Računarska grafika 2

13M111RG2

3 OpenGL

OpenGL

- OpenGL je
  - API (Applications Programming Interface)
  - softverski interfejs ka grafičkom hardveru
  - sloj između programera i grafičkog hardvera
  - platformski je nezavisan
- Omogućava opis scena na najnižem nivou

OpenGL

Na kursu ćemo se baviti ovim verzijama.

- Istorijat :
  - 1980 – IrisGL (**Silicon Graphics**)
  - 1992, 1997, 1998 – OpenGL verzije 1.0, 1.1, 1.2
  - 2001, 2002, 2003 – OpenGL verzije 1.3, 1.4, 1.5
  - 2004, 2006 – OpenGL verzije 2.0, 2.1
  - 2008 – OpenGL verzija 3.0 (**Khronos Group**)
  - 2009(maj), 2009(avgust) – OpenGL verzije 3.1, 3.2
  - 2010(mart) – OpenGL verzije 3.3, 4.0
  - 2010(jul) – OpenGL verzija 4.1
  - 2011(avgust) – OpenGL verzija 4.2
  - 2012 – OpenGL verzija 4.3
  - 2013 – OpenGL verzija 4.4
  - 2014 – OpenGL verzija 4.5
  - 2017 – OpenGL verzija 4.6

Khronos grupa

Active Standards [edit]

- 3D Commerce, aligns the 3D asset workflow for online retail
- COLLADA, a file-format intended to facilitate interchange of 3D assets
- EGL, an interface between Khronos rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system[4]
- glTF, a file format specification for 3D scenes and models[5]
- NNEF reduces machine learning deployment fragmentation by enabling a rich mix of neural network training tools and inference engines to
- OpenCL, a cross-platform computation API.[7]
- OpenGL, a cross-platform computer graphics API
- OpenGL ES, a derivative of OpenGL for use on mobile and embedded systems, such as cell phones, portable gaming devices, and more
- OpenGL SC, a safety critical profile of OpenGL ES designed to meet the needs of the safety-critical market
- OpenVG, an API for accelerating processing of 2D vector graphics
- OpenVX, Hardware acceleration API for Computer Vision applications and libraries
- OpenXR, an open and royalty-free standard for virtual reality and augmented reality applications and devices
- SPIR, an intermediate compiler target for OpenCL and Vulkan
- SYCL, a single-source C++ DSEL for heterogeneous computing
- Vulkan, a low-overhead computer graphics API
- WebGL, a JavaScript binding to OpenGL ES within a browser on any platform supporting the OpenGL or OpenGL ES graphics standards

Preuzeto sa Wikipedije: https://en.wikipedia.org/wiki/Khronos_Group, 07.11.2019.

