Computação Gráfica e Realidade Virtual

JOGL
Primitivas Gráficas 2D

Prof. Dr. Fernando Kakugawa
fernando.kakugawa@animaeducacao.com.br

Introdução à biblioteca OpenGL

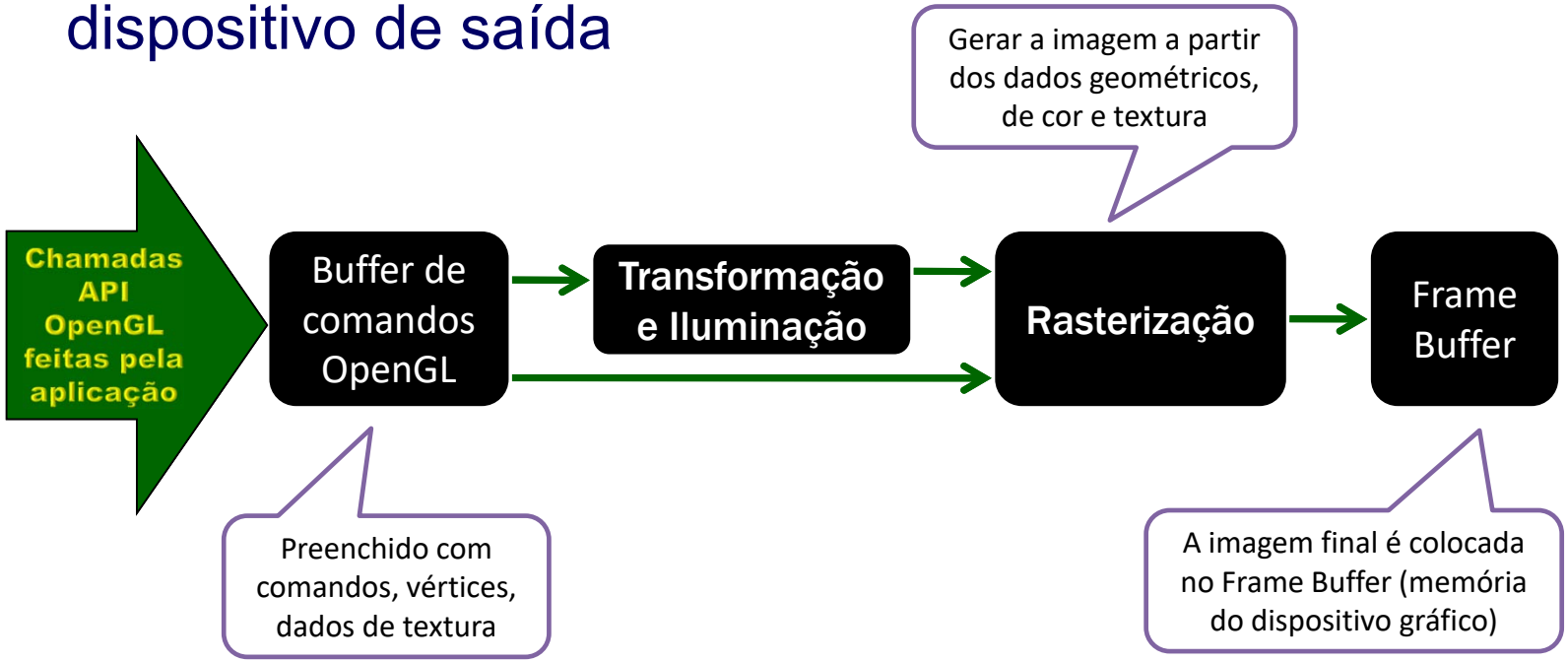
- OpenGL (*Open Graphics Library*):
  - API para programação Gráfica
  - Possui rotinas gráficas e de modelagem 2D/3D extremamente portável e rápida
  - Definida como uma interface para hardware gráfico
  - Produz imagens com alto realismo

Máquina de Estados

- OpenGL é uma máquina de estados que controla um conjunto específico de operações sobre desenhos
- O desenho das primitivas é afetado pelo estado atual das variáveis de controle
- Um conjunto de estados que definem como desenhar
  - Cor, fonte de luz, espessura da linha, do ponto, textura, transformações geométricas
- **Vantagem:**
  - Redução no número de parâmetros necessários para definição do objeto
  - Não é preciso especificar todos os atributos para cada primitiva a ser desenhada

Pepiline de Visualização

- Representa o fluxo de informação gráfica, como é processado da CPU para a apresentação no dispositivo de saída



Bibliotecas

- **GLU** - *OpenGL Utility Library* - Especificar matrizes para projeção
- **GLUT** - *OpenGL Utility Toolkit* - Toolkit independente de plataforma (elementos GUI - Interface Gráfica)
  - **Desenho de caracteres e primitivas 3D**
- Na JOGL existe uma classe/interface para cada biblioteca
  - Pacote **com.jogamp.opengl.GL2:** Interface GL
  - Pacote **com.jogamp.opengl.glu.GLU:** Class GLU
  - Pacote **com.jogamp.opengl.util.gl2.GLUT**: Class GLUT

O que é JOGL?

- É um binding para a API OpenGL desenvolvido em Java
  - Um binding representa uma forma de utilizar funcionalidades de uma biblioteca (nativa ou não) do SO com base numa linguagem diferente da que foi utilizada nessa biblioteca
- Mantém a portabilidade do OpenGL
- Inconsistente com a POO (natureza procedural do OpenGL)
- JOGL faz acesso a API OpenGL via chamadas **JNI** (*Java Native Interface*)

Interface GLDrawable/GLEventListner

- Fornece um mecanismo baseado em eventos (GLEventListner) para executar a renderização OpenGL

- O GLEventListner implementa 4 métodos:
  - **init():** Chamado uma única vez quando o contexto OpenGL é criado
  - **dispose():** Chamado quando o contexto OpenGL é destruído
  - **display():** Chamado para renderizar a imagem
  - **reshape():** Chamado quando a janela é redimensionada

INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE

Instalação e Configuração do Ambiente

- Arquivo com as bibliotecas necessárias
  - `JOGL_Win64.zip`
- Copiar o arquivo para a pasta DOCUMENTOS e **extrair**
  - Segue o site caso precisem fazer o download: `jogamp-all-platforms.7z`

https://jogamp.org/wiki/index.php/Downloading_and_installing_JOGL

Instalação e Configuração do Ambiente

- Organize os arquivos de acordo com o SO:

64-bit Windows

```
gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-windows-amd64.jar
jogl-all-natives-windows-amd64.jar
```

64-bit Linux

```
gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-linux-amd64.jar
jogl-all-natives-linux-amd64.jar
```

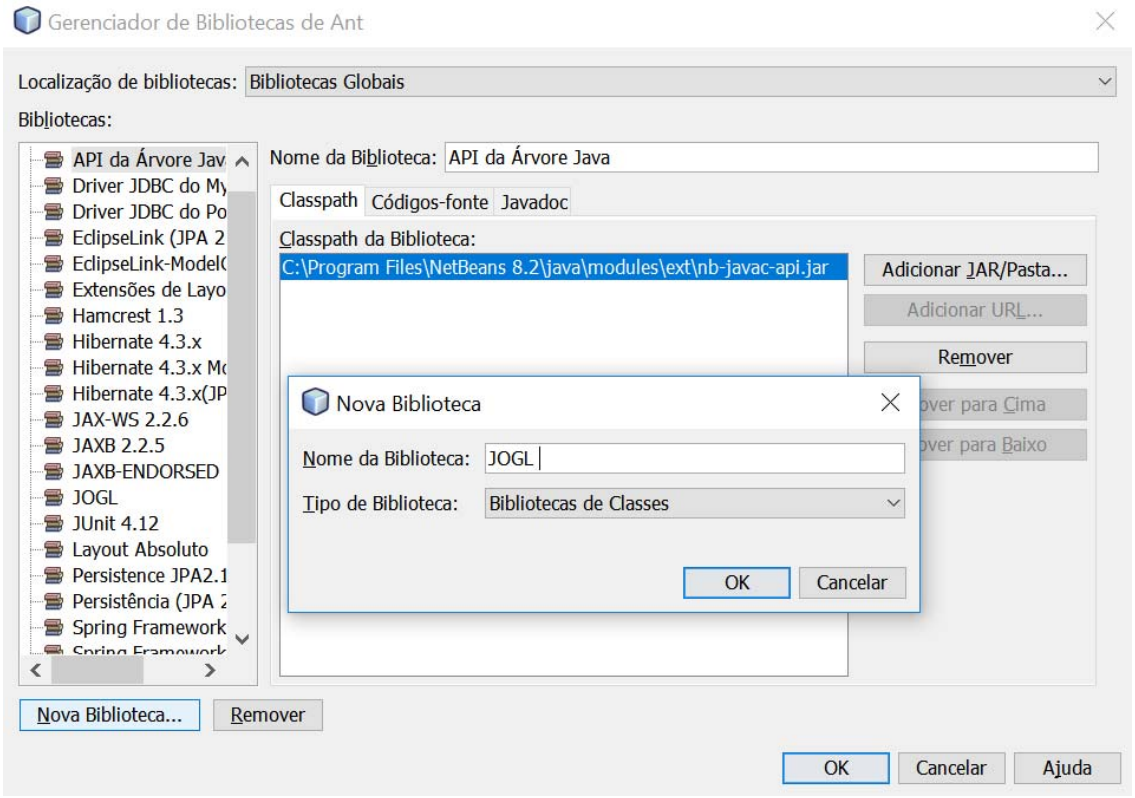
32/64-bit Mac

```
gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-macosx-universal.jar
jogl-all-natives-macosx-universal.jar
```

Configuração JOGL no Netbeans

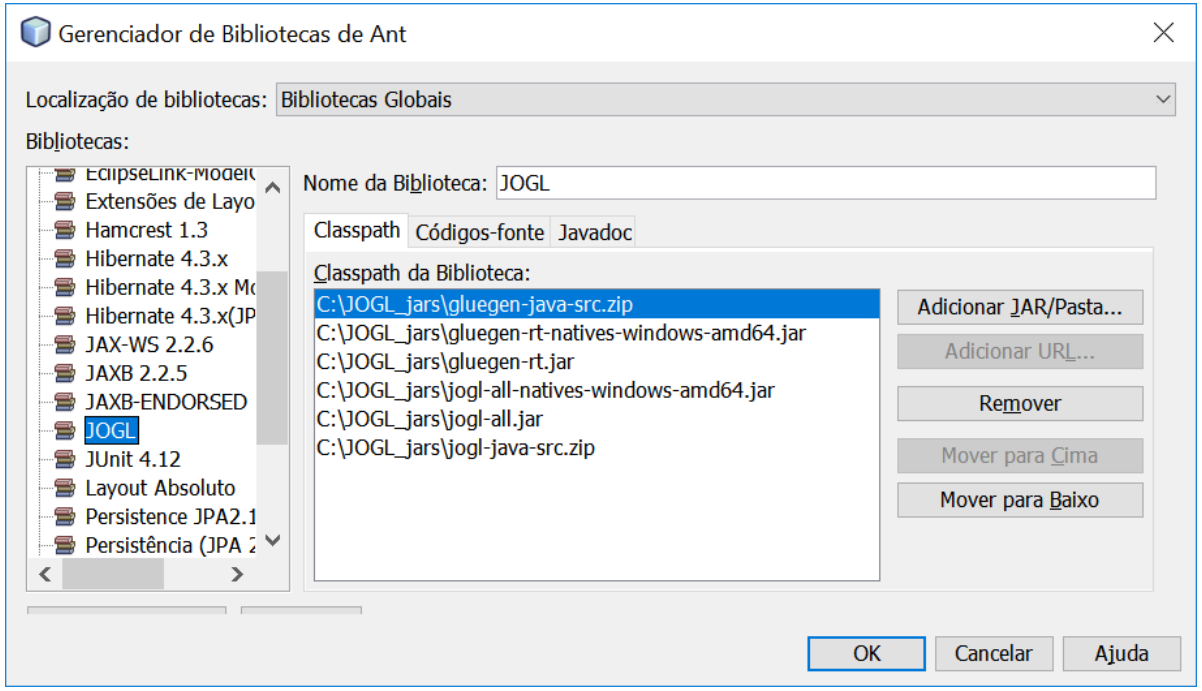
- Abra o Netbeans → Menu Ferramentas → Bibliotecas

