OpenGL זה...

לא:
- שפת תכנות
- ספריית תוכנות
- ממשק חלונאות

```
glBegin(GL_POLYGON);
  glColor3f(0, 1, 0);
  glVertex3f(-1, -1, 0);
  glVertex3f(-1, 1, 0);
  glVertex3f(1, 1, 0);
  glVertex3f(1, -1, 0);
glEnd();
```

כן:
- הגדרה של ממשק, מסמך.
- Application Programming Interface
- סטנדרט - Cross Language, Cross Platform
- מגדיר בערך 250 פקודות
- "האסמבלי של גרפיקה ממוחשבת"

OpenGL זה...

- **Open** - סטנדרט פתוח, כל אחד יכול לממש.
- **GL** - Graphics Library
- ספריות נלוות:
  - **GLU** - OpenGL Utility - ספריית עזר סטנדרטית.
  - **GLUT** - OpenGL Utility Toolkit - מימוש בסיסי של GUI, תפריטים, חלונות, קלט מעכבר ומקלדת (crossplatform!)
  - **לכל מערכת הפעלה**
    - **WGL** - ספריית עזר לחלונות
    - **GLX** - ספריית עזר ליוניקס, לינוקס
    - **AGL, CGL** - ספריית עזר למקינטוש
    - **JOGL, LWJGL** - פורט ל Java

למה זה טוב?

- גרפיקה ב"דוס"

```
asm
   mov ax, 0013h
   int 10h
end;
```

Mode 13h (Mode X)

כתיבה ישירה ל **0xA0000** (VGA Block)

תלת מימד? אין, ניתן לממש בתוכנה

חבילות תוכנה proprietary (graphics של פסקל?)

אבסטרקציה מהחומרה וממערכת ההפעלה

לא חלק מ OpenGL:

יצירת חלונות

אינטראקציה עם המשתמש (...?)

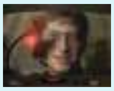
OpenGL - מימושים

- ATI, nVidia, Matrox, 3dfx (RIP)
- **opengl32.dll**
- **libGL.so**

Stubים לחומרה

- Microsoft - opengl32.dll
- התקנת default
- Windows XP SP2

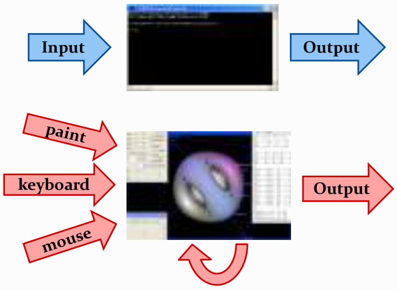
מימוש בתוכנה



- Linux
- libGL.so – mesa 3D - מימוש חומרה/תוכנה

Event Driven Programming

OpenGL is a state machine on top of which we program
(עוד בהמשך)



הסינטקס

glFunction2f(float a, float b);

```
glVertex3f(1.0f, 2.0f, 3.0f);
glVertex3i(1,1,1);
glRotated(90.0, 0.0, 0.0, 1.0);
```

glFunction2fv(float *v);

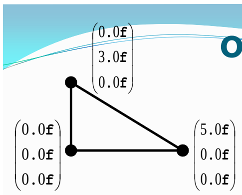
```
float v[] = {1.0f, 2.0f, 3.0f};
glVertex3fv(v);
```

וואריאציות עם int:
glFunc2i(int a, int b)
לא מומלץ להשתמש.

GL_CONSTANT

```
glEnable(GL_DEPTH_TEST);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

הסינטקס



- קורדינטות (...) glVertex
  - בדרך כלל float
  - אין הגבלה על ערכים

- צבע (...) glColor
  - ערכי Red, Green, Blue
  - בין 0.0 ל 1.0
  - בדרך כלל float

1.0f, 1.0f, 1.0f
1.0f, 0.0f, 0.0f
0.0f, 1.0f, 0.0f
0.0f, 0.0f, 1.0f
0.0f, 0.0f, 0.0f

- קבועים - uint, 16 או 32 ביט.

