

## בפרקים הקודמים...

- סטנדרט שמגדיר API להתמשקות עם מערכות גרפיקה
- אבסטרקציה מחומרה ומערכת הפעלה
- סינטקס: `glFunction2f(float a, float b);`
- OpenGL כמכונת מצבים
- צבע נוכחי, מטריצה נוכחית, האם לבצע תאורה?
- Per Vertex Operations - בציור של נקודה נדגמים המשתנים
- צבע נוכחי, נורמל, תאורה, טקסטורה
- ציור בסיסי - `glBegin() ... glEnd()`
- Back Face Culling
- טרנספורמציות `GL_MODELVIEW`, `GL_PROJECTION`
- Z-Buffer

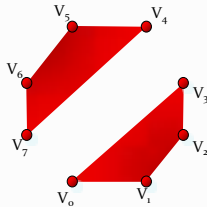


## glBegin(), glEnd()

`glBegin(GL_QUADS)`

V<sub>0</sub>  
V<sub>1</sub>  
V<sub>2</sub>  
V<sub>3</sub>  
V<sub>4</sub>  
V<sub>5</sub>  
V<sub>6</sub>  
V<sub>7</sub>

מרובעים בין כל 4 נקודות עוקבות



`glEnd()`

איך נקבע הצבע של הפוליגון?

## והיום...

- צבע ותאורה
- Display Lists

## Shading Models

`glShadeModel(GL_FLAT);`

`glBegin(GL_QUADS);`

`glColor3f(1.0,0.0,0.0);`

`glVertex3f(0.0,0.0,0.0);`

`glColor3f(0.0,1.0,0.0);`

`glVertex3f(1.0,0.0,0.0);`

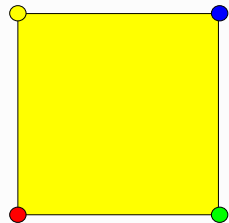
`glColor3f(0.0,0.0,1.0);`

`glVertex3f(1.0,1.0,0.0);`

`glColor3f(1.0,1.0,0.0);`

`glVertex3f(0.0,1.0,0.0);`

`glEnd();`



איזה נקודה נבחרת?

## צבעים וחומרים

כאשר לא משתמשים בתאורה

`glDisable(GL_LIGHTING);`

הצבע שמקבל כל Vertex הוא הצבע הנוכחי.

`glColor3f(float r, float g, float b);`

`glColor4f(float r, float g, float b, float a);`



כל ערך בין 0.0 ל 1.0

**r** - ערך האדום

**g** - ערך הירוק

**b** - ערך הכחול

**a** - ערך השקיפות (ברירת מחול = 1.0)

## צבעים וחומרים

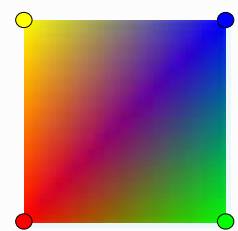
ללא תאורה



## Shading Models

```

glShadeModel (GL_SMOOTH) ;
glBegin (GL_QUADS) ;
glColor3f (1.0,0.0,0.0) ;
glVertex3f (0.0,0.0,0.0) ;
glColor3f (0.0,1.0,0.0) ;
glVertex3f (1.0,0.0,0.0) ;
glColor3f (0.0,0.0,1.0) ;
glVertex3f (1.0,1.0,0.0) ;
glColor3f (1.0,1.0,0.0) ;
glVertex3f (0.0,1.0,0.0) ;
glEnd () ;
    
```

