PPKE ITK

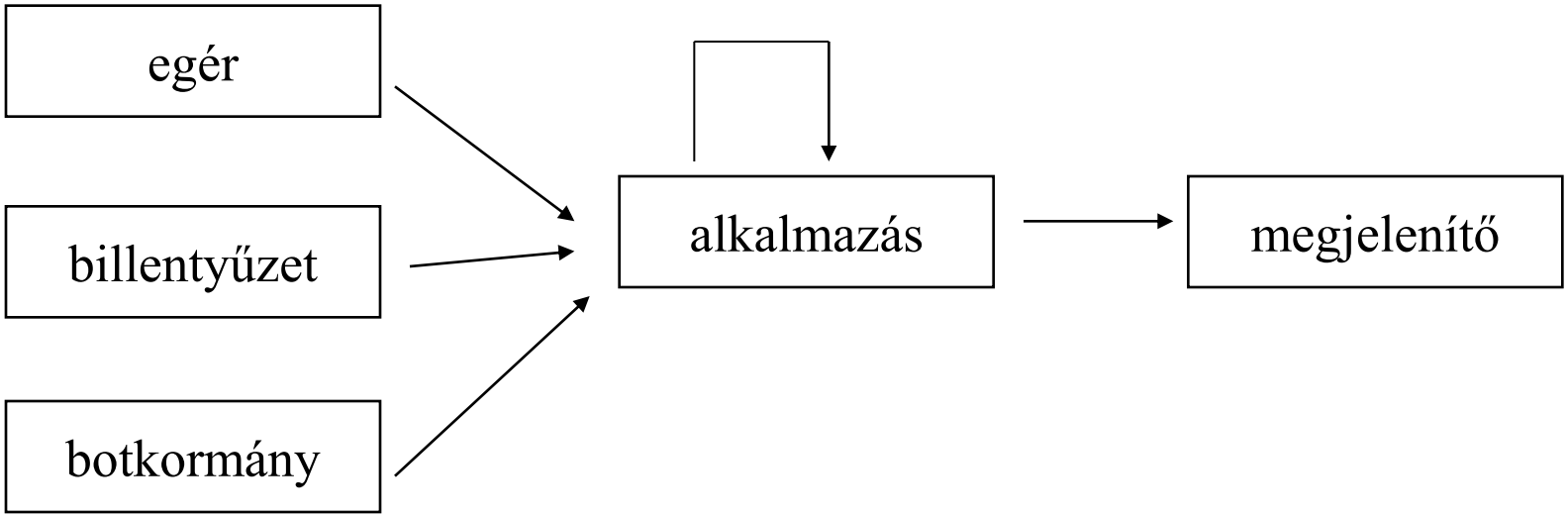
A számítógépes grafika alapjai

Grafikus szoftver alrendszer, OpenGL alapok

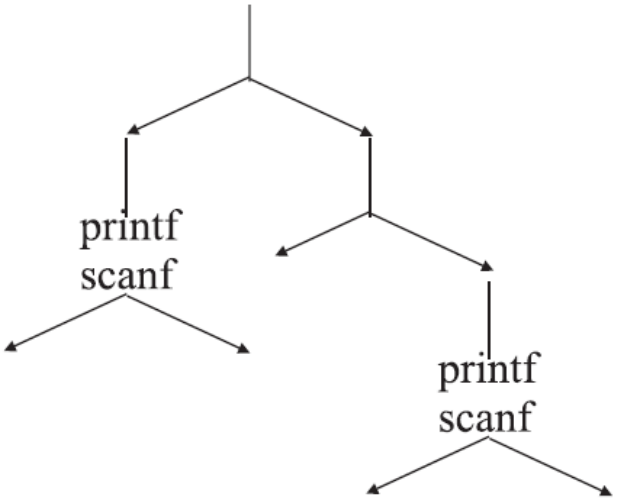
Előadó: Benedek Csaba

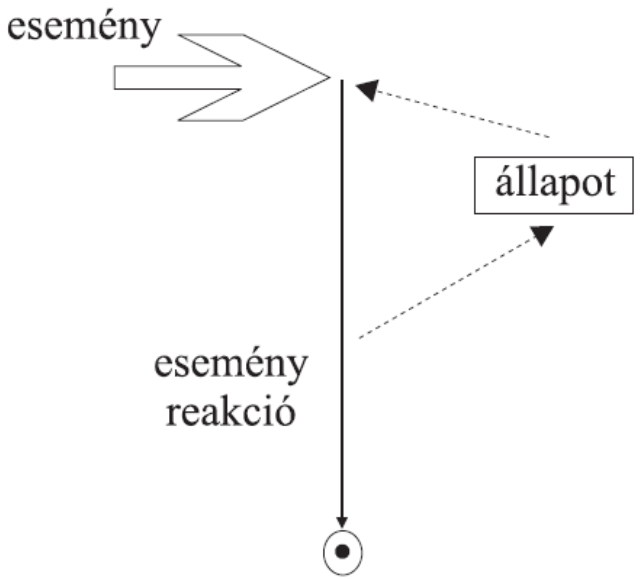
Tananyag : Szirmay-Kalos László Benedek Csaba