Nova Biblioteca...→
Nomear JOGL
OK



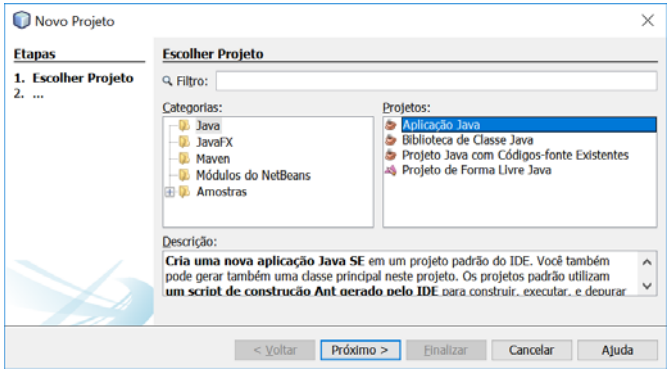
Configuração JOGL no Netbeans

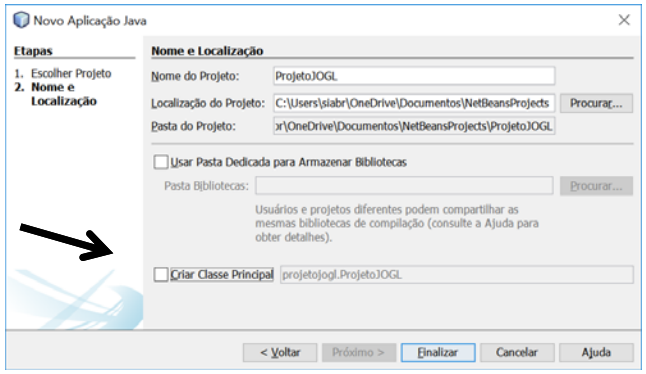
- Selecione JOGL → Botão “Adicionar JAR/Pasta”

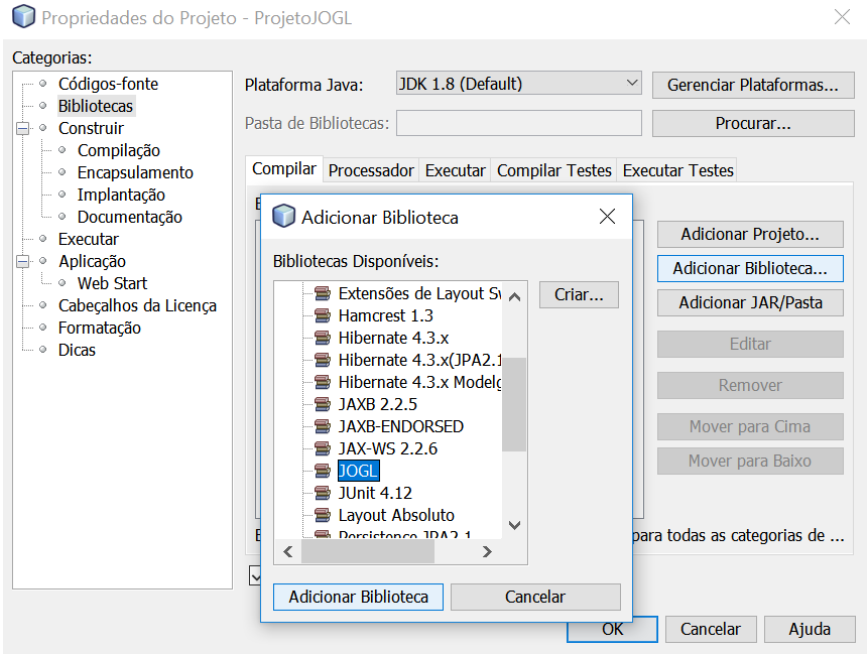


Configuração JOGL no Netbeans

- Menu arquivo Novo projeto -> nomeie como **ProjetoJOGL**
- Desmaque “Criar Classe Principal”
- Botão Direito no ProjetoJOGL -> Propriedades -> Bibliotecas
- Botão “Adicionar Biblioteca” – Escolher JOGL
- Adicione e OK

