

# JOGL

JOGL זה מעטפת JNI ל-OpenGL. אנחנו נעבוד עם גירסה 2 של OpenGL.

## אובייקטים בסיסיים

- GLCanvas מתפקד כמו Canvas - משטח עליו ניתן לצייר.
- GLJPanel - מתפקד כמו JPanel. איטי יחסית.
- GLEventListener - מנגנון callback.
- Animator - הלולאה הראשית שכל הזמן קוראת ל-paint.

## מבנה תוכנית

1. יוצרים Animator.
  2. מוסיפים אירוע סגירה שקורא ל-stop() על animator.
  3. יוצרים GLCanvas.
  4. מוסיפים listener canvas ל-GLEventListener.
  5. מוסיפים את canvas ל-frame.
  6. animator.start();
  7. חשוב לעשות canvas.requestFocus(); - אחרת המקשים לא יעבדו (כי החלון לא יהיה בפקוס)
- ל-event listener יש כל מיני מתודות - החשובה שבהם היא display, שבה צריך לעשות:
- `GL2 gl = drawable.getGL().getGL2();` - כדי לקבל את אובייקט הציור המתאים.
  - `gl.glClear (...);` - לנקות את המסך (מהאובייקטים הקודמים)
  - `gl.glLoadIdentity();` - לאפס את מטריצת הטרנספורמציות.
  - `gl.glTranslatef(x, y, z);` - להזיז את מטריצת הטרנספורמציות.
  - `gl.glRotatef(theta, rotationAxisX, rotationAxisY, rotationAxisZ);` - לסובב את מטריצת הטרנספורמציות.
  - כדי לצייר:
1. `gl.glBegin(GL2.GL_TRIANGLES);` - כדי לצייר משולשים.
  2. `gl.glColor3f(r, g, b);` - כדי לקבוע את הצבע של הקודקוד.
  3. `gl.glVertex3f(x, y, z);` - כדי לצייר את הקודקוד.
- נשים ♡: הקואורדינטות יסובבו לפי מטריצת הטרנספורמציות.
4. חוזרים על שלבים 2 ו-3 עוד פעמיים (סה"כ 3 פעמים) כדי לצייר משולש.
  5. חוזרים על שלבים 2-4 כדי לצייר כמה משולשים שרוצים.
  6. `gl.glEnd();` - כדי לסיים את ציור המשולשים.
- עוד מתודה חשובה היא reshape, שנקראת כאשר גודל החלון משתנה. שם צריך להגדיר את projection:
- `gl.glMatrixMode(GL2.GL_PROJECTION);` - מגדירים שאנחנו עובדים עכשיו על מטריצת projection.

- `gl.glLoadIdentity();` - מאפסים את מטריצת ה-projection.
- `gl.glFrustum (...);` - הגדרת ה-clipping.
- `gl.glMatrixMode(GL2.GL_MODELVIEW);` - חוזרים לעבוד על מטריצת הטרנספורמציות.

## ציור קווים

עם `gl.glBegin(GL.GL_LINES);` אפשר לצייר קווים - בין כל שני קודקודים הוא יצייר קו. אפשר להשתמש ב-`gl.glVertex2f(x, y);` כדי לצייר את הקודקודים בדו-מימד.

## שינוי נקודת הייחוס כדי לצייר אובייקטים מסויימים

אפשר להשתמש בפקודה `gl.glPushMatrix();` כדי "לשנות את נקודת הייחוס". עושים את זה כדי לזוז למקום המתאים כשרוצים לצייר אובייקט לפי נקודת ייחוס מסויימת. אחרי זה אפשר לעשות `gl.glTranslatef (...);`, `gl.glScalef (...);` ו-`gl.glRotatef (...);` כדי לבצע טרנספורמציות על אותו אובייקט. בסיום הציור של האובייקט עושים `gl.glPopMatrix();` כדי לחזור לנקודת הייחוס הקודמת.

## Display Lists

`gl.glCallList ( list );` מאפשר לצייר אובייקט שמור (לפי מטריצת הטרנספורמציות הנוכחית). כדי ליצר אובייקט כזה:

1. `list = gl.glGenerateList(1);` - מייצר אובייקט אחד. אפשר לבקש כמה אובייקטים רציפים - למשל `list = gl.glGenerateList(2);` ייצר לנו שני אובייקטים עם אינדקסים `list + 1`.

2. `gl.glNewList(list, GL2.GL_COMPILE);` - מתחיל ליצר את האובייקט. `GL_COMPILE` אומר לקמפל את האובייקט כדי שיהיה אפשר לצייר אותו בצורה יותר מהירה.

3. מתחילים לצייר כרגיל עם `gl.glBegin (...)`.

## Quad Strip

```
gl.glBegin(GL2.GL_QUAD_STRIP);
```

```
quad מקבל בכל פעם ארבעה קודקודים ומצייר אותם
```

```
quad strip מקבל בכל פעם 2 קודקודים ומצייר עוד ריבוע משני הקודקודים האחרונים.
```

אם למשל יש לנו כדור, אנחנו יכולים לחלק אותו לרצועות שכל אחת מקיפה את הכדור. יוצרים quad strip ועוברים על כל רצועה עם קואורדינטות פולריות:

```
for (a = 0f, b <= 360f, b += db) {  
    if (color > 0) {  
        gl.glColor3f(1f, 0f, 0f);  
    } else {  
        gl.glColor3f(1f, 1f, 1f);  
    }  
  
    x = radius * COS(b) * COS(a);  
    y = radius * SIN(b) * COS(a);  
    z = radius * SIN(a);  
    gl.glVertex3f(x, y, z);  
  
    x = radius * COS(b) * COS(a + da);  
    y = radius * SIN(b) * COS(a + da);  
    z = radius * SIN(a + da);  
    gl.glVertex3f(x, y, z);  
  
    color = 1 - color;  
}
```