Interaktív program struktúra

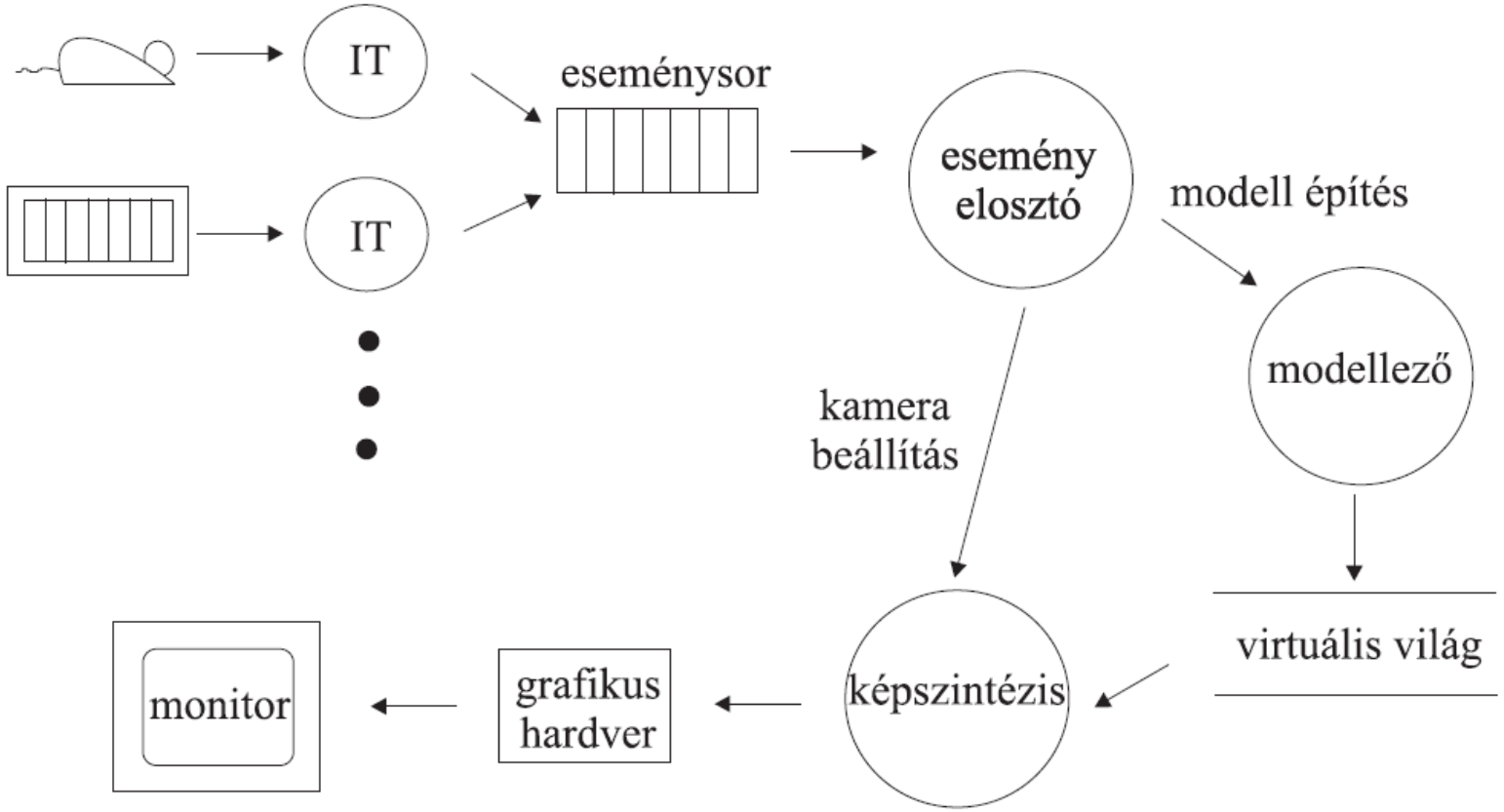


Programvezérelt vs eseményvezérelt programozás

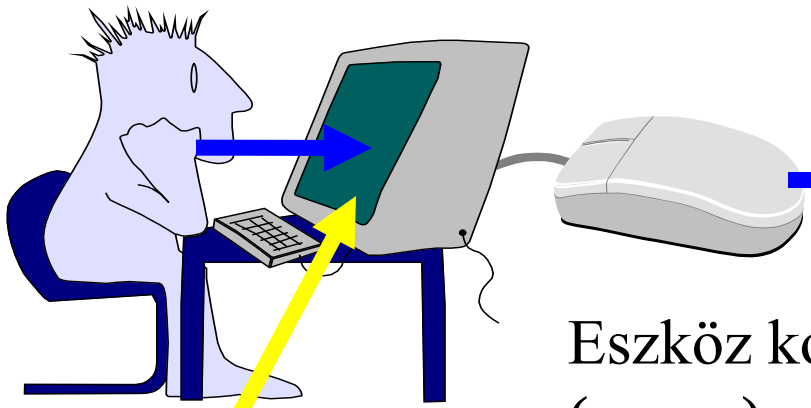




Interaktív program struktúra



2D grafikus rendszerek (funkcionális modell)



Bemeneti csővezeték

T_V^{-1}

T_M^{-1}

Eszköz-képtér

Képtér-világ

Világ-lokális mod.

Eszköz koord.
(xe, ye)

Kurzor
(xp, yp)

kamera

Virtuális világ

Kép frissités

rasztertár

Pixel műveletek

Raszteri-záció

Vágás

Világ-képtér

Modellezési transzf.

Kimeneti csővezeték

T_V

T_M

Grafikus hardver illesztése, programozása

- Grafikus könyvtárak
  - A grafikus hardver szolgáltatásait ezen keresztül lehet elérni
  - Hierarchikus rétegek, szabványosított interfészek
  - Fő irányelvek:
    - eszközfüggetlenség
    - rajzolási állapot
  - 2-D példák:
    - Ablak, menü, egér: Win32 API
    - Rajzolás: Ms-Windows GDI, GDI+, DirectDraw
  - 3-D példák: **OpenGL** (platformfüggetlen), DirectX (csak MS-Windows)

OpenGL

- Értelmező szótár:
  - Open = hardver és szoftver platform független
  - GL = Graphics Language
- Szül: 1992 Silicon Graphics (SG), GL néven
- OpenGL: közös specifikáció SG, IBM, Intel, MS stb
- Elérhetőség:
  - Futtatható verzió Windows operációs rendszer része
  - Ingyenes fejlesztői verzió
  - Végfelhasználói verzió: videokártya gyártók illesztő programjaiban implementált
- Szintaxis: gl{FunctionName}, pl `glClearColor()`