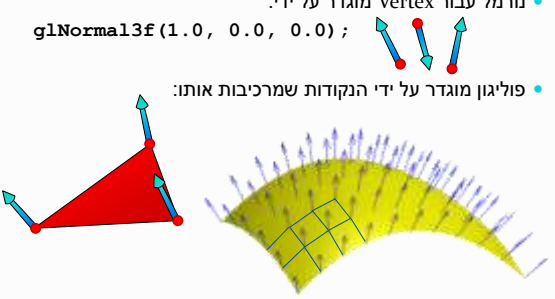


## צבעים וחומרים

הדבר הראשון שצריך בשביל תאורה הוא נורמל לכל פוליגון

- נורמל עבור Vertex מוגדר על ידי:
 

```
glNormal3f (1.0, 0.0, 0.0) ;
```
- פוליגון מוגדר על ידי הנקודות שמרכיבות אותו:



## צבעים וחומרים

עם תאורה



## נורמלים מנורמלים

- אורכו של נורמל חייב להיות 1.0
- אחרת...
- אפשר להגיד ל OpenGL לדאוג לזה בעצמו:
 

```
glEnable (GL_NORMALIZE)
```
- מתי להשתמש בזה?
- מתי לא להשתמש בזה?
- מה קורה ב Scaling ?



## נורמלים וכיווניות פוליגון

נתקלנו בעבר בסוגיית כיווניות הפוליגונים.. Front, Back facing



ההחלטה האם פוליגון הוא Front או Back Facing

**איננה קשורה** לנורמלים של הפוליגון

- נורמל מוגדר עבור Vertex בודד
- כיווניות הפוליגון מוגדרת עבור הפוליגון אחרי שהוא כבר הוגדר
- בזמן הגדרת ה Vertex הראשון של הפוליגון, לא ידועה הכיווניות אבל כן מוגדר הנורמל.

## פרמטרי תאורה

כל פרמטרי התאורה נשלטים על ידי פונקציה אחת:

```
glLightf(light, pname, param)
glLightfv(light, pname, param[])
```

- `light` - איזה מקור לשנות `GL_LIGHT0..GL_LIGHT7`
- `pname` - Paramere name - איזה פרמטר רוצים לשנות
- `param`
- ערך הפרמטר (מספר יחיד)
- מערך עם ערכי הפרמטר (צבע, ווקטור, נקודה...) - 4 ערכים

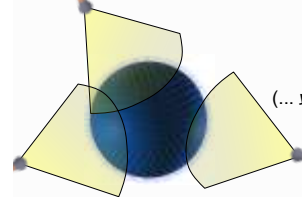
## תאורה

התחלת עבודה עם תאורה:

```
glEnable(GL_LIGHTING);
```

אפשר מקור אור מספר #:

```
glEnable(GL_LIGHT#);
(GL_LIGHT7 עד GL_LIGHT0)
```



- 8 מקורות או אפשריים (`GL_LIGHT7` עד `GL_LIGHT0`)
- כל מקור אור:
- מאופשר או לא מאופשר
- ממוקם בנקודה מסוימת בחלל
- זוכר את הפרמטרים שלו (צבע ...)

## פרמטרי תאורה

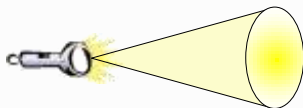
של **Attenuation**  
**Point Light**

- `GL_CONSTANT_ATTENUATION`
- `GL_LINEAR_ATTENUATION`
- `GL_QUADRATIC_ATTENUATION`

ברירת המחדל:  $(K_c, K_l, K_q) = (1, 0, 0)$

להפוך **Point Light**  
ל **Spot Light**

- `GL_SPOT_DIRECTION`
- `GL_SPOT_EXPONENT`
- `GL_SPOT_CUTOFF`



## פרמטרי תאורה

הצבעים של התאורה.  
כל אחד ב **RGBA**

- `GL_AMBIENT`
- `GL_DIFFUSE`
- `GL_SPECULAR`

`GL_POSITION` - מיקום / כיוון

```
float v[] = { 1.0f, 1.0f, 1.0f, 0.0f };
gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, v, 0);
```



