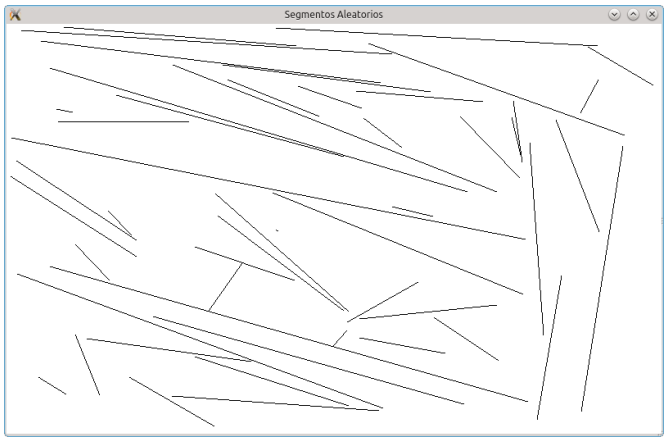
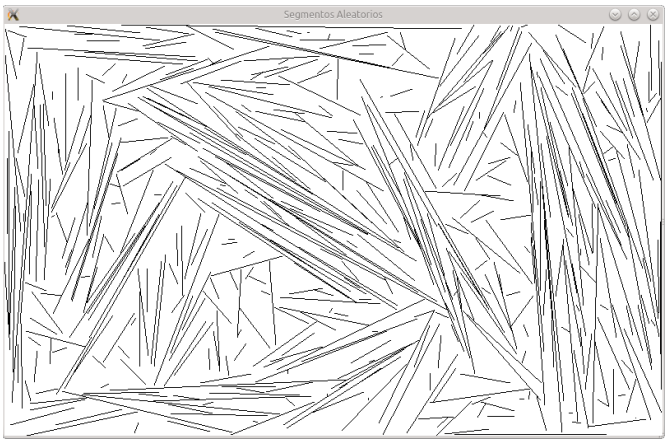
Para este exercício, você precisará apenas gerar e pintar “segmentos” (linhas delimitadas). Para pintar um segmento usando o OpenGL, usa-se os seguintes comandos:

```
       glColor3f(0.0, 0.0, 0.0);  // Seta a cor do seg. (Red, Green, Blue, entre 0.0 e 1.0)
       glBegin(GL_LINES);         // Indica que um segmento será iniciado
       glVertex2f(x1, y1);        // Seta a posição inicial do segmento (inteiros)
       glVertex2f(x2, y2);        // Seta a posição final do segmento (inteiros)
       glEnd();                   // Finaliza a criação do segmento
```

Neste trabalho seu objetivo será ler, da linha de comando, a largura da janela (`argv[1]`), a altura da janela (`argv[2]`) e uma quantidade de segmentos (`argv[3]`). Você deverá gerar a quantidade de segmentos especificada na linha de comando de acordo com as seguintes regras:

- Os pontos inicial e final de cada segmento devem ser gerados aleatoriamente (`rand`);
- Os pontos inicial e final de cada segmento devem estar dentro da janela;
- Não pode haver interseção entre dois segmentos quaisquer;

Exemplo de Execução:

`$ ./segmentos 800 500 50`	`$ ./segmentos 800 500 500`
	

Dicas:

- Para a geração de números aleatórios:
  - Inclua as bibliotecas `time.h` e `stdlib.h`
  - Inicialize a geração com `srand(time(NULL))` (apenas uma vez no seu código).
  - Para gerar um número aleatório: `rand()`

- Para saber se há interseção entre dois segmentos:
  - Interseção entre linhas é muito fácil (basta saber se elas são paralelas ou não).
  - Mas entre segmentos é um pouco mais complicado.
  - Bryce Boe propõe uma solução bem “simples” (a mais simples que encontrei) para o problema (em python):
    - `http://www.bryceboe.com/2006/10/23/line-segment-intersection-algorithm/`

```
def ccw(A,B,C):
      return (C.y-A.y)*(B.x-A.x) > (B.y-A.y)*(C.x-A.x)

def intersect(A,B,C,D):
      return ccw(A,C,D) != ccw(B,C,D) and ccw(A,B,C) != ccw(A,B,D)
```

ENTREGA DO LABORATÓRIO

Envie, até 02/02/2023 às 23:59, o código-fonte para horacio@icomp.ufam.edu.br com o assunto “Entrega do 11o Laboratório de LPA”.