GLU (OpenGL Utilities)

- OpenGL-hez kapcsolódó szabványos kiegészítő csomag
  - Pl: hasznos programozást könnyítő rutinok transzformációs mátrixok beállítására, felületek teszcellációjára
  - Szintaxis: glu{FunctionName}, pl `gluPerspective()`

DirectX

- OpenGL alternatívája
- Számítógépes játékok többsége ezt használja a színtér megjelenítésére
- Csak Windows platformon használható, Windows XP-től kezdve az operációs rendszer szerves része
- C++, C#, VB stb támogatja az API szintű programozását

OpenGL vs DirectX

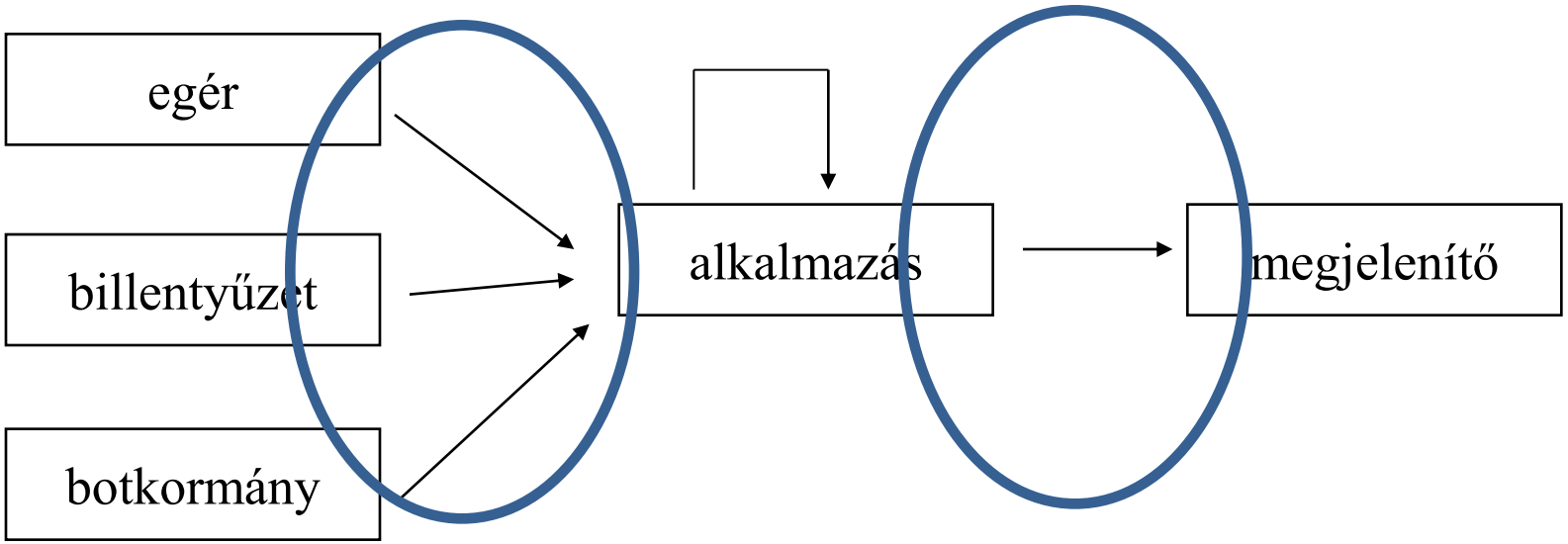
- Összehasonlítás: állandó vitaforrás
  - Melyik a gyorsabb - ez sokszor a gyártók által implementált illesztő programtól függ.
  - Melyik az egyszerűbben kezelhető – felhasználó és alkalmazás függő
- Tények:
  - OpenGL platformfüggetlen, DirectX nem
  - Számítógépes játékok 90%-a DirectX-t használ
  - DOOM III-at OpenGL-ben írták ☺

Mit NE várjunk el az OpenGL-től?