**Directional Light** ←  $w=0.0$

$x, y, z$  - כיוון האור

**Point Light** ←  $w=1.0$

$x, y, z$  - מיקום האור

## צבעים וחומרים



הצבעים של החומר.  
כל אחד ב **RGBA**

- `GL_AMBIENT`
- `GL_DIFFUSE`
- `GL_SPECULAR`
- `GL_EMISSION`
- `GL_AMBIENT_AND_DIFFUSE`

`GL_SHININESS` - חזקת ה  $\cos()$  של specular - מספר

```
float v[] = {1.0f, 0.0f, 0.0f, 1.0f};
gl.glMaterialfv(GL.GL_FRONT_AND_BACK,
GL.GL_AMBIENT_AND_DIFFUSE, v, 0);
```

## צבעים וחומרים

כאשר **משתמשים בתאורה**

הצבע שמקבל כל **Vertex** הוא **החומר הנוכחי**

```
glMaterial(face, pname, param)
glMaterial(face, pname, param[])
```

`face` - איזה פוליגונים משנים `GL_FRONT, GL_BACK`, מה שמשמשים בדרך כלל

`pname` - שם הפרמטר

`param` - ערך או מערך ערכים




## מיקום תאורה

- מיקום התאורה הוא יחסי למודל (GL\_MODELVIEW)
- נקבע כאשר קוראים ל `glLight(GL_POSITION)`

```

paintEvent()
{
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    glLight(GL_POSITION,0,0,1,1)
    glRotate(model_rotation)
    renderModel()
}

paintEvent()
{
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    glRotate(model_rotation)
    glLight(GL_POSITION,0,0,1,1)
    renderModel()
}
    
```



צללים? אין?

## חומרים - קיצור דרך

בעיה: בכל פעם שרוצים לשנות את צבע החומר יש לבצע קריאה מורכבת ל `glMaterial()` עם המערכים הדרושים. הפתרון: להשתמש בצבע הנוכחי `(glColor)`.

אפשר קיצור הדרך:

```

glEnable(GL_COLOR_MATERIAL)

// בחירה איזה חלק של החומר יושפע על ידי הצבע הנוכחי:
glColorMaterial(GL_FRONT_AND_BACK,
                GL_AMBIENT_AND_DIFFUSE)

// זה:
float v[] = {1.0f, 0.0f, 0.0f, 1.0f};
gl.glMaterialfv(GL.GL_FRONT_AND_BACK,
                GL.GL_AMBIENT_AND_DIFFUSE, v, 0);

// הופך לזה:
gl.glColor3f(1.0f, 0.0f, 0.0f, 1.0f);
    
```

## Display Lists

רשימות תצוגה הם שיטה להאצת התצוגה

```

paintEvent()
{
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    glTranslate(left wheel)
    renderWheel()
    glTranslate(right wheel)
    renderWheel()
    glTranslate(front wheel)
    glColor(red)
    renderWheel()
}

renderWheel()
{
    glColor(blue)
    glBegin(GL_QUADS)
    for(...) // render tire
        glVertex(...)
    glEnd()
    glColor(gray)
    glBegin(GL_TRIANGLE_STRIP)
    glVertex(center)
    for(...) // front
        glVertex(...)
    glEnd()
    glBegin(GL_TRIANGLE_STRIP)
    glVertex(center)
    for(...) // back
        glVertex(...)
    glEnd()
}
    
```



## סיכום תאורה

כאשר מופעלת תאורה. צבע של **Vertex** מושפע מ:

עבור כל מקור אור:

- היחס בין כיוון הנורמל לכיוון ממנו מגיעה אור המקור וכיוון העין
- פרמטרי התאורה
- Ambient, Diffuse, Specular Intensities
- סוג המקור: כיווני, נקודה, פנס
- החומר ממנו עשוי ה Vertex
- Ambient, Diffuse, Specular Reflections
- Emmission, shininess
- אפשר לקצר על ידי שימוש ב `GL_COLOR_MATERIAL`
- טקסטורה? צללים? **Shading and Shadows**

מה עם צבע הפוליגון?