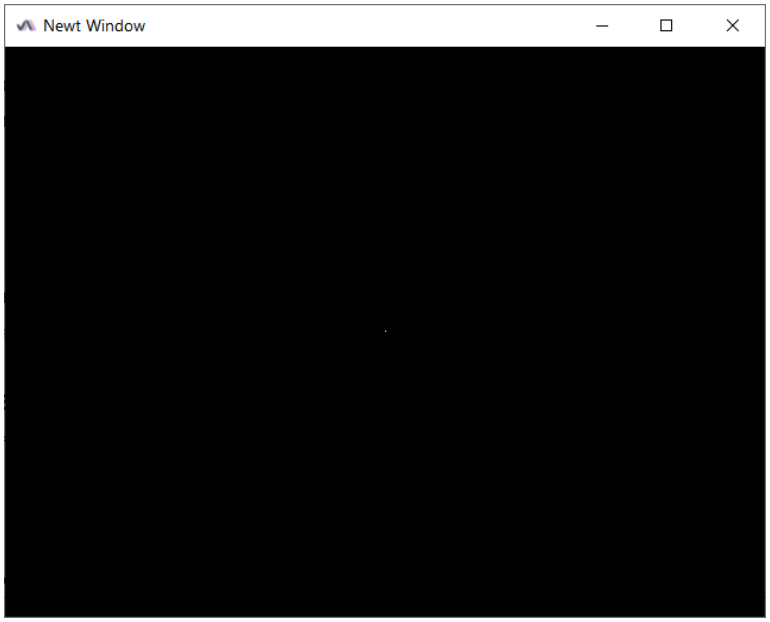






Configuração JOGL no Netbeans

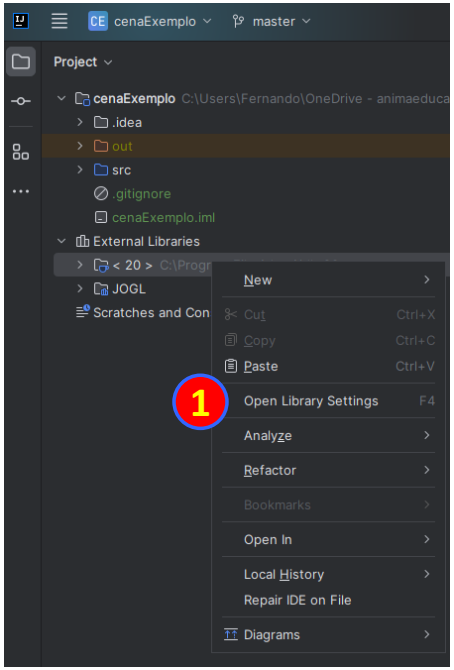
- Copie os arquivos `Renderer.java`, `Cena.java` e `KeyBoard.java` (pasta src) para dentro do projeto ProjetoJOGL
- Menu Executar → Executar projeto
- Deve aparecer uma janela como essa:



Configuração JOGL no IntelliJ

Abra o seu projeto no IntelliJ;

Nas bibliotecas em EXTERNAL LIBRARIES, clique com o direito, apresentará um pop-up com um menu;



1. Nesse menu clique em OPEN LIBRARY SETTINGS

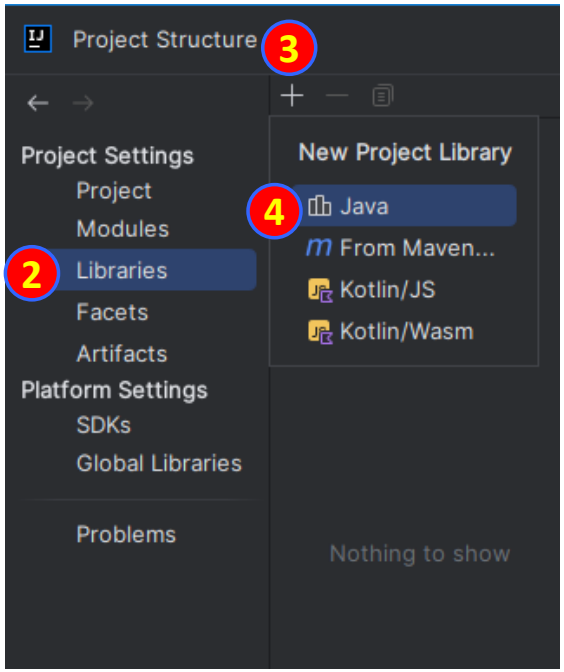
Abrirá uma nova janela;