- Az OpenGL **NEM**
  - geometriai modellező rendszer
  - bemeneti eszközkezelő, ablakozó
  - interaktív rajzoló vagy grafikus rendszer
- Viszont **IGEN**
  - alkalmas különböző modellező, rajzoló rendszerek grafikai igényeinek a kielégítésére

Alkalmazás:

- OpenGL „csak” megjelenítést támogat, kell hozzá ablakozó, eseménykezelő rendszer
- Az OpenGL platformfüggetlen, de az alkalmazás ettől még nem!



- Eseménykezelés: „operációs rendszer” pl Ms-Windows vagy XWindow

- Ablakozás: „operációs rendszer” pl Ms-Windows vagy XWindow
- 3-D grafikus ábrázolás: OpenGL

Ablakkezelés Windows-ban (részlet)

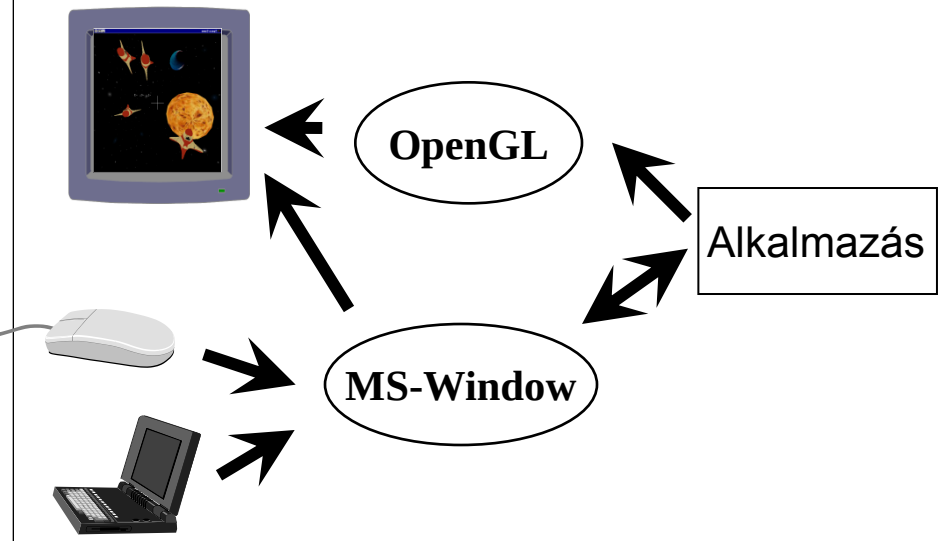
```
//---------------------------------------------------------------------
void InitWindowClass( HANDLE hInstance, HANDLE hPrevInstance ) {
//---------------------------------------------------------------------
        WNDCLASS wndclass;
         strcpy(szClassName, "grafika");
            if ( !hPrevInstance ) {
                        wndclass.style = CS_HREDRAW | CS_VREDRAW;
                        wndclass.lpfnWndProc = WndProc; // ablakkezel˝o f¨uggv´eny
                        wndclass.hInstance = hInstance; // program azonos´ıt´o
                        wndclass.hIcon = LoadIcon( hInstance, IDI_APPLICATION );
                        wndclass.hCursor = LoadCursor( NULL, IDC_ARROW );
                        wndclass.hbrBackground = GetStockObject( WHITE_BRUSH );
                        wndclass.lpszMenuName = "windowsmenu"; // men¨un´ev az er˝oforr´as f´ajlban
                        wndclass.lpszClassName = szClassName; // oszt´alyn´ev
                        wndclass.cbClsExtra = 0;
                        wndclass.cbWndExtra = 0;
                        if ( !RegisterClass( &wndclass ) ) exit( -1 );
            }
}
//---------------------------------------------------------------------
void InitWindow( HANDLE hInstance, int nCmdShow ) {
//---------------------------------------------------------------------
HWND hwnd = CreateWindow( szClassName, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
CW_USEDEFAULT, NULL, NULL, hInstance, NULL );
if ( ! hwnd ) exit( -1 );
ShowWindow(hwnd, nCmdShow ); // ablak megjelen´ıt´ese
UpdateWindow( hwnd ); // ´erv´enytelen´ıt´es
}
// egy Windows program itt indul
int PASCAL WinMain( HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow ) {
//---------------------------------------------------------------------
            InitWindowClass(hInstance, hPrevInstance);
            InitWindow( hInstance, nCmdShow );
            AppStart( );
            return 0;
}
```