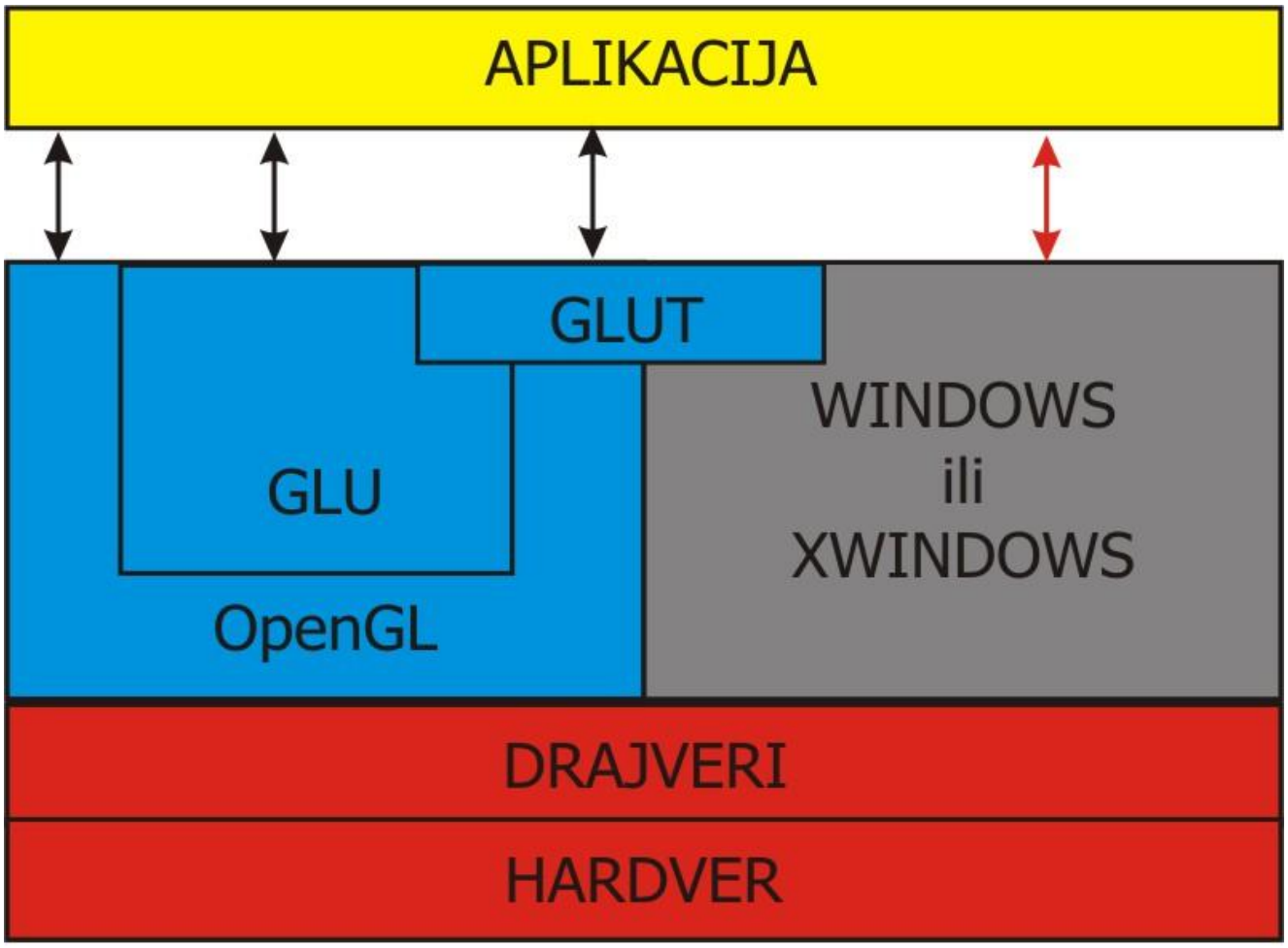
OpenGL prateće biblioteke (C/C++)

- OpenGl – gl.h
- OpenGL Utility Library (GLU) – glu.h
  - Transformacija koordinata
  - Osnovni objekti
  - NURBS
  - Sastavni deo OpenGL distribucije
- GLUT – glut.h – nije deo OpenGL distribucije
