

...זה OpenGL

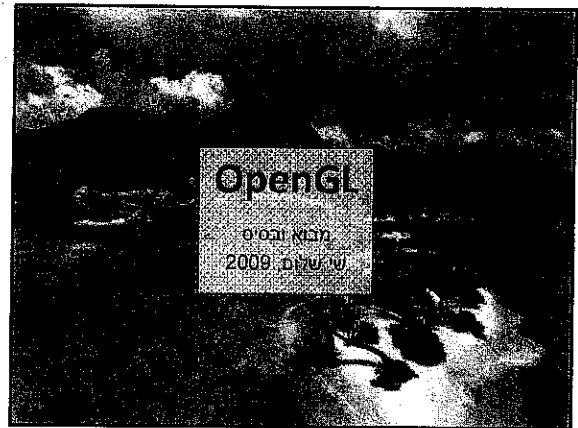
```
gl.glBegin(GL_POLYGON);
gl.glColor3f(0, 1, 0);
gl.glVertex3f(-1, -1, 0);
gl.glVertex3f(-1, 1, 0);
gl.glVertex3f(1, 1, 0);
gl.glVertex3f(1, -1, 0);
gl.glEnd();
```

לא:

- שפת תכנות
- ספריית תוכנות
- ממשק חלונות

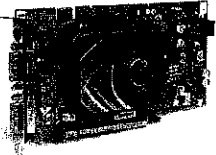
כן:

- הגדרה של ממשק, מסמך
- 352 עמודים נכון לרסא 3.1
- Application Programming Interface
- Cross Language, Cross Platform
- מגדיר בערך 250 פקודות
- "האסמבלי של גרפיקה ממוחשבת"

...זה OpenGL

- אלגוריתמים בסיסיים ממומשים בחומרה
- Rasterization
- Z-Buffer
- Lighting, Materials
- Texturing
- ממשק מהתוכנה לחומרה
- אבל החומרה כל הזמן מתקדמת...
- מודל פרוצדורלי. מבצעים קריאות לפקודות והן מתבצעות אחת אחרי השניה.
- בניגוד ל...?



...זה OpenGL

- Open - סטודרט פתוח, כל אחד יכול לממש
- GL - Graphics Library
- ספריות נלוות:
- OpenGL Utility - GLU
- ספריית עזר סטודרטית
- OpenGL Utility Toolkit - GLUT
- מימוש בסיסי של GUI
- תפריטים, חלונות, קלט מעכבר ומקלדת
- WGL - ספריית עזר לחלונות
- GLX - ספריית עזר לינוקס, לינוקס
- AGL, CGL - ספריות עזר למק.
- OpenGL Extension Wrangler Library - GLEW
- JOGL, LWJGL - פורט ל Java
- OpenGL Shader Language - GLSL
- OpenGL ES

תמונה, PDA וכו'

למה זה טוב?

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

גרפיקה ב"דוס"

Mode 13h (Mode X)

כתיבה ישירה ל VGA Block 0A0000


תלת מימד? (ידי)

חבילות תוכנה (graphics) proprietary של פסקל?

לא חלק מ OpenGL:

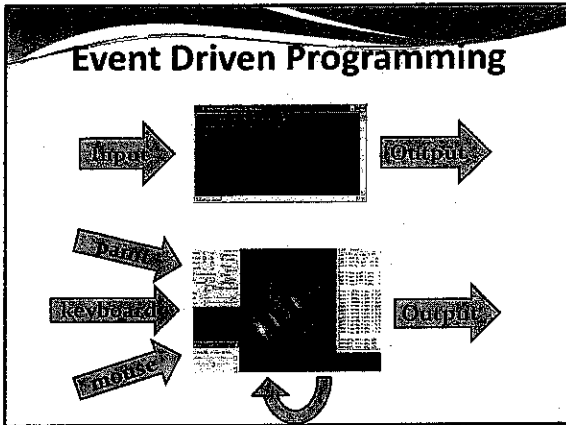
יצירת חלונות

אינטראקציה עם המשתמש (...?)



גרסאות

- Version 1.0 - 1992: first version by SGI
 - Basic geometry primitives, flat and Gouraud shading, Parametric textures,
 - alpha and stencil buffers and more
- Version 1.1 - 1996: Vertex Arrays, Texture enhancements
- Version 1.2 - 1998: Extension mechanism, 3D textures, RGBA pixel format
- Version 1.3 - 2001: Compressed textures, Cube-maps, multitextures, multisamples
- Version 1.4 - 2002: Automatic MipMap, Fog coordinates,
- Version 1.5 - 2003: Vertex Buffers, Occlusion queries, Full set of Boolean operators
- Version 2.0 - 2004: GLSL 1.0
- Version 2.1 - 2006: GLSL 1.2, extensions
- Version 3.0 - 2008: Geometry shader, new GLSL, many extensions



OpenGL - מימושים

- ATI, nVidia, Matrox, 3dfx (RIP)
 - opengl32.dll
 - libGL.so
- Stub לוחמה
 - opengl32.dll - Microsoft
 - default התקנת
 - Windows XP SP2
- Linux
 - mesa 3D - libGL.so - מימוש חומרה/תוכנה

הסינטקס

קורדינטות (...)

