

Uvodni čas

- Ciljevi:
- Načini ispitivanja i polaganja vežbi
 - Način izvođenja vežbi
 - Uvod u OpenGL/SharpGL
 - Osnovne grafičke primitive

Načini ispitivanja

- Projektni zadatak (dve kontrolne tačke) - **50 poena**
- Teorijski test 1 (radi se na KT1) - **10 poena**
- Teorijski test 2 (radi se na KT2) - **10 poena**

Pravila polaganja

- Svako radi za sebe (svako prepisivanje, predavanje tuđeg rada će se kažnjavati)
- Uslov za upis ocene bez usmenog:
 $PZ + TT1 + TT2 \geq 51 \wedge TT1 + TT2 > 14$

Literatura

- Skripta za vežbe (biće okačeno na GIM-u)
- OpenGL - Redbook <http://www.openglprogramming.com/red/>
- Dave Schreiner, Graham Sellers, John Kessenich, Bill Licea-Kane, *OpenGL Programming Guide (Eighth Edition)*, Addison Wesley Professional, 2013.

Uvod u OpenGL

- API koji služi za pristupanje funkcijama GPU
- Alternativa IrisGL-u
- Aktuelna verzija: 4.6
- Kreirao Silicon Graphics, razvija Chronos Group
- Automat stanja

Luxo Jr.

- Pixar Animation Studio, 1986.



Uvod u OpenGL

- Client-Server sistem
- Softver napisan u OpenGL:
 - Adobe After Effects, Photoshop, ArtRage
 - 3D Studio Max, Maya, Blender

OpenGL vs DirectX

- OpenGL je open source za razliku od DirectX u čiji izvorni kod ima uvid samo Microsoft
- OpenGL je multiplatformska biblioteka za razliku od DirectX koji se izvršava isključivo na Microsoft platformama

SharpGL

- Objektno-orientisana paradigma
- OpenGL u .NET
- Sadrži sve funkcionalnosti OpenGL-a

OpenGL in C++

```
glLineWidth(3.0f);  
glPointSize(2.0f);
```

SharpGL

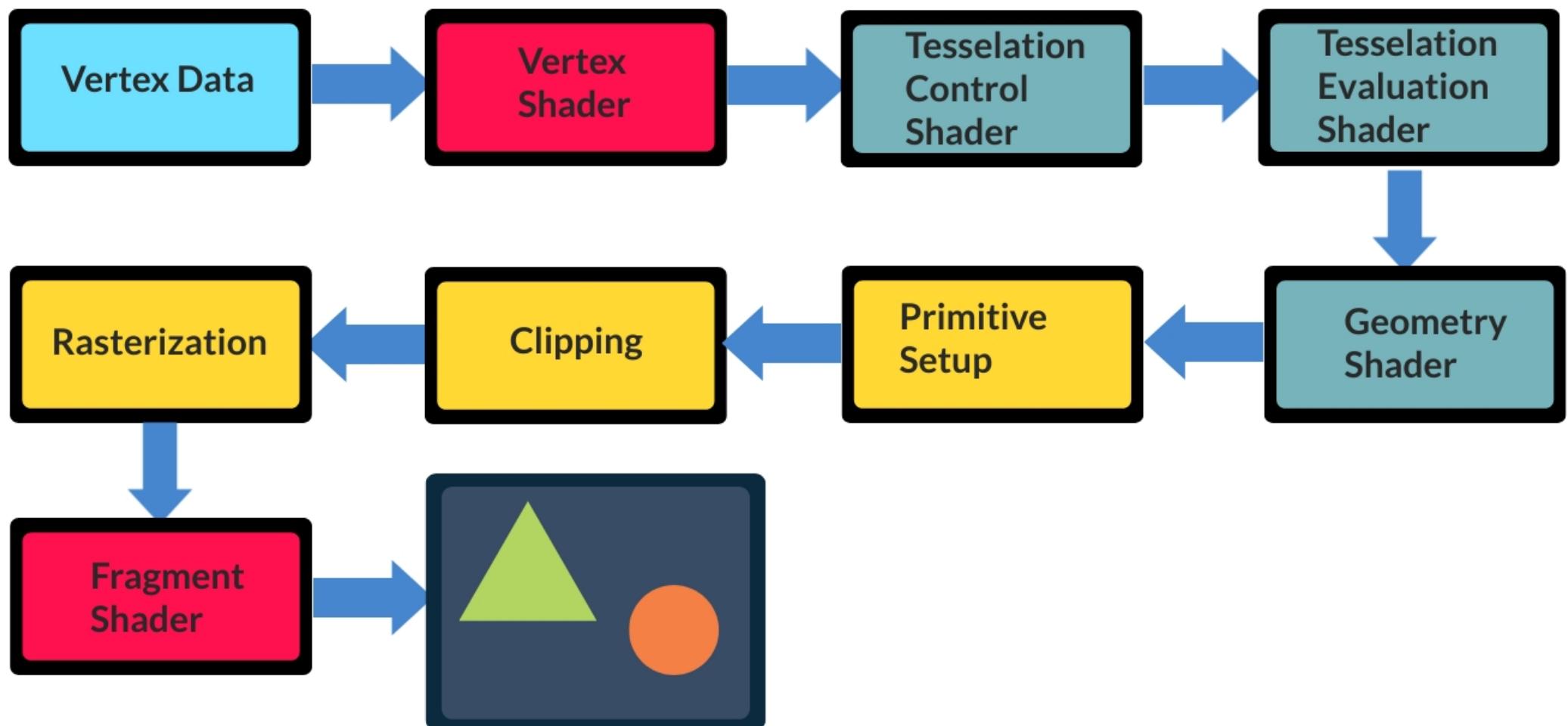
```
OpenGL gl =  
openGLCtrl1.OpenGL;
```

```
gl.LineWidth(3.0f);  
gl.PointSize(2.0f);
```

Rendering Systems

- Rasterization
- Ray-casting
- Ray-tracing

OpenGL Pipeline



Terminologija

- Verteks
- Primitiva/Poligon
- Objekat/Model
- Fragment
- Pixel
- Framebuffer

Definicije

- **void glPointSize(GLfloat size);**

Sets the fixed size, in pixels, that will be used for points when GL_PROGRAM_POINT_SIZE is not enabled.

- **void glLineWidth(GLfloat width);**

Sets the fixed width of lines. The default value is 1.0. width is the new value of line width and must be greater than 0.0, otherwise an error is generated.

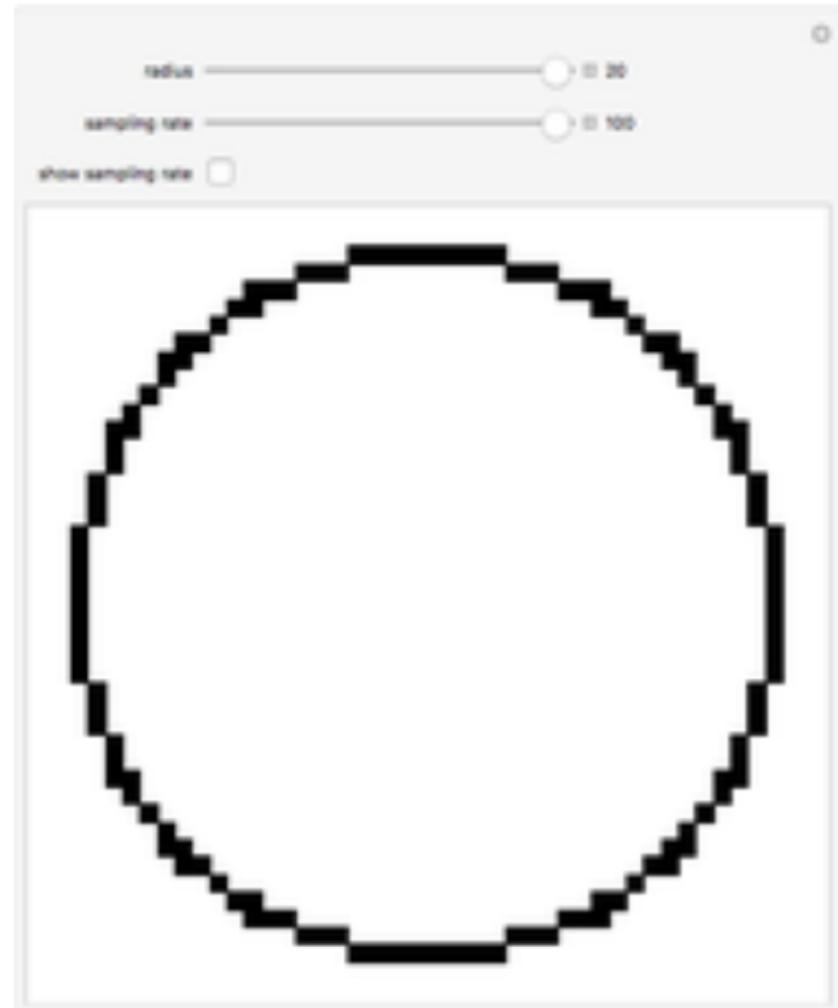
- **void glPolygonMode(GLenum face, GLenum mode);**

Controls the drawing mode for a polygon's front and back faces. The parameter face must be GL_FRONT_AND_BACK; while mode can be GL_POINT, GL_LINE, GL_FILL to indicate whether the polygon should be drawn as points, outlined, or filled. By default, both the front and back faces are drawn filled.

Tipovi primitiva

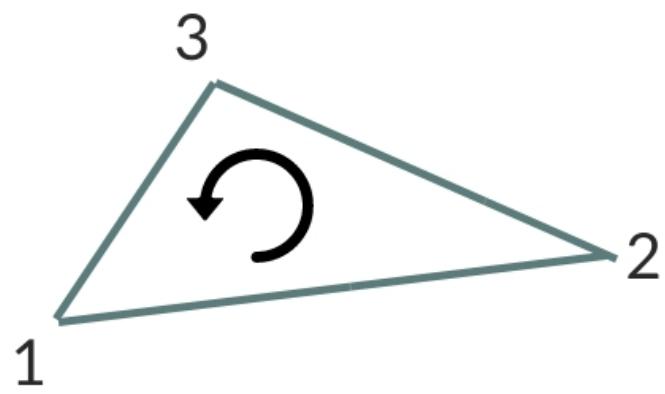
- Points
- Lines
- Line Strips
- Line Loops
- Triangles
- Triangle Strips
- Triangl Fans

- Circle Approximation using Bezier Curves
- De Casteljau Algorithm

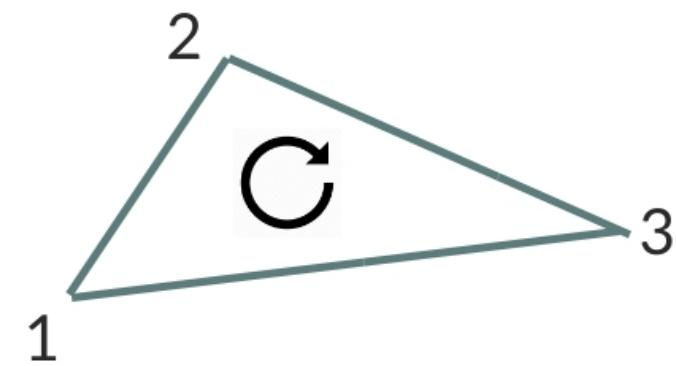


Orijentacija u OpenGL-u

- Definisano redosledom navođenja verteksa
- Lice poligona u OpenGL-u je definisano sa **CCW (Counter-Clock Wise)** orijentacijom
- Orijentacija lica poligona može se definisati pozivom funkcije **glFrontFace(GLint mode)**
- Podrazumevana orijentacija je CCW