  - Rutine za rad sa prozorima (platformski nezavisne)
  - Nije deo OpenGL distribucije
  - Razvoj obustavljen u verziji 3.7 (2000.)
  - Postoje zamene (freeGLUT, GLFW)
- Druge
  - Qt toolkit, FLTK , ...

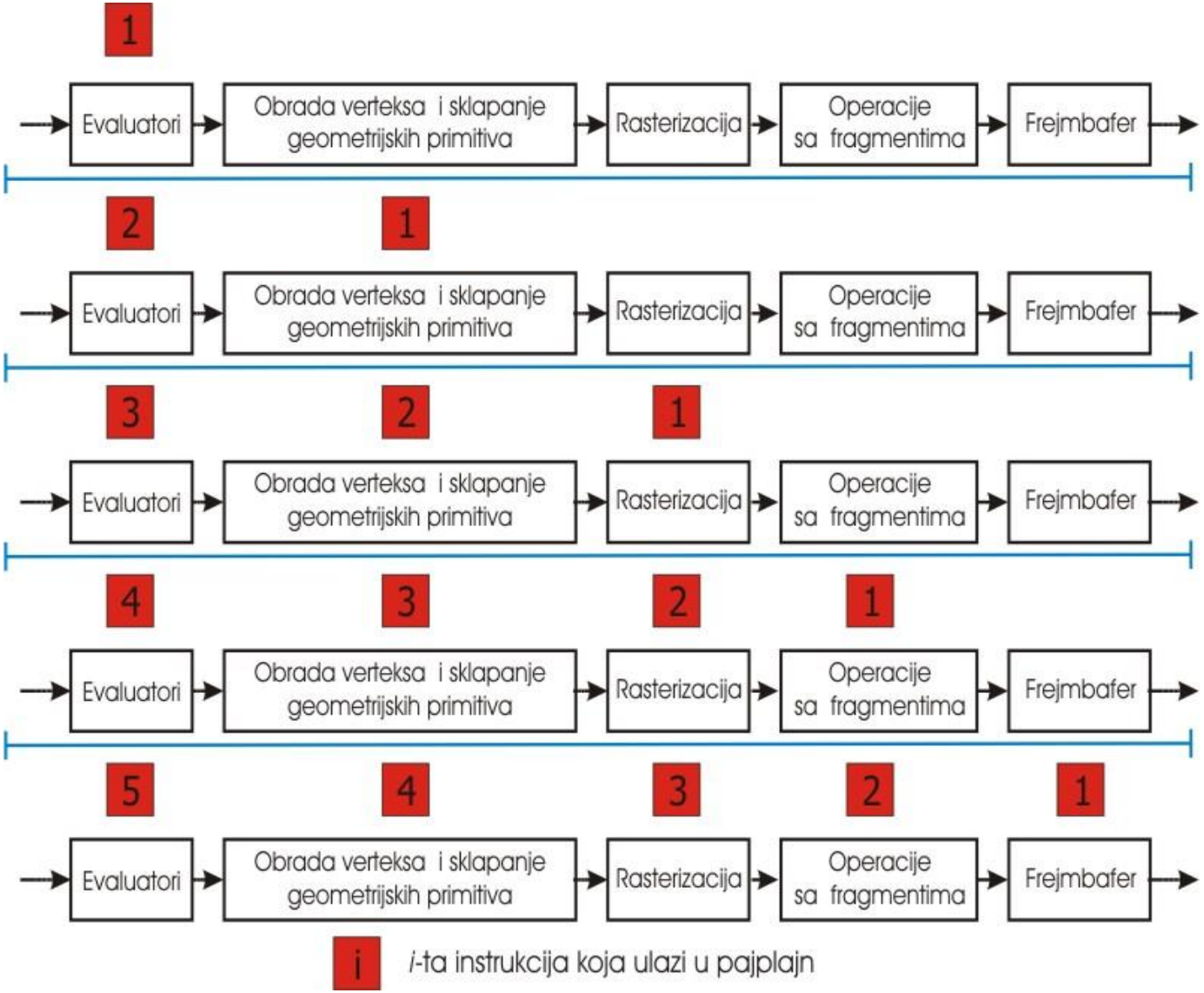
OpenGL prateće biblioteke (Java)

