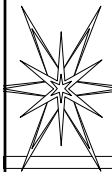


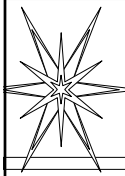
# OpenGL

<http://www.opengl.org/>



# Bevezetés

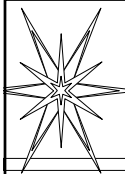
- Tematika
  - OpenGL
    - Primitívek létrehozása
    - ...
  - Előadás feldolgozása
- Hasznos oldalak
  - <http://www.opengl.org>
  - <http://www.opengl.org/documentation/specs/glut/index.html>
  - <http://www.mesa3d.org>



## Bevezetés

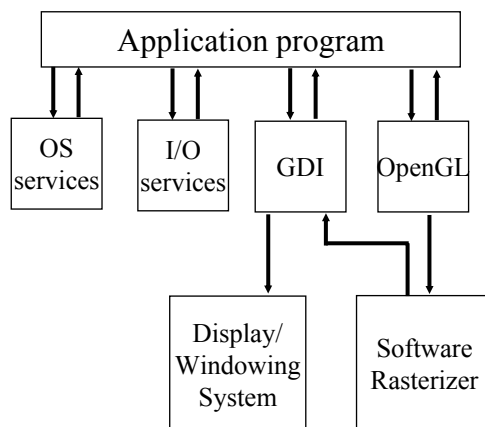
- OpenGL szabvány
  - Szoftveres felület a grafikus hardverhez
  - 3D-s grafikus és modellező könyvtár
    - Portábilis
    - Nagyon gyors
    - Gyorsabb, mint egy ray tracer
  - SGI által tervezett és optimalizált algoritmusok
- Hogyan működik az OpenGL
  - lépéseket kell megadni ahhoz, hogy megkapjuk az adott nézetet vagy megjelenést
  - 200-nál több parancs és függvény
    - Grafikus primitívek, megvilágítás, árnyékolás, textúrázás, keveredés, átlátszóság, animálás, stb.
  - Nincs ablak kezelés
  - Nincs OpenGL file formátum

3

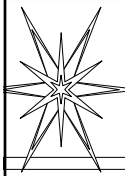


## Bevezetés

- Általános megvalósítások
  - Szoftveres
  - GDI
    - Szöveg kiírás
    - 2D-s vonal rajzolás
    - ...
  - Microsoft
  - SGI
    - MMX-enhanced
    - Hivatalosan nem támogatott
  - Mesa
    - Conformance teszt



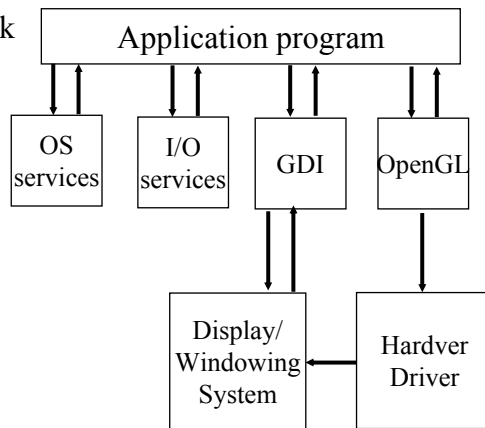
4



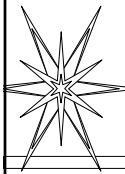
## Bevezetés

- Hardveres megvalósítások

- Speciális hardver eszköz meghajtó
- Minden OpenGL API fv. hívás a hardver meghajtóhoz megy
- Néha az OpenGL funkció szoftveresen van megvalósítva a meghajtó programban
- Más funkciók pedig egyenesen a hardvert használják

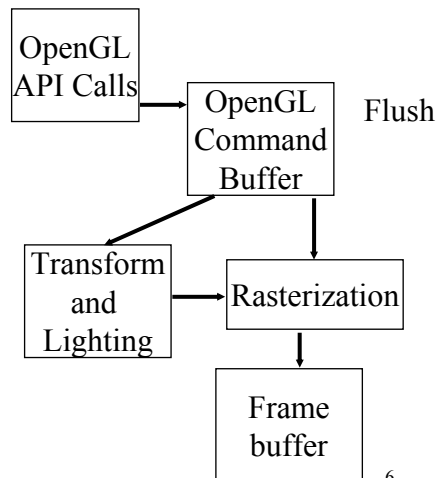


5

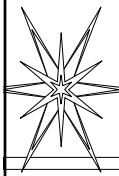


## OpenGL pipeline

- OpenGL alkalmazás fv. hívások
- Parancs puffer
  - Vertex adat
  - Textúra adat
  - ...
- Transzformáció és megvilágítás
  - Az objektumok geometriáját leíró pontok újraszámítása
- Raszterizálás
  - Színes kép készül



6

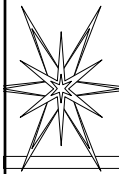


## OpenGL állapotgép

- Minden parancs azonnali hatással van az aktuális renderelési állapotra
  - Flag-ek
    - A kód be van kapcsolva?
    - Engedélyezett a megvilágítás?
  - Numerikus értékek
  - Függvények segítségével állíthatóak és lekérdezhetőek



7



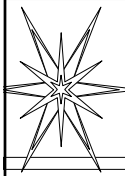
## Függvény elnevezési szabályok

- <Lib prefix><Root command><Optional arg. count><Optional arg. type>

gl Color 3 f

**glColor3f(...)**

8



## OpenGL utility toolkit (GLUT)

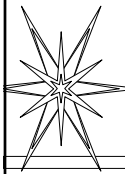
- Kezdetekben AUX
  - Kiegészítő library
- Kiváltotta a GLUT
  - Pop-up menük
  - Más ablakok kezelése
  - Joystick támogatás

<http://www.opengl.org>

- Első program

```
#include <GL/glut.h>
void RenderScene(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}
void SetupRC(void) {
    glClearColor(0.0f, 0.0f, 1.0f, 1.0f);
}
void main (void) {
    glutInitDisplayMode(GLUT_SINGLE |
                        GLUT_RGB);
    glutCreateWindow("Simple");
    glutDisplayFunc(RenderScene);
    SetupRC();
    glutMainLoop();
}
```

9



## Linux Makefile (RedHat 7.3)

- ```
# Makefile for OpenGL examples
APPS = simple
OBJ = $(APPS).o
SRC = $(APPS).c
CFLAGS = $(C_OPTS) -I/usr/include
LIBS = -L/usr/X11R6/lib -lX11 -lXi -lXmu -lglut -lGL -lGLU -lm -pthread

application:$(APPS)

clean:rm -f $(APPS) *.raw *.o core a.out

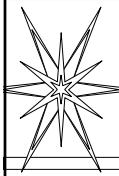
realclean:    clean
              rm -f *~ *.bak *.BAK

.SUFFIXES: c o
.c.o:
    $(CC) -c $(CFLAGS) $<$

(APPS): $(OBJ)
    $(CC) -o $(APPS) $(CFLAGS) $(OBJ) $(LIBS)

depend:      makedepend -- $(CFLAGS) $(SRC)
```

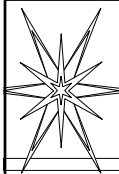
10



## OpenGL használata

- Programozási könyvtár
  - opengl32.dll
  - glu32.dll
  - opengl32.lib
  - glu32.lib
  - gl.h, glu.h
- Adattípusok
  - GLbyte
  - GLshort
  - GLint, GLsizei
  - GLfloat, GLclampf
  - GLdouble, GLclampd
  - GLubyte, GLboolean
  - GLushort
  - GLuint, GLenum, GLbitfield

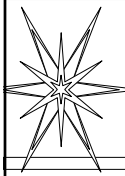
11



## Függvények az első programban

- *glutInitDisplayMode(unsigned int mode)*
  - Inicializálja a megjelenítési módot
  - *mode*
    - GLUT\_SINGLE
    - GLUT\_DOUBLE
    - GLUT\_RGB, GLUT\_RGBA
    - GLUT\_DEPTH
    - GLUT\_STENCIL
    - GLUT\_ACCUM
    - GLUT\_ALPHA
- *glutDisplayFunc(void \*(f)(void))*
  - Beállítja a callback fv.-t az aktuális ablakon
    - Átméretezés
    - Előtérbe kerülés
    - *glutPostRedisplay*
  - *glFlush* és *glutSwapBuffers* nem hívódik meg automatikusan ezután

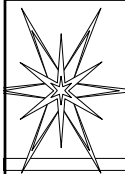
12



## Függvények az első programban

- *void glutMainLoop(void)*
  - Elindítja a GLUT eseménykezelő ciklusát
  - Nem tér vissza csak a program befejezése után
- *void glClear(GLbitfield mask);*
  - Az adott puffereket törli
  - A puffer egy tárolási terület a kép információ számára
  - Mask
    - GL\_COLOR\_BUFFER\_BIT
    - GL\_DEPTH\_BUFFER\_BIT
    - GL\_STENCIL\_BUFFER\_BIT
    - GL\_ACCUM\_BUFFER\_BIT

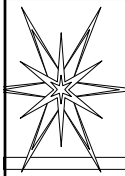
13



## Függvények az első programban

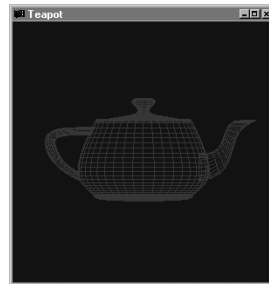
- *void glFlush(void)*
  - Az OpenGL pipeline-t és puffereket üríti
  - A várakozó parancsok végrehajtódnak
- *void glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf a)*
  - Beállítja a kitöltési értéket, amit a red, green, blue és alpha (szín) pufferek törlésekor használ majd
  - Az értékek [0.0f, 1.0f] között lehetnek

14

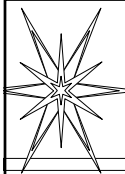


## Teáskanna

```
▪ #include <GL/glut.h>
void RenderScene(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glutWireTeapot(0.5f);
    glFlush();
}
void SetupRC(void) {
    glClearColor(0.0f, 0.0f, 1.0f, 1.0f);
}
void main (void) {
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Teapot");
    glutDisplayFunc(RenderScene);
    SetupRC();
    glutMainLoop();
}
```

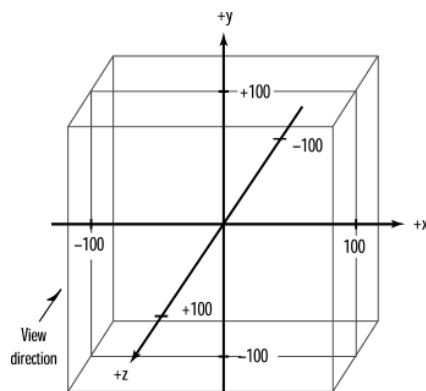


15



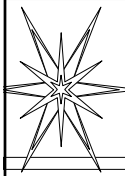
## Vonalak, pontok, poligonok

- Az objektumok és a látvány egyszerű, kicsi alakzatokból épülnek fel
  - Primitívek
    - Egy vagy 2 dimenziós objektumok
- 3D-s festővászon
  - OpenGL parancsok
  - OpenGL függvények