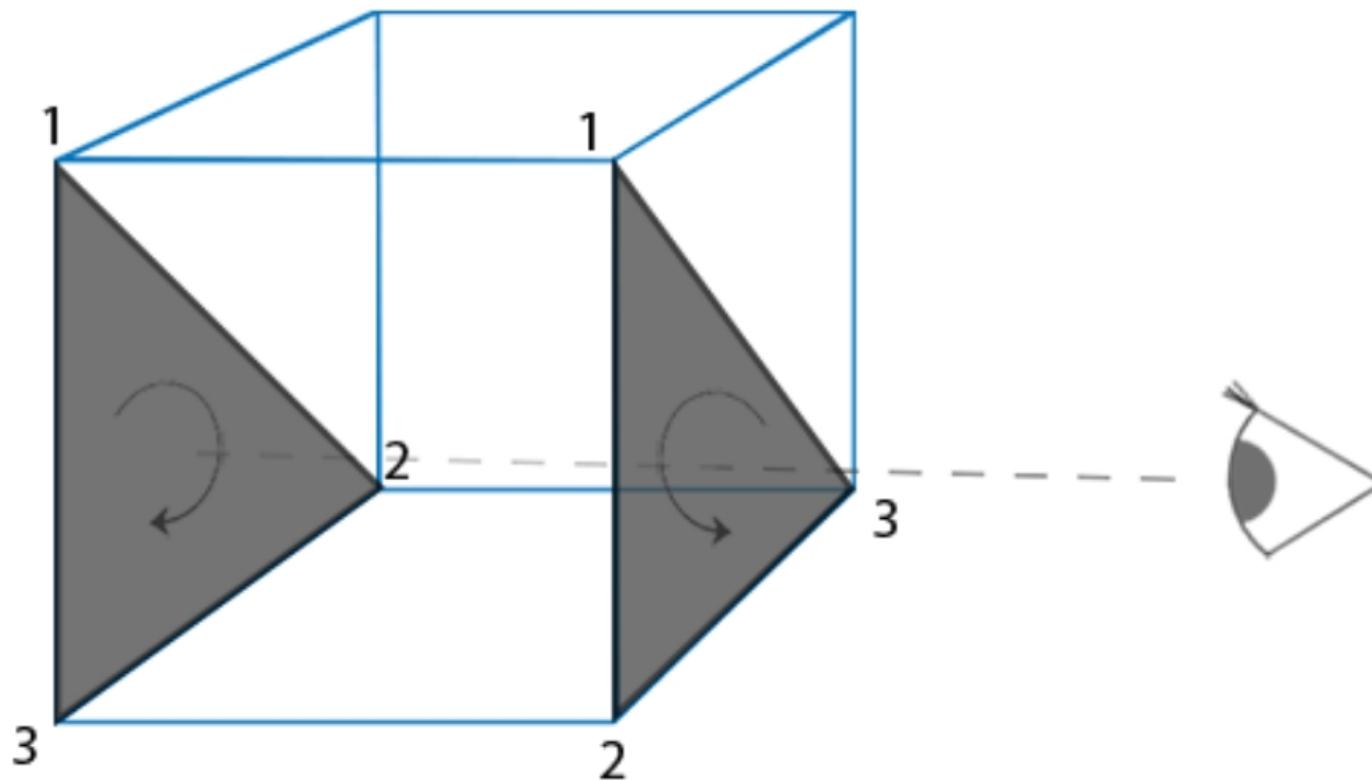


Front poligon

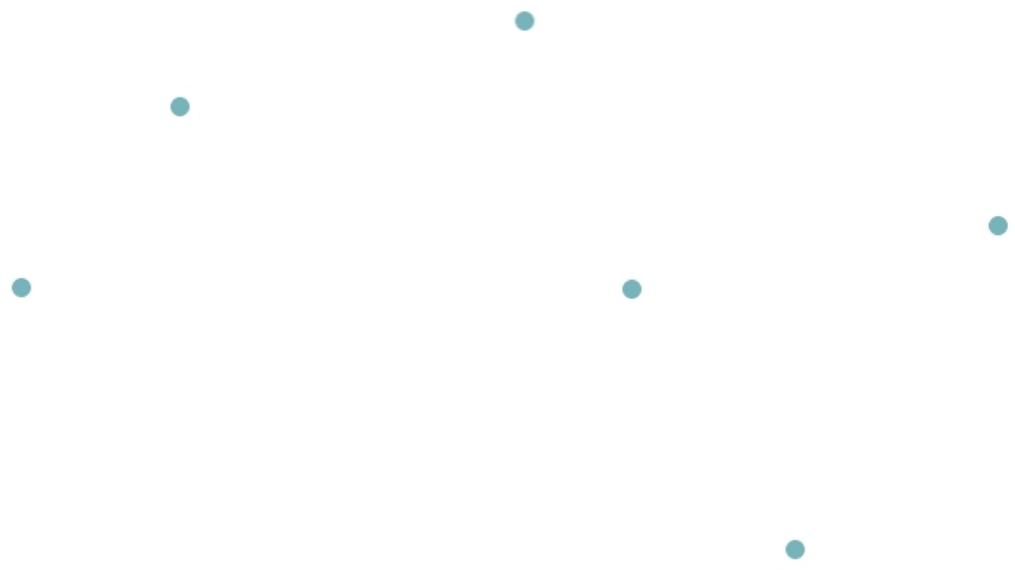


Back poligon

Face culling



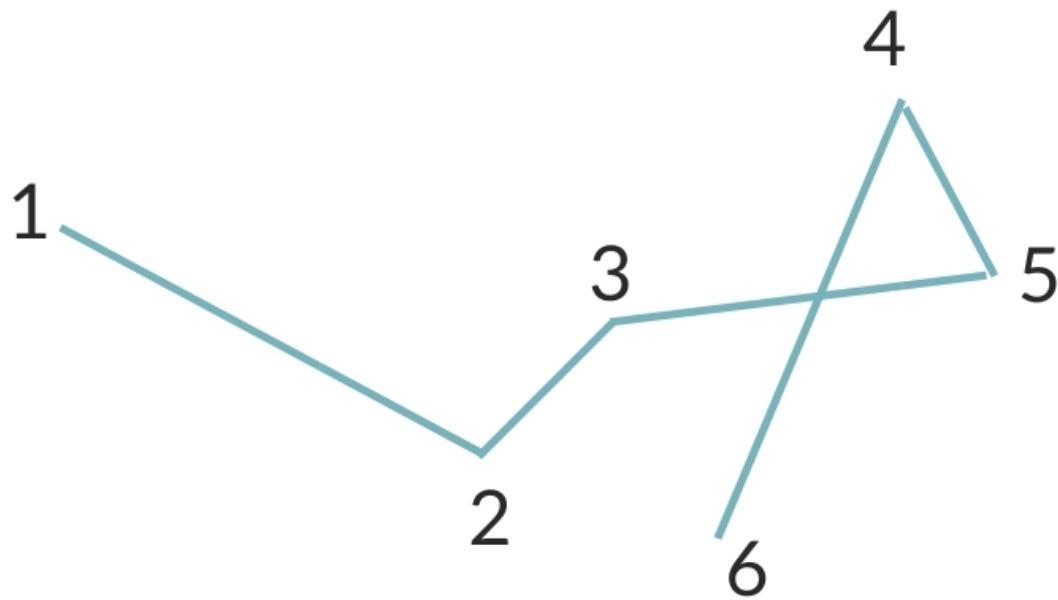
OpenGL primitives - GL_POINTS



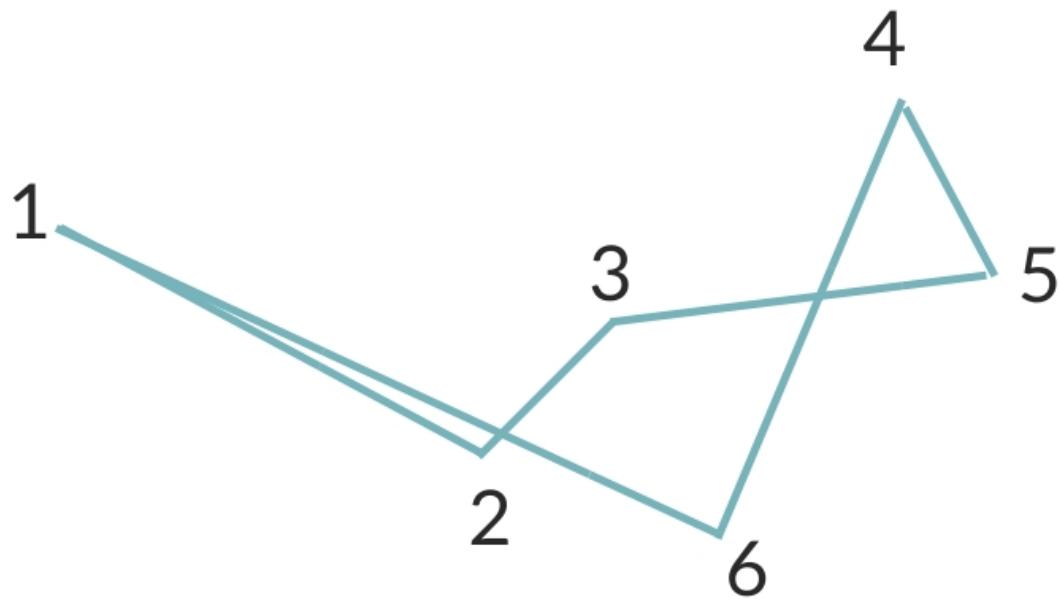
OpenGL primitives - GL_LINES



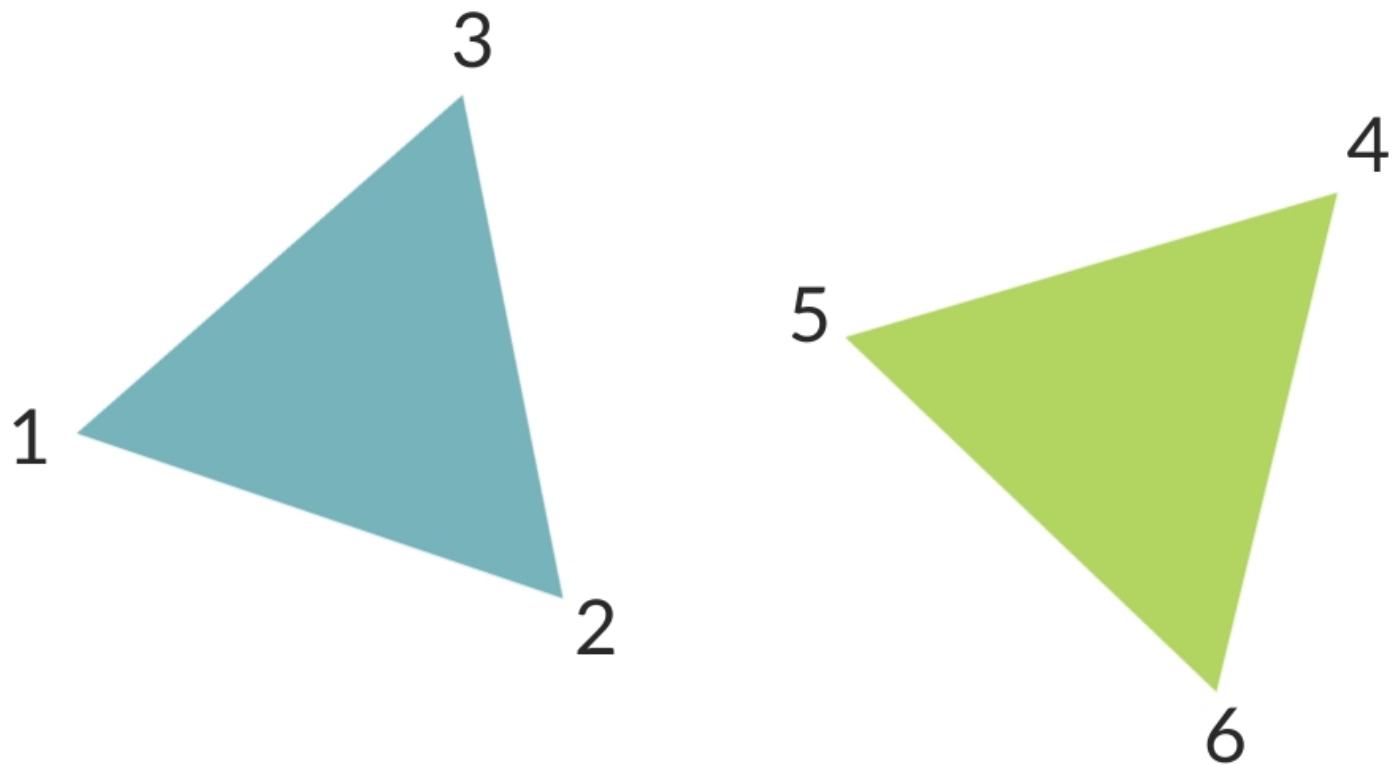
OpenGL primitives - GL_LINE_STRIP



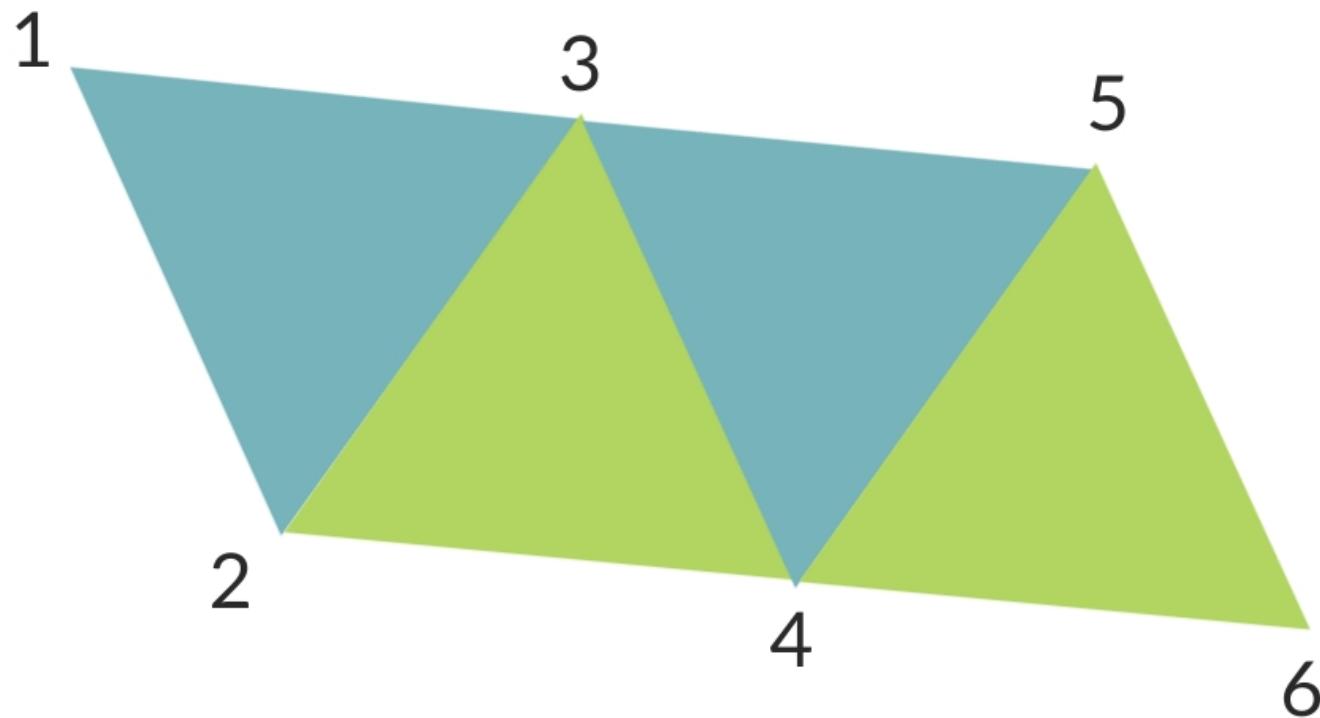
OpenGL primitives - GL_LINE_LOOP



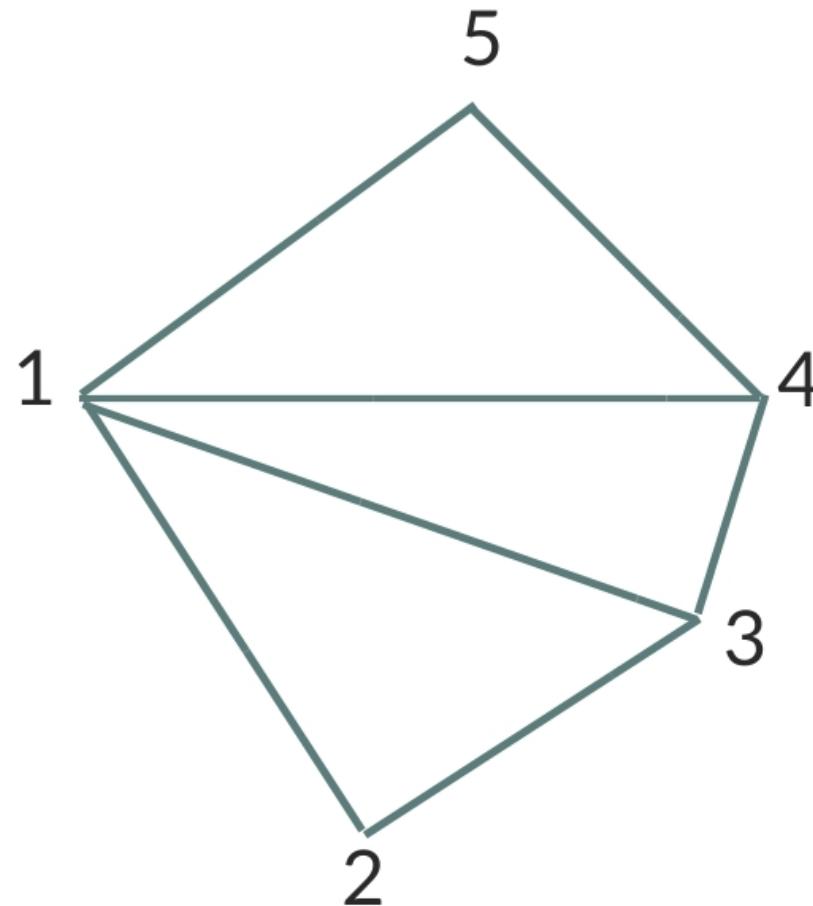
OpenGL primitives - GL_TRIANGLES



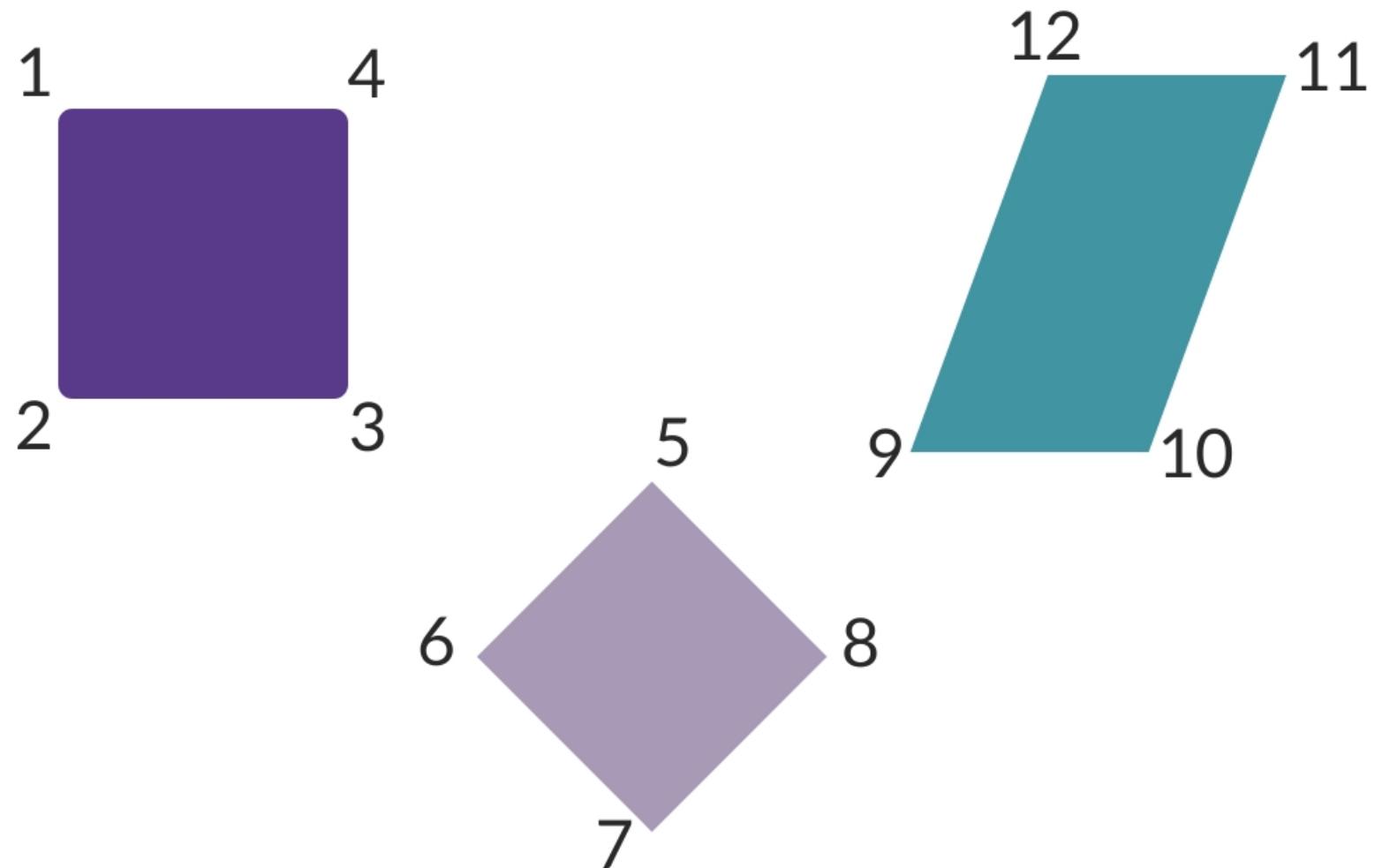