- Dve najpopularnije biblioteke
  - JOGL (http://jogamp.org)
  - LWJGL (https://www.lwjgl.org)
- Podržavaju i OpenCL, OpenAL, ...
- Suštinski koriste JNI i predstavljaju omotač

Struktura OpenGL programa



OpenGL 1.x-2.x – tok podataka



Primitive

- Osnove primitive su
  - tačke
  - linije
  - poligoni
- Sve primitive su opisane pomoću temena :
  - tačka se zadaje jednim temenom
  - linija se opisuje pomoću krajnjih temena (duž)
  - poligon se opisuje nizom temena
- Teme (eng. *vertex*)
  - interna predstava tačke u hardveru
  - npr. dva ili tri broja u pokretnom zarezu koji definišu koordinate
  - sva računanja se obavljaju nad trodimenzionalnim temenima

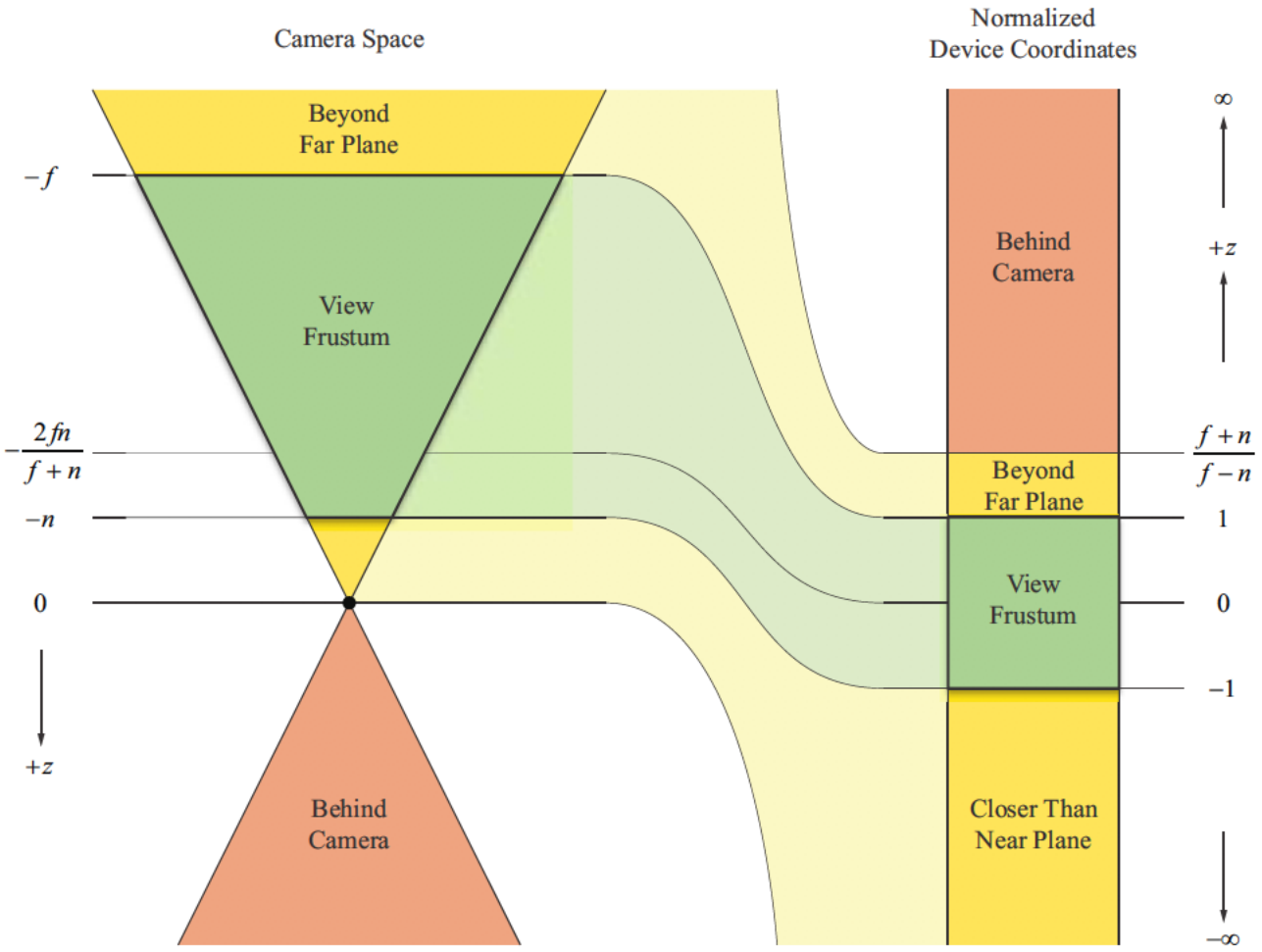
Definisanje primitive

- Sve primitive su opisane pomoću temena
- Teme se zadaje pomoću nekoliko parametara
  - Koordinate (x, y, z, w)
  - Podaci o vektoru normale u temenu (n_x, n_y, n_z)
  - Podaci o teksturi koja se koristi
  - Boja (RGBA ili indeks)
  - Podaci o ivici (edge-flag)
- Svi ovi podaci procesiraju se zajedno
- Detalji o geometrijskim podacima će biti kasnije objašnjeni

Temena (vertices)

- OpenGL radi sa homogenim koordinatama [x, y, z, w]
- Ukoliko w nije navedeno, podrazumeva se w=1.0
  - ukoliko je w različito od 0, homogene koordinate se mapiraju u [x/w, y/w, z/w] 3D koordinate
  - w=0.0 označava zamišljenu tačku u beskonačnosti
- OpenGL “ne ume najbolje da se snađe” kada je w<0.0 (specifikacija dopušta proizvođačima da sami odluče šta da rade)
- Sve tačke se tretiraju kao 3D
- Ukoliko z koordinata nije specificirana, podrazumeva se da je njena vrednost 0.0

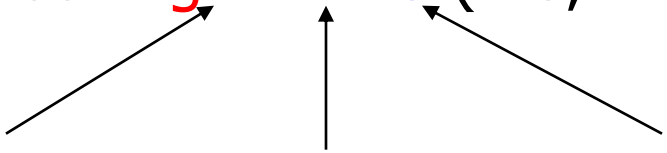
Perspektivna projekcija
- preslikavanje dubine -



OpenGL sintaksa

tip	prefiks	primer
poziv procedure	gl	glEnd()
konstanta	GL_	GL_POINTS
tip podataka	GL	GLfloat

Tipična komanda : glVertex3f(1.0, 2.0, 1.0)



prefiks naziv komande broj i tip argumenata
(tri float argumenta)

OpenGL sintaksa

OpenGL tip	Interna predstava	C tip	Sufiks
GLbyte	8-bitni ceo broj	char	b
GLshort	16-bitni ceo broj	short	s
GLint, GLsizei	32-bitni ceo broj	long	i
GLfloat, GLclampf	32-bitni broj u pok. zarezu	float	f
GLdouble, GLclampd	64-bitni broj u pok. zarezu	double	d
GLubyte, GLboolean	8-bitni pozitivan ceo broj	unsigned char	ub
GLushort	16-bitni pozitivan ceo broj	unsigned short	us
GLuint, GLenum, GLbitfield	32-bitni pozitivan ceo broj	unsigned long unsigned int	ui