## Display Lists - שימוש

- רשימת תצוגה מזהה על ידי מספר (אינדקס)
- אינדקסי הרשימות מוקצים על ידי הפונקציה:

```
int first = glGenLists(range)
```

ניתן להקצות מספר אינדקסים בו זמנית, מוחזר הערך הראשון (טווח רציף)

- התחלת פקודות של רשימה:

```
glNewList(index, mode)
// mode: GL_COMPILE, GL_COMPILE_AND_EXECUTE
```

- סיום פקודות הרשימה:

```
glEndList()
```

- מחיקת רשימת תצוגה:

```
glDeleteLists(first, range)
```

## Display Lists

רשימת תצוגה היא "הקלטה" של פקודות OpenGL

```

initOpenGL()
{
    m_wheelList = glGenLists(1)
    glNewList(m_wheelList, GL_COMPILE)
    renderWheel()
    glEndList()
}

paintEvent()
{
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    glTranslate(left wheel)
    glCallList(m_wheelList)
    glTranslate(right wheel)
    glCallList(m_wheelList)
    glTranslate(front wheel)
    glColor(red)
    glCallList(m_wheelList)
}
    
```

הקלטה

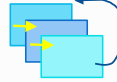
ניגון

## Display Lists

- רשימות התצוגה נשמרות ב Context של OpenGL. אם עוברים ל Context אחר, הם לא יעבדו.
- עם זאת, אפשר לשתף רשימות.. עם תמיכה של הסביבה.
- ישנן פקודות OpenGL שלא מוקלטות אלא מבוצעות באופן מיידי. `glReadPixels`, `glRenderMode`, `glSelectBuffer`, `glVertexPointer`, `glFeedbackBuffer`, `glFinish`, `glFlush`, `glIsEnabled`, `glFlush`, `glGet`
- הקוד שיוצר את רשימת התצוגה הוא קוד רגיל. עם `if`, `for`, `while` וכו'. מה שמוקלט זה רק הקריאות ל OpenGL, על פי הסדר שבוצעו, עם הערכים שהועברו.
- קריאות עם מערכים `glVertexfv (v)` - המערך מועתק ונשמר עם הרשימה.

## Display Lists

- הסיבה העיקרית לשימוש ב Display List - **שיפור ביצועים**
- הפקודות נשמרות במצב "מקומפל" ונשלחות בצורה אופטימלית לכרטיס המסך
- אי אפשר לשנות Display List. רק למחוק או לכתוב מחדש.
- כיוון רשימות - ניתן לקרוא ל רשימת תצוגה A בתוך הקלטה של רשימת תצוגה B.
- מה שיוקלט זה הקריאה ל `glCallList`, לא הפקודות עצמן של A וקורסיה?
- ה State של OpenGL לא מושפע מביצוע רשימות
- ה State לפני הקריאה נשאר בתוך הקריאה
- שינוי שבוצע בתוך הקריאה, ישרוד גם אחרי הקריאה.



## Vertex Buffer Objects

- השיטה "האולטימטיבית" לשיפור ביצועים
- מקצים זיכרון על גבי כרטיס המסך (buffer) ובו מאחסנים נתונים כגון קוארדינטות, חומרים, נורמלים וכו'
- בעת רינדור כל פריים (frame) כרטיס המסך ניגש ישירות למידע המקומי
- מדלגים על ציורו הבקבוק הגדול ביותר - העתקות זכרון מהזיכרון המרכזי לכרטיס המסך



## Vertex Arrays

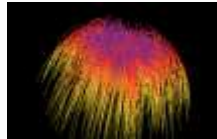
- שיטה נוספת לשיפור ביצועים - **ציור של מודל גדול**
- קריאות חוזרות ונשנות ל `glColor ()`, `glNormal ()`, `glVertex ()`
- במקום זה שומרים את כל המידע במערכים גדולים ונותנים לכרטיס מסך לרוץ עליו לבד.

```
float color_array[], vertices[]
initOpenGL() {
    glEnableClientState(...)
    glColorPointer(3, GL_FLOAT, 0, color_array)
    glVertexPointer(3, GL_FLOAT, 0, vertices)
}
paintEvent() {
    glDrawArrays(GL_QUADS, 0, size)
}
```