OpenGL primitives - GL_TRIANGLE_STRIP



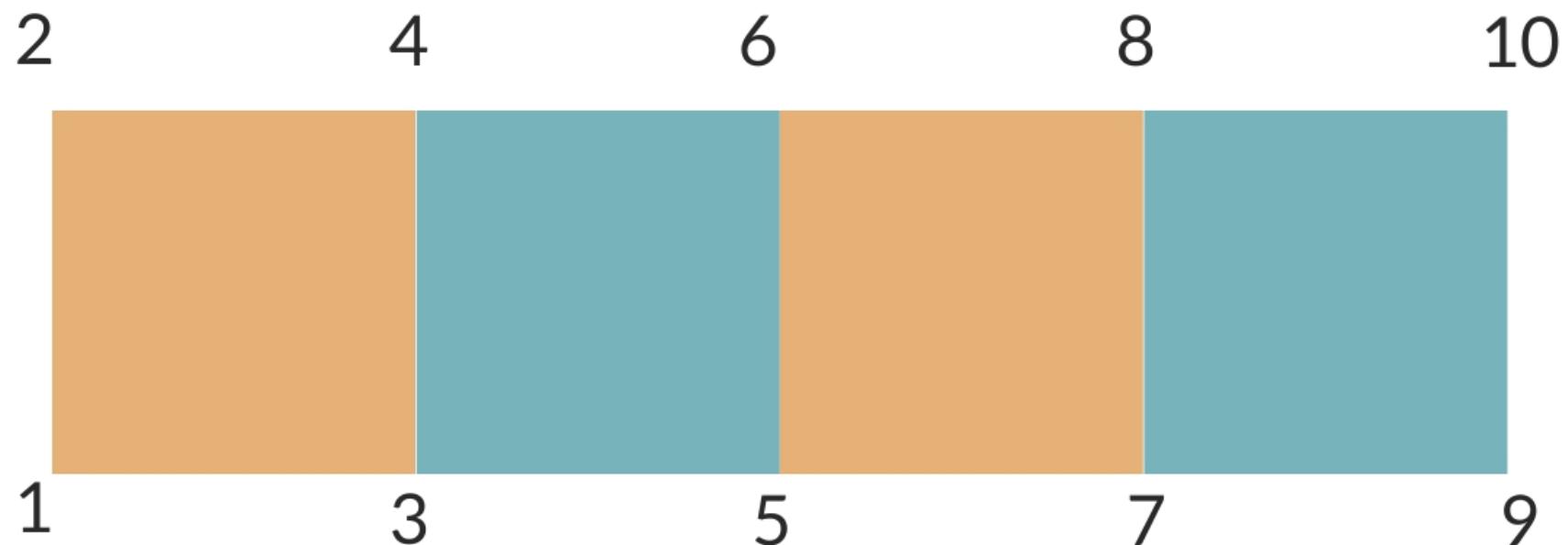
OpenGL primitives - GL_TRIANGLE_FAN



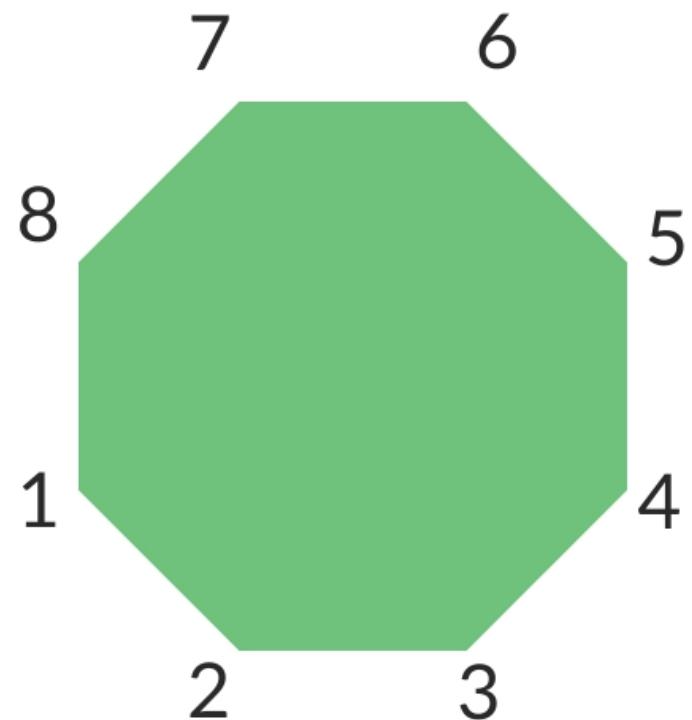
OpenGL primitives - GL_QUADS



OpenGL primitives - GL_QUAD_STRIP



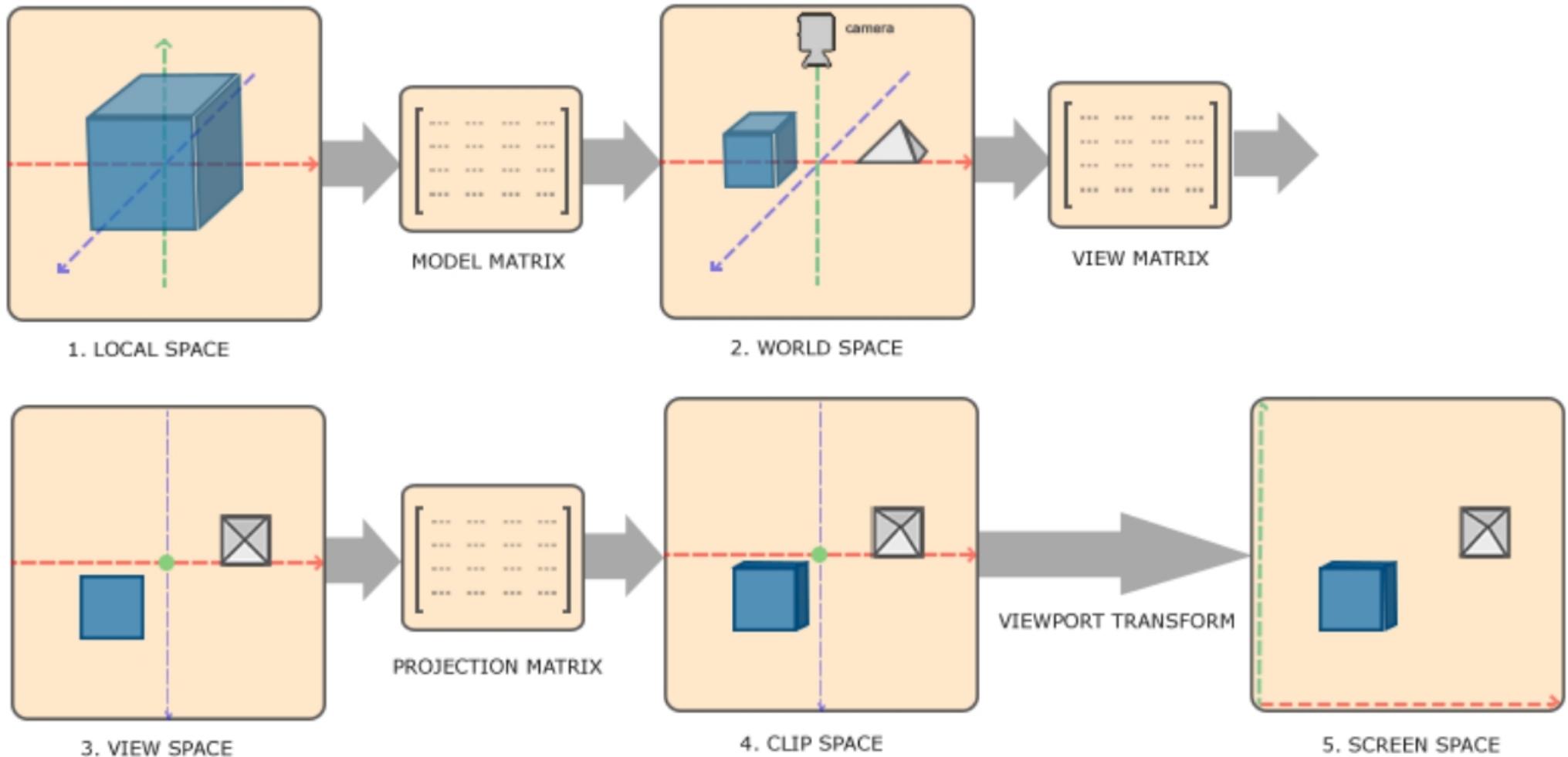
OpenGL primitives - GL_POLYGON



Quadric objekti

- Cylinder (BaseRadius, Height...)
- Disk (InnerRadius, OuterRadius,...)
- Sphere (Radius, Slices, Stacks)

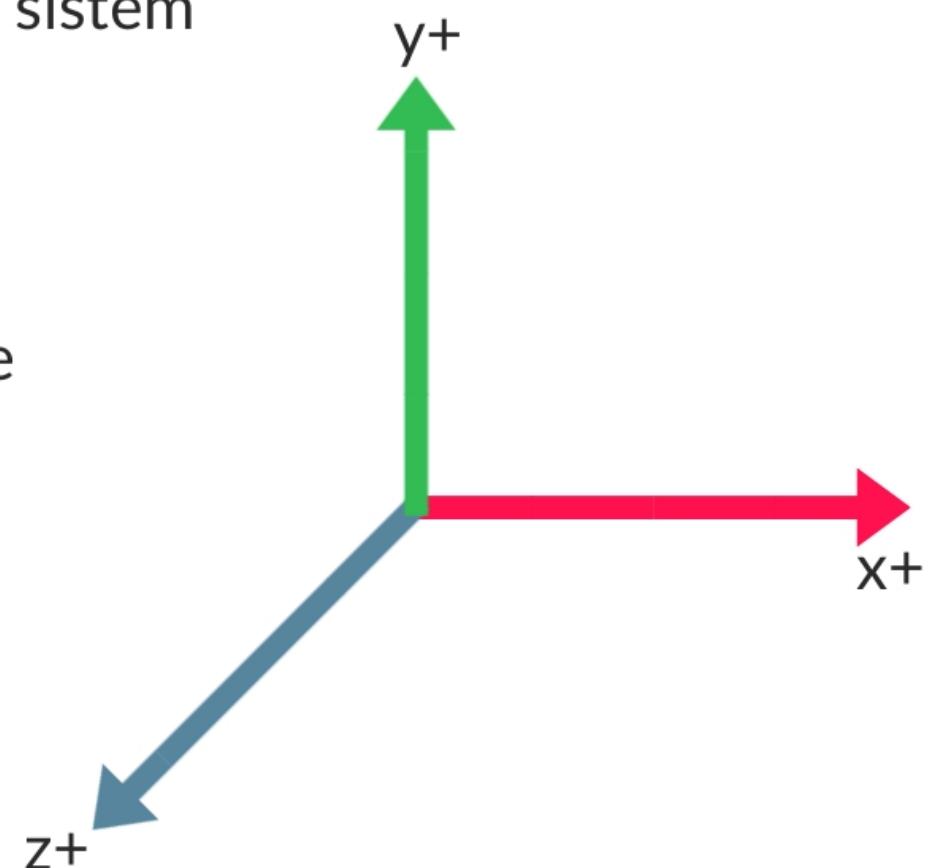
Matrice transformacije



Koordinatni sistem u OpenGL

- Dekartov desni pravougli koordinatni sistem