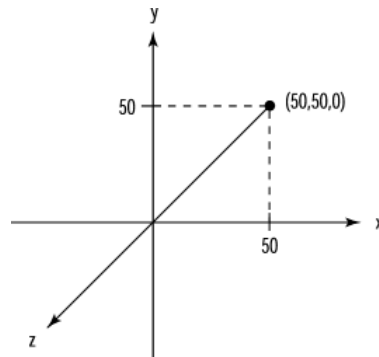
16



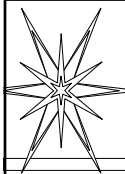


## Vertex

- *glVertex*
  - 2, 3 vagy 4 paraméter
  - Különböző típusok
  - Példa
    - 2D  
`glVertex2f(50.0f, 50.0f)`
    - 3D  
`glVertex3f(50.0f, 50.0f, 0.0f)`
    - 3D homogén koordináták  
`glVertex4f(50.0f, 50.0f, 0.0f, 1.0f)`



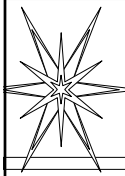
17



## Vertex

- `void glBegin(GLenum mode)`
  - Milyen primitívet kell létrehozni
  - *mode*
    - GL\_POINTS
    - GL\_LINES
    - GL\_LINE\_STRIP
    - GL\_LINE\_LOOP
    - GL\_TRIANGLES
    - GL\_TRIANGLE\_STRIP
    - GL\_TRIANGLE\_FAN
    - GL\_QUADS
    - GL\_QUAD\_STRIP
    - GL\_POLYGON
- `void glEnd(void)`
  - A vertex-ek listájának a végét jelzi. A `glBegin()`-nel párban kell használni
    - glVertex
    - glColor
    - glIndex
    - glNormal
    - glEvalCoord
    - glCallList
    - glCallLists
    - glTextCoord
    - glEdgeFlag
    - glMaterial

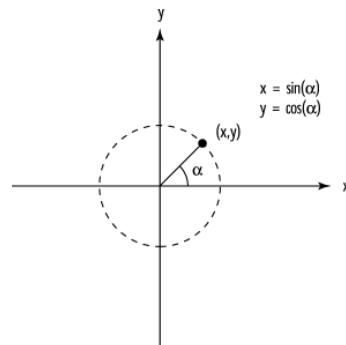
18



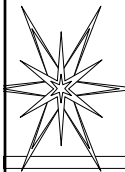
## Vertex

- Pontok rajzolása
  - $0^\circ$ - $360^\circ$
  - A szögek radiánban vannak megadva
  - 3-szor rajzolja ki a kört
    - A z koordinátát alig-alig változtatva
  - *glutSpecialFunc*
    - Regisztrálja azt a fv-t, amely a speciális billentyűk leütésekor hívódik meg
  - *glRotatef* később

- Példa: points.c



19

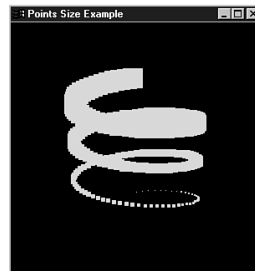


## Vertex - pont méret

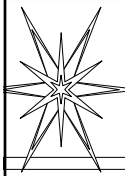
- `void glPointSize(GLfloat size)`
  - *size* a pont közelítő átmérője
  - Nem mindegyik méret támogatott
- Példa

```
GLfloat sizes[2];
GLfloat step;
glGetFloatv(GL_POINT_SIZE, sizes);
glGetFloatv(
    GL_POINT_SIZE_GRANULARITY, &step);
```
- A `glBegin` és `glEnd`-en kívül szabad meghívni
- Ha kisebb vagy nagyobb értéket állítunk be a megengedettnél, akkor a legkisebb/legnagyobb értéket használja

- Példa: pointsz.c



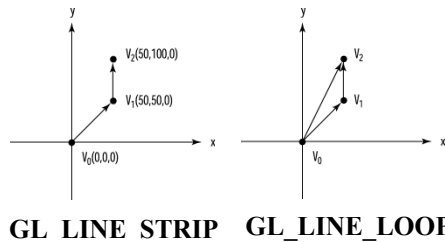
20



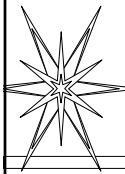
## Vertex - vonal rajzolás

- `glBegin(GL_LINES)`  
`glVertex3f(0.0f, 0.0f, 0.0f);`  
`glVertex3f(50.0f, 50.0f, 50.0f);`  
`glEnd();`
  - Páratlan számú vertex esetén az utolsót nem veszi figyelembe
- `glBegin(GL_LINE_STRIP)`  
`/* GL_LINE_LOOP */`  
`glVertex3f(0.0f, 0.0f, 0.0f); /*v1*/`  
`glVertex3f(50.0f, 50.0f, 50.0f); /*v2*/`  
`glVertex3f(50.0f, 100.0f, 0.0f); /*v3*/`  
`glEnd();`

- Példa: `lines.c`



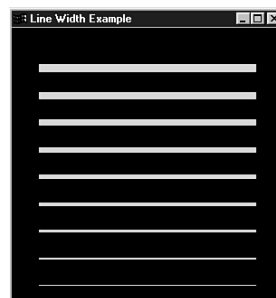
21



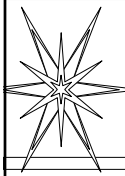
## Vertex - vonal vastagság

- `glLineWidth(GLfloat width)`
- Példa  
`GLfloat sizes[2];`  
`GLfloat step;`  
`glGetFloatv(`  
`GL_LINE_WIDTH_RANGE, sizes);`  
`glGetFloatv(`  
`GL_LINE_WIDTH_GRANULARITY,`  
`&step);`

- Példa: `linesw.c`



22

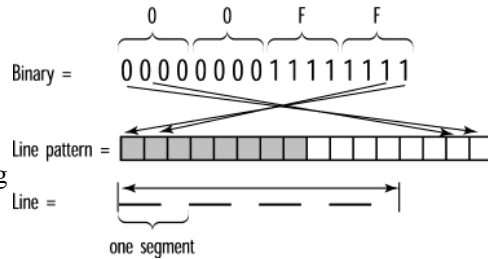


## Vertex - vonal „szaggatás”

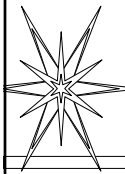
- `glEnable(GL_LINE_STIPPLE)`
- `glDisable (GL_LINE_STIPPLE)`
- `void glLineStipple(`  
`GLint factor,`  
`GLushort pattern)`
- *Pattern*
  - 16 bit-es érték
  - 0-s bit a least significant
- *factor*
  - A minta szélességét adja meg
  - Pl. 5 esetén mindegyik bit a mintában 5 pixel-t jelent

▪ Példa: `lstipple.c`

Pattern = 0X00FF =255

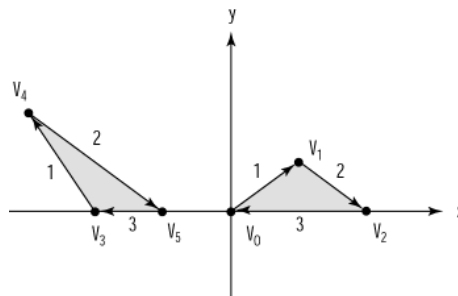


23



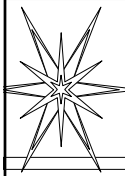
## Vertex - háromszög

- `glBegin(GL_TRIANGLES);`  
`glVertex2f(0.0f, 0.0f); /*V0*/`  
`glVertex2f(25.0f, 25.0f); /*V1*/`  
`glVertex2f(50.0f, 0.0f); /*V2*/`  
  
`glVertex2f(-50.0f, 0.0f); /*V3*/`  
`glVertex2f(-75.0f, 50.0f); /*V4*/`  
`glVertex2f(-25.0f, 0.0f); /*V5*/`  
`glEnd();`



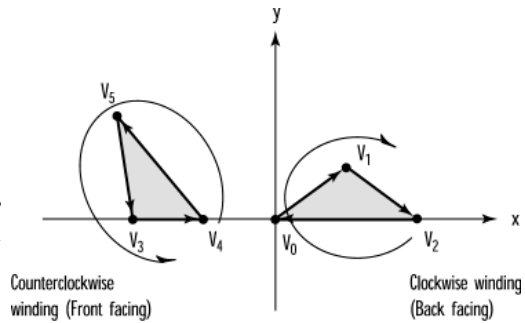
- Mindegyik poligon előállítható háromszögekből
- A videokártyák optimalizálva vannak a háromszögek kirajzolására

24

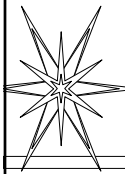


## Vertex - „irányítottság”

- OpenGL-ben alapértelmezésben órajárással ellentétes irányítottságú előlnézet
- Jobb-sodrású
- Fontos pl. a színezésnél, hátsó terület elrejtésénél
- `glFrontFace(GL_CW)`
  - `GL_CW`
  - `GL_CCW`

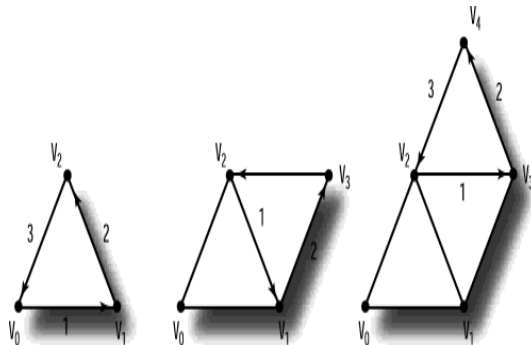


25

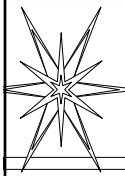


## Vertex - háromszögsík

- `GL_TRIANGLE_STRIP`
  - Megőrzi az irányítottságot mindegyik háromszögre
- Előnyök
  - Az első három pont megadása után egy új háromszöghöz csak egy pontot kell megadni
  - Kevesebb vertex, gyorsabb végrehajtás a transzformációknál



26



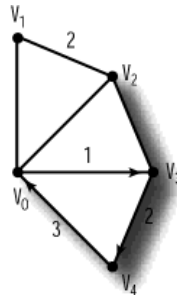
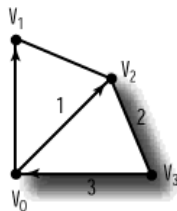
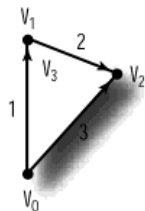
## Vertex – háromszög-legyező

- TRIANGLE\_FAN

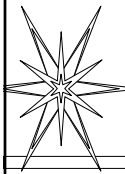
- Egy középpontnál kapcsolódó háromszögek csoportja