```
#define GL_DEPTH_BUFFER_BIT     0x00000100
#define GL_COLOR_BUFFER_BIT     0x00004000
#define GL_DEPTH_TEST           0x0B71
```

התחלת עבודה

יצירת פרוייקט SWT, שימוש ב JOGL

```
final Display display = new Display();
Shell shell = new Shell(display);
final GLCanvas canvas = new GLCanvas(comp, SWT.NONE, data);
final GLContext context =
  GLDrawableFactory.getFactory().createExternalGLContext();

GL gl = context.getGL();

gl.glEnable(GL.GL_DEPTH_TEST);
gl.glColor3f(1.0f, 0.0f, 0.0f);
gl.glNormal3f(...);
gl.glVertex3f(...);
...
```

OpenGL כמכונת מצבים

- OpenGL היא מכונת מצבים.
- ה"מצב" בה נמצאת המכונה נשמר ב **GLContext**
- מה כולל ה"מצב" של מכונת המצבים?
  - צבע נוכחי בו מציירים
  - מטריצות Model, Projection
  - שיטת הציור
    - פוליגונים, קווים, קווים+פוליגונים
  - תאורה
    - איפה יש גופי תאורה, מה המצב שלהם
  - טקסטורות
  - ועוד ועוד...
- מכונת המצבים מתחילה ממצב default

למה זה טוב?

- ניהול מספר חלונות בו זמנית
- "בערך" OOP
- המשתמש לא צריך לזכור שום דבר
- אינטראקציה מהירה עם החומרה

OpenGL כמכונת מצבים

שליטה על מכונת המצבים
משתנים ספציפיים

```
glColor3f()    –  צבע נוכחי
glNormal3f()   –  נורמל נוכחי
glMatrixMode() –  באיזה מטריצה שולטים
glLineWidth()  –  עובי קו
glLightf()     –  פרמטרי תאורה
glMaterial()   –  החומר הנוכחי איתו עובדים
...
```

אפשר לקבל בחזרה את הערכים מהמכונה

glGet...(...)
GL_CURRENT_COLOR, GL_CURRENT_NORMAL, GL_MATRIX_MODE, GL_LINE_WIDTH, GL_LIGHT1 ...

קריאה יקרה! לא להשתמש באופן קבוע

OpenGL כמכונת מצבים

מצב בינארי - מודלק או מכובה

glEnable(), glDisable()

GL_DEPTH_TEST – האם ה"זִי באפר" מופעל
GL_LIGHTING – האם לבצע תאורה?
GL_LIGHT1 – האם מופעלת תאורה מס' 1
GL_CULL_FACE
...

לקבל את המצב הנוכחי מהמכונה:

glIsEnabled()

דוגמא

```
// set the current color to blue.
glColor3f(0.0f, 0.0f, 1.0f);
// render the cow.
renderCow();
// set the current color to green.
glColor3f(0.0f, 1.0f, 0.0f);
// render ground.
renderGround();
```

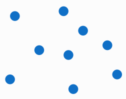
Per Vertex Operations

- **Vertex - קודקוד, נקודה.**
- הנקודה היא **הפרימיטיב** הבסיסי ביותר והחשוב ביותר.
- רוב הפעולות מתבצעות ונשמרות עבור נקודה.
- קו - מוגדר <u>באופן מלא</u> על ידי 2 נקודות
  - צבע, עובי, קיווקו
  - תכונות הקו מוגדרות על ידי תכונות הנקודות
- פוליגון - מוגדר <u>באופן מלא</u> על ידי 3 נקודות ומעלה
  - צבע, טקסטורה, נורמל
  - תכונות הפוליגון מוגדרות על ידי תכנות הנקודות

$$v = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

נקודות

- עוד שם: Vertex
- ציור נקודה <u>**דורש את פרמטרי הציור**</u>
  - צבע, חומר
  - טקסטורה
  - נורמל
- גודל מינימאלי של נקודה הוא פיקסל
- יכול להיות גם גדול יותר - `glPointSize()`
- כמה נקודות יכולות להיות קיימות באותה נקודה במרחב
- כמה נקודות יכולות להיות מצוירות באותו פיקסל
  - גם אם הן לא באותה נקודה במרחב.