2. No canto esquerdo clique em LIBRARIES;

3. Na parte superior clique no símbolo +;

Abrirá um pop-up menu

4. Clique em JAVA

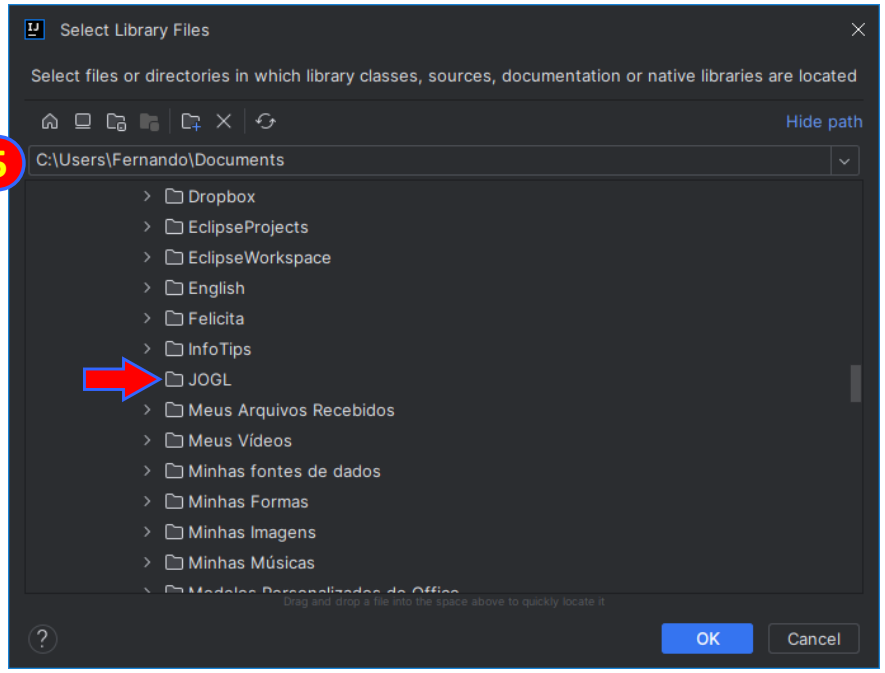


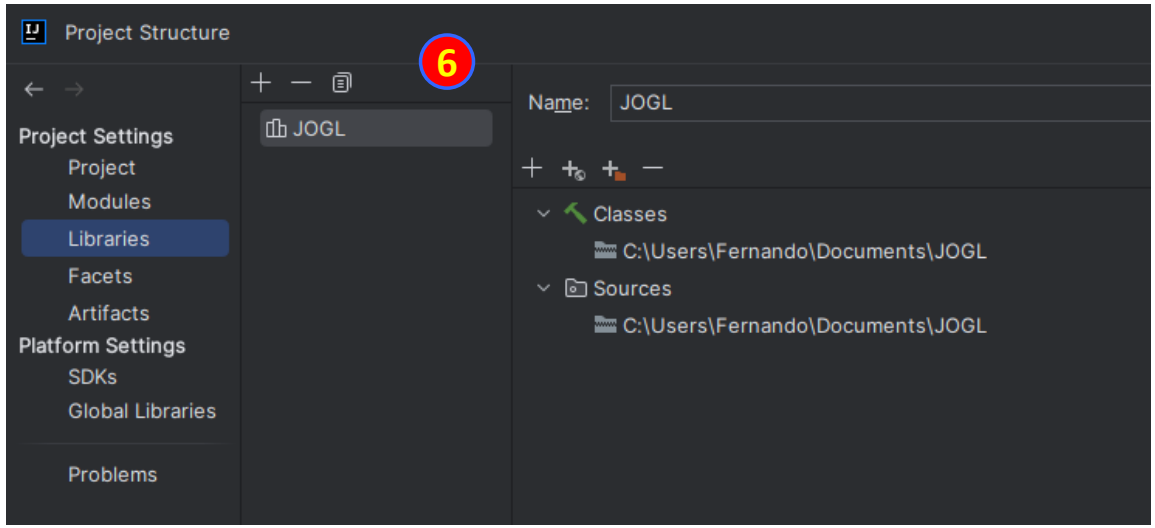
Configuração JOGL no IntelliJ

5. Indique o caminho onde você deixou a pasta JOGL com os arquivos da biblioteca;

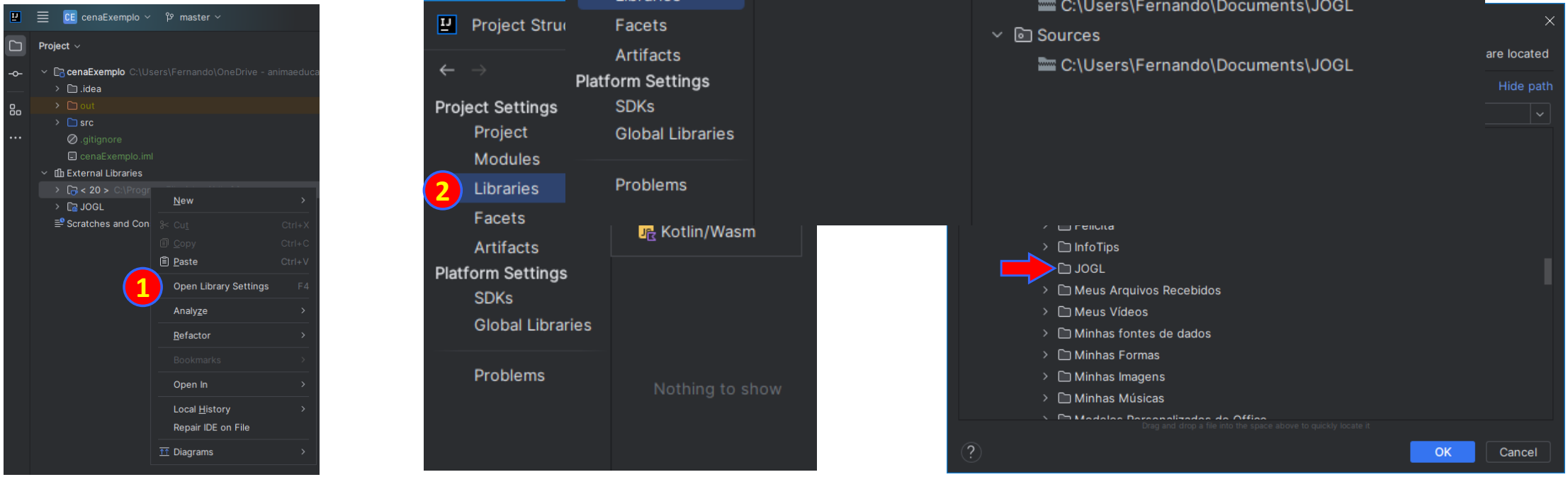
Após indicar o caminho, clique em OK;

6. O JOGL estará configurado na IDE para o seu projeto;





Configuração JOGL no IntelliJ



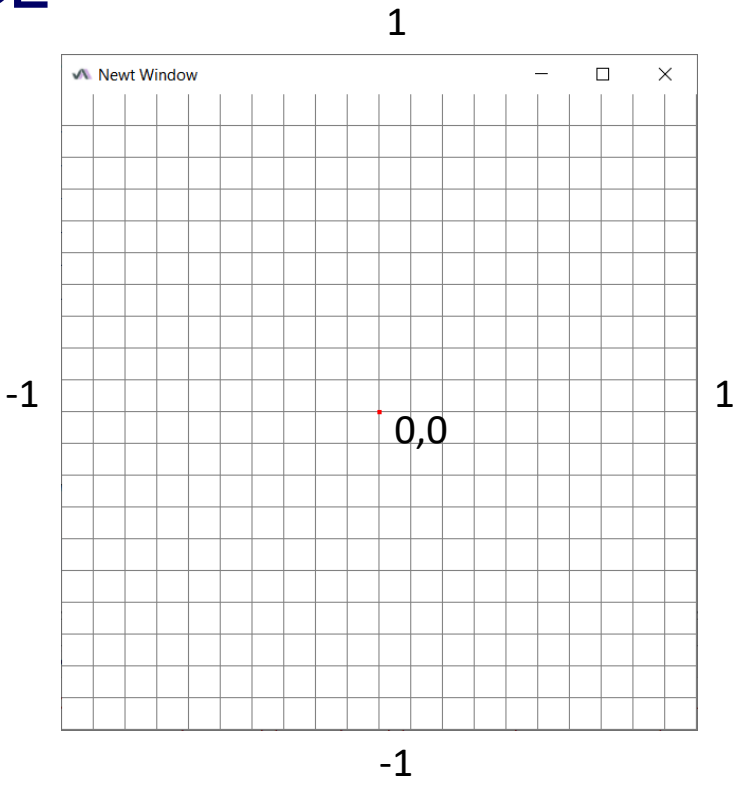
SRU e Primitivas gráficas

Sistema de Referência do Universo - SRU

- Universo
  - Região do espaço utilizada em uma aplicação
- Como a descrição geométrica de um modelo normalmente envolve coordenadas geométricas, é preciso adotar um sistema de referência que irá definir uma origem em relação à qual todos os posicionamentos do universo são descritos
- SRU
  - Consiste em 3 eixos ortogonais entre si (x, y, z) e uma origem (0, 0, 0)