- Az első vertex a középpont

- Órajárással megegyező



27



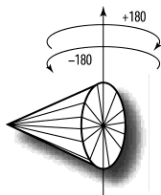
## Vertex - szolid objektumok

- Színezés

- A színek pontonként van megadva
- `glShadeModel(GL_FLAT)`
  - `GL_SMOOTH`

- Rejtett felszín eltávolítás

- Az alját később rajzolja ki
- Mélység ellenőrzés
  - Pixel z koordinátája meghatározza
  - `glEnable(GL_DEPTH_TEST)`
  - `glDisable()`



Példa: `triangle.c`

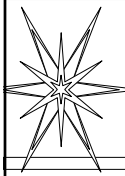
- Belső/külső oldalak kirajzolásának letiltása

- Alapértelmezéskor kirajzolja
- `glEnable(GL_CULL_FACE)`
- `glCullFace(GL_FRONT)`
  - `GL_BACK`

- Poligon módok

- Alapértelmezésben szolid
- `glPolygonMode(GL_BACK, GL_LINE)`
- `GL_POINT`, `GL_FILL`
- `GL_FRONT`, `GL_FRONT_AND_BACK`

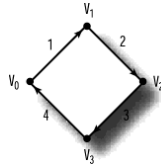
28



## Vertex - más primitívek

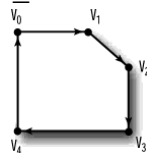
- Négy oldalú poligon

- `GL_QUADS`



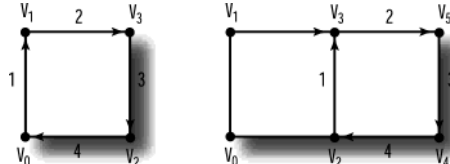
- Általános poligon

- `GL_POLYGON`



- Négy oldalú csík

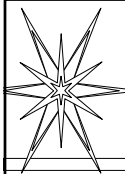
- `GL_QUAD_STRIP`



- Téglalap

- `glRectf(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2,)`

29



## Poligonok kitöltése

- Texture mapping később

- Szaggatott vonal minta

- `glEnable(GL_POLYGON_STIPPLE)`

- `glPolygonStipple(pBitmap)`

- `GLubyte *pBitmap`

- 32x32

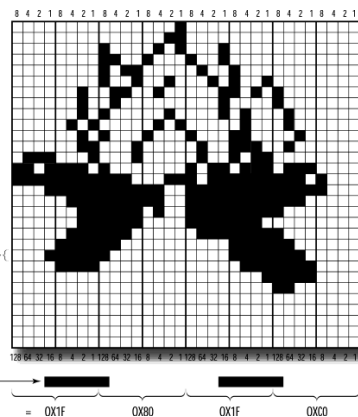
- A legmagasabb helyiértékű bit elsőként

- Alulról felfele haladva

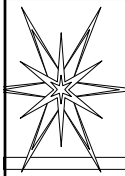
- A minta nem forog a poligonnal

- Csak egyszerű poligon kitöltésre használjuk

Példa: `pstipple.c`

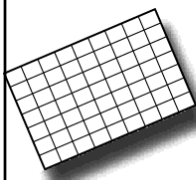


30

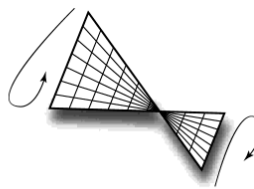


## Poligonokra vonatkozó szabályok

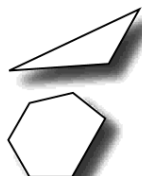
- Minden pont egy síkon van
- A poligonok élei nem metszhetik egymást
- Konvex-ek



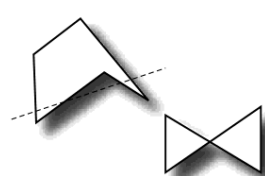
Planar polygon



Nonplanar polygon

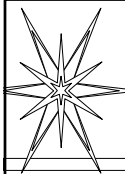


Valid polygons



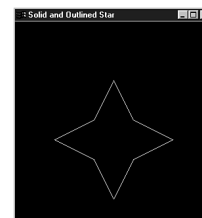
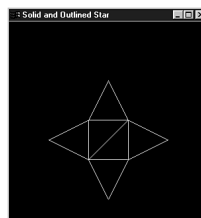
Invalid polygons

31



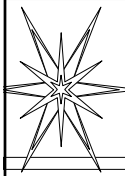
## Parcellák és élek

- Konvex poligonokból előállítható nem konvex poligon
- Amikor kitöltjük a poligonokat, akkor nem látjuk az éleket
- *edge flag*
  - `glEdgeFlag(True)`
  - `False`
  - *Poligon mód: `GL_LINE`, `GL_POINT`*
- Példa: `star.c`



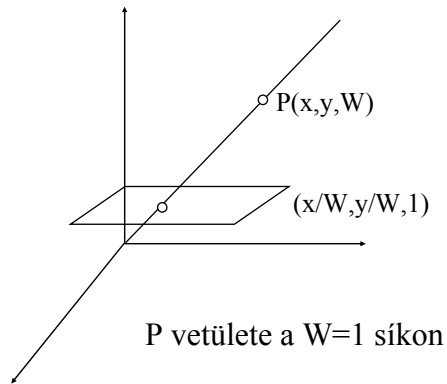
32



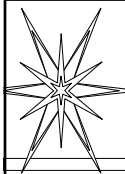


## Homogén koordináták

- $(x,y) \rightarrow (x,y,W)$
- $(x,y,W) = (x',y',W')$ , ha  $\exists \alpha: x' = \alpha x, y' = \alpha y, W' = \alpha W$
- Egy ponthoz végtelen sok  $(x,y,W)$  tartozik
- $(0,0,0)$  nem megengedett
- Ha  $W \neq 0$ ;  $(x/W, y/W, 1)$  a szokásos jelölése  $(x,y)$ -nak
- Ha  $W = 0$ ;  $(x,y,0)$  végtelen távoli pont



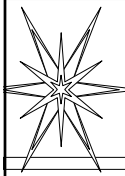
33



## Koordináta transzformációk

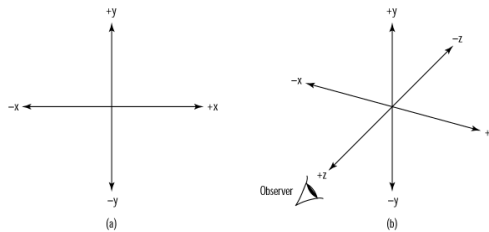
- OpenGL transzformációk
  - Viewing
    - A néző vagy a kamera helyének a megadása
  - Modellező (Modeling)
    - Az objektumok mozgatása
  - Modelview
    - Dualitás a nézeti és a modellező transzformációk között
- Projekció
  - A nézet vágása és méretezése
- Nézeti
  - A végleges eredmény ablakra való leképezése

34

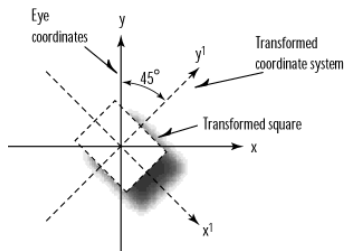


## Szem koordináták

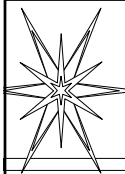
- A megfigyelő nézőpontja
- Virtuálisan rögzített koordináta rendszer



A monitorra merőleges  
(ahogy a megfigyelő látja)



35

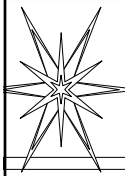


## Viewing transzformációk

- Ez hajtódik végre először
- Nézőpont meghatározása
  - $(0, 0, 0)$
  - A z tengely negatív része felé nézve
  - A pozitív értékek a z tengelyen a megfigyelő mögött vannak
  - `void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz)`
- A megfigyelő pont transzformálása
- Ezt kell legelőször definiálni

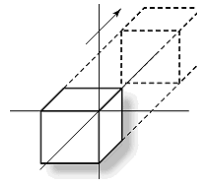


36



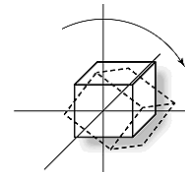
## Modellező transzformációk

- A modell vagy annak egy részének a manipulálására használjuk
- A végső kinézetet befolyásolja



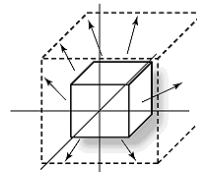
Translation

(a)



Rotation

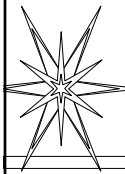
(b)



Scaling

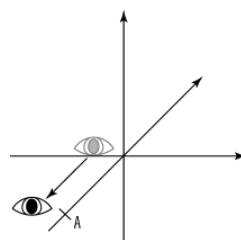
(c)

37



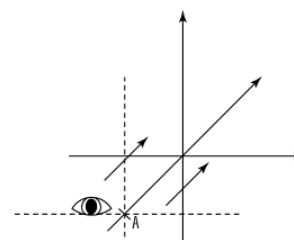
## Modelnézeti dualitás

- A viewing transzformáció és a modellező transzformációk duálisak



Moving the observer

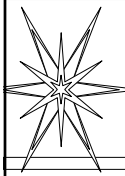
(a)



Moving the coordinate system

(b)

38



# Projekció/Nézeti transzformáció

- Utolsóként van alkalmazva
- Definiálja a nézeti teret és vágósíkokat
  - Ortogonális
    - 2D-s rajzok
    - CAD
  - Perspektivikus
    - Valós élet



Orthographic



Perspective

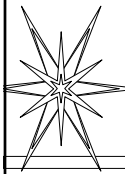


Orthographic



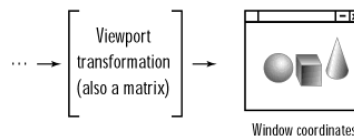
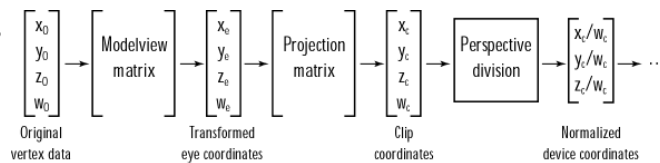
Perspective

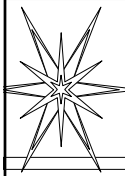
- Nézeti transzformáció
  - 2D-s leképezés az ablakra
  - *glViewport()*



# Mátrixok

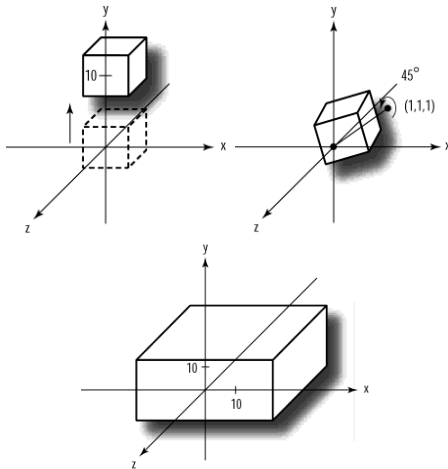
- Transzformációs sor
  - Modellnézeti
  - Projekciós
  - Nézeti