- Pozitivna rotacija se radi CCW
- Verteks predstavljen sa tri koordinate

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Homogeni koordinatni sistem

- Homogena koordinata - w
- Uvedena radi uniformne predstave matrica transformacija
- Sve transformacije moguće predstaviti 4x4 matricom
- Vrednost homogene koordinate najčešće je jednaka 1

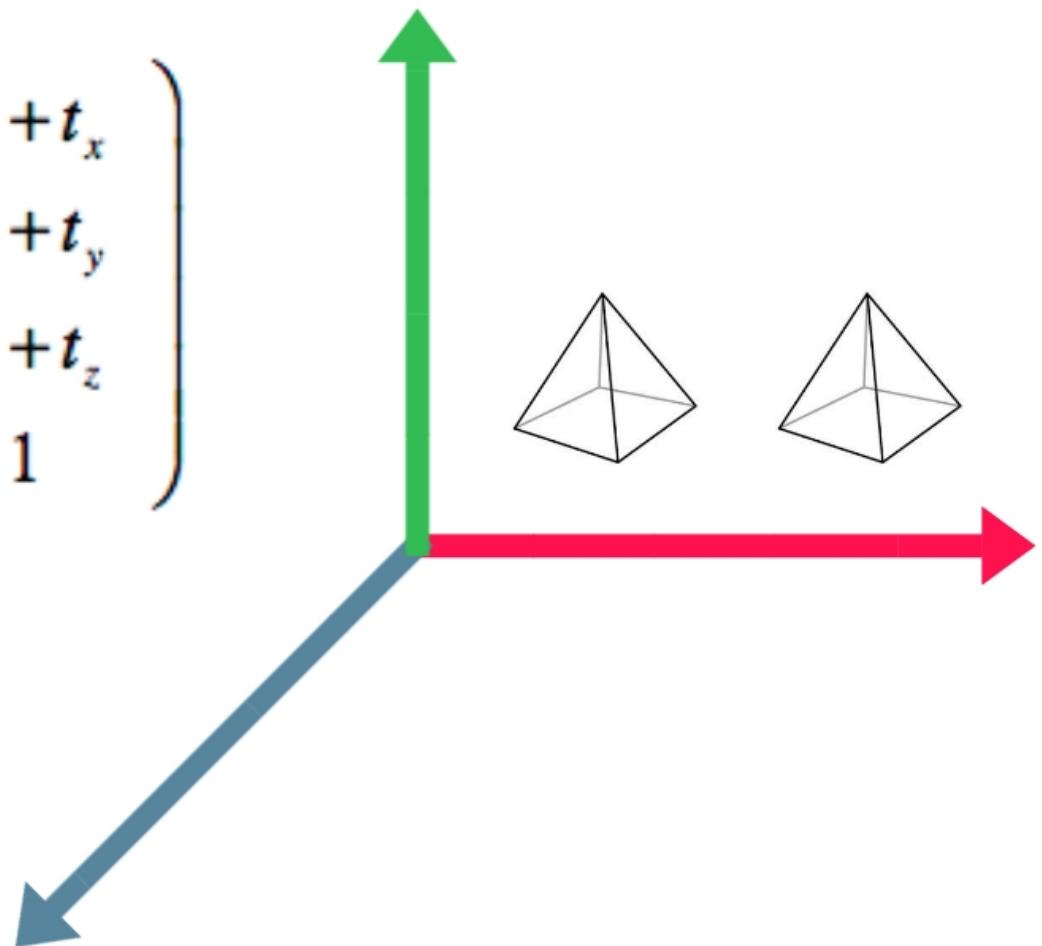
$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

3D Transformacije

- Translacija
- Skaliranje
- Rotacija
- Primenuju se matricama (linearnim transformacijama)