- float כלל
- אין הגבלה על ערכים
- glNormal() - ווקטורים

צבע (...)

- Red, Green, Blue ערכי
- בין 0.0 ל 1.0
- float כלל

```

const int GL_DEPTH_BUFFER_BIT = 0x00000100
const int GL_COLOR_BUFFER_BIT = 0x00004000
const int GL_DEPTH_TEST = 0x0B71
    
```

הסינטקס

```

GLfloat a, GLfloat b;
glFunction2f(a, b);
glVertex3f(1.0f, 2.0f, 3.0f);
glVertex3i(1, 1, 1);
glRotated(90.0, 0.0, 0.0, 1.0);

GLfloat *v;
glFunction2fv(v);
float v[] = {1.0f, 2.0f, 3.0f};
glVertex3fv(v);
    
```

ומראיציית עם int: int a, int b
לא מומלץ להשתמש.

GL_CONSTANT

```

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

float - ג'א

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 מצוירים
- מטריצות Model, Projection
- שיטת הציור
- פוליגונים, קווים, קווים+פוליגונים
- תאורה
- איפה יש גופי תאורה, מה המצב שלהם
- טקסטורות
- ועוד ועוד ועוד...

למה זה טוב?

- ניהול מספר חלונות בזמנית
- "בערך" OOP
- המשתמש לא צריך לדבר שום דבר
- אינטראקציה מהירה עם החומרה

מכונת המצבים מתחילה ממצב default מוגדר היטב

התחלת עבודה

יצירת פרויקט 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_DEPTH_TEST);
gl.glColor3f(1.0f, 0.0f, 0.0f);
gl.glNormal3f(...);
gl.glVertex3f(...);
    
```

glGetError();
חאט קוד טראש
לא חומרה

OpenGL כמכונת מצבים

שליטה על מכונת המצבים
מצב בינארי - מודלק או מכונה

`glEnable()`, `glDisable()`

האם הדי באפר מופעל - `GL_DEPTH_TEST`
האם לבצע תאורה? - `GL_LIGHTING`
האם מופעלת תאורה מס' 1 - `GL_LIGHT1`
`GL_CULL_FACE`

...
לקבל את המצב הנוכחי מהמכונה:
`glIsEnabled()`

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` ...

אני, חלד קורה ב'ק' פיקצ'ורי `paint`:

Per Vertex Operations

Vertex - קודקוד, נקודה.

- הנקודה היא הפרימיטיב הבסיסי ביותר והחשוב ביותר.
- רוב פעולות הציור מתבצעות ונשמרות עבור נקודה.
- קו - מוגדר באופן מלא על ידי 2 נקודות
- צבע, עובי, קיווק
- תכונות הקו מוגדרות על ידי תכונות הנקודות
- פוליגון - מוגדר באופן מלא על ידי 3 נקודות ומעלה
- צבע, טקסטורה, נורמל
- תכונות הפוליגון מוגדרות על ידי תכונות הנקודות

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

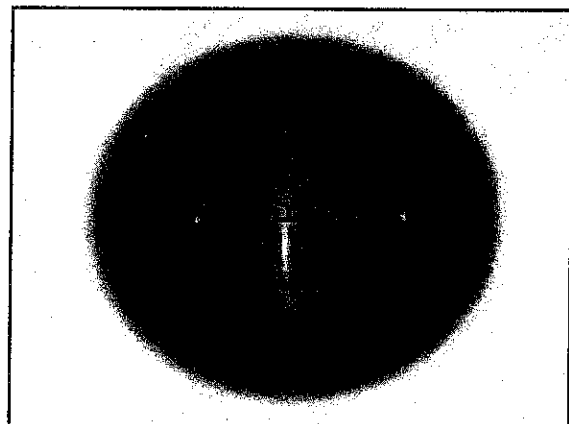
דוגמא

```
// 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();
```

נקודות

- עוד שם: Vertex
- ציור נקודה דוגם את פרמטרי הציור
- 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 הם לא באותה נקודה במרחב.