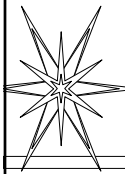


## Modelnézeti mátrix

- 4x4-es mátrix
- A vertex koordinátákat transzformálja
- `void glTranslatef(GLfloat x, GLfloat y, GLfloat z)`
- `void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)`
- `glScale(GLfloat x, GLfloat y, GLfloat z)`

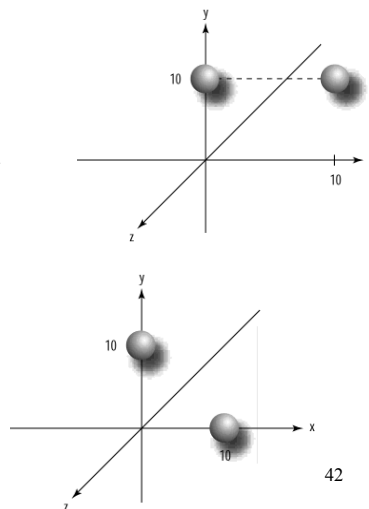


41

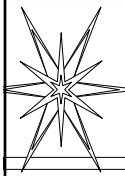


## Az egység mátrix

- A transzformációkat egymásután használva azok hatása összegződik
- A két utasítás segítségével a Modelview mátrix-ba betöltjük az egység mátrixot
  - `glMatrixMode(GL_MODELVIEW)`  
`glLoadIdentity()`
  - `GL_MODELVIEW,`  
`GL_PROJECTION, GL_TEXTURE`



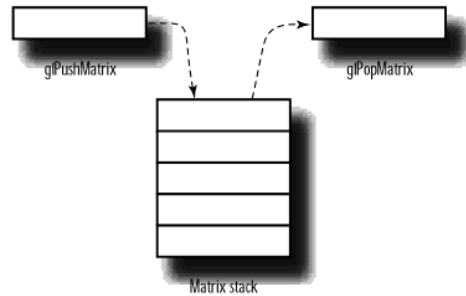
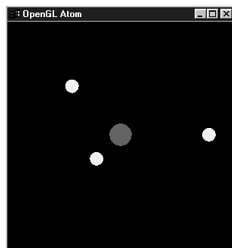
42



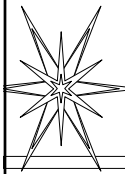
## Mátrix veremek

- `glGet(GL_MAX_MODELVIEW_STACK_DEPTH)` (Microsoft 32)
- `glGet(GL_MAX_PROJECTION_STACK_DEPTH)` (Microsoft 2)
- `GL_STACK_OVERFLOW`
- `GL_STACK_UNDERFLOW`
- `void glPushMatrix(void)`
- `void glPopMatrix(void)`

Példa: atom.c

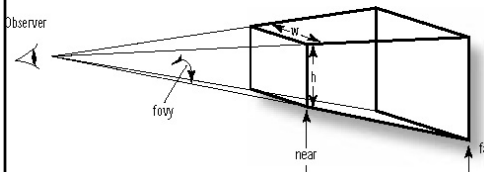
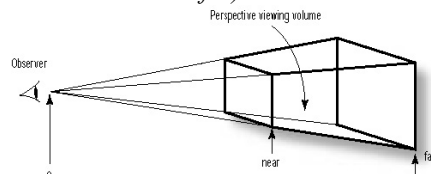


43



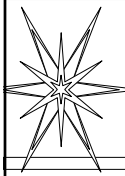
## Vetületek használata

- A vetület mátrix meghatározza a nézeti tér méretét és alakját
  - Ortozonális
    - Példa: ortho.c
  - Perspektivikus
    - Példa: perspect.c
- Perspektív
  - Piramis szelet
    - `glFrustum(GLdouble left, ..., GLdouble far)`



- `void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble znear, GLdouble zfar)`
  - `fovy` field of view
  - `aspect`

44

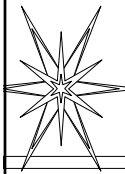


## Perspektív vetület

- // Change viewing volume and viewport. Called when window is resized
- void ChangeSize(GLsizei w, GLsizei h) {
  - GLfloat fAspect;
  - // Prevent a divide by zero
  - if (h == 0) h = 1;
  - // Set Viewport to window dimensions
  - glViewport(0, 0, w, h);
  - fAspect = (GLfloat)w/(GLfloat)h;
  - // Reset coordinate system
  - glMatrixMode(GL\_PROJECTION);
  - glLoadIdentity();
  - // Produce the perspective projection
  - gluPerspective(60.0f, fAspect, 1.0, 400.0);
  - glMatrixMode(GL\_MODELVIEW);
  - glLoadIdentity();

Példa: solar.c

45

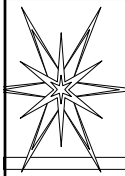


## Mátrix műveletek

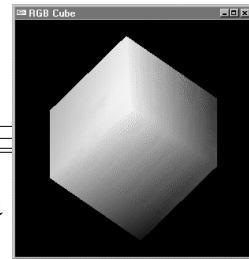
- Gyorsabbak a magasabb szintű műveletek
- Saját transzformáció végrehajtása
  - $GLfloat\ m[] = \{$ 
    - $1.0, 0.0, 0.0, 10.0,$
    - $0.0, 1.0, 0.0, 0.0,$
    - $0.0, 0.0, 1.0, 0.0,$
    - $0.0, 0.0, 0.0, 1.0\}$
  - $glMatrixMode(GL_MODELVIEW);$
  - $glMulMatrix(m);$
- Más transzformációk
  - Árnycék rajzolás
  - ...
- Mátrix betöltés
  - $GLfloat\ m[] = \{$ 
    - $1.0, 0.0, 0.0, 0.0,$
    - $0.0, 1.0, 0.0, 0.0,$
    - $0.0, 0.0, 1.0, 0.0,$
    - $0.0, 0.0, 0.0, 1.0\}$
  - $glMatrixMode(GL_MODELVIEW);$
  - $glLoadMatrix(m);$

$$\begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$$

46

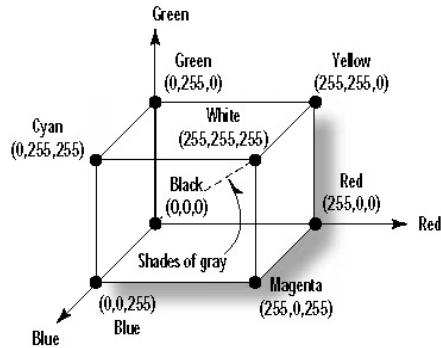


# Szín, fény, anyag

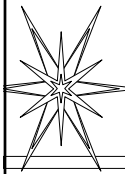


- Szín mélység
  - 4, 8, 16, 24
  - 32 - true color
    - 32 bit-es cím határra igazítja a pixel adatot
    - Pixelenként 1 byte-ot veszítünk
    - Gyorsabb
  - 16 bit
    - Gyakorlatilag ugyanaz mint a 24 bit
    - Memória megtakarítás
    - gyorsítás

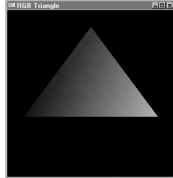
## ▪ RGB kocka



47

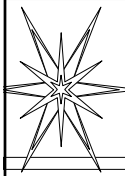


# Szín

- A rajzolási szín beállítása
  - `void glColor<x><t>(r, g, b, a)`
    - `<x>` argumentumok száma
      - alfa - átlátszóság
    - `<t>` argumentumok típusa
      - b, d, f, i, s, ub, ui, us
  - `glColor3f`
    - Intenzitás
  - `glColor3ub`
    - `glColor3ub(0, 255, 128) = RGB(0, 255, 128)`
- Ha minden vertex-re külön definiáljuk a színt, akkor mi lesz a köztes területtel?
  - 
  - `glShadeModel()`
    - `GL_FLAT` – utolsó vertex színe
      - `GL_POLYGON` esetében az első vertex színe
    - `GL_SMOOTH`

48



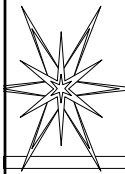


## Fény

Példa: jet.c

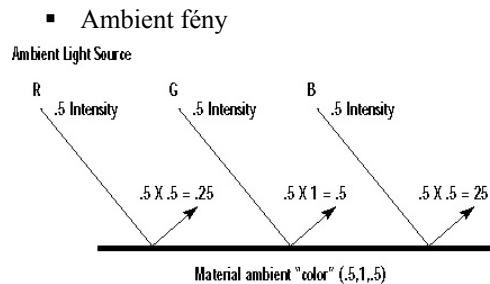
- A tárgyak színét sok minden befolyásolja
  - Ambiens – környező
    - Van forrás
    - A sugarak iránytalanok
    - Az objektumokat megvilágítva ilyen fényvel olyan, mint ha minden irányból minden felületet érne a fény
  - Diffúz - szórt
    - Bizonyos irányból jön
    - Egyenletesen szétszóródik a felszínen
  - Spekuláris - tükröződő
    - Irányított
    - Bizonyos irányokból élesen tükröződik
    - Fényes foltot hagy a felületen
- Egy egyszerű forrás előállítható a három típus segítségével

49

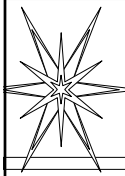


## Anyag

- Egy poligon piros
- A poligon egy olyan anyagból készült, amely a piros fényt tükrözi
  - Definiálni kell az anyag tükrözési tulajdonságait az ambiens, diffúz és a spekuláris fényforrásokra
  - Fény kibocsátási tulajdonság
    - Hátsó lámpa
    - ...



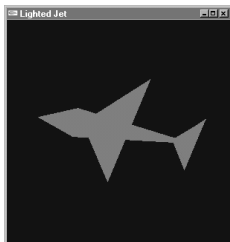
- Diffúz és Spekuláris
  - Az intenzitás függ a fény
    - Beesési szögétől
    - Távolságtól
    - Hígulási tényezők (kód)<sub>50</sub>



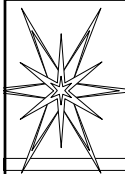
## Fény hozzáadás I.