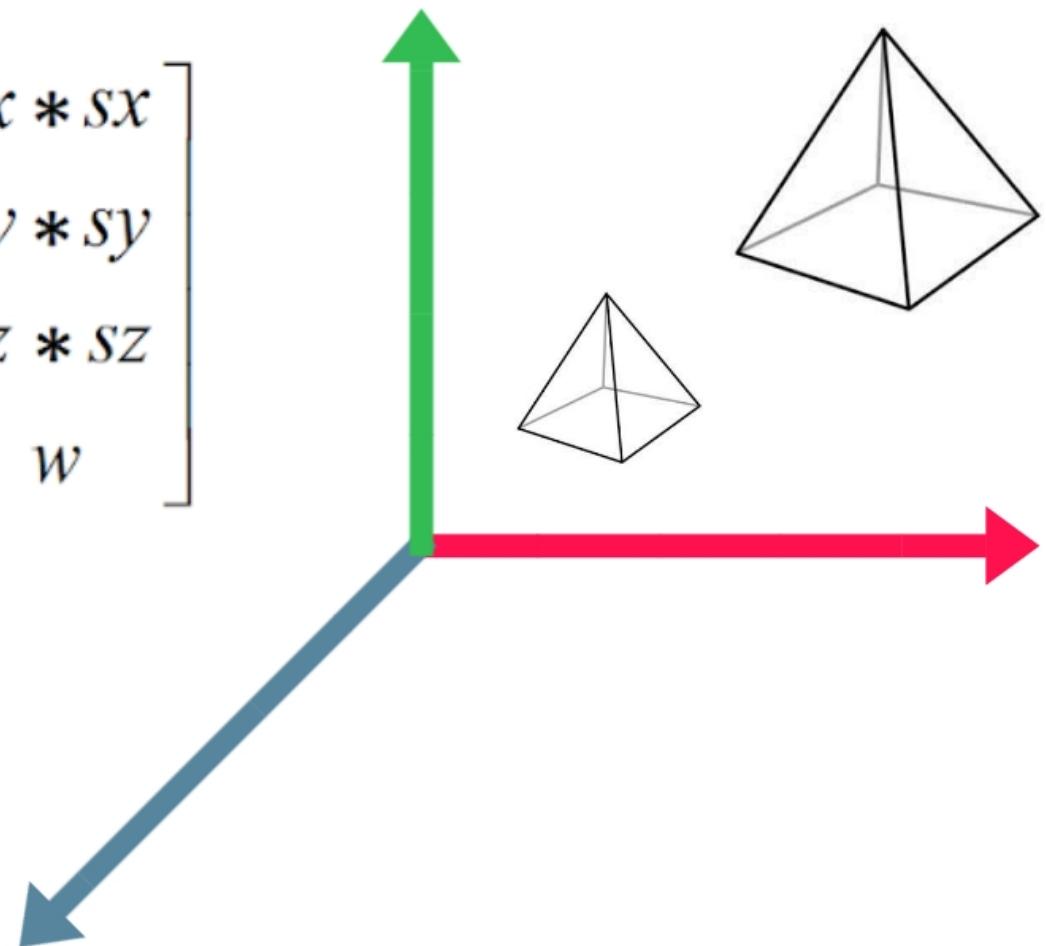
3D translacija

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix} = \begin{pmatrix} v_x + t_x \\ v_y + t_y \\ v_z + t_z \\ 1 \end{pmatrix}$$



3D skaliranje

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x * sx \\ y * sy \\ z * sz \\ w \end{bmatrix}$$



3D rotacija

- Sastoji se iz više elementarnih rotacija
- Ojlerovi uglovi, axis-angle representation, quaternions
- Extrinsic and intrinsic rotations

X-axis rotation	Y-axis rotation	Z-axis rotation
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

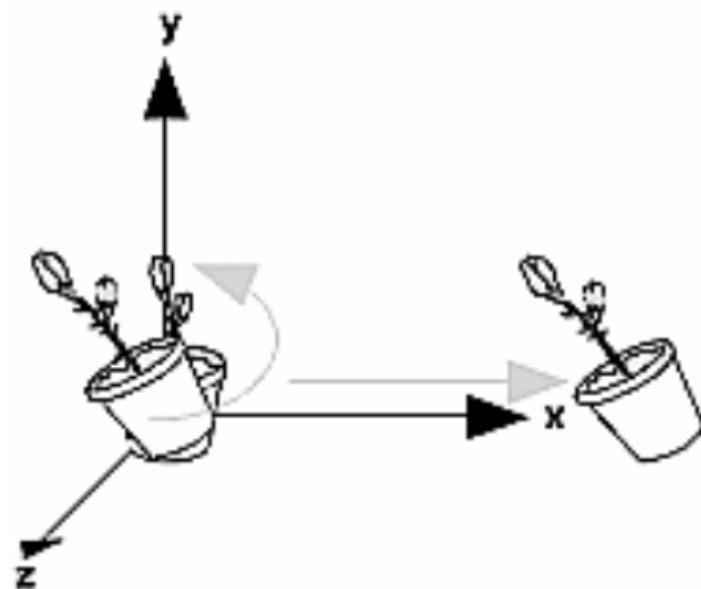
Primer

$$\begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} =$$
$$= \begin{bmatrix} x_{\text{tran}} \\ y_{\text{tran}} \\ z_{\text{tran}} \\ w \end{bmatrix}$$

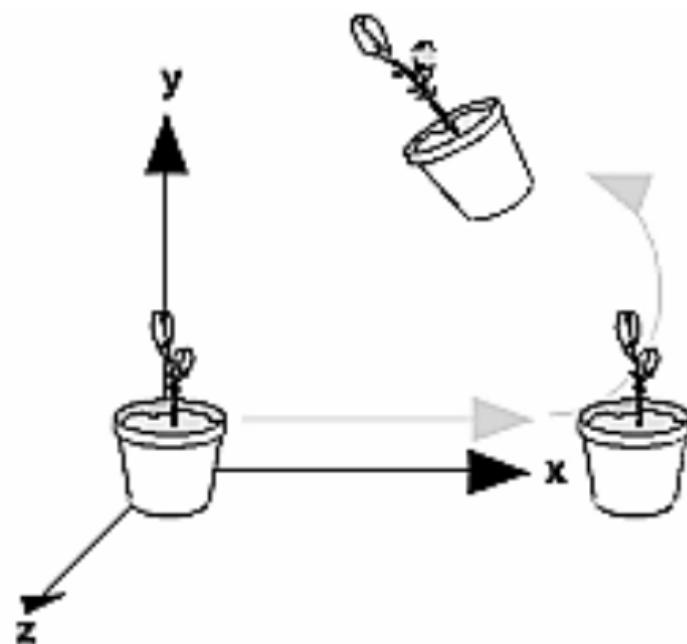
Matrični stek u OpenGL

- Kumulativnost
- Primenuju se u redosledu suprotnom od navođenja
- `glPushMatrix()` - dodavanje matrice transformacije na stek
- `glPopMatrix()` - skidanje matrica transformacija sa steka

- Redosled primene transformacija je bitan



Rotate then Translate



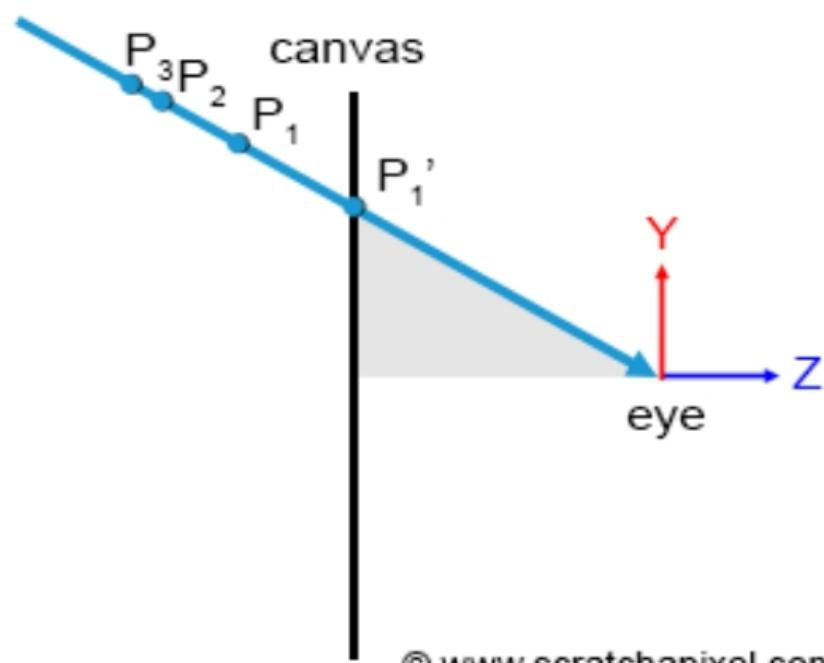
Translate then Rotate

3D transformacije u OpenGL

- `glTranslatef(GLfloat x, GLfloat y, GLfloat z);`
- `glScalef(GLfloat x, GLfloat y, GLfloat z);`
- `glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);`

Problem vidljivosti

- Za realističan prikaz je potrebno odrediti da li je objekat zaklonjen sa tačke posmatranja drugim objektom



© www.scratchapixel.com

Hidden surface determination

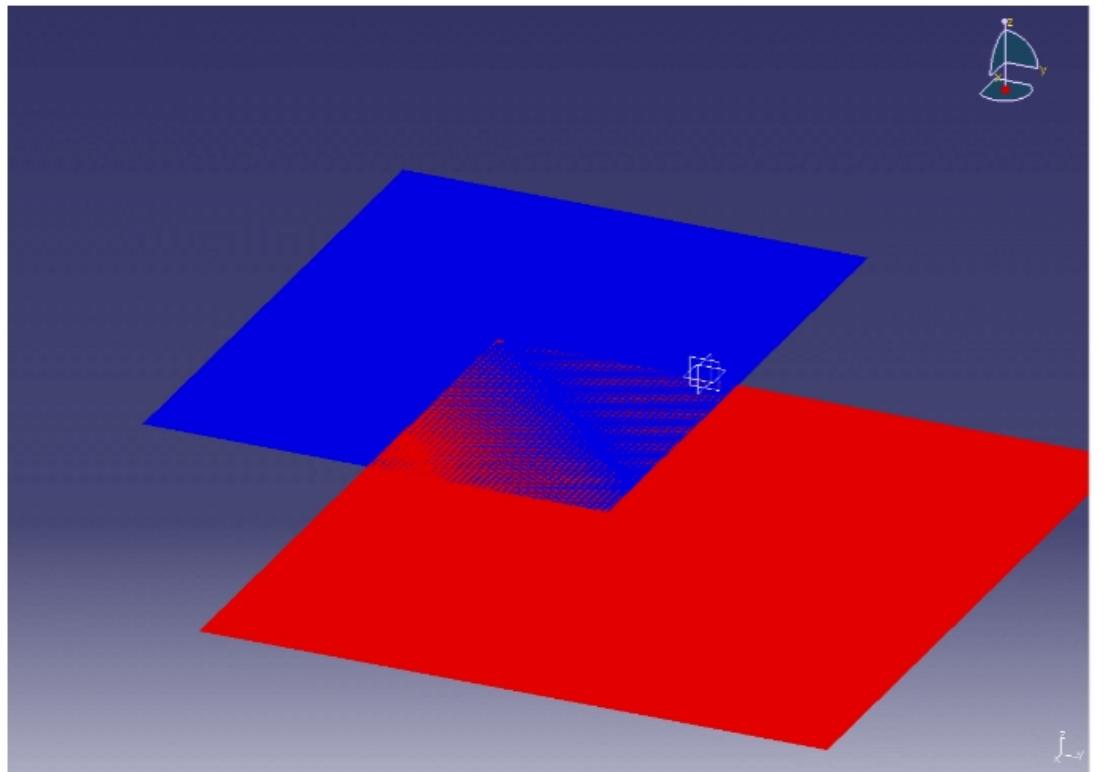
- Z-buffering
- Painter's algorithm
- Binary Space Partitioning (BSP)
- Ray Tracing (implicitno rešava problem)

Z buffer

- Može se odnositi na strukturu podataka ili na metod rešavanja problema vidljivosti
- Čuva podatak o "dubini" (**z** vrednosti) svakog piksela
- 16-bitni, 24-bitni ili 32-bitni z-buffer

Z fighting

- Situacija kada se dve ravni nađu na istoj dubini (**z**-vrednosti)



Backface culling

- Metoda za određivanje koji poligoni idu na renderovanje na osnovu orientacije sa definisane tačke posmatranja
- Jedan od metoda određivanja orientacije jeste formulom:

$$(V_0 - P) \cdot N >= 0$$

OpenGL arhitektura

- CPU-GPU saradnja
- Klijent-server arhitektura
- Klijent - aplikacija koja koristi OpenGL API i izvršava se na CPU
- Server - implementacija OpenGL graphics engine-a koja se izvršava na GPU

Različiti mehanizmi iscrtavanja u OpenGL-u

- Immediate režim (`glBegin(mode)/glEnd()`)
- Display list
- Vertex Array mehanizam
- Vertex Buffer Objects (VBOs)

Display List

- Kompajlira pozive OpenGL funkcija i čuva u memoriji na serverskoj strani (GPU)
- **GLuint glGenLists(int num)** - kreira niz praznih display listi od kojih je svakoj listi dodeljen jedinstveni indeks. Parametar **num** predstavlja broj display listi koji će se kreirati.
- **void glNewList(GLuint listIndex, GLenum mode)** - nakon poziva se definišu komande koje će se čuvati u Display listi. Parametar mode može biti GL_COMPILE ili GL_COMPILE_AND_EXECUTE
- **void glEndList()** - označava kraj definisanja komandi koje će se čuvati u display listi
- **void glCallList(GLuint listIndex)** - izvršava Display listu sa zadatim indeksom

Vertex Array mehanizam

- Scena definisana pomoću nizova
- OpenGL prepoznaće šest različitih tipova nizova:
 - niz verteksa - **glVertexPointer()**
 - niz normala - **glNormalPointer()**
 - niz boja verteksa - **glColorPointer()**
 - niz koordinata tekstura - **glTexturePointer()**
 - indeksni niz - **glIndexPointer()**
 - edge flag niz - **glEdgeFlagPointer()**