OpenGL sintaksa – primer

Specifikacija temena :

- **`void glVertex{2 3 4}{s i f d}[v](TYPEcoords);`**
  - Temena se mogu specificirati sa 2, 3 ili 4 koordinate
  - **`TYPEcoords`** su koordinate temena predstavljene u formatu koji odgovara sufiksima komande

- Primer :

```
glVertex2s(1,0);          <-- [1.0, 0.0, 0.0, 1.0]
glVertex3d(4.0,2.5,0.3);  <-- [4.0, 2.5, 0.3, 1.0]
glVertex4f(1.f,0.8f,0.f,3.f);  <-- [1.0, 0.8, 0.0, 3.0]

GLfloat koord[3]{1.0f, 5.0f, 18.4f};
glVertex3fv(koord);       <-- [1.0, 5.0, 18.4, 1.0]
```

OpenGL – boja temena

Boja se zadaje pomoću jedne od sledećih komandi (pre zadavanja temena):

```
void glColor3{b i f d ub us ui}(TYPE r, TYPE g, TYPE b)
void glColor3{b i f d ub us ui}v(TYPE *color)
void glColor4{b i f d ub us ui}(TYPE r, TYPE g, TYPE b, TYPE a)
void glColor4{b i f d ub us ui}v(TYPE *color)
```

Primer (ekvivalentne naredbe, sve postavljaju crnu boju) :

```
glColor3f(0.0f, 0.0f, 0.0f);
glColor3b(-128, -128, -128);
glColor3ub(0, 0, 0);
unsigned int boja[3] = {0, 0, 0};
glColor3uiv(boja);
```

OpenGL **begin/end** par

- glBegin() / glEnd() – specificira vrstu primitive
  i kraj njenog definisanja
- Između njih se pomoću glVertex*() specificira lista temena
- Temena koja se specificiraju između ovih komandi
  čine jednu primitivu
- Postoji ograničenje u vidu komandi koje moguće upotrebiti
  između glBegin() i glEnd()
- Primer :

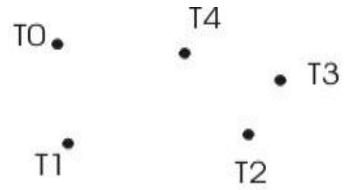
```
glBegin(GL_POLYGON);
        glVertex2f(0.0f, 0.0f);
        glVertex2f(1.0f, 0.0f);
        glVertex2f(0.5f, 0.866f);
glEnd();
```

OpenGL – glBegin()

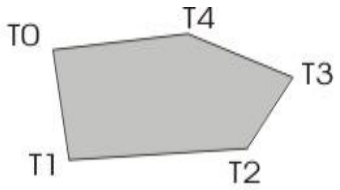
- **`void glBegin(GLenum mode);`**
  - označava početak sekvence komandi za opis primitive
  - **`mode`** određuje tip primitive

vrednost	značenje
GL_POINTS	crta individualne tačke
GL_LINES	parovi tačaka se interpretiraju kao individualne linije
GL_LINE_STRIP	svi segmenti su povezani u liniju
GL_LINE_LOOP	kao i prethodno, samo što je dodat segment između poslednje i prve tačke
GL_TRIANGLES	po tri tačke obrazuju jedan trougao
GL_TRIANGLE_STRIP	kao i prethodno, samo što jedna tačka može biti teme više trouglova
GL_TRIANGLE_FAN	po tri tačke obrazuju trouglove pri čemu nulta tačka pripada svakom trouglu
GL_QUADS	po četiri tačke obrazuju četvorouglove koji se crtaju
GL_QUAD_STRIP	kao i prethodno, samo što jedna tačka može biti u više četvorouglova
GL_POLYGON	sve tačke obrazuju jedan poligon

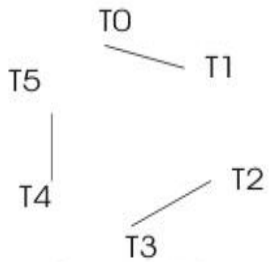
OpenGL – glBegin()