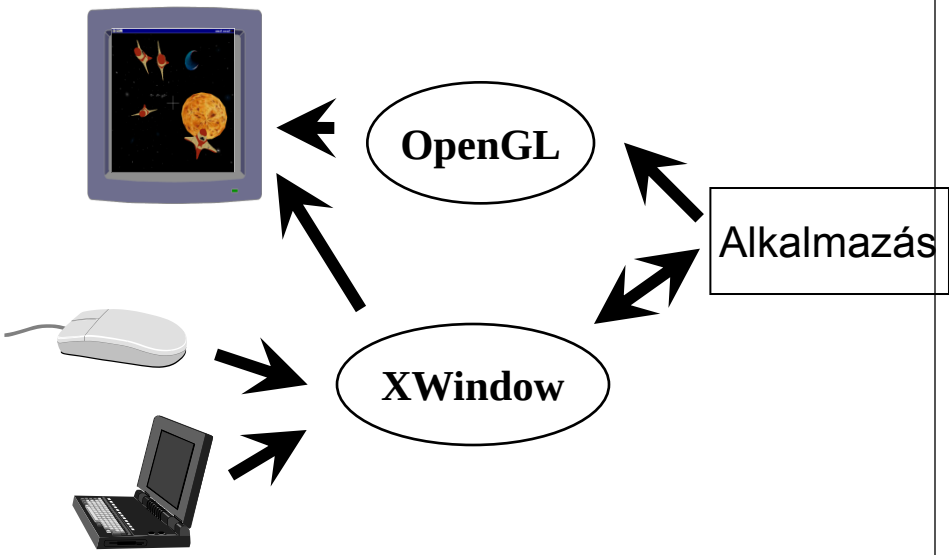
GLUT= OpenGL Utility Toolkit

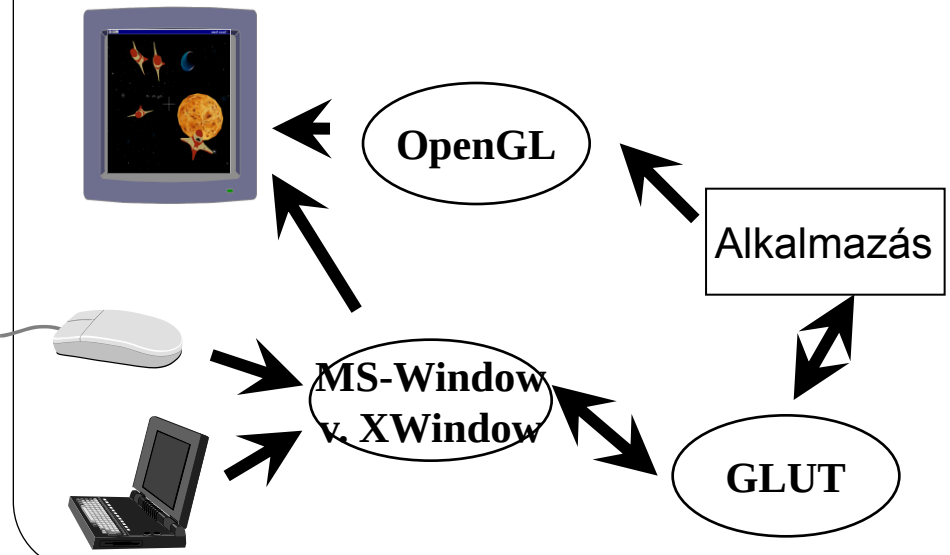
- platformfüggetlen eseménykezelés és ablakozás ☺
- egyszerű inteface ☺
- használja az MS-Windows vagy XWindow rutinokat, de erről az alkalmazás nem kell, hogy tudjon ☺
- csak kis méretű, egyszerű programok megírására alkalmas ☹