Vertex Array mehanizam

- `glEnableClientState(GLenum cap)/glDisableClientState(GLenum cap)`
- Nizovi definisani parametrom `cap` (capabilities):
 - `GL_VERTEX_ARRAY`
 - `GL_COLOR_ARRAY`
 - `GL_EDGE_FLAGS_ARRAY`
 - `GL_NORMAL_ARRAY`
 - `GL_INDEX_ARRAY`
 - `GL_TEXTURE_COORD_ARRAY`
 - `GL_SECONDARY_COLOR_ARRAY`
 - `GL_FOG_COORD_ARRAY`

Vertex Array mehanizam

- `glVertexPointer(GLint size, GLenum type, GLsizei stride, const void* pointer);`
- Parametri:
 - **size** - broj koordinata kojim je definisana pozicija verteksa
 - **type** - tip podatka koordinate verteksa
 - **stride** - byte offset između dva verteksa u nizu
 - **pointer** - pokazivač na prvu koordinatu prvog verteksa u nizu

Funkcije za iscrtavanje

- **glDrawArrays(GLenum mode, GLint first, GLsizei count)** - enkapsulira iscrtavanje u jedan poziv funkcije
 - **mode** - tip grafičke primitive
 - **first** - početni indeks niza od kog krećemo interpretaciju
 - **count** - broj verteksa
- **glDrawElements(GLenum mode, GLsizei count, GLenum type, const void* indices)** - prednost u odnosu na glDrawArrays je što ne moramo da definišemo tačku više pute
 - **mode** - tip grafičke primitive
 - **count** - broj verteksa
 - **type** - tip vrednosti elemenata indeksnog niza
 - **indices** - pokazivač na niz indeksa

Prednosti i mane

- Display list - podaci i pozivi funkcija vezani za scenu se kompajlirani čuvaju na serverskoj strani (GPU), ali nakon što kompajliramo gubimo mogućnost menjanja scene
- Vertex Array mehanizam - prednost u odnosu na glBegin/glEnd način iscrtavanja jeste manji broj poziva funkcija (glDrawArray), kao i manji broj verteksa potrebnih za definisanje scene korišćenjem indeksnih nizova (glDrawElements). Mana je što se nizovi čuvaju na klijentskoj strani, što daje slabije performanse od display liste

Vertex Buffer Objects(VBO)

- Rešava problem statičnosti scene prilikom korišćenja display listi i rešava problem čuvanja scene (predstavljene nizovima) na strani aplikacije prilikom korišćenja vertex array mehanizma
- VBOs predstavljaju baferizovane objekte koji služe za čuvanje podataka o sceni na serverskoj strani (GPU)
- Za razliku od display listi podaci u ovim baferima mogu se čitati i upisivati nove vrednosti
- Vertex array mehanizam je moguće koristiti za definisanje podataka (nizovima) i načina pristupa podacima (referenciranje/dereferenciranje nizova)

Vertex Buffer Objects(VBO)

- glGenBuffers (GLsizei n, GLuint* buffers)
- glBindBuffer (GLenum target, GLuint buffer)
- glBufferData (GLenum target, GLsizeiptr size, const void* data, GLenum usage)

Učitavanje modela

- Vrlo teško modelovati kompleksne objekte u OpenGL-u
- 3DS Max, Maya, Blender
- 3D model file formats:
 - Wavefront OBJ (.obj, često ide uz .mtl fajl)
 - Collada (.dae)
 - 3DS Max (.3ds)
 - Flimbox (.fbx)

Assimp biblioteka

- Open Asset Import Library
- Koristi se za učitavanje kompleksnih modela iz različitih fajl formata
- Model se učitava u **scene** objekat

Assimp scene

- Higerarhijski model (Node hierarchy)
- Strukture kojim su definisani modeli:
 - rootNode
 - Meshes
 - Materials
 - Cameras
 - Textures
 - Animations
 - Flags

Node hierarchy

- rootNode predstavlja korenski čvor hijerarhije
- Svaki čvor može imati jednog roditelja
- Node strukture referenciraju jedan ili više Mesh struktura
- Svaki čvor definiše lokalni koordinatni sistem za Mesh
- Rekurzivno procesovanje čvorova

Mesh struktura

- Čuvaju se u nizu unutar **scene** objekta
- Obavezno sadrže podatke o poziciji verteksa i orijentaciji poligona (Vertices, Faces)
- Mogu sadržati podatke o normalama, tangentama, skeletima (Normals, Tangents, Bones)

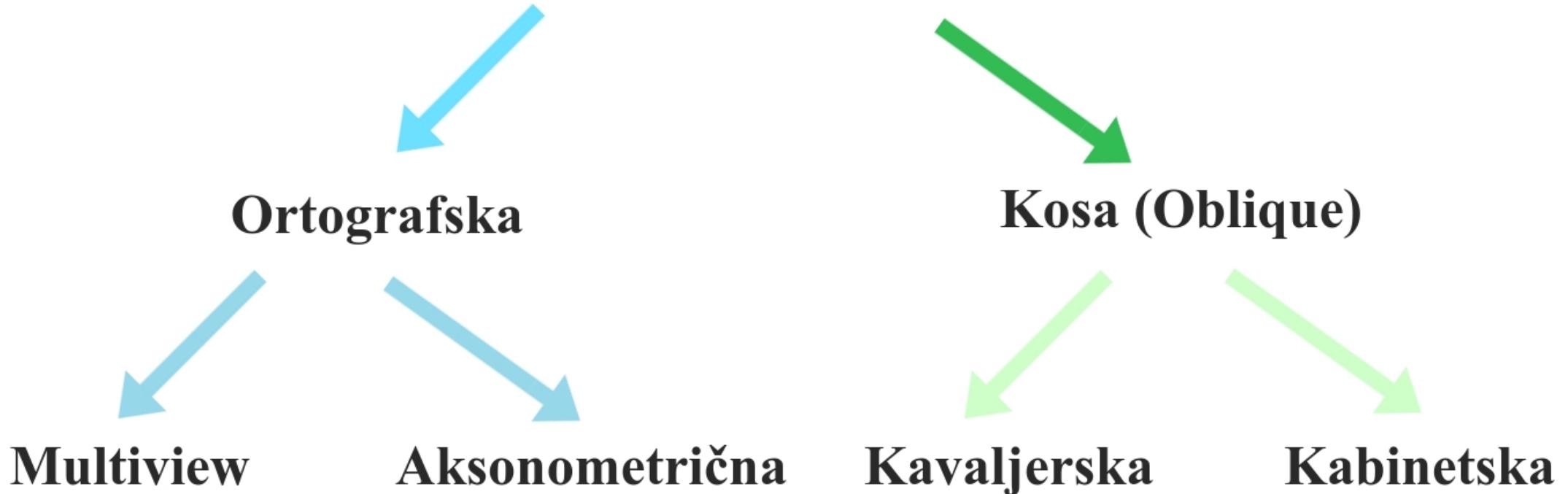
Projekcije

- Paralelne
- Perspektivna

Paralelna projekcije

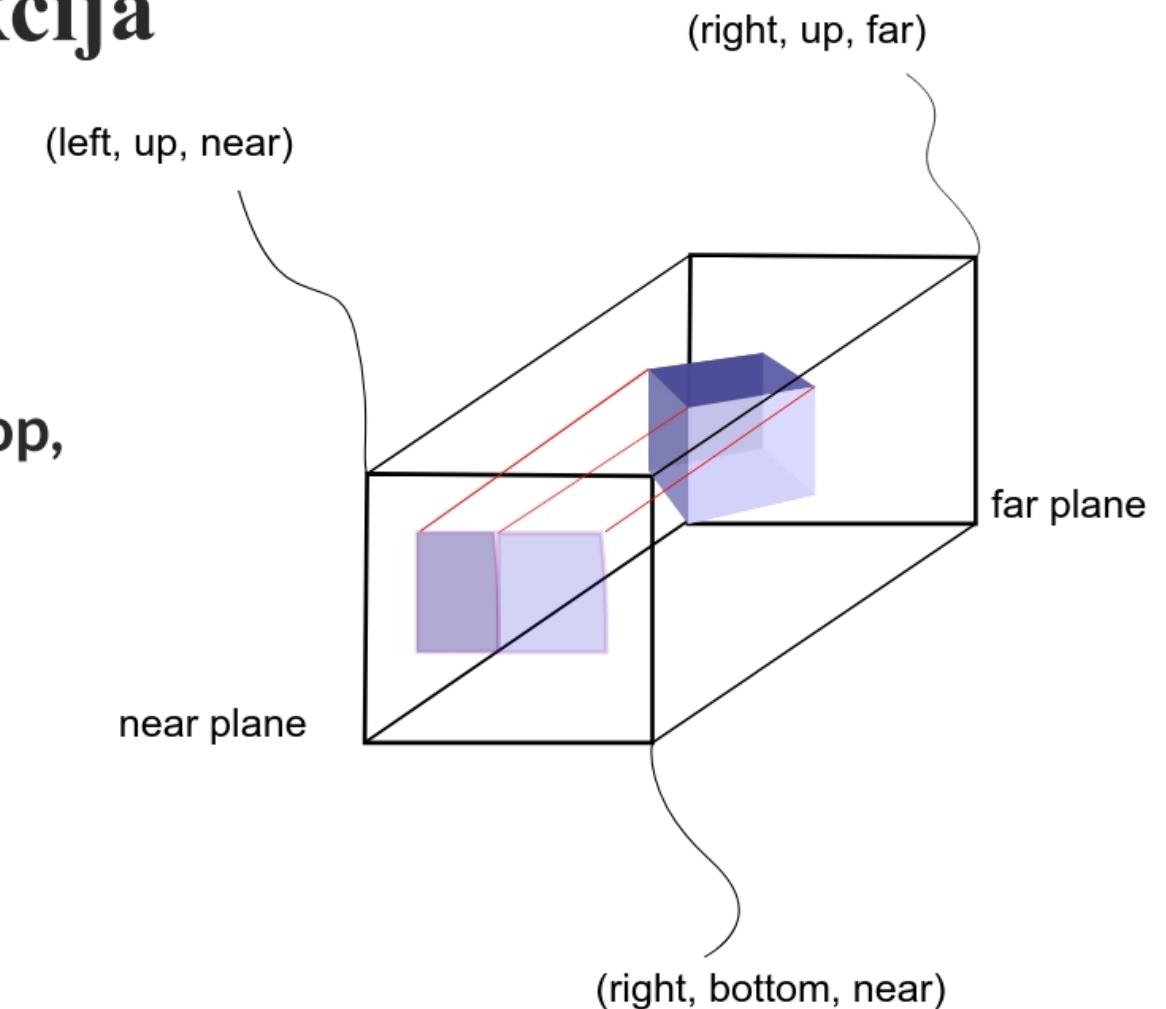
- Tačke scene se projektuju povlačenjem paralelnih linija na ravan projekcije (viewing plane)
- Vrednosti z koordinata se odbacuju
- Manji nivo realizma, ali održavaju se relativni odnosi između objekata (normale, paralele)
- Uglavnom se koristi u arhitekturi, građevini..