- Megvilágítás engedélyezése
  - `glEnable(GL_LIGHTING)`
  - Használja az anyag tulajdonságokat, a fény paramétereiket a vertex-ek színének kiszámításánál
- Megvilágítási modell
  - `glLightModelfv(GLenum, const GLfloat *params)`
    - `GL_LIGHT_MODEL_AMBIENT`
    - `GL_LIGHT_MODEL_LOCAL_VIEWER`
    - `GL_LIGHT_MODEL_TWO_SIDE`
    - Default (0.2, 0.2, 0.2, 1.0)
  - Példa

```
GLfloat ambientLight[] =
{ 1.0f, 1.0f, 1.0f, 1.0f };
glEnable(GL_LIGHTING);
glLightModelfv(
GL_LIGHT_MODEL_AMBIENT,
ambientLight);
```



51

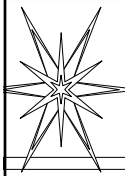


## Fény hozzáadás II.

- Anyag tulajdonságok
  - `glMaterialfv(GLenum, GLenum, 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`
      - `GL_COLOR_INDEXES`
- Példa

```
GLfloat gray[] = {
0.75f, 0.75f, 0.75f, 1.0f };
glMaterialfv(GL_FRONT,
GL_AMBIENT_AND_DIFFUSE,
gray);
glBegin(GL_TRIANGLES);
glVertex3f(-15.0f, 0.0f, 30.0f);
glVertex3f(0.0f, 15.0f, 30.0f);
glVertex3f(0.0f, 0.0f, -56.0f);
glEnd();
```

52

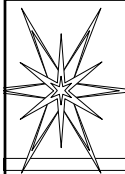


## Fény hozzáadás III.

- Color tracking
  - Az anyagi tulajdonságokat csak `glColor` hívásával állítjuk be
  - `glEnable(GL_COLOR_MATERIAL)`
  - `glColorMaterial(GLenum, GLenum)`
    - face
      - GL\_FRONT
      - GL\_BACK
      - GL\_FRONT\_AND\_BACK
- mode
  - GL\_EMISSION
  - GL\_AMBIENT
  - GL\_DIFFUSE
  - GL\_SPECULAR
  - GL\_AMBIENT\_AND\_DIFFUSE
- Példa

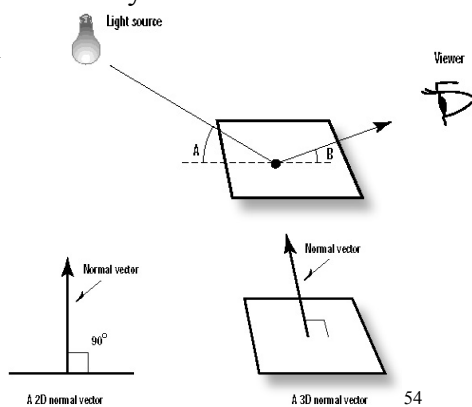
```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT,
GL_AMBIENT_AND_DIFFUSE);
glColor3f(0.75f, 0.75f, 0.75f);
glBegin(GL_TRIANGLES);
glVertex3f(-15.0f, 0.0f, 30.0f);
glVertex3f(0.0f, 15.0f, 30.0f);
glVertex3f(0.0f, 0.0f, -56.0f);
glEnd();
```

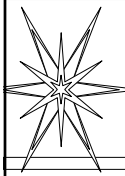
Példa: `ambient.c`



## Fényforrások használata

- Az OpenGL legalább 8 független fényforrást támogat
  - A fényforrást elhelyezhetjük egy végtelen távoli pontban
  - Az objektumhoz közel is elhelyezhetjük
  - Reflektor forrást adhatunk meg
    - Tölcsér
    - Karakterisztikák
      - Melyik irányban
- Melyik a felfele mutató irány?



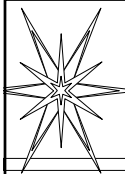


## Normál egységvektorok

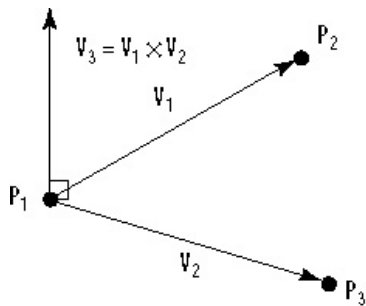
- `glNormal3f(GLfloat, GLfloat, GLfloat)`
  - A normál vektorát definiálja a következő vertex-nek vagy halmaznak
- `glEnable(GL_NORMALIZE)`
  - Automatikusan egységnyire normalizálja a vektorokat
  - Sebesség csökkenés
- `glScale()`
  - Módosítja a normálisok hosszát
- `glEnable(GL_RESCALE_NORMALS)`
  - OpenGL 1.2
  - A normal vektorok nem egységnyiek, de skálázhatóak egy értékkel (`glScale`)
  - Kevesebb műveletet kell végrehajtani
- Példa
 

```
void ReduceToUnit(float vector[3]) {
    float length;
    length = (float)sqrt((vector[0]*vector[0]) +
        (vector[1]*vector[1])+(vector[2]*vector[2]));
    if (length == 0.0f) length = 1.0f;
    vector[0] /= length;
    vector[1] /= length;
    vector[2] /= length;
}
```

55



## Egységvektorok kiszámítása

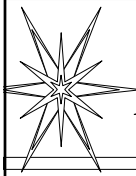


- Példa
 

```
void calcNormal(float v[3][3], float out[3]) {
    float v1[3],v2[3];
    static const int x = 0;
    static const int y = 1;
    static const int z = 2;
    v1[x] = v[0][x] - v[1][x];
    v1[y] = v[0][y] - v[1][y];
    v1[z] = v[0][z] - v[1][z];
    v2[x] = v[1][x] - v[2][x];
    v2[y] = v[1][y] - v[2][y];
    v2[z] = v[1][z] - v[2][z];
    out[x] = v1[y]*v2[z] - v1[z]*v2[y];
    out[y] = v1[z]*v2[x] - v1[x]*v2[z];
    out[z] = v1[x]*v2[y] - v1[y]*v2[x];
    ReduceToUnit(out);}

```

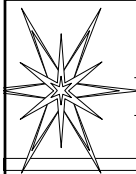
56



## A fényforrás beállítása

- `glLightfv(GLenum, GLenum, const GLfloat *)`
    - `light`
      - `GL_LIGHT0 ... GL_MAX_LIGHTS`
    - `pname`
      - `GL_AMBIENT`
      - `GL_DIFFUSE`
      - `GL_SPECULAR`
      - `GL_POSITION`
      - `GL_SPOT_DIRECTION`
      - `GL_SPOT_EXPONENT`
      - `GL_SPOT_CUTOFF`
      - `GL_CONSTANT_ATTENUATION`
      - `GL_LINEAR_ATTENUATION`
      - `GL_QUADRATIC_ATTENUATION`
  - Példa

```
GLfloat ambientLight[] = {0.3f,
0.3f, 0.3f, 1.0f};
GLfloat diffuseLight[] = {0.7f,
0.7f, 0.7f, 1.0f};
GLfloat lightPos = {-50.0f,
50.0f, 100.0f, 1.0f };
glLightfv(GL_LIGHT0,
GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0,
GL_DIFFUSE, diffuseLight);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0,
GL_POSITION, lightPos);
```
- Példa: litjet.c 57

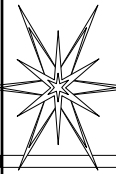


## Fényhatások - Tükröződés

- Tükröződő fény
    - Példa

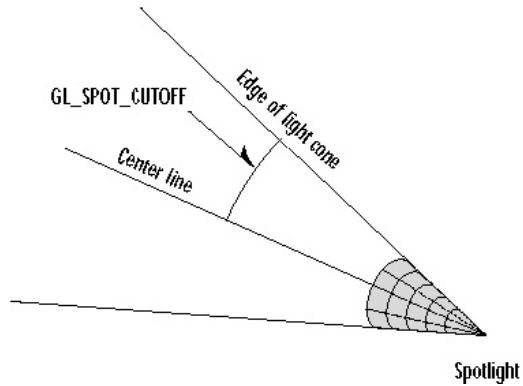
```
GLfloat ambientLight[] = { 0.3f,
0.3f, 0.3f, 1.0f };
GLfloat diffuseLight[] = { 0.7f,
0.7f, 0.7f, 1.0f };
GLfloat specular[] = { 1.0f, 1.0f,
1.0f, 1.0f};
...
glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT0,
GL_AMBIENT,ambientLight);
glLightfv(GL_LIGHT0,
GL_DIFFUSE,diffuseLight);
glLightfv(GL_LIGHT0,
GL_SPECULAR,specular);
glEnable(GL_LIGHT0);
```
  - Tükröződő visszaverődés
    - Példa

```
GLfloat specref[] = { 1.0f, 1.0f,
1.0f, 1.0f };
...
glEnable(
GL_COLOR_MATERIAL);
glMaterialfv(GL_FRONT,
GL_SPECULAR,specref);
glMateriali(GL_FRONT,
GL_SHININESS,128);
```
    - `glMateriali(GLenum, GLenum, GLint)`
      - 1-128
- Példa: shinyjet.c 58



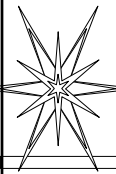
## Reflektorfény

- `GLfloat spotDir[] = { 0.0f, 0.0f, -1.0f };`
- `glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spotDir);`
  - irány
- `glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 60.0f);`
  - szög
- `glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 100.0f);`
  - $(\max \{p \cdot d, 0\})^e$



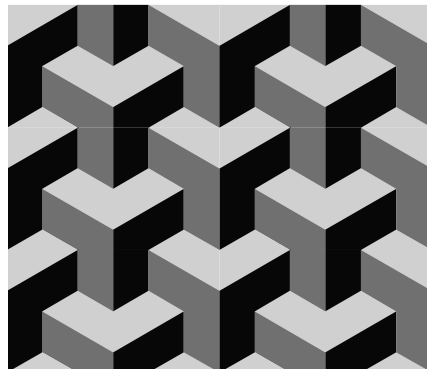
Példa: spot.c

59



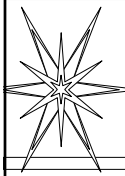
## Textúrák

- Képek feszítése a poligonokra
- Nagy számításigényű
- Textúra ~ kép
  - 1D-s textúra
  - 2D Windows .bmp
  - 3D (volume) MRI
    - 256x256x256
    - OpenGL 1.2



Példa: tex1d.c

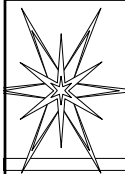
60



## Textúrák definiálása

- 1D textúrák definiálása
  - `void glTexImage1D(target, level, components, width, border, format, type, pixels)`
    - `target`: GL\_TEXTURE\_1D
    - `level`: szint, alap 0, (mipmap)
    - `components`:
      - színek száma 1, 3, 4
    - `width`: méret, 2 hatványa
    - `border`: keret pixelek száma 0, 1, 2
    - `format`: szín típusa GL\_RGB, GL\_COLOR\_INDEX
    - `type`: GL\_BYTE, GL\_FLOAT
    - `pixels`: GLvoid \*
- 2D textúrák definiálása
  - `void glTexImage2D(target, level, components, width, height, border, format, type, pixels)`
    - `height`: magasság, 2 hatvány

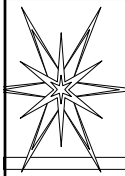
61



## Textúrák

- Textúra módok
  - GL\_MODULATE
    - Hozzáigazítja a textúrát világítás és szín információhoz
    - Legtöbbször ezt használjuk
  - GL\_DECAL
    - Nem változik a textúra
  - GL\_BLEND
    - Egy vagy két komponensű textúra esetén használható
    - A textúra kép keveredik:
      - textúra színnel,
      - megvilágítási és szín információkkal.
- Példa
  - `glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, mode);`
  - `GLfloat rgba[4];`
  - `glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, rgba);`