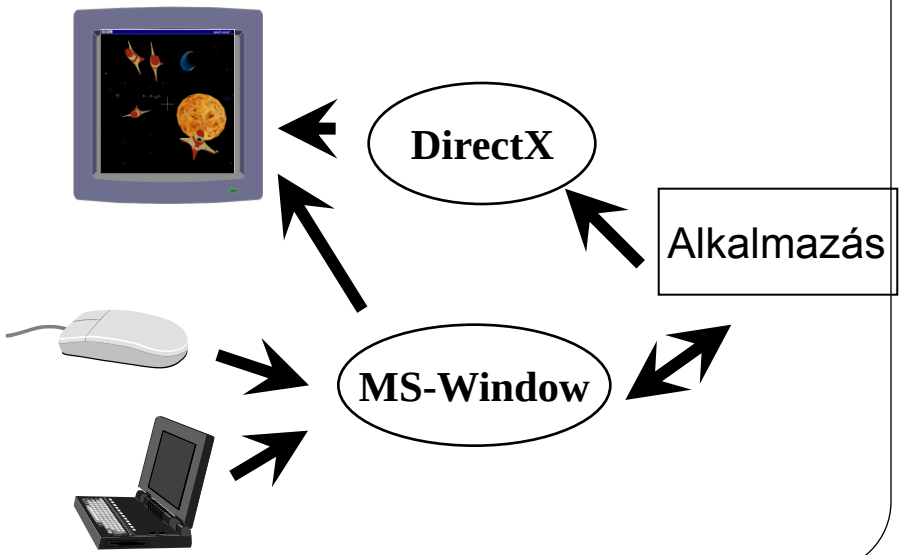
- Laboron ezt fogjuk használni!

Input/Output kezelés





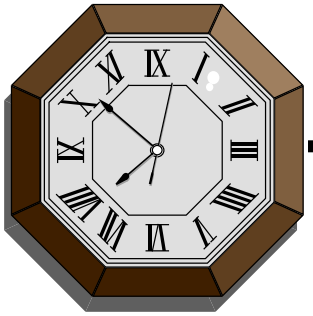


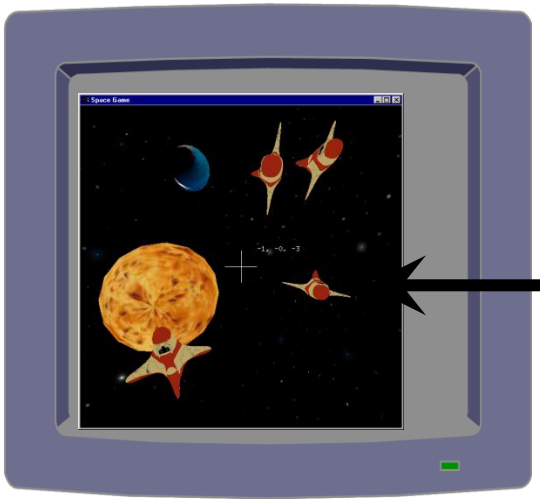


GLUT jellemzői

- egyszerre több ablak
- callback függvény alapú eseménykezelés
- időzítő (timer) üresjárat (idle) kezelés
- számos előre definiált tömör és drótváz test (pl teáskanna = `glutWireTeapot()` parancs)
- egyszerű menük