Paralelna projekcije



Ortogonalna projekcija

`glOrtho(left, right, bottom, top,
far, near)`



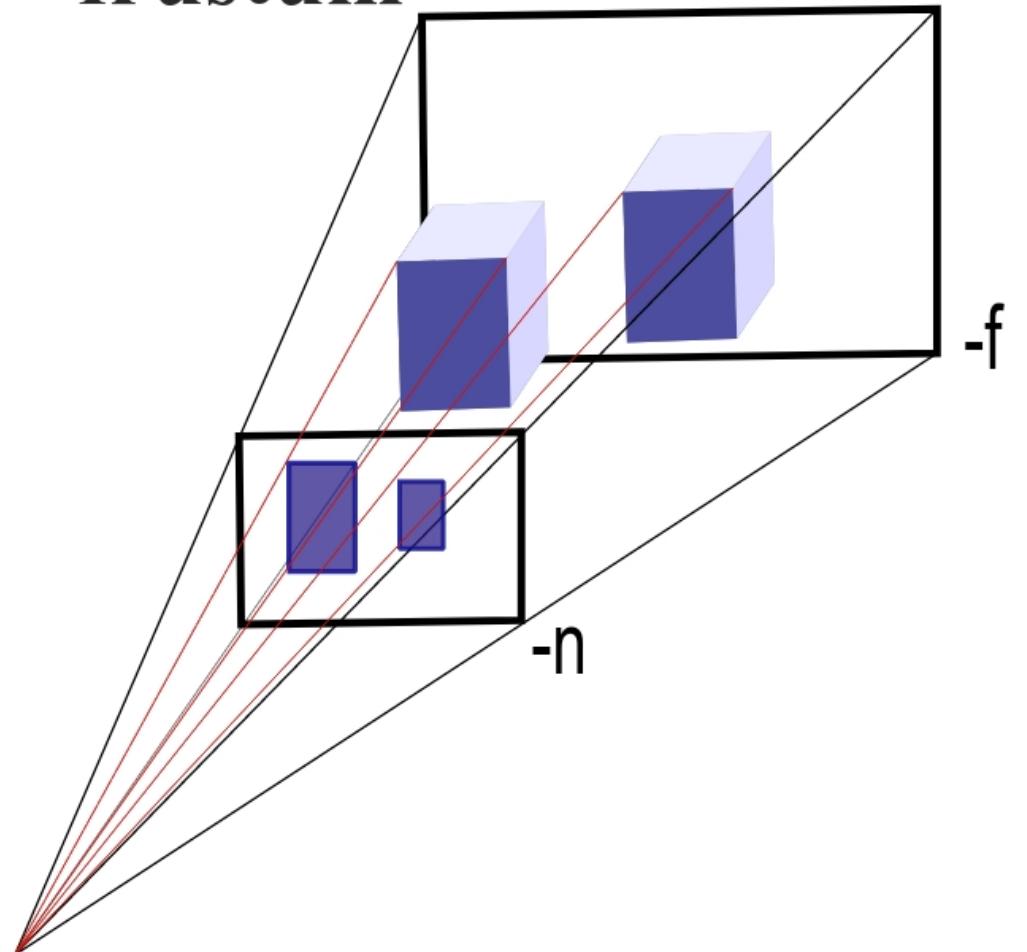
Perspektivna projekcija

- Tačke scene se projektuju na ravan projekcije (viewing plane) povlačenjem linije do centra projekcije (tačke posmatranja)
- Približnije ljudskom načinu posmatranja, objekti koji su udaljeniji su prikazani kao umanjeni
- Tipovi:
 - One point
 - Two point
 - Three point

Perspektivna projekcija - frustum

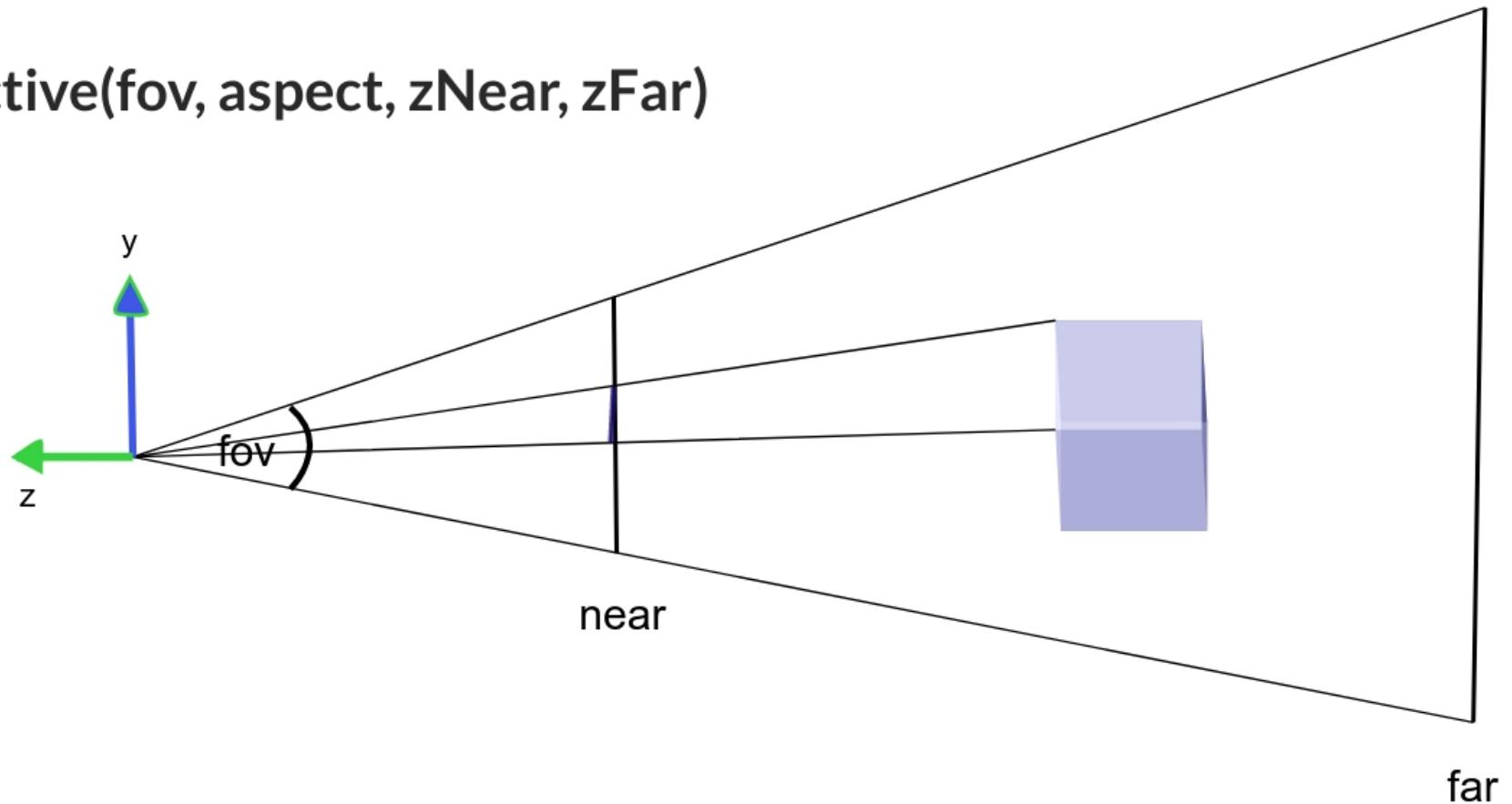
- Perspective division
- Tačke nestajanja(Vanishing points)

`glFrustum(left, right, bottom, top,
far, near)`



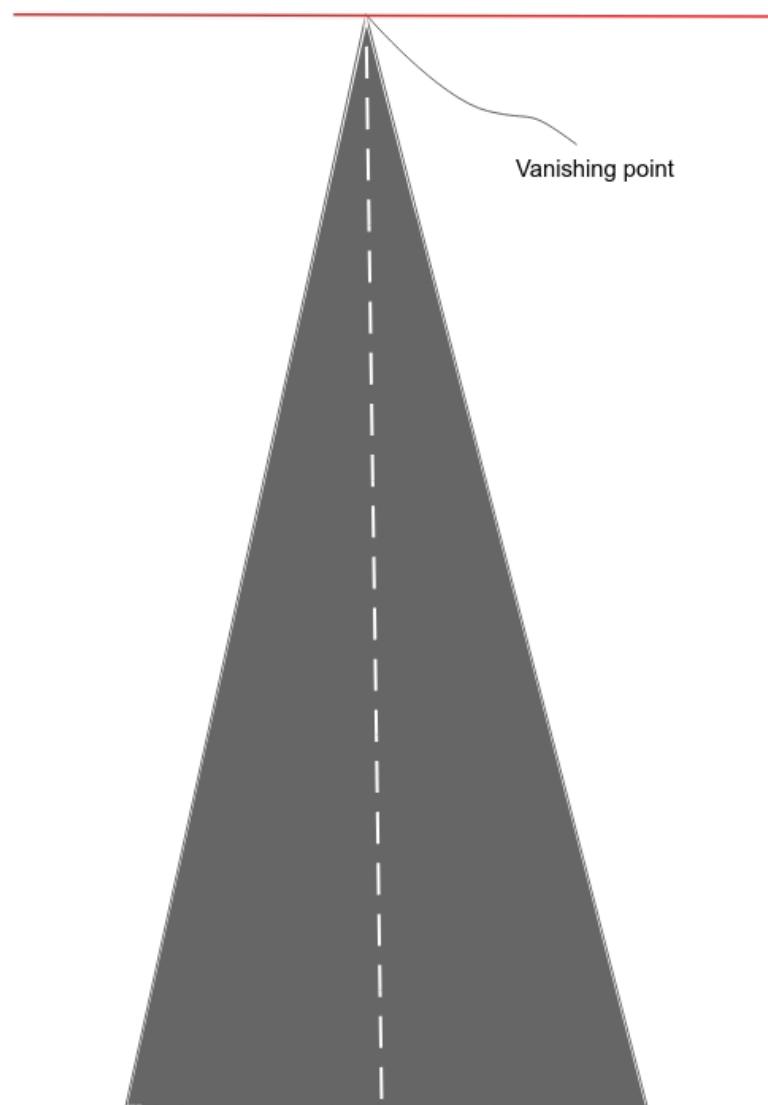
Perspektivna projekcija

`gluPerspective(fov, aspect, zNear, zFar)`



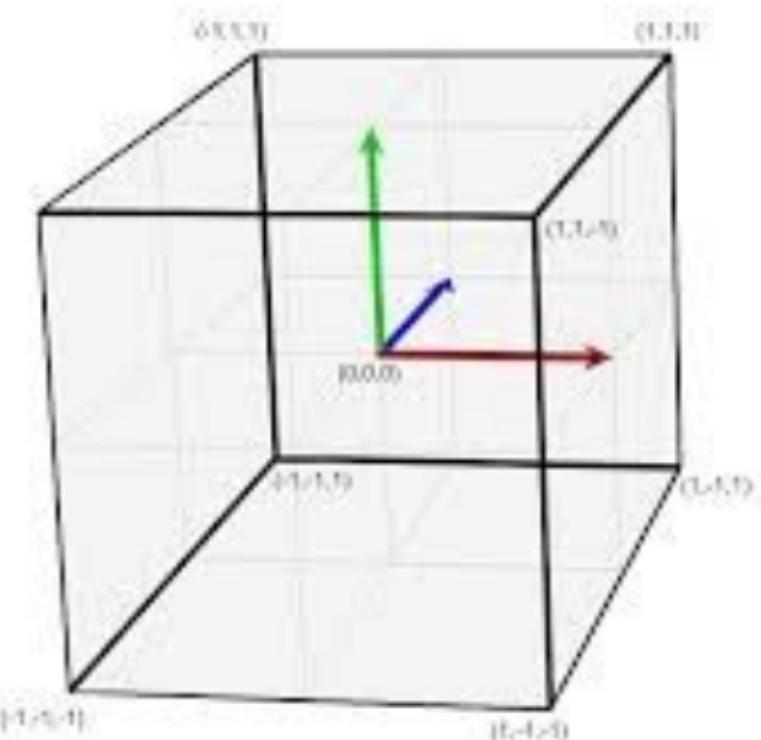
Perspektivna projekcija

- Paralelne linije se spajaju u tačkama nestajanja



Normalized Device Coordinates

- Nakon primene matrice projekcije scena se prebacuje u NDC koordinatni sistem



Clipspace

Viewport transform

- Vrši se mapiranje NDC koordinata u koordinate prozora (screen coordinates, window coordinates)
- Definiše se površina na kojoj se vrši renderovanje (najčešće četvorougao)
- `glViewport(GLint x, GLint y, GLsizei width, GLsizei height)`
 - `x, y` - donji levi ugao viewporta
 - `width, height` - širinu i visinu viewporta

Iscrtavanje teksta u OpenGL-u

- Može biti zahtevan proces
- Različite tehnike za kreiranje karaktera (3D mesh, Bitmap Font - glyphs, Bezier curves..)