כל המידע נשלח בבת אחת לכרטיס המסך (יותר יעיל)

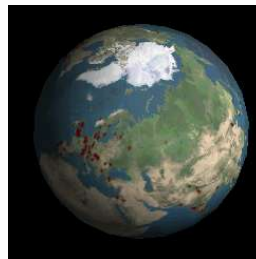
## טקסטורות

- הפעלת שימוש בטקסטורות
- `glEnable (GL_TEXTURE_2D)`
- `glEnable (GL_TEXTURE_1D)`
- `glEnable (GL_TEXTURE_1D)`



## טקסטורות

### הדבקת תמונה על אובייקט

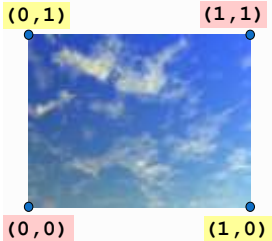


- פרמטריזציה
- קורדינטות טקסטורה
- שילוב Shading

## טקסטורות

• כאשר הטקסטורה כבר נמצאת בידי הכרטיס מסך, הרזולוציה המקורית שלה מאבדת את משמעות.

• התמונה ממופה ל"קורדינטות טקסטורה" (s,t) בין 0 ל 1.



• כל התייחסות עתידית לטקסטורה תהיה על פי קורדינטות אלה.

• `m_tex.bind()` בוחרת את הטקסטורה להיות הטקסטורה הנוכחית

## טקסטורות - JOGL

לפני שאפשר להשתמש בתמונה כטקסטורה, כרטיס המסך צריך להכיר אותה.

• JOGL מקל מאוד על יצירה ושימוש בטקסטורות  
`glTexImage2D, gluBuild2DMipmaps, glGenTextures, glBindTexture, glDeleteTextures`

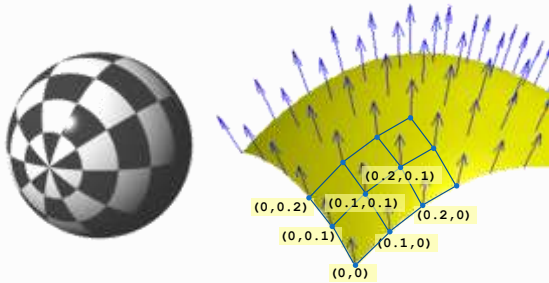
• יצירת טקסטורה בעזרת JOGL:

```
File f = new File("img.png");
Texture m_tex;
m_tex = TextureIO.newTexture(f, true);
```

גודל התמונה: אורך ורוחב חייבים להיות חזקה של  $2^m \times 2^m$ !  
 אחרי שלב זה אפשר כבר להשתמש בטקסטורה!

## קורדינטות טקסטורות

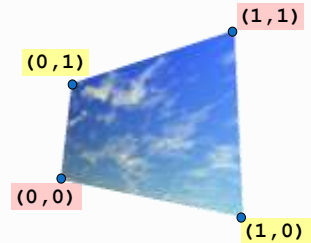
מפרמטריזציה לקורדינטות



## טקסטורות - שימוש

• בזמן ציור, עבור כל Vertex מספקים:  
 • צבע (חומר), נורמל, וכו'  
 • קורדינטות הטקסטורה גם הן פרמטר שניתן לספק עבור Vertex:

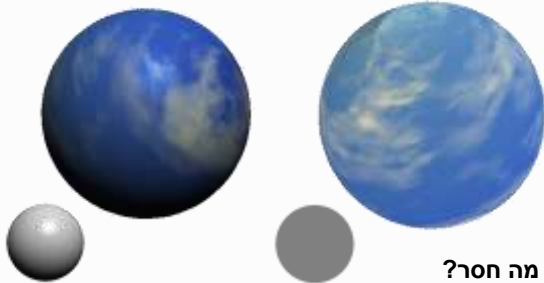
```
m_tex.bind()
glBegin(GL_QUADS)
glTexCoord2f(0,0)
glVertex3f(1,2,0)
glTexCoord2f(1,0)
glVertex3f(2,2,0)
glTexCoord2f(0,1)
glVertex3f(1,0,1)
glTexCoord2f(1,1)
glVertex3f(1,1,1)
glEnd()
```



## טקסטורות

תאורה, Shading

המצב עד כה:



מה חסר?

## Mip Maps



מאחורי הקלעים תהליך טעינת הטקסטורה יוצר לכל טקסטורה mip-maps

mip-map - סדרה של תמונות ברזולוציה הולכת וקטנה, החל מ  $2^m \times 2^m$  עד  $1 \times 1$  פיקסלים



אופטימיזציה של התהליך  
 • ממזער את מספר פעולות ה scaling  
 • מכניס רק כמה רזולוציה שצריך.

שקוף למשתמש ב JOGL!

```
Texture t = TextureIO.newTexture(f, true);
```

## טקסטורות

עוד פרמטרים...

```
glTexParameterf(GL_TEXTURE_2D, pname, param)
    • pname - איזה פרמטר לשנות
    • param - לאיזה ערך לשנות

pname = GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T
```



GL\_REPEAT



GL\_CLAMP

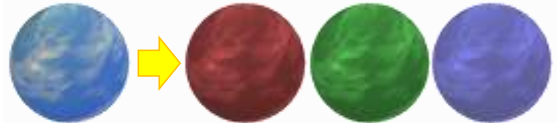
## טקסטורות

• הוספת shading לטקסטורות:

```
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glLight(...)
```

```
glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE, GL_MODULATE)
```

• `glTexEnvf` - שליטה על פרמטרי סביבה... עוד שימושים?  
• עירוב עוד צבע..



## GLU Quadrics

• אתחול:

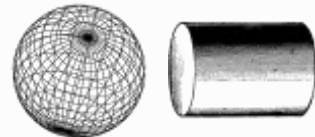
```
GLU glu = new GLU();
GLUQuadric q = glu.gluNewQuadric();
קביעת פרמטרים - איך ליצור את האובייקטים:
    • האם ליצור נורמל עבור כל Vertex?
    glu.gluQuadricNormals(q, GLU_FLAT); //GLU_SMOOTH
    • האם ליצור קורדינטות טקסטורה? (אוטומטית!)
    glu.gluQuadricTexture(q, true);
    • באיזה סגנון לצייר את האובייקט?
    glu.gluQuadricDrawStyle(q, GLU_FILL);
```

## GLU

OpenGL היא ספריית העזר ל OpenGL

• ממוש "מעל" OpenGL, לא ישירות בחומרה. מבצע קריאות OpenGL.  
• פרוצדורות נפוצות, דרכים קלות יותר לעשות דברים.  
`gluLookAt()`, `gluPerspective()`, `gluOrtho2D()`

• GLU Quadrics - פונקציות שמציירות אובייקטים נפוצים.



## GLSL

רוצים לתכנת מחדש את הכרטיס מסך לעשות דברים שהוא לא עושה בדרך כלל

- Rendering מיוחד ומזר
- חישוב צללים
- דפורמציות



- אפשרות 1: לכתוב קוד ישירות לכרטיס מסך
- לכתוב באסמבלי ספציפי לכרטיס ול Vendor...

• אפשרות 2: לכתוב ב GLSL!