```
glVertex3f(1.0, 0.0, 1.0);
```

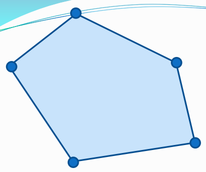
קווים

- קו מתוח בין שתי נקודות

- `glLineStipple()` - מגדיר קווקו.
- `glLineWidth()` - עובי הקו

פוליגון

- **שטח סגור בין כמה נקודות**

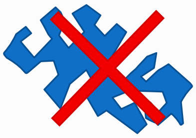


- `glPolygonMode()` **-** איך לצייר את הפוליגון
  - רק קווים, רק נקודות, מלא בצבע
  - Front, Back ?

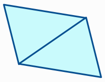
פוליגון

מגבלות:

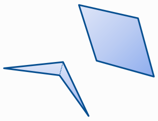
- **חייב להיות קמורים**



- אסור לגבולות 2 פוליגונים להיחתך (...)



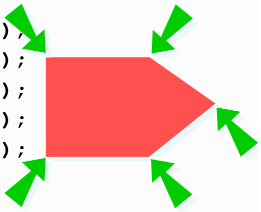
- צריכים להיות באותו מישור (אחרת...)



ציור - הלכה למעשה

- `glBegin()` - התחל ציור
- `glEnd()` - סיים ציור

```
glBegin(GL_POLYGON);
  glVertex2f(0.0, 0.0);
  glVertex2f(4.0, 0.0);
  glVertex2f(6.0, 1.5);
  glVertex2f(4.0, 3.0);
  glVertex2f(0.0, 3.0);
glEnd();
```



ציור - הלכה למעשה

- `glBegin()` – מסמן את התחלת הגדרת נקודות הציור.
- `glEnd()` – מסמן את סיום ההגדרה

- `glVertex()` - מותר להשתמש רק בין `glBegin()` ל `glEnd()`
  - לא עושה כלום בכל מקום אחר.

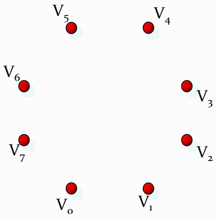
- עוד פונקציות שמותר להשתמש בתוך בלוק הציור (אבל לא רק שם)
  - `glColor()`
  - `glNormal()`
  - `glTexCoord()`
  - `glMaterial()`

- בלוקי `glBegin()` - `glEnd()` הם מה שמבוצע בתוך קוד ה paint Event.

glBegin() , glEnd()

```
glBegin(GL_POINTS)
  V0
  V1
  V2
  V3
  V4
  V5
  V6
  V7
glEnd()
```

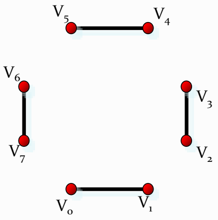
צייר נקודות



glBegin() , glEnd()

```
glBegin(GL_LINES)
  V0
  V1
  V2
  V3
  V4
  V5
  V6
  V7
glEnd()
```

צייר קווים בודדים בין נקודה אי-זוגית לנקודה זוגית

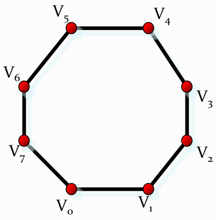


glBegin() , glEnd()

```
glBegin(GL_LINE_LOOP)
  V0
  V1
  V2
  V3
  V4
  V5
  V6
  V7
glEnd()
```

צייר קווים בין כל זוג נקודות שכנות

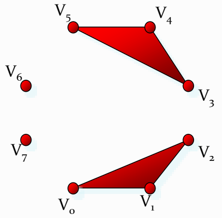
הבדל מפוליגון?



glBegin(), glEnd()