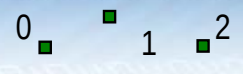
Janela de Visualização

- Entendendo a janela de visualização do OpenGL

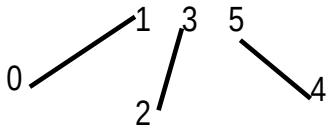


Primitivas Gráficas

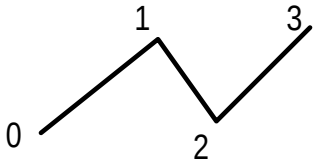
- Tipos de Primitivas OpenGL



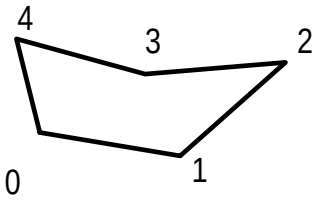
GL_POINTS



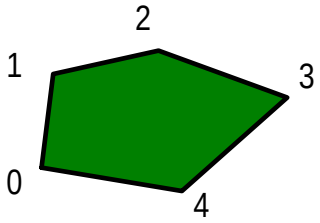
GL_LINES



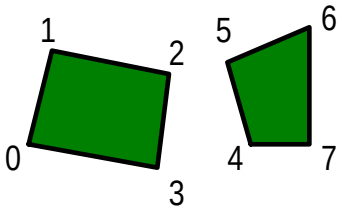
GL_LINE_STRIP



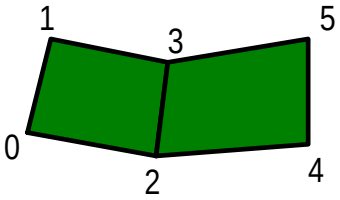
GL_LINE_LOOP



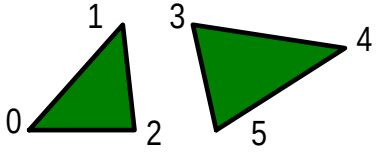
GL_POLYGON



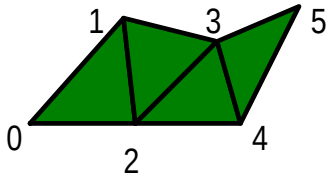
GL_QUADS



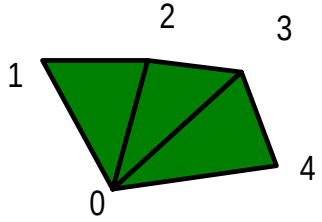
GL_QUAD_STRIP



GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN

Primitivas Gráficas

VALOR	DESCRIÇÃO
GL_POINTS	Para desenhar pontos
GL_LINES	Para desenhar segmentos de linha
GL_LINE_STRIP	Para desenhar segmentos de linha conectados
GL_LINE_LOOP	Para desenhar segmentos de linha conectados, unindo o primeiro ao último
GL_POLYGON	Para desenhar um polígono convexo
GL_TRIANGLES	Para desenhar triângulos
GL_TRIANGLE_STRIP	Para desenhar triângulos conectados
GL_TRIANGLE_FAN	Para desenhar triângulos a partir de um ponto central
GL_QUADS	Para desenhar quadriláteros
GL_QUAD_STRIP	Para desenhar quadriláteros conectados

Primitivas Gráficas

- Como usar uma primitiva 2D:

```
gl.glBegin(GL2.GL_PRIMITIVA);
  gl.glVertex2f( x , y);
  .......
gl.glEnd();
```

- O comando **glVertex** indica a posição de um vértice

Modelo de Cor

- Modelo RGB (Red, Green, Blue)

```
gl.glColor3f( r, g, b);
```

1 – intensidade máxima

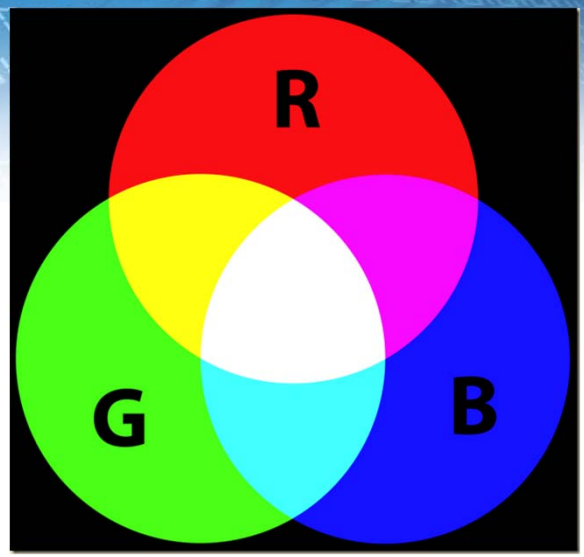
0 - intensidade mínima da cor;

`gl.glColor3f(1, 0, 0)` - representa cor vermelho

`gl.glColor3f(0.5f, 0, 0)` - representa cor vermelho (metade da intensidade)

Modelo de Cor

- **Cores Básicas**
  - Vermelho ➔ 1, 0, 0
  - Verde ➔ 0, 1, 0
  - Azul ➔ 0, 0, 1
  - Branco ➔ 1, 1, 1
  - Preto ➔ 0, 0, 0



- O comando **`glClearColor`** - limpa a janela de visualização e estabelece a nova cor de fundo
- O comando **`glClear`** - pinta o fundo da janela com a cor estabelecida no comando **`glClearColor`**