`glVertex3f(1.0, 0.0, 1.0);`



פוליגון

- שטח סגור בין כמה נקודות

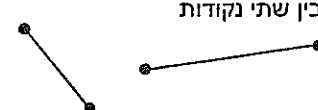


- `glPolygonMode()` - איך לצייר את הפוליגון
- רק קווים, רק נקודות, מלא בצבע

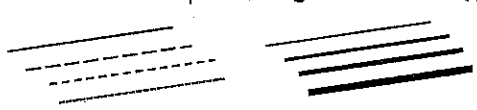


קווים

- קו מתוח בין שתי נקודות



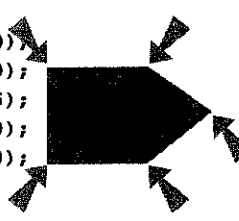
- `glLineStipple()` - מגדיר קווקו.
- `glLineWidth()` - עובי הקו



ציור - הלכה למעשה

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

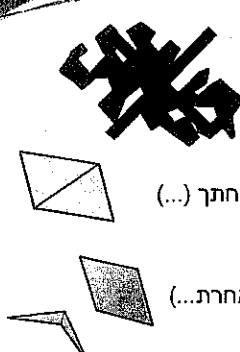
```
glBegin(GL_POLYGON);
glVertex2f(0.0, 0.0);
glVertex2f(4.0, 0.0);
glVertex2f(6.0, 1.5);
glVertex2f(4.0, 3.0);
glVertex2f(0.0, 3.0);
glEnd();
```



פוליגון

מגבלות:

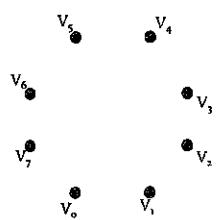
- חייב להיות קמור
- אסור לגבולות 2 פוליגונים להיחתך (...)
- צריכים להיות באותו מישור (אחרת...)



`glBegin()`, `glEnd()`

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

ציור נקודות



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

צייר קווים בין כל זוג נקודות שכנות
הבדל מפוליגון?

glBegin(), glEnd()

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

צייר קווים בודדים בין נקודה א-זוגית לנקודה זוגית

glBegin(), glEnd()

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

צייר פוליגון

glBegin(), glEnd()

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

משולשים בין נקודות כל שלשה בודדת

Front, Back Facing Polygon

אלא אם מוגדר אחרת:

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

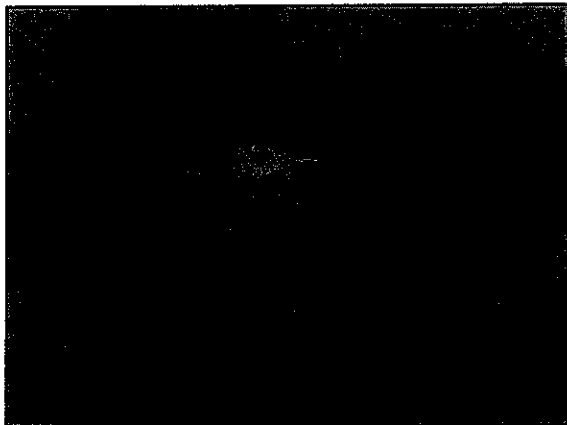
משנה הגדרה - glFrontFace()

glBegin(), glEnd()

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

מרובעים בין כל 4 נקודות עוקבות

ר"ר
38 נאיות
שחון



Back Face Culling

פוליגונים שמכוונים אחורנית לא מרונדרים

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

מתי להשתמש? מתי לא להשתמש?

טרנספורמציות

$$\begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glTranslate(dx, dy, dz) glLoadIdentity() glScale(sx, sy, sz)

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

glRotate(α, 1, 0, 0) - X glRotate(α, 0, 1, 0) - Y glRotate(α, 0, 0, 1) - Z

טרנספורמציות

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

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

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

- "המטריצה הנוכחית" - חלק מהמצב של OpenGL.
- כל זמן נתון יש מטריצת טרנספורמציה בלבד.
- glLoadIdentity() - העבר למטריצה הנוכחית את טרנספורמצית היחידה.
- glTranslate() - הכפל את המטריצה הנוכחית בטרנספורמצית הזזה: מרחק ב 3 מימדים.
- glScale() - הכפל את המטריצה הנוכחית בטרנספורמצית מתחה - ב 3 מימדים, סביב הראשית.
- glRotate() - הכפל את המטריצה הנוכחית בטרנספורמצית סיבוב - זווית סביב ציר (ווקטור).

טרינספורמציות

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?

טרינספורמציות

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.