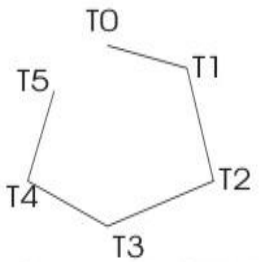
GL_POINTS



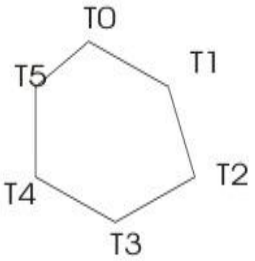
GL_POLYGON



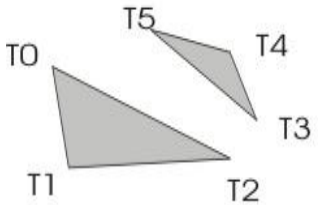
GL_LINES



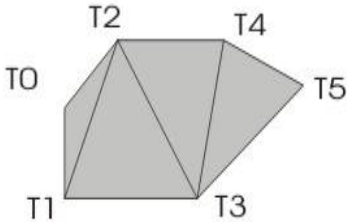
GL_LINE_STRIP



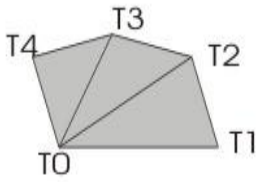
GL_LINE_LOOP



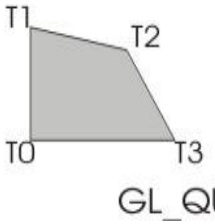
GL_TRIANGLES

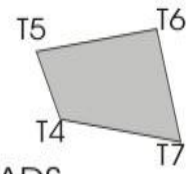


GL_TRIANGLE_STRIP

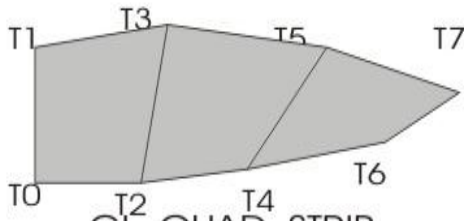


GL_TRIANGLE_FAN





GL_QUADS



GL_QUAD_STRIP

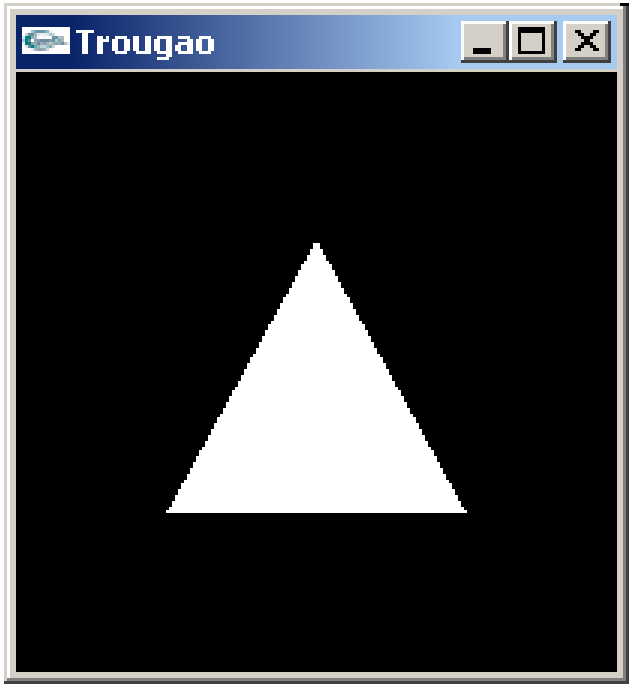
OpenGL – glBegin()/glEnd()

- Komande koje se mogu koristiti unutar glBegin()/glEnd()