```
glBegin(GL_TRIANGLES)
  V0
  V1
  V2
  V3
  V4
  V5
  V6
  V7
glEnd()
```

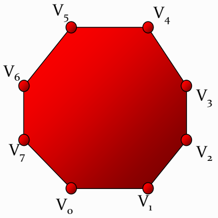
משולשים בין
נקודות כל שלשה
בודדת



glBegin(), glEnd()

```
glBegin(GL_POLYGON)
  V0
  V1
  V2
  V3
  V4
  V5
  V6
  V7
glEnd()
```

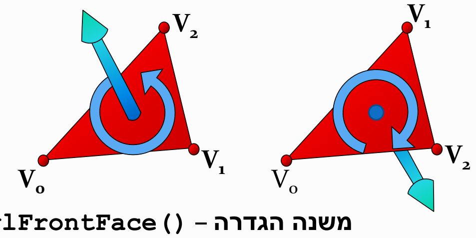
צייר פוליגון



Front, Back Facing Polygon

אלא אם מוגדר אחרת:

- נגד כיוון השעון (CCW) ⟺ פוליגון פונה קדימה (front)
- עם כיוון השעון (CW) ⟺ פוליגון פונה אחורה (back)



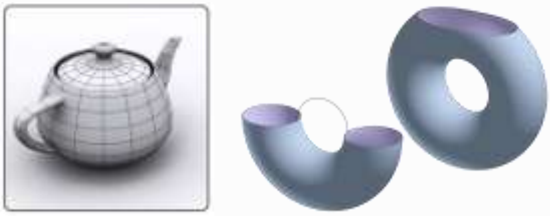
משנה הגדרה – `glFrontFace()`

Back Face Culling

פוליגונים שמכוונים אחורנית לא מרונדרים

```
glEnable(GL_CULL_FACE);
glCullFace(...)
```

מתי להשתמש? מתי לא להשתמש?



טרנספורמציות

- הדרך להזיז ולשנות את האלמנטים בחלל היא להפעיל עליהם טרנספורמציה.
- ניזכר: נקודה וקטור של 4 מספרים
- טרנספורמציה היא מטריצה 4x4 שמוכפלת בוקטור.

$$v = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \quad T(v) = Av = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix}$$

טרנספורמציות

$$\begin{pmatrix}1&0&0&dx\\0&1&0&dy\\0&0&1&dz\\0&0&0&1\end{pmatrix}$$

`glTranslate(dx,dy,dz)`

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

`glLoadIdentity()`

$$\begin{pmatrix}sx&0&0&0\\0&sy&0&0\\0&0&sz&0\\0&0&0&1\end{pmatrix}$$

`glScale(sx,sy,sz)`

טרנספורמציות

$$\begin{pmatrix}1&0&0&0\\0&\cos\alpha&-\sin\alpha&0\\0&\sin\alpha&\cos\alpha&0\\0&0&0&1\end{pmatrix}$$

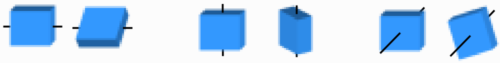
`glRotate(α,1,0,0)` - X

$$\begin{pmatrix}\cos\alpha&0&\sin\alpha&0\\0&1&0&0\\-\sin\alpha&0&\cos\alpha&0\\0&0&0&1\end{pmatrix}$$

`glRotate(α,0,1,0)` - Y

$$\begin{pmatrix}\cos\alpha&-\sin\alpha&0&0\\\sin\alpha&\cos\alpha&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

`glRotate(α,0,0,1)` - Z



טרנספורמציות

"המטריצה הנוכחית" - חלק מהמצב (State) של OpenGL.

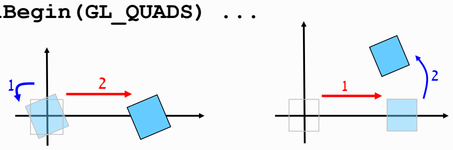
בכל זמן נתון יש מטריצת טרנספורמציה בזכרון.

- `glLoadIdentity()` - העבר למטריצה הנוכחית את טרנספורמציית היחידה.
- `glTranslate()` - הכפל את המטריצה הנוכחית בטרנספורמציית הזזה: מרחק ב3 מימדים
- `glScale()` - הכפל את המטריצה הנוכחית בטרנספורמציית מתיחה - ב 3 מימדים, סביב הראשית
- `glRotate()` - הכפל את המטריצה הנוכחית בטרנספורמציית סיבוב - זווית סביב ציר (ווקטור)

שימוש בטרנספורמציות

סדר הפעלת הטרנספורמציות משנה!

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotate(60,0,0,1);          glTranslate(1,0,0);
glTranslate(1,0,0);          glRotate(60,0,0,1);
glBegin(GL_QUADS) ...
```