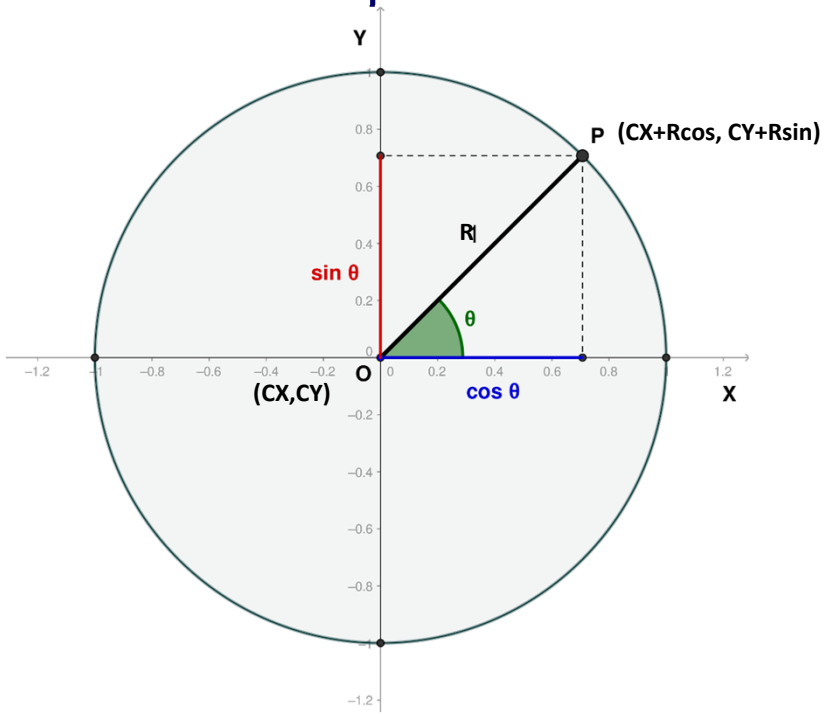
```
gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
gl.glClear( GL2.GL_COLOR_BUFFER_BIT );
```

Exemplo: Desenhando Círculos

- Podem ser desenhados utilizando primitivas gráficas como pontos, linhas e polígonos
- A equação paramétrica do circulo implementado é:

$x = CX+R.\cos(\theta)$,
$y = CY+R.\sin(\theta)$,
$0 <= \theta <= 2\pi$

Onde (CX,CY) é o centro e R é o raio do círculo



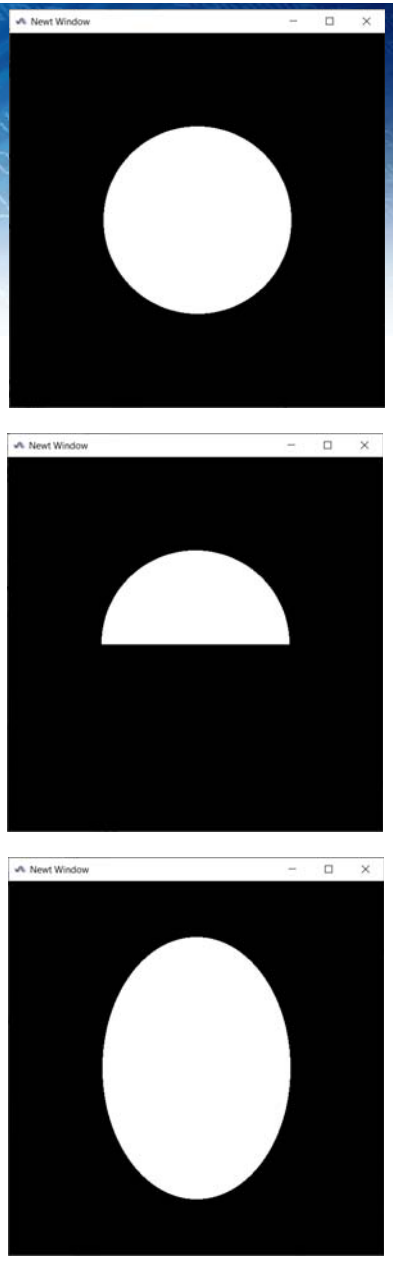
Desenhando Círculos

```
//Para 1/2 circulo: Math.PI
double limite = 2*Math.PI;
double i, cX, cY, rX, rY;

cX = 0;
cY = 0;
//Valores diferentes geram elipses
rX = 0.5f;
rY = 0.5f;

gl.glBegin(GL2.GL_POLYGON);
   for(i=0 ; i < limite; i+= 0.01) {
      gl.glVertex2d(cX + rX * Math.cos(i),
                    cY + rY * Math.sin(i) );

   }
gl.glEnd();
```



KEEP
CALM
AND
VAMOS
PRATICAR

Exercícios

1 – Crie 5 pontos na tela e altere o tamanho do ponto

Pesquisar a função `glPointSize()`

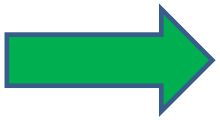
2 – Crie uma linha amarela na tela e altere a sua espessura

Pesquisar a função `glLineWidth()`

Exercício 3

- Implementar as primitivas gráficas vistas em sala de aula. O arquivo deve apresentar as primitivas dispostas como na imagem a seguir.

Usar as primitivas:



GL_POINTS

GL_LINES

GL_LINE_LOOP

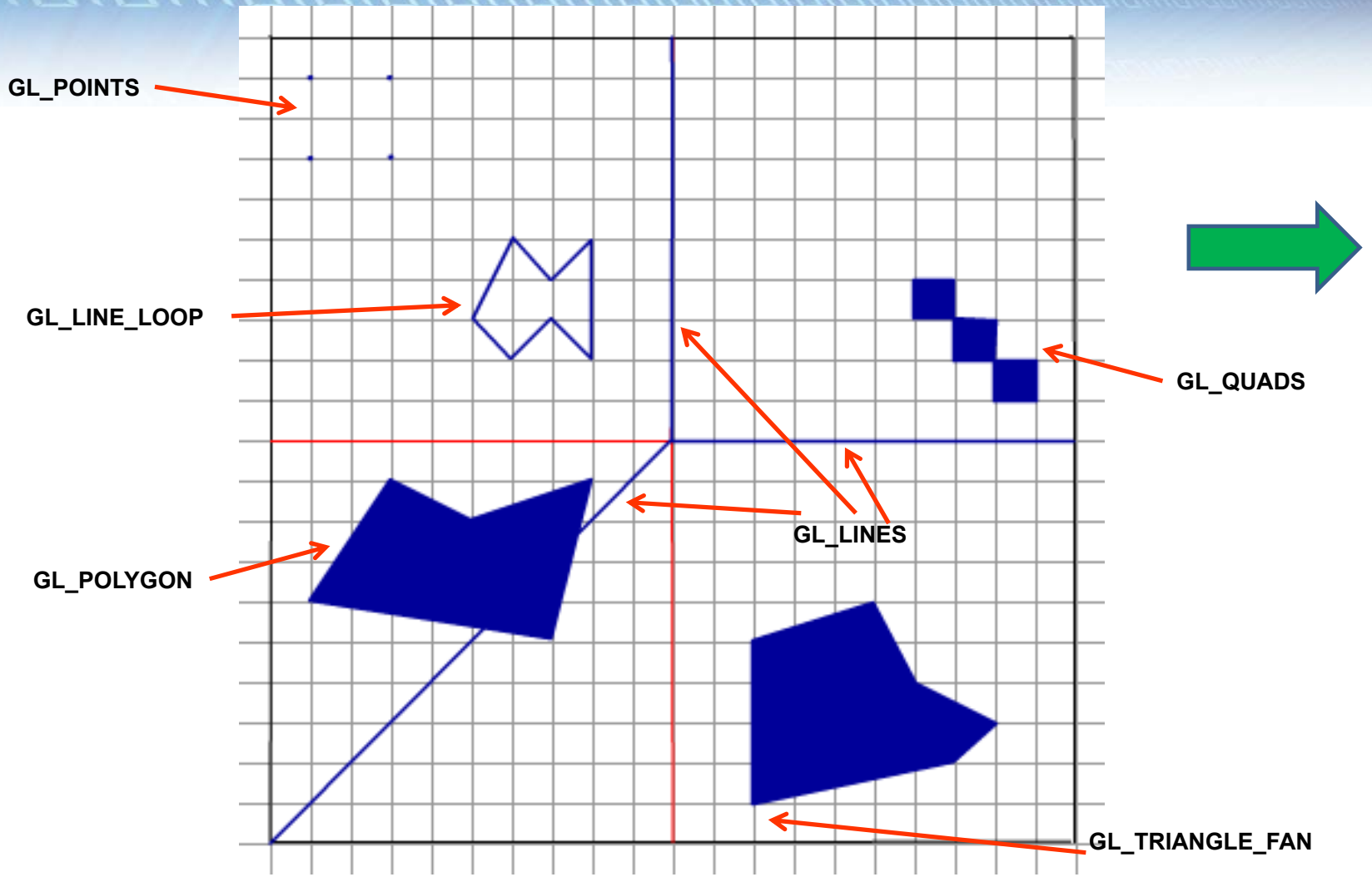
GL_POLYGON

GL_QUADS

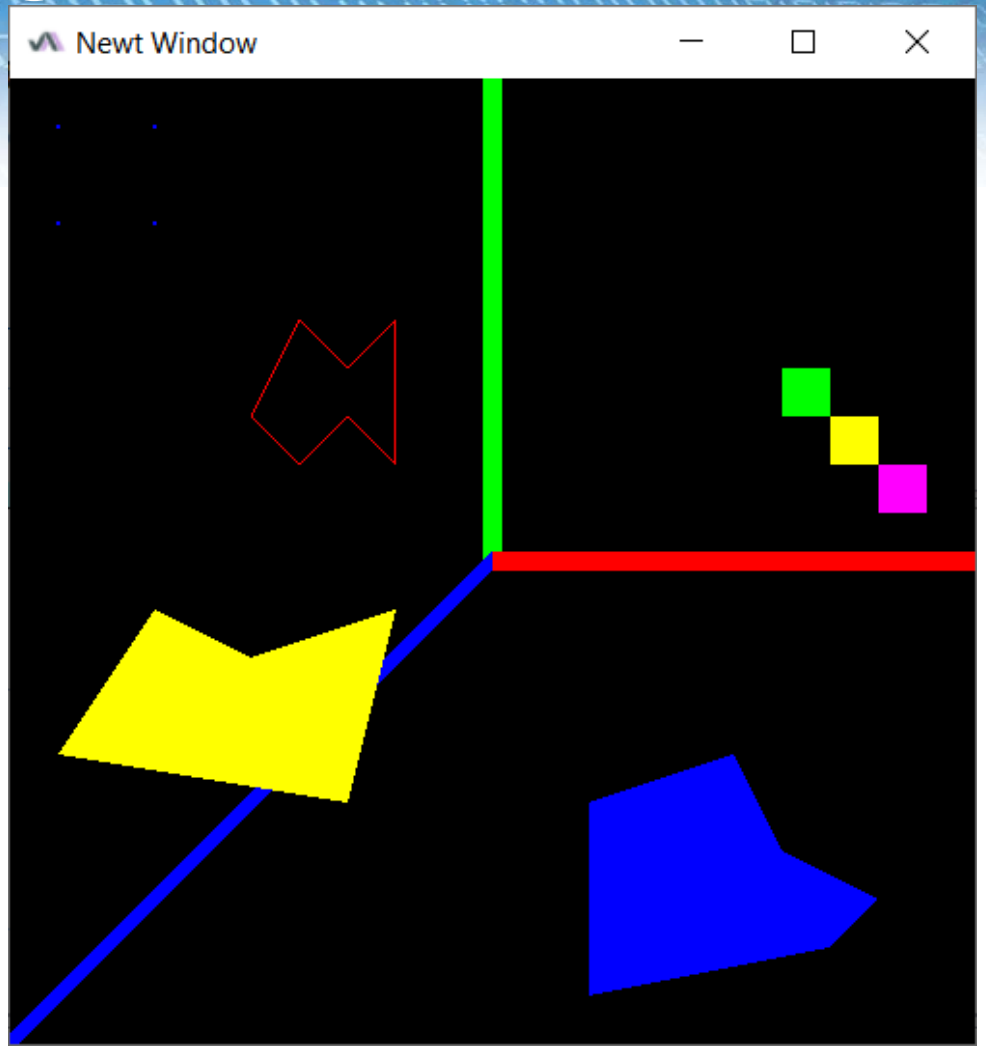
GL_TRIANGLE_FAN

Os pontos de um polígono devem ser no sentido anti-horário!!

Exercício 3 – Guia p/ definir coordenadas



Exercício 3



Material elaborado por:

Prof. Ms. Simone de Abreu

siabreu@gmail.com

Prof. Dr. Fernando Kakugawa

fernando.kakugawa@animaeducacao.com.br