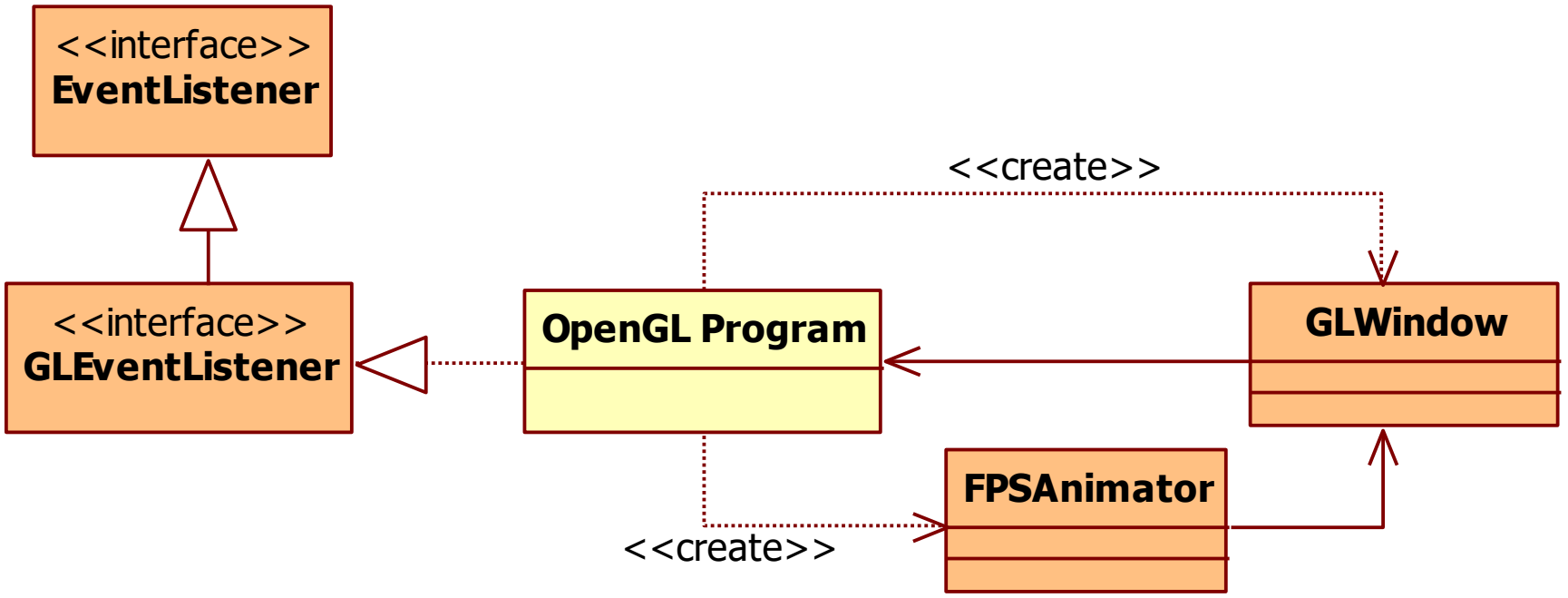
naziv	namena
glVertex*()	zadaje koordinate tačke
glColor*()	postavlja tekuću boju
glIndex*()	postavlja tekući indeks
glNormal*()	postavlja koordinate vektora normale
glTexCoord*()	postavlja koordinate teksture
glEdgeFlag*()	kontroliše crtanje ivica
glMaterial*()	postavlja osobine materijala
glArrayElement()	objedinjeno zadavanje parametara
glEvalCoord*(), glEvalPoint*()	parametarski generiše koordinate temena
glCallList(), glCallLists()	izvršava displej listu (liste)

OpenGL – primer (jezik C)

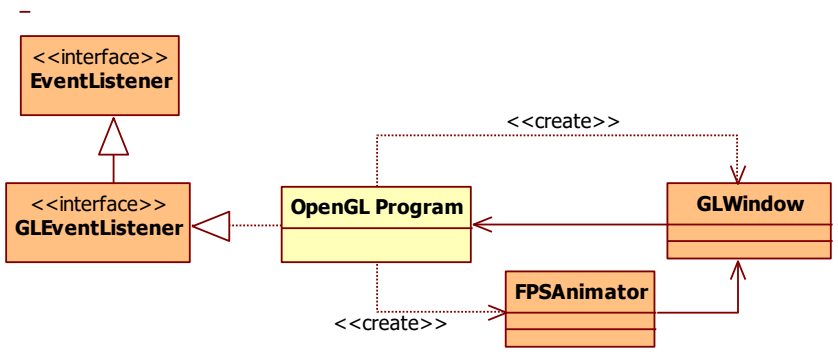
```
main() {
    Inicijalizuj_prozor();
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f (1.0f, 1.0f, 1.0f);

    glBegin(GL_POLYGON);
        glVertex2f(-0.5f, -0.5f);
        glVertex2f(0.5f, -0.5f);
        glVertex2f(0.0f, 0.5f);
    glEnd();

    glFlush();
    Održavaj_prozor_na_ekranu();
}
```



OpenGL – Izgled Java programa (JOGL)