Input/Output kezelés





Operating system Windows

init.
callback regisztráció

GLUT

callback-ek

main

DisplayFunc

KeyboadFunc

SpecialFunc

Reshape

MousePress

MouseMotion

IdleFunc

applikáció

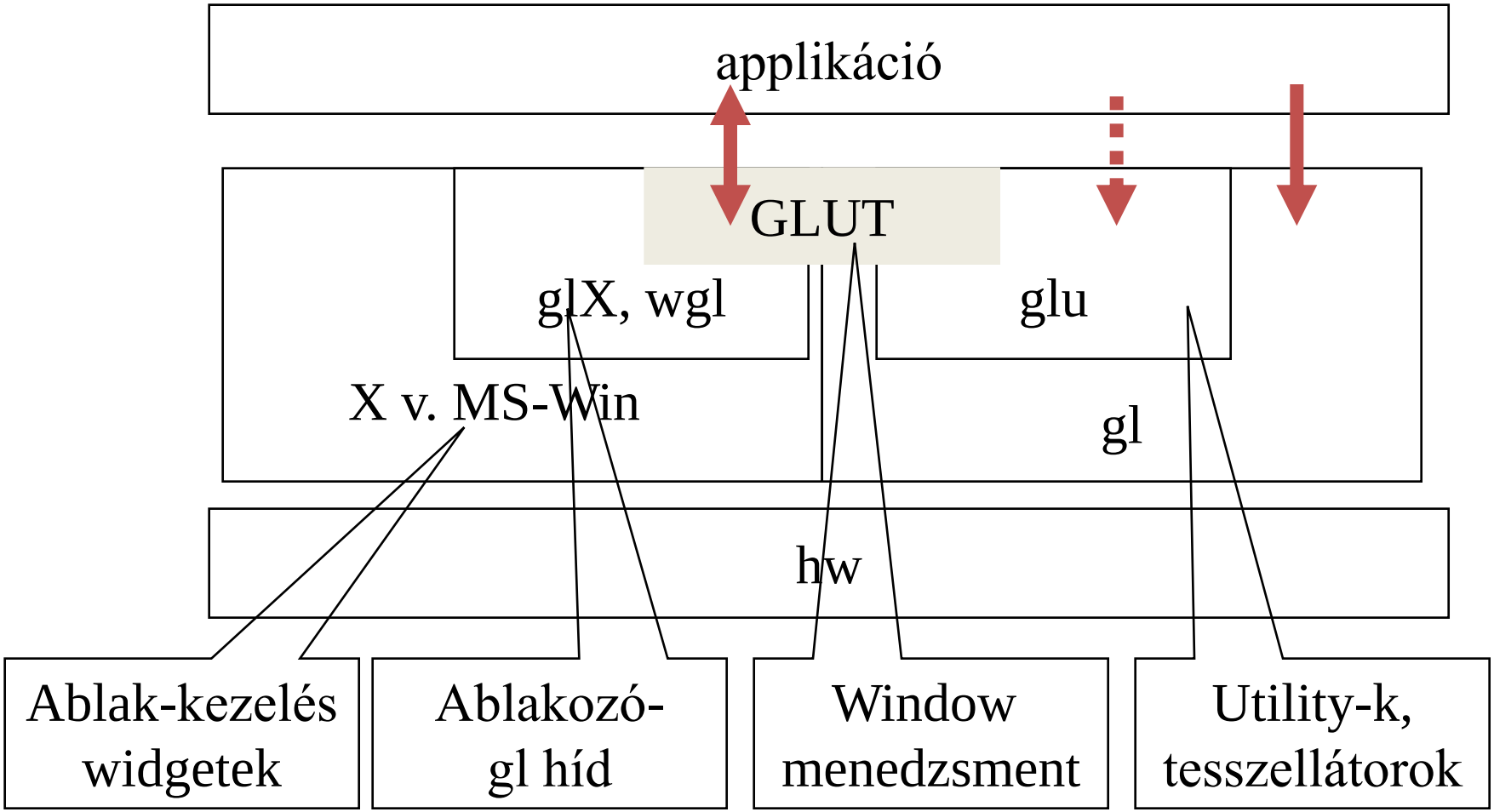
OpenGL

grafikus hardver