- `glCompileShader()` שפת תכנות דמוית C
- `glUseProgram()` מקומפלת בדרייבר
- תוכנה רצה על הכרטיס מסך

## GLU Quadrics

• ציור:

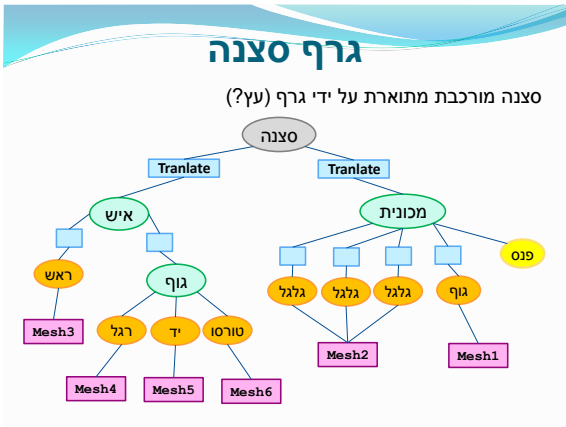
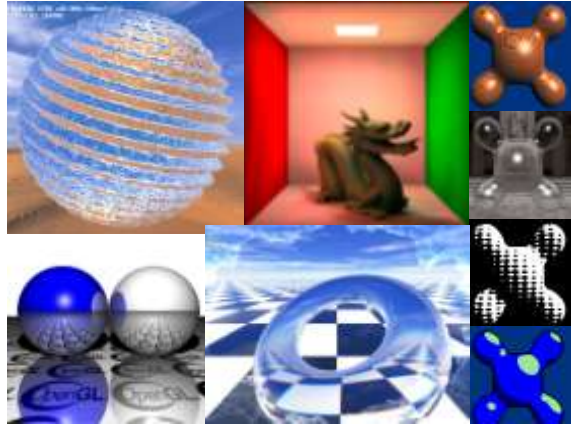
```
glu.gluSphere(q, radius, slices, stacks)
glu.gluCylinder(q, r1, r2, h, slices, stacks)
glu.gluDisk(q, ...)
glu.gluPartialDisk(q, ...)
```

• מה קורה בקריאה ל `gluSphere`?

```
glBegin(GL_QUADS)
glNormal(...); glTexCoord(...);
glVertex(...);
glNormal(...); glTexCoord(...);
glVertex(...);
glNormal(...); glTexCoord(...);
glVertex(...);
glEnd()
```

```
GLU glu = new GLU();
GLUQuadric q =
    glu.gluNewQuadric();
glu.gluQuadricTexture(q,
    true);
glu.gluQuadricNormals(q,
    GLU_FLAT);
m_text.bind();
glu.gluSphere(q,
    2.0, 40, 40);
```

# בניית אפליקציה



## מודלים

- כשם שישנם פרמטרים רבים לקבצי תמונות, ישנם פרמטרים רבים לקבצי מודלים תלת מימדיים.
- שפות הגדרת סצנה: VRML, X3D, DWF, U3D, 3DS
- סצנה מורכבת
- מספר רב של אובייקטים, גופי תאורה
- טרנספורמציות
- מבנה היררכי

## מודלים

מבנה טיפוסים של קובץ מודל:

- קובץ טקסטואלי
- רשימה של Vertex
- רשימה של פוליגונים
- שלושות או רביעיות של אינדקסי Vertex

```

OFF
8 6 0
-0.500000 -0.500000 0.500000
0.500000 -0.500000 0.500000
-0.500000 0.500000 0.500000
0.500000 0.500000 0.500000
-0.500000 0.500000 -0.500000
0.500000 0.500000 -0.500000
-0.500000 -0.500000 -0.500000
0.500000 -0.500000 -0.500000
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
    
```

קבצי OBJ - קצת יותר מורכבים

## מודלים

פורמט נוסף של קובץ לתיאור תלת מימדי:

- תיאור Mesh פשוט: OFF, OBJ, PLY, PLY2, STL
- בדרך כלל אובייקט אחד
- ללא טרנספורמציות, או תאורה
- תיאור של Vertexים ופוליגונים
- נורמלים, קוארדינטות טקסטורה.
- חומרים? טקסטורות?