62



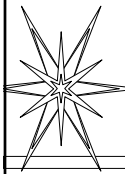
## Textúra filterek

- Textúra filterek
  - Pixelek interpolálása
  - `GL_TEXTURE_MIN_FILTER`
    - Poligon kisebb mint a textúra kép
  - `GL_TEXTURE_MAG_FILTER`
    - Poligon nagyobb mint a textúra kép
  - Filterek
    - `GL_NEAREST`, `GL_LINEAR`,  
`GL_NEAREST_MIPMAP_NEAREST`,  
`GL_NEAREST_MIPMAP_LINEAR`,  
`GL_LINEAR_MIPMAP_NEAREST`,  
`GL_LINEAR_MIPMAP_LINEAR`

- Példa

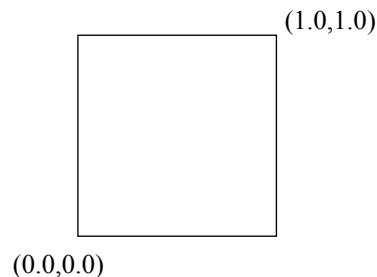
```
glTexParameteri(
GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,
filter);
glTexParameteri(
GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER,
filter);
```

63



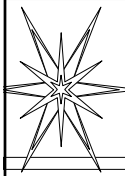
## Textúrák koordináták

- Textúra koordináták
  - *glTexCoord1d(GLdouble)*  
*glTexCoord1f(GLfloat)*  
*glTexCoord1i(GLint)*
  - *glTexCoord2d(GLdouble, GLdouble)*  
*glTexCoord2f(GLfloat, GLfloat)*  
*glTexCoord2i(GLint, GLint)*



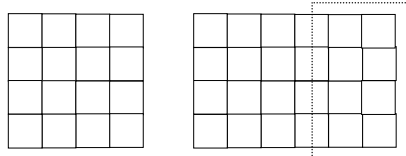
64



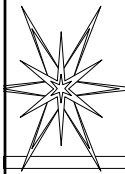


## Textúrák

- Textúra csomagolás
  - A textúra koordináták 0.0-1.0 között vannak
  - Koordináták kívül kerülnek
    - összekapcsolódnak
    - ismétlődnek
  - További keret (border) pixelek:
    - a keret pixeleket használja majd
- `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_S, wrap)`
- `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_T, wrap)`
- `wrap`
  - `GL_CLAMP`
  - `GL_REPEAT`



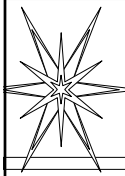
65



## Textúra példa (2D)

- `Glubyte *bits`  
`BITMAPINFO *info`  
  
`bits = LoadDIBITMAP("pot.bmp", &info);`  
`glTexImage2D(GL_TEXTURE_2D, 0, 3, info->bmiHeader.biWidth,`  
`info->bmiHeader.biHeight, 0, GL_BGR_EXT,`  
`GL_UNSIGNED_BYTE, bits);`  
  
`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);`  
`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);`  
  
`glEnable(GL_TEXTURE_2D);`  
  
`glColor3f(1.0, 1.0, 1.0);`  
`gluSolidTeapot(10.0);`

66



## Textúrák

- Több textúra
  - Textúra objektumok
    - Számokkal való azonosítás
    - Memóriában lehet tartani őket
  - Kezelésük
    - `glGenTextures`
    - `glBindTexture`
      - textúra műveletek eltárolódnak az adott objektumban
    - `glDeleteTexture`

- Példa

```
GLuint tex_obj;
```

```
glGenTextures(1, &text_obj);
```

```
glBindTexture(
```

```
GL_TEXTURE_ID, tex_obj);
```

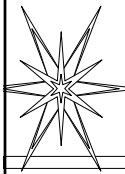
```
...
```

```
glTexImage1D(...);
```

```
...
```

```
glDeleteTexture(1, &tex_obj);
```

67



## Textúrák

- Automatikusan a memóriában tartja a nemrégén használt textúrákat
- Nem hatékony
  - 8 MB textúra memória ↔ 30 MB textúra adat
- `glPrioritizeTextures()`
  - 0.0 legalacsonyabb
  - 1.0 legmagasabb

- Példa

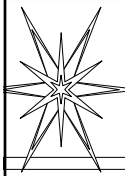
```
GLuint text_objs[5];
```

```
GLclampf tex_priors[5];
```

```
glPrioritizeTextures(5,
```

```
tex_objs, tex_priors);
```

68

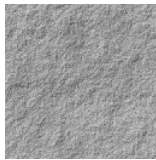


## Textúrák

- Egyszerű textúra beolvasó

- texture.c
- bitmap.c

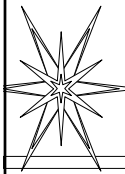
Példa: terrain.c



- Példa

```
glGenTextures(1, &texture);
glBindTexture(type, texture);
glTexParameterf(type,
GL_TEXTURE_MAG_FILTER,
magfilter);
glTexParameterf(type,
GL_TEXTURE_MIN_FILTER,
minfilter);
glTexParameterf(type,
GL_TEXTURE_WRAP_S, wrap);
glTexParameterf(type,
GL_TEXTURE_WRAP_T, wrap);
glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,
GL_MODULATE);
```

69



## Textúrák

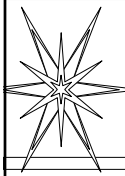
- Megvilágítás és textúrák

- GL\_MODULATE, GL\_BLEND
- GL\_DECAL
- Normálisok beállítása
- Példa: terrain5.c

- Tükröződő fény

- Világos pixel-t kapunk és nem fehérét
- Kétszer kell kirajzolni az objektumokat
  - Textúra nélkül
  - Textúrával
- Blending később
- Példa: specular.c

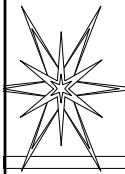
70



## Vizuális hatások

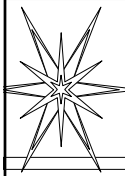
- Blending – keveredés
  - Pixel szintű vezérlés az RGBA szín tárolásánál a szín pufferben
  - Nem használható indexelt szín módban és indexelt színű ablakokban
  - `glEnable(GL_BLEND)`  
`glDisable(GL_BLEND)`
  - `glBlendFunc(GLenum source, GLenum target)`
    - `GL_ONE, GL_ZERO = Disable(GL_BLEND)`
    - `Source` = glColor-ral beállított értékek
    - `Target` = color pufferben lévő érték

71



## Vizuális hatások

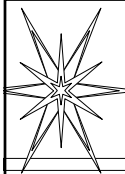
- Átlátszóság
  - `glEnable(GL_BLEND);`  
`glBlendFunc(GL_SRC_ALPHA,`  
`GL_MINUS_SRC_ALPHA);`
  - $Rd = Rs * As + Rd * (1 - As)$   
 $Gd = Gs * As + Gd * (1 - As)$   
 $Bd = Bs * As + Bd * (1 - As)$
  - A forrás alfa komponensét használjuk
    - Nincs szükség olyan kártyára, amely támogatja az alfa színsíkot a szín pufferben
  - A mélységi puffer teszt meggátolhat a megfelelő eredmény elérésében
    - Az átlátszó objektumokat hátulról kezdjük kirajzolni és a nem<sub>72</sub> átlátszó objektumok után



## Blending függvények

- |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>▪ <b>Forrás színre</b><ul style="list-style-type: none"><li>▪ <code>GL_ZERO</code><ul style="list-style-type: none"><li>▪ 0,0,0</li></ul></li><li>▪ <code>GL_ONE</code><ul style="list-style-type: none"><li>▪ a forrás szín nem változik</li></ul></li><li>▪ <code>GL_DST_COLOR</code><ul style="list-style-type: none"><li>▪ a forrás színt megszorozza a cél pixel színével</li></ul></li><li>▪ <code>GL_ONE_MINUS_DST_COLOR</code><ul style="list-style-type: none"><li>▪ a forrás színt megszorozza a (1,1,1,1-cél szín) értékkel</li></ul></li><li>▪ <code>GL_SRC_ALPHA</code><ul style="list-style-type: none"><li>▪ a forrás színt megszorozza a forrás alfa értékével</li></ul></li><li>▪ <code>GL_ONE_MINUS_SRC_ALPHA</code></li></ul></li></ul> | <ul style="list-style-type: none"><li>▪ <b>Cél színre</b><ul style="list-style-type: none"><li>▪ <code>GL_ZERO</code><ul style="list-style-type: none"><li>▪ 0, 0, 0</li></ul></li><li>▪ <code>GL_ONE</code><ul style="list-style-type: none"><li>▪ A cél szín nem változik</li></ul></li><li>▪ <code>GL_SRC_COLOR</code><ul style="list-style-type: none"><li>▪ Cél színt megszorozza a forrás pixel színével</li></ul></li><li>▪ <code>GL_ONE_MINUS_SRC_COLOR</code><ul style="list-style-type: none"><li>▪ A cél színt megszorozza a (1,1,1,1-forrás szín) értékkel</li></ul></li><li>▪ <code>GL_SRC_ALPHA</code><ul style="list-style-type: none"><li>▪ A cél színt megszorozza a forrás alfa értékével</li></ul></li><li>▪ <code>GL_ONE_MINUS_SRC_ALPHA</code></li></ul></li></ul> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
- `GL_DST_ALPHA`
  - `GL_ONE_MINUS_DST_ALPHA`
  - `GL_SRC_ALPHA_SATURATE`
    - a min[forrás szín, (1-dest\_alpha)]
- `GL_DST_ALPHA`
  - `GL_ONE_MINUS_DST_ALPHA`
  - `GL_SRC_ALPHA_SATURATE`

73

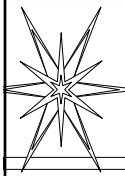


## Vizuális hatások

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glPushMatrix();
glTranslatef(0.0, 0.0, -15.0);
glRotatef(-45.0, 0.0, 1.0, 0.0);
glDisable(GL_BLEND);
glColor3f(1.0, 1.0, 0.0);
glutSolidTeapot(1.0);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0, 0.0, -10.0);
glRotatef(45.0, 0.0, 1.0, 0.0);
glEnable(GL_BLEND);
glColor4f(1.0, 1.0, 1.0, 0.25);
glutSolidTeapot(1.0);
glPopMatrix();
```

Példa: blendpot.c





## Vizuális hatások

### ▪ Köd

#### ▪ *glFogi(GLenum, params)*

##### ▪ GL\_FOG\_MODE

▪ GL\_LINEAR  
(mélység árnyékolás)

▪ GL\_EXP  
(erős köd és felhő)

▪ GL\_EXP2  
(füst és pára)

##### ▪ GL\_FOG\_COLOR

▪ Szín, RGBA tömb

##### ▪ GL\_FOG\_START

▪ A legközelebbi pont, ahol a ködöt alkalmazni kell

##### ▪ GL\_FOG\_END

▪ A legtávolabbi pont, amittől a legnagyobb értékkel állítja elő a ködöt

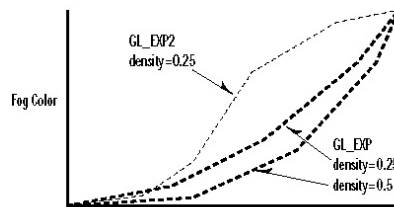
##### ▪ GL\_FOG\_DENSITY

▪ 0.0-nál nagyobb

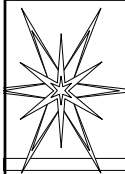
▪ Tipikusan 0.1-nél kisebb

##### ▪ GL\_FOG\_INDEX

▪ Indexelt szín módban a köd színének a sorszám



75



## Vizuális hatások

### ▪ Példa