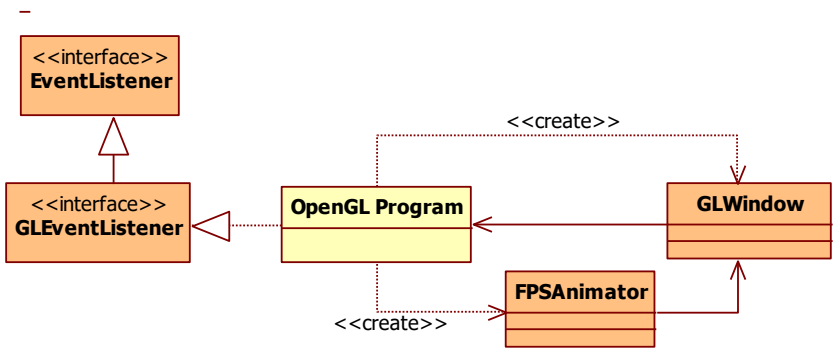


JOGL – jednostavan program




```
package OpenGL_L01;
import javax.media.opengl.GLAutoDrawable;
import javax.media.opengl.GLCapabilities;
import javax.media.opengl.GLEventListener;
import javax.media.opengl.GLProfile;
import com.jogamp.newt.event.WindowAdapter;
import com.jogamp.newt.event.WindowEvent;
import com.jogamp.newt.opengl.GLWindow;
import com.jogamp.opengl.util.FPSAnimator;
import javax.media.opengl.GL;
import javax.media.opengl.GL2;
import javax.media.opengl.glu.GLU;

public class OpenGLBasic implements GLEventListener {  // Renderer
   private static String TITLE = "JOGL 2 - Basic OpenGL demo";  // title
   private static final int WINDOW_WIDTH = 640;  // width of the drawable
   private static final int WINDOW_HEIGHT = 480; // height of the drawable
   private static final int FPS = 60; // animator's target frames per second
```

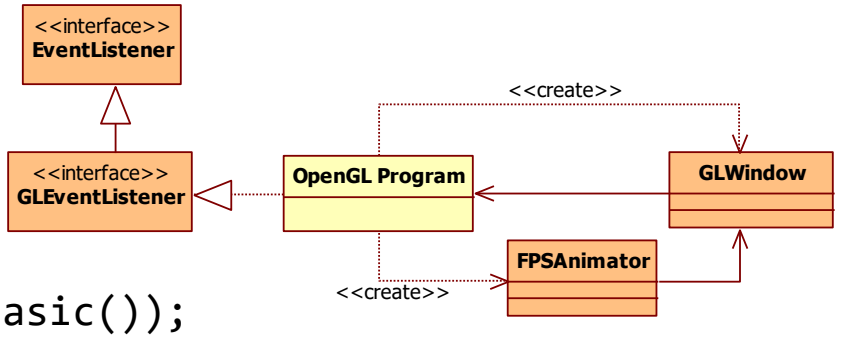
JOGL – jednostavan program




```
/** Constructor */
public OpenGLBasic() {}

/** The entry main() method */
public static void main(String[] args) {
    // Get the default OpenGL profile,
    // reflecting the best for your running platform
    GLProfile glp = GLProfile.getDefault();

    System.out.println(glp.getGLImplBaseClassName());
    System.out.println(glp.getImplName());
    System.out.println(glp.getName());

    // Specifies a set of OpenGL capabilities, based on your profile.
    GLCapabilities caps = new GLCapabilities(glp);

    caps.setAlphaBits(8);
    caps.setDepthBits(24);
    System.out.println(caps);

    // Create the OpenGL rendering canvas
    GLWindow window = GLWindow.create(caps);
```

JOGL – jednostavan program

```
    // Create a animator that drives canvas' display() at the specified FPS.
    final FPSAnimator animator = new FPSAnimator(window, FPS, true);

    window.addWindowListener(new WindowAdapter() {
       public void windowDestroyNotify(WindowEvent arg0) {
          // Use a dedicate thread to run the stop() to ensure that the
          // animator stops before program exits.
          new Thread() {
             public void run() {
                animator.stop(); // stop the animator loop
                System.exit(0);
             }
          }.start();
       };
    });

    window.addGLEventListener(new OpenGLBasic());
    window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    window.setTitle(TITLE);
    window.setVisible(true);
    animator.start();
}
```



JOGL – jednostavan program

```
/** Called back by the drawable to render OpenGL graphics */
@Override
public void display(GLAutoDrawable drawable) {
    render(drawable);
}

/** Called back before the OpenGL context is destroyed. */
@Override
public void dispose(GLAutoDrawable drawable) { }

/** Called back by the drawable when it is first set to visible,
and during the first repaint after the it has been resized. */
@Override
public void reshape(GLAutoDrawable drawable,
                    int x, int y, int weight, int height) { }
```

JOGL – jednostavan program

```
    /** Called back immediately after the OpenGL context is initialized */
    @Override
    public void init(GLAutoDrawable drawable)
    {
        GL2 gl = drawable.getGL().getGL2();
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
        gl.glColor3f(1.0f, 1.0f, 1.0f);
    }

    /** Render the shape (triangle) */
    private void render(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();

        gl.glClear(GL.GL_COLOR_BUFFER_BIT);

        gl.glBegin(GL.GL_TRIANGLES);
         gl.glVertex2f(-0.5f, -0.5f);
         gl.glVertex2f(0.5f, -0.5f);
         gl.glVertex2f(0, 0.5f);
        gl.glEnd();
    }
}
```