Iscrtavanje teksta u SharpGL-u

- `glDrawText(int x, int y, float r, float g, float b, string font, float fontSize, string text)`
- `glDrawText3D(string font, float fontSize, float deviation, float extrusion, string text,)`

Kamera u OpenGL-u

- `gluLookAt(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)` - definiše poziciju, smer gledanja i orijentaciju kamere
- Parametri:
 - eye - pozicija kamere
 - center - tačka u koju kamera gleda
 - up - definiše orijentaciju kamere (šta je gore)
- LookAt funkcija kreira view matrix koja vrši transformaciju verteksa iz World Space u View Space koordinatni sistem

Kamera u OpenGL-u

- LookAt funkcija kreira matricu transformacije koja se može razložiti na matricu rotacije i matricu translacije

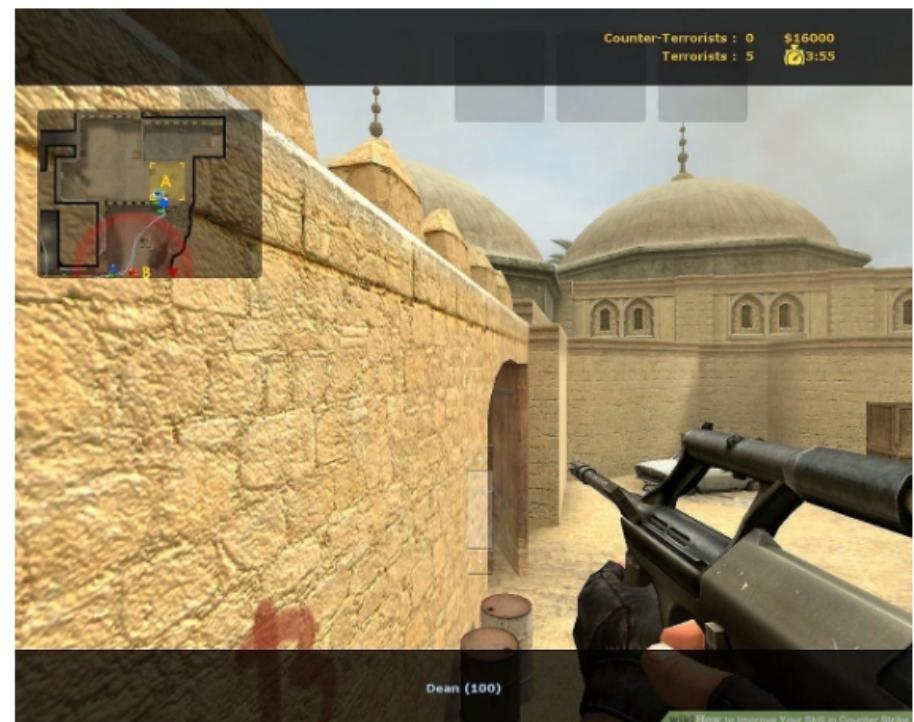
$$M_{\text{view}} = M_R M_T = \begin{pmatrix} r_0 & r_4 & r_8 & 0 \\ r_1 & r_5 & r_9 & 0 \\ r_2 & r_6 & r_{10} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r_0 & r_4 & r_8 & r_0t_x + r_4t_y + r_8t_z \\ r_1 & r_5 & r_9 & r_1t_x + r_5t_y + r_9t_z \\ r_2 & r_6 & r_{10} & r_2t_x + r_6t_y + r_{10}t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- `glRotate(-rotX, 1, 0, 0);`
`glRotate(-rotY, 0, 1, 0);`
`glRotate(-rotZ, 0, 0, 1);`
`glTranslate(-eyeX, -eyeY, -eyeZ);`

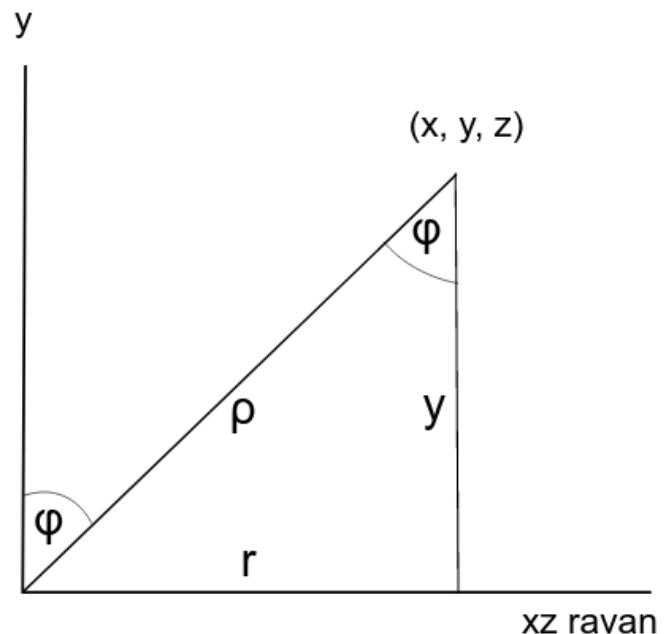
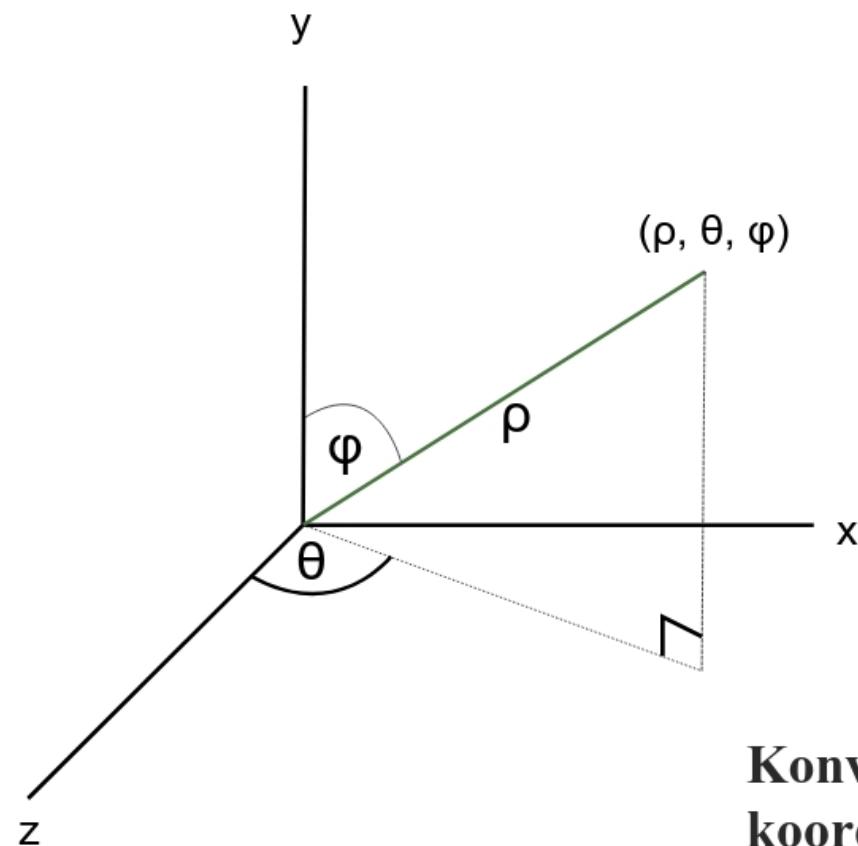
[http://www.songho.ca/opengl/gl_camera.html]

FPS Camera

- First Person Shoot camera
- Rotiramo kameru pomeranjem miša
- Pozicija kamere se menja pritiskom tastera na tastaturi(WASD, strelice)
- Česta pojava u game developmentu



Sferični koordinatni sistem



Konverzija u Kartezijev
koordinatni sistema:

$$x = \rho \sin \varphi \cos \theta$$
$$y = \rho \cos \varphi$$
$$z = \rho \sin \varphi \sin \theta$$

Animacija

- Tehnika prikazivanja više uzastopnih slika (frejmova) u pokušaju "oživljavanja" scene
- **DispatcherTimer class**

A timer that is integrated into the Dispatcher queue which is processed at a specified interval of time and a specified priority.

[<https://docs.microsoft.com/en-us/dotnet/api/system.windows.threading.dispatchertimer?redirectedfrom=MSDN&view=net-5.0>]

Boja verteksa

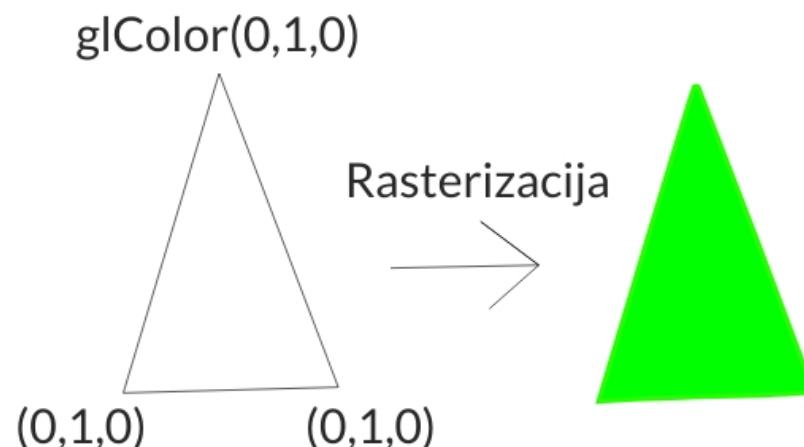
- `glColor(float r, float g, float b);`
- Materijali
- Shading modeli
- Modeli osvetljenja (normale)
- Teksture

Shading modeli

- **void glShadeModel(GLenum mode)**
- **mode:** • GL_FLAT
 - GL_SMOOTH
- Definiše način određivanja boja piksela prilikom rasterizacije

Flat shading

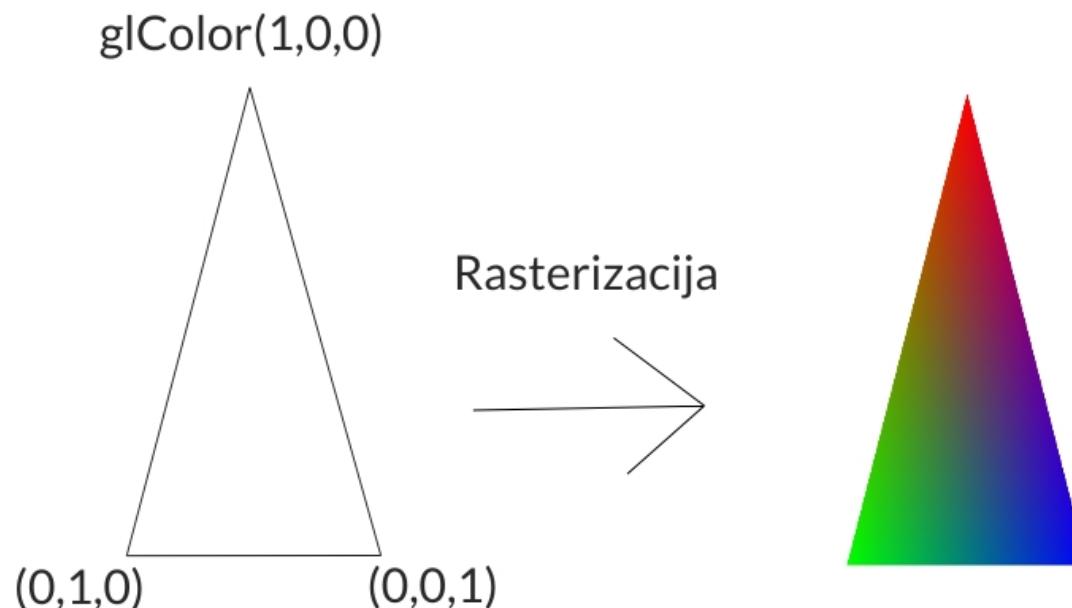
- Konstantno senčenje
- Prilikom rasterizacije flat shaded poligona uzima boju jednog od verteksa poligona i dodeljuje tu boju svim pikselima/fragmentima rasterizovanog poligona



[<https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glShadeModel.xml>]

Flat shading

- Prilikom rasterizacije poligona boje piksela se određuju interpolacijom između boja verteksa



Reflektivna svojstva objekata/materijala

- Svaki objekat, odnosno materijal od kog je sačinjen taj objekat, ima reflektivna svojstva
- Objekat/materijal delom apsorbuje svetlosne zrake, delom ih odbija
- Odbijeni zraci svetlosti od objekta koji stižu do našeg oka predstavljaju boju kojom ćemo videti taj objekat
- Kako bi dobili realističnije scene, potrebno je da simuliramo interakciju svetlosti i materijala objekata

Materijali u OpenGL-u

- Materijali definišu reflektivna svojstva objekata
- Potrebno je definisati model kojim će se simulirati različiti materijali kroz interakciju sa modelom osvetljenja
- Komponente koje čine model materijala:
 - Ambijentna
 - Difuzna
 - Spekularna
 - Shininess
 - Emisiona
- Svaka od komponenti interaguje sa odgovarajućom komponentom svetlosnog modela
- Primeri materijala:
[<http://devernay.free.fr/cours/opengl/materials.html>]

Materijali u OpenGL-u - glMaterial

- **void glMaterial(GLenum face, GLenum pname, const GLfloat * params)**
 - **face** - GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
 - **pname** - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_EMISSION, GL_SHININESS, GL_AMBIENT_AND_DIFFUSE
 - **params** - definiše vrednost/boju dodeljenu komponentama

<https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glMaterial.xml>

Materijali u OpenGL - Color Tracking

- Color Tracking mehanizam omogućuje da boja određene komponente materijala bude određena sa glColor() pozivom

void glColorMaterial(GLenum face, GLenum mode)

- face - GL_FRONT, GL_BACK, GL_FRONT_AND_BACK
- mode - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE

- Olakšava definisanje materijala objekata (na nivou verteksa) u odnosu na glMaterial() metodu