```
static float fog_color[4] = {0.0, 0.0, 0.0, 0.0};
```

```
glEnable(GL_DEPTH_TEST);
```

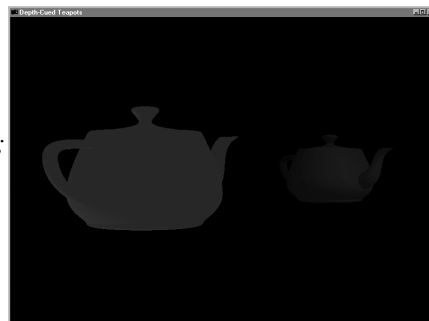
```
glEnable(GL_FOG);
```

```
glFogfv(GL_FOG_COLOR, fogcolor);
```

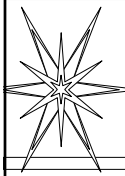
```
glFogi(GL_FOG_MODE, GL_LINEAR);
```

```
glFogf(GL_FOG_START, 10.0);
```

```
glFogf(GL_FOG_END, 20.0);
```

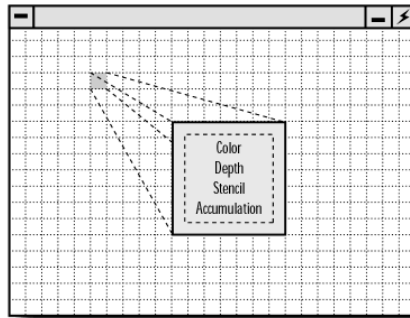


76

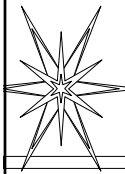


## Pufferek

- Mindegyik puffer sajátos képességgel bír
  - Color
  - Depth
  - Stencil
  - Accumulation
- Két dimenziós tömbök
  - A sorok és oszlopok száma megegyezik az ablak kliens területén lévővel
  - Más tagoltsággal illetve típussal rendelkeznek



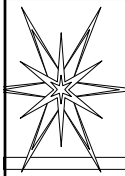
77



## A pufferek beállítása GLUT-ban

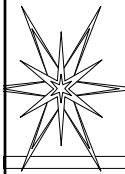
- Puffer kiválasztása
  - `void glutInitDisplayMode(mode)`
    - `GLUT_RGB`, `GLUT_RGBA`
    - `GLUT_INDEX`
    - `GLUT_SINGLE`
    - `GLUT_DOUBLE`
    - `GLUT_ACCUM`
    - `GLUT_ALPHA`
    - `GLUT_DEPTH`
    - `GLUT_STENCIL`
    - `GLUT_MULTISAMPLE`
    - `GLUT_STEREO`
    - `GLUT_LUMINANCE`
- Csak a szükséges puffereket válasszuk ki
  - Korlátozott memória méret
  - Ha olyan kombinációt (Microsoft) választunk ki, amit a videó kártya nem támogat, akkor a szoftveres implementáció „kihagyja” a hardveresen gyorsított funkciókat

78



## Pufferek

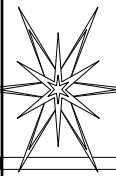
- Szín puffer
  - Pixel szín információk
  - A legközelebbi szín
  - Az általános Microsoft OpenGL nem támogatja az alfa komponenst
- Dupla Puffer
  - Animáció
  - Csak a szín pufferre van hatása
    - Nem kapunk még egy mélység puffert vagy összegző puffert
  - Egy pixel formátum kiválasztása dupla pufferelemmel → OpenGL kiválaszt egy háttér puffert rajzolásra
  - *glDrawBuffer(GLenum)*
    - GL\_FRONT (látható)
    - GL\_BACK (láthatatlan)
    - GL\_FRONT\_AND\_BACK



## Pufferek

- Mélység puffer
  - Távolság értékek
  - Mindegyik érték a pixelek távolságát jelenti a nézőponttól
  - 16, 32 bites értékek
  - Nem látható felület eltávolítására használjuk általában
  - *glEnable(GL\_DEPTH\_TEST)*
  - *glDisable(GL\_DEPTH\_TEST)*
- Mélység összehasonlítás
  - A pixelek z pozícióját összehasonlítja a mélység puffer értékével
  - Ha az összehasonlítás igaz, akkor a pixel a mélység mentén van tárolva
  - *glDepthFunc()*
  - GL\_LESS alapértelmezés
    - Igaz, ha a forrás z kisebb a mélység z értékénél





## Pufferek

- Összehasonlítás
  - GL\_NEVER
  - GL\_LESS
  - GL\_EQUAL
  - GL\_LEQUAL
  - GL\_GREATER
  - GL\_NOTEQUAL
  - GL\_GEQUAL
  - GL\_ALWAYS
- Mélység értékek
  - Leképezett Z koordináták,
  - A csonka nézet elejétől a nézet hátsó lapjáig megy [0.0, 1.0]
  - GL\_EQUAL, GL\_NOTEQUAL esetén néha szükség van a mélység értékek tartományának a megváltoztatására
  - *GLfloat near, far;*  
*glDepthRange(near, far);*
  - Iniciális mélységi értékek beállítása  
*GLfloat depth;*  
*glClearDepth(depth);*

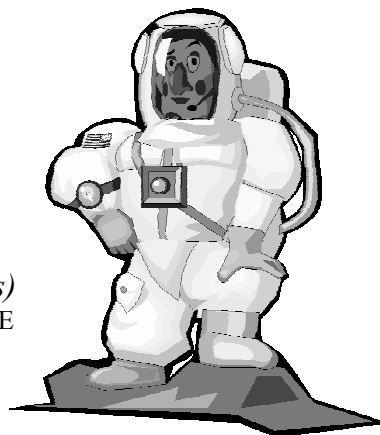
Példa: glut.c

81

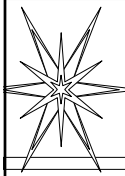


## Vonalakon és háromszögeken túl

- Másodfokú felület
  - *gluQuadricOrientation(*  
*GLUquadricObj \*obj,*  
*GLenum orientation)*
    - GLU\_OUTSIDE,
    - GLU\_INSIDE
  - *gluQuadricTexture(*  
*GLUquadricObj \*obj,*  
*GLboolean textureCoords)*
    - GLU\_TRUE, GLU\_FALSE



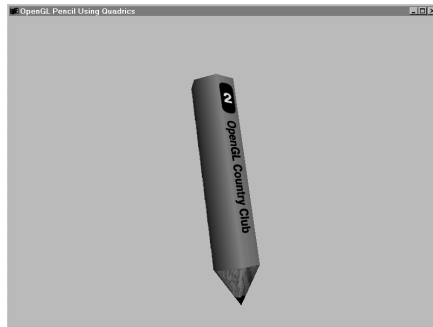
82



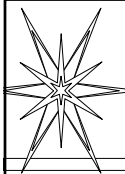
## Hengerek rajzolása

- `void gluCylinder(  
GLUquadricObj *obj,  
GLdouble baseRadius,  
GLdouble topradius,  
GLdouble height, GLint slices,  
GLint stacks)`
  - *slices*: hány részből áll a henger körben
  - *stacks*: felosztás (alulról felfele), reflektor fény esetén
  - Textúra
    - Elülső éltől helyezi rá (0, R, 0)
    - A képnek fejfelé lefele kell állnia

- Kúp rajzolás
  - `gluCylinder()`
  - (top/bottom)Radius = 0



83

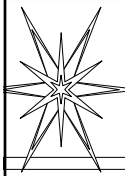


## Korong rajzolás

- `void gluDisk(  
GLUquadricObj *obj,  
GLdouble innerRadius,  
GLdouble outerRadius,  
GLint slices, GLint loops)`
  - *innerRadius*
    - belső üres terület
  - *Loops*
    - 1 - kör,
    - 2 – csavaralátét
  - Textúra
    - A textúra teteje a korong tetejére kerül

- Korong részlet
  - `void gluPartialDisk(  
GLUquadricObj *obj,  
GLdouble innerRadius,  
GLdouble outerRadius,  
GLint slices,  
GLint loops,  
GLdouble startAngle,  
GLdouble sweepAngle)`
    - *startangle*
      - clockwise szög fokban
    - *sweepangle*
      - körszelet szöge

84

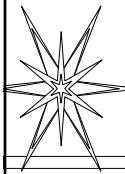


## Gömb rajzolása

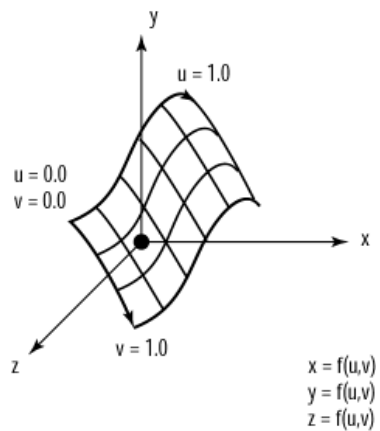
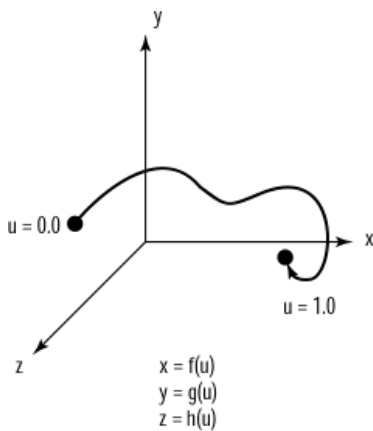
- `void gluSphere(`  
`GLUquadric *obj,`  
`GLdouble radius,`  
`GLint slices, GLint stacks)`
- Textúrák
  - A szélességi és hosszúsági koordinátákat használja. A világ térkép tökéletesen ráilleszkedik.



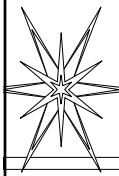
85



## Görbék és felületek

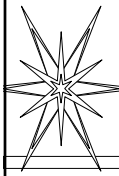
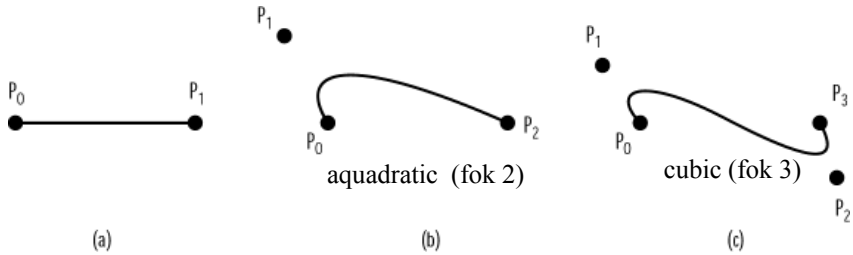


86

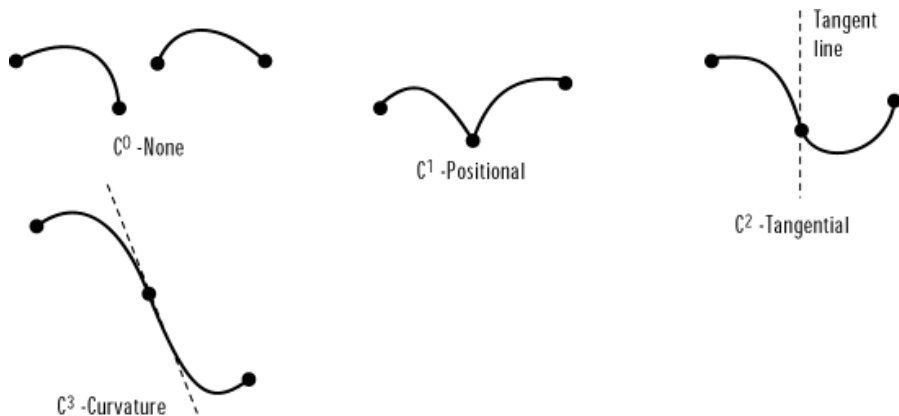


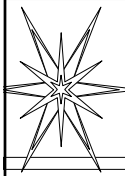
# Görbék és felületek

- Rang
  - A kontrol pontok száma
- Fok = Rang - 1



# Folytonosság



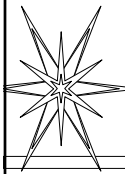


## Kiértékelők

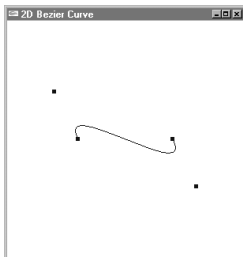
- Bézier görbék rajzolása
  - Kontrol pontok
  - Parametrikus  $u$  és  $v$
- Alkalmas kiértékelő fv.
- Generálja a pontokat, amelyek a görbét vagy a felszínt alkotják



89



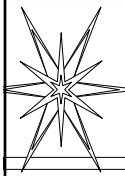
## 2D-s görbe I.



Példa: bezier.c