טרנספורמציות

מה היא ה"מטריצה נוכחית" ?

יש כמה מטריצות נוכחיות ששמורות כל הזמן על ידי OpenGL:

- `GL_MODELVIEW` - משנה, מזיזה, מסובבת את המקום בו הולכים לצייר (אחרי השינוי)
- `GL_PROJECTION` - משנה, מזיזה, מסובבת את המצלמה!

איך עוברים ביניהן?

`glMatrixMode(...);`

- פעולות מטריצה שיבואו אחרי קריאה זו ישנו את המטריצה שנבחרה.
- בכל זמן נתון אפשר לשנות רק את אחת מהמטריצות.

טרנספורמציות

פונקציות מתקדמות

- `glLoadMatrix()` - החלף את המטריצה הנוכחית במטריצה נתונה. הכללה של `glLoadIdentity()`.
- `glMultMatrix()` - הכפל את המטריצה הנוכחית במטריצה נתונה. הכללה של `glTranslate()`, `glScale()`, `glRotate()`

$$\begin{bmatrix}m_1&m_5&m_9&m_{13}\\m_2&m_6&m_{10}&m_{14}\\m_3&m_7&m_{11}&m_{15}\\m_4&m_8&m_{12}&m_{16}\end{bmatrix}$$

למה זה שימושי?

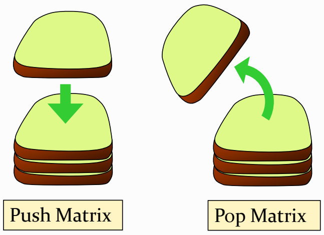
טרנספורמציות

OpenGL מספקת "מחסנית" של מטריצות.

`glPushMatrix()` - דחוף למחסנית עותק של המטריצה הנוכחית.

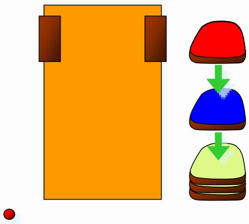
`glPopMatrix()` - הוצא מהמחסנית את המטריצה העליונה והחלף את המטריצה הנוכחית בה.

למה זה טוב?



Push / Pop Matrix

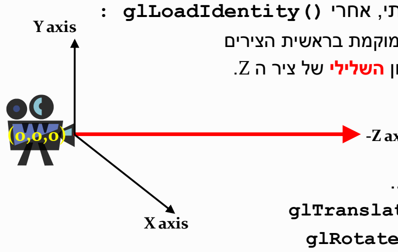
```
render_car()
{
  glTranslatef(c_x,c_y,c_z);
  render_car_body();
  glPushMatrix();
    glTranslate(f_x,f_y,f_z);
    glPushMatrix();
      glTranslate(l_x,l_y,l_z);
      render_wheel();
    glPopMatrix();
    glPushMatrix();
      glTranslate(r_x,r_y,r_z);
      render_wheel();
    glPopMatrix();
  glPopMatrix();
  . . .
}
```