Push / Pop Matrix

```
render_car()
{
    glTranslatef(x,y,z);
    render_car_body();
    ...
}
```

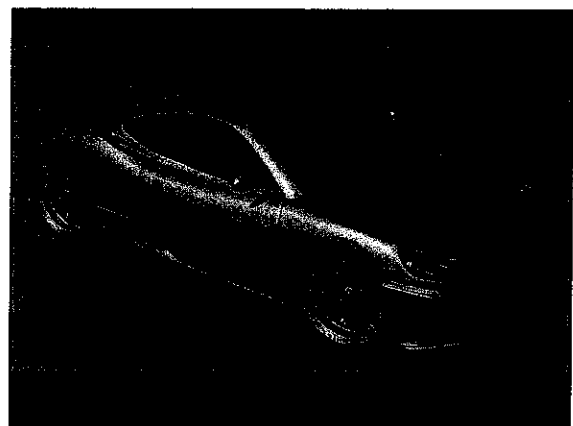
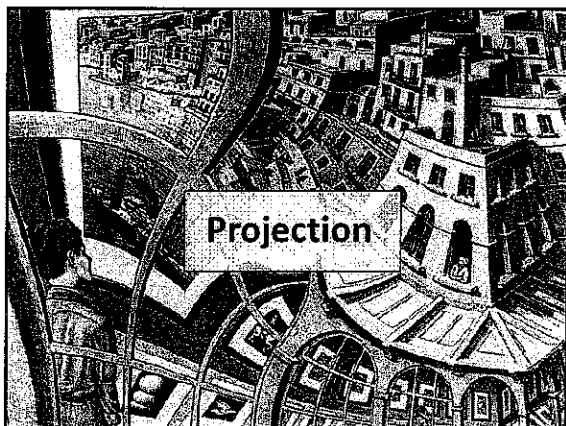
טרינספורמציות

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?

הבידוד של פוסט האדים מאלו של פוסט האדים האחרים, איך זה הפעולה הקודמת שמתבצעת.



Viewing Volume

• Orthographic - מראה אורתוגרפי

קובייה

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

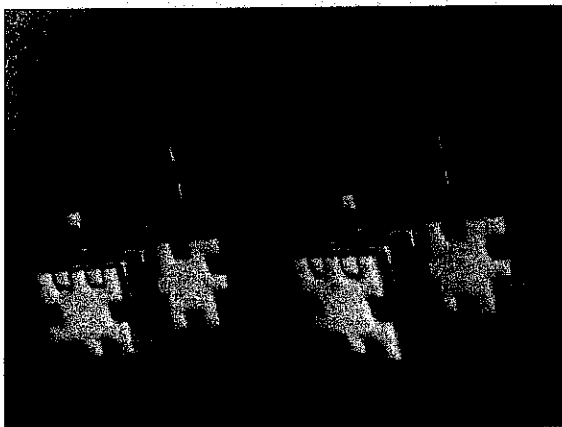
- שינוי מיקום המצלמה?
- שינוי כיוון המצלמה?

Projection

מטריצת `GL_PROJECTION` שולטת על המצלמה ומה שרואים בה

- מיקום התחלתי, אחרי `glLoadIdentity()`
- המצלמה ממוקמת בראשית הצירים
- מכוננת לכיוון השלילי של ציר ה-Z

מה יקרה אם...
 • נזיזת? `glTranslate()`
 • לסובב? `glRotate()`



Viewing Volume

• Perspective - מראה פרספקטיבה

פירמידה קטומה

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

`gluPerspective(viewAngle, aspectRatio, -near, -far);`

`gluLookAt(eye, center, up);`

Projection

פרטים אחרונים...

- לשים לב ל `Aspect Ratio` של המסך.
- לפני שינוי 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?

Perspective Distortion

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...

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 התחלת.

Handwritten notes: ציבג, לנקי, ב frame, ציבג באיזה, פולגון, OR

חשוב לנקות את ה buffer של ה תחבונ 'ארוחה' כל הסצנה.

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_EQUAL)` ; 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 לשמת `GL_EQUAL` מ

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...)

סדר ה אובייקטים הם כל סדר הכולם כיקור.

Double Buffering

אם את הבאפר הנוכחי כרגע ניקינים 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...

Z-Buffer

Z-Fighting

שני משטחים שנמצאים בדיוק באותו מישור יריב על מי יהיה מוצג לפני מי.

יבחר יאילו 2 אובייקטים (אחד ל כל אובייקט)

Shading Models

```
glShadeModel (GL_FLAT);
glBegin(GL_QUADS);
[Image showing a flat shaded square]
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);
[Image showing a sphere with no shading]
```

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

a - ערך השקיפות (ברירת מחדל = 1.0)

עוד מידע...

- הספר האדום - OpenGL Programming Guide
- Nate Robins OpenGL tutorials
- NeHe tutorials
- MSDN!
- האתר הרישמי: <http://www.opengl.org>
- Google is your friend

Shading Models

```
glShadeModel (GL_SMOOTH);
glBegin(GL_QUADS);
[Image showing a smooth shaded square]
glColor3f (1.0,1.0,0.0);
glVertex3f(0.0,1.0,0.0);
glEnd();
```

טור

Polygon Subdivision

• אלגוריתם להחלקת אובייקטים פוליגונים.

• בכל מעבר, כל פוליגון הופך ל 4 פוליגונים.