- ```
void RenderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMap1f(GL_MAP1_VERTEX_3, //Type of data generated
    0.0f, // Lower u range
    100.0f, // Upper u range
    3, // Distance between points in the data
    nNumPoints, // number of control points
    &ctrlPoints[0][0]); // array of control points
    glEnable(GL_MAP1_VERTEX_3);
    glBegin(GL_LINE_STRIP);
    for(i = 0; i <= 100; i++) {
        glEvalCoord1f((GLfloat) i);
    }
    glEnd();
    DrawPoints();
    glutSwapBuffers();
}
```

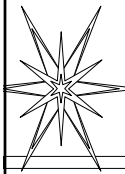
90



## 2D-s görbe II.

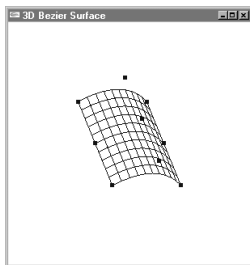
- `glMapGrid1f(GLint, GLfloat, GLfloat)`
  - un: felosztás száma
  - u1,u2: alsó és felső értékek u irányban
- `glEvalMesh1(GLenum, GLint, GLint)`
  - mode: GL\_POINT, GL\_LINE, GL\_FILL
  - i1,i2: első és utolsó integer érték az u tartományra
- ```
void RenderScene(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glMap1f(GL_MAP1_VERTEX_3,  
            100.0f,  
            100.0f,  
            3,  
            nNumPoints,  
            &ctrlPoints[0][0]);  
    glEnable(GL_MAP1_VERTEX_3);  
    glMapGrid1d(100,0.0,100.0);  
    glEvalMesh1(GL_LINE,0,100);  
    DrawPoints();  
    glutSwapBuffers();  
}
```

91

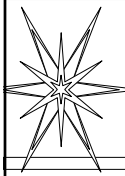


## 3D-s felszín

- Hasonló a 2D-s görbékhez
  - u és v tartományon definiálni kell a pontokat
- ```
void RenderScene(void) {  
    glClear(GL_COLOR_BUFFER_BIT);  
    glMatrixMode(GL_MODELVIEW); glPushMatrix();  
    glRotatef(45.0f, 0.0f, 1.0f, 0.0f);glRotatef(60.0f, 1.0f, 0.0f, 0.0f);  
    glMap2f(GL_MAP2_VERTEX_3, // Type of data generated  
            0.0f, // Lower u range  
            10.0f, // Upper u range  
            3, // Distance between points in the data  
            3, // Dimension in u direction (order)  
            0.0f, // Lower v range  
            10.0f, // Upper v range  
            9, // Distance between points in the data  
            3, // Dimension in v direction (order)  
            &ctrlPoints[0][0][0]); glEnable(GL_MAP2_VERTEX_3);  
    glMapGrid2f(10,0.0f,10.0f,10,0.0f,10.0f);  
    glEvalMesh2(GL_LINE,0,10,0,10);  
    DrawPoints();glPopMatrix();glutSwapBuffers();}
```



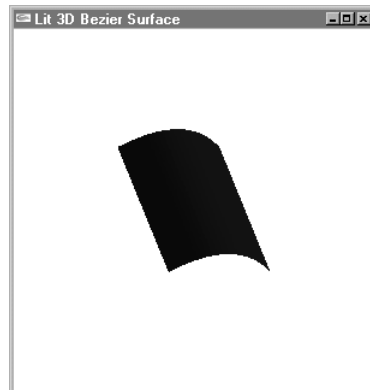
92



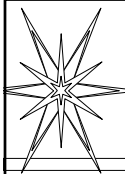
## Megvilágítás

- `glEvalMesh2(GL_FILL, 0, 10, 0, 10);`
- `glEnable(GL_AUTO_NORMAL)`
- Engedélyezni kell a megvilágítást

Példa: bezier3d.c

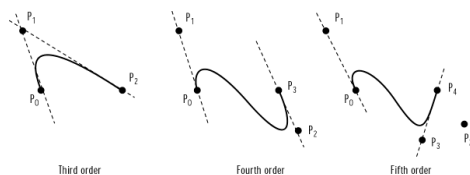


93



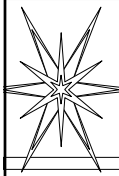
## Non Uniform Rational B-Spline

- Bézier görbe kvadratikus esetben elég sima
- Több pont esetén már simaság csökken



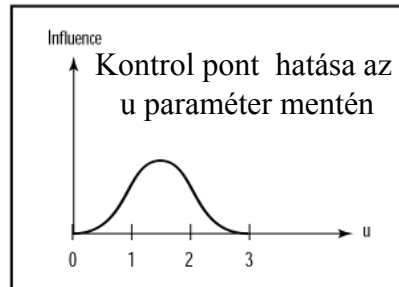
- B-spline-ok (bi-cubic splines)
- Szegmensekre van szétbontva a görbe
- Az adott szegmens alakját a legközelebbi 4 pont határozza meg
- Egy hosszabb görbe sok kontrol ponttal simább
- A csomópontokkal a szegmensek között C3 folytonosságot mutat
- A görbének nem feltétlenül kell átmennie bármelyik kontrol ponton

94

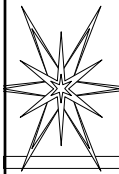


## Csomók

- Két csomópont értéket definiálunk minden kontrolponthoz
- Az értékek tartománya megegyezik az  $u$  és  $v$  értelmezéstartományával
- Nem csökkenőek
- A csomópontok meghatározzák a kontrolpontok hatását az  $u$  és  $v$  tartományon belül
- 0-3 közötti pontok vannak hatással a görbe alakjára



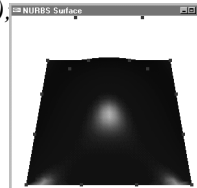
95



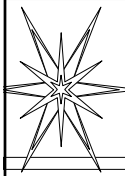
## NURBS felszín kreálása

- ```
GLUnurbsObj *pNurb = NULL;  
...  
pNurb = gluNewNurbsRender();  
...  
if (pNurb)  
gluDeleteNurbsRender(pNurb);
```
- NURBS tulajdonságok
  - ```
gluNurbsProperty(pNurb,  
GLU_SAMPLING_TOLERANCE,  
25.0f);
```
  - ```
gluNurbsProperty(pNurb,  
GLU_DISPLAY_MODE,  
(GLfloat)GLU_FILL);
```
- ```
gluBeginSurface(pNurb);  
gluNurbsSurface(pNurb,  
8, Knots, //No. knots and knots array u  
8, Knots, //No. knots and knots array v  
4*3, //distance between control p. in u  
3, //distance between control p. in v dir  
&ctrlPoints[0][0][0],  
4,4, //u and v order of surface  
GL_MAP2_VERTEX_3);  
gluEndSurface(pNurb);
```

Példa: nurbs.c



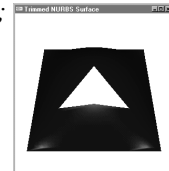




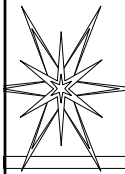
## Kivágás I.

- Kivágni egy szeletet egy NURBS felszínből
  - `gluBeginSurface(pNurb);`  
`gluNurbsSurface(pNurb,`  
`8, Knots, //No. knots and knots array u`  
`8, Knots, //No. knots and knots array v`  
`4*3, //distance between control p. in u`  
`3, //distance between control p. in v dir`  
`&ctrlPoints[0][0][0],`  
`4,4, //u and v order of surface`  
`GL_MAP2_VERTEX_3);`
- `glEndSurface(pNurb);`

- `// Outer area, include entire curve`  
`gluBeginTrim (pNurb);` ←
- `gluPwlCurve (pNurb, 5,`  
`&outsidePts[0][0], 2,`  
`GLU_MAP1_TRIM_2);`
- `gluEndTrim (pNurb);`
- `// Inner triangular area`  
`gluBeginTrim (pNurb);`
- `gluPwlCurve (pNurb, 4,`  
`&insidePts[0][0], 2,`  
`GLU_MAP1_TRIM_2);`
- `gluEndTrim (pNurb);`



Példa: nurbt.c <sup>97</sup>



## Kivágás II.

- `gluNurbsCurve(`  
`GLUnurbsObj *nObj, GLint nknots,`  
`GLfloat *knot, GLint stride, GLfloat`  
`*ctlArray, GLint order, GLenum type)`
  - nknots: csomók száma
  - knot: tömb
  - stride: eltolás a kontrol pontok között a görbén
  - ctlArray: kontrol pontok
  - order: NURBS felület rangja
  - type: GL\_MAP2\_VERTEX\_3, GL\_MAP2\_VERTEX\_4, ...

- `gluPwlCurve(`  
`GLUnurbsObj *nObj,`  
`GLint count,`  
`GLfloat *array,`  
`GLint stride,`  
`GLenum type)`
  - count: tömbben lévő elemek száma
  - array: határ pontok
  - stride: eltolás a pontok között a görbén
  - type: GLU\_MAP1\_TRIM\_2, GLU\_MAP1\_TRIM\_3 (w)