## Model-View-Controller

**Controller** - אתחול, ניהול קלט/פלט, ניתוב הודעות...

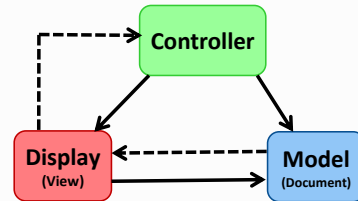
- מממשת את main()
- בונה את ה GUI
- מקבלת אירועים מהמשתמש - עכבר, מקלדת
- מנתב אירועים למחלקות המתאימות
- גורם לשינויים במודל
- גורם לשינויים בתצוגה

Main.java •

## Model-View-Controller

GUI Design Pattern - MVC לתכנון של אפליקציות

הפרדה בין המידע, התצוגה של המידע ונתיבי השליטה



## Model-View-Controller

**Display** - מציג את המודל למשתמש

- דיבור עם OpenGL
- רינדור מודלים, הפעלת טרנספורמציות
- נקודת מבט המשתמש - לא מושפע מהמודל
- לטובב את המודל או לטובב את ה Projection?
- יכולים להיות כמה - אספקטים שונים של המודל
- מקבל עידכונים מהמודל ומה Controller

Renderer.java •

## Model-View-Controller

**Model** - שומר את כל המידע הרלוונטי לגבי מה שקורה

באפליקציה, מה שעתידי לקרות, מה שקרה

- גרף הסצנה
- מודלים, mesh
- מצב העולם
- איפה נמצא המשתמש עכשיו?
- איפה נמצאים דברים אחרים?

Player.java •

Enemy.java •

World.java, Board.java •

## מתכון שימוש ב OpenGL

שינוי גודל חלון:

- אתחול מטריצת ה Projection, חישוב מחדש.

```

void resizeEvent()
{
    float fAspect = (float)width / (float)height;
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, fAspect, 0.5f, 400.0f);
    glMatrixMode(GL_MODELVIEW);
}
  
```

## מתכון שימוש ב OpenGL

אתחול

- אתחול משתני state שלא הולכים לשתנות הרבה או בכלל.

```

void initOpenGL()
{
    glEnable(GL_DEPTH_TEST);
    glClearColor(1.0);
    glClearDepth(1.0);
    glClearColor(0.8f, 0.2f, 1.0f, 1.0f);
    glEnable(GL_TEXTURE_2D);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    float v[] = { 0.2f, 0.8f, 1.0f, 0.0f };
    float s[] = { 0.8f, 0.8f, 0.8f, 0.0f };
    glLightfv(GL_LIGHT0, GL_POSITION, v, 0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, s, 0);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
    ...
}
  
```

## מתכון שימוש ב OpenGL

משתמש שמסובב את המודל על המסך

```
void mouseMoveEvent(event e)
{
    if (!e.pressedButton())
        return;
    float dx = e.pos.x - m_lastPos.x;
    float dy = e.pos.y - m_lastPos.y;
    glRotated(dx, 0, 1, 0);
    glRotated(dy, 1, 0, 0);

    m_lastPos = e.pos;
    canvas.repaint()
}
```

אנימציה כללית?

## מתכון שימוש ב OpenGL

ציור:

בכל Frame מציירים את כל הסצנה מחדש!  
המטריצה הנוכחית היא תמיד `GL_MODELVIEW`

```
void paintEvent() // paint a single frame.
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity(); // GL_MODELVIEW
    glRotate(...);
    glTranslate(...);
    glColor3f(1.0, 0.0, 0.0);
    renderModel1();

    glTranslate(...);
    renderModel2();
    context.swapBuffers()
}
```



## עוד לינקים

<https://jogl-demos.dev.java.net/>