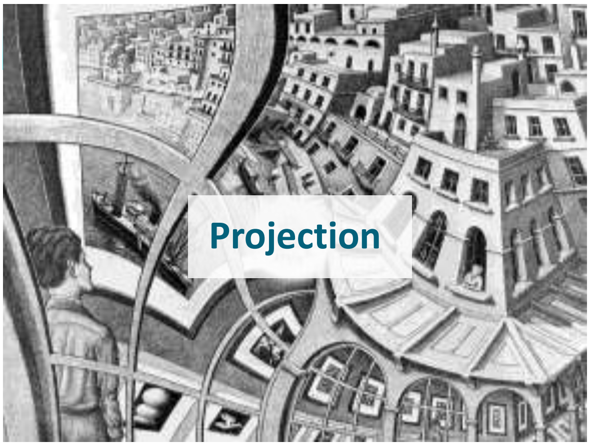
Projection

מטריצת GL_PROJECTION שולטת על המצלמה ומה שרואים בה

- מיקום התחלתי, אחרי `glLoadIdentity()` :
  - המצלמה ממוקמת בראשית הצירים
  - מכוונן לכיוון **השלילי** של ציר ה Z.



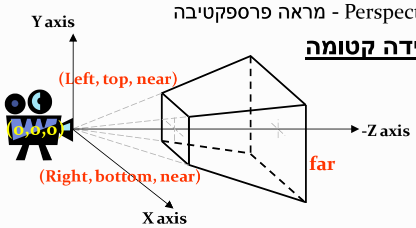
- מה יקרה אם..
  - נזיז? `glTranslate()`
  - נסובב? `glRotate()`



Viewing Volume

- Perspective - מראה פרספקטיבי

<u>פירמידה קטומה</u>

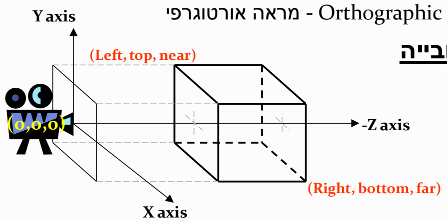



```
glFrustum(left, right, bottom, top, -near, -far);
```
<u>או:</u>
```
gluPerspective(viewAngle, aspectRatio, -near, -far);
```
<u>או:</u>
```
gluLookAt(eye, center, up);
```

Viewing Volume

- Orthographic - מראה אורטוגרפי

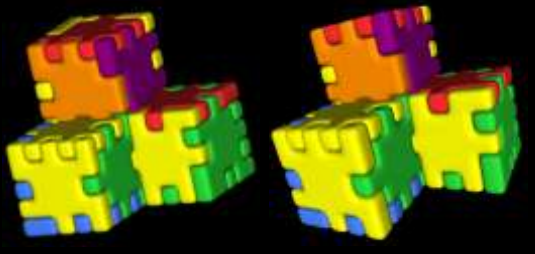
<u>קובייה</u>




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

- שינוי מיקום המצלמה?
- שינוי כיוון המצלמה?

Viewing Volume



Projection

פרטים אחרונים...

- לשים לב ל Aspect Ratio של המסך.
- לפני שינויי מטריצה, לוודא שהמטריצה הנוכחית היא `GL_PROJECTION`
- להתחיל מ Identity
- לשנות כאשר משתנה גודל החלון.
- `glViewport()` - מגדיר את תחומי החלון בו עובדים.
- גודל ה"מסך" – left, right, top bottom מגדירים מה גודל האובייקטים שאפשר להציג
- האם לסובב את האובייקט ימינה או את המצלמה שמאלה?

באפרים

- חלק משמעותי מהמצב הנוכחי ("state") של OpenGL הוא הבאפרים הנוכחיים.
- מימדי כל באפר הם כמימדי שטח החלון עליו מציירים.
- **Color Buffer** - שומר את הצבע הנוכחי עבור כל פיקסל.

כאשר מציירים, בסופו של דבר צבע הפיקסלים מגיע לבאפר זה והוא מוצג בחלון התצוגה.

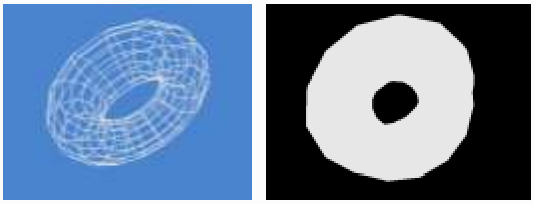
- **Depth Buffer** - ה Z-Buffer של הסצנה. שומר עבור כל פיקסל את עומק הנקודה המוצגת.
- **Accumulation Buffer, Stencil Buffer**

`glClear(...)` - מנקה את הבאפרים לערך התחלתי.

Color Buffer

שומר את הצבע הנוכחי של כל פיקסל

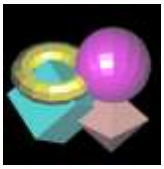
- `glClearColor(…)` - באיזה צבע מתנקה באפר הצבע כאשר מנקים אותו עם `glClear()`
- יש לנקות באפר זה לפני כל פריים. אחרת...



Z-Buffer

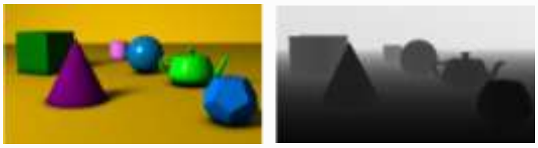
מנקודת המבט של הצופה רואים חלקים מסוימים של כל אובייקט

ה Z-Buffer הוא המקום בו נשמרים המידע על העומק של כל פיקסל.



`glEnable(GL_DEPTH_TEST);`

מפעיל את ה Z-Buffer של OpenGL. (אחרת...)

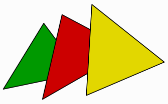


Z-Buffer

- `glClearDepth(1.0);`

כאשר מנקים את ה Z-Buffer, לאיזה ערך הוא מתנקה?

- כמעט אף פעם לא שונה מ 1.0



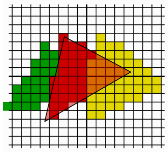
- `glDepthFunc(GL_LEQUAL);`

מגדיר את פונקציה ההשוואה בין שני ערכים

פוליגון 1 - עומק = 0.3

פוליגון 2 - עומק = 0.2 ➔ יבחר לתצוגה.

פוליגון 3 - עומק = 0.4



- כמעט אף פעם אין סיבה לשנות מ `GL_LEQUAL`

Double Buffering

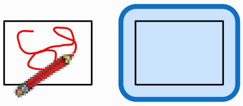
אם את הבאפר הנוכחי כרגע ניקינו כדי לצייר עליו מחדש..
איך הוא עדיין מוצג על המסך?

Double Buffering:

- יש 2 סטים של באפרים
- כאשר מציירים על אחד, השני מוצג בחלון התצוגה.
- כאשר מסיימים לצייר, יש לקרוא ל

```
canvas.swapBuffers();
```

אחרת...



צבעים וחומרים

כאשר <u>לא</u> משתמשים בתאורה

```
glDisable(GL_LIGHTING);
```

הצבע שמקבל כל Vertex הוא <u>הצבע הנוכחי</u>.

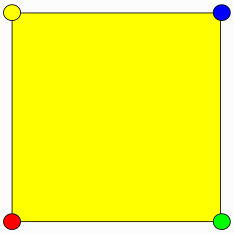
```
glColor3f(float r, float g, float b);
glColor4f(float r, float g, float b, float a);
```

כל ערך בין 0.0 ל 1.0

(ברירת מחדל = 1.0)

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();
```

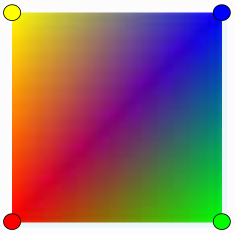


איזה נקודה נבחרת?

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();
```

Gouraud!



עוד מידע...

- הספר האדום - OpenGL Programming Guide
- Nate Robins OpenGL tutorials
- NeHe tutorials
- **MSDN** !
- האתר הרישמי: http://www.opengl.org
- http://en.wikipedia.org/wiki/Java_OpenGL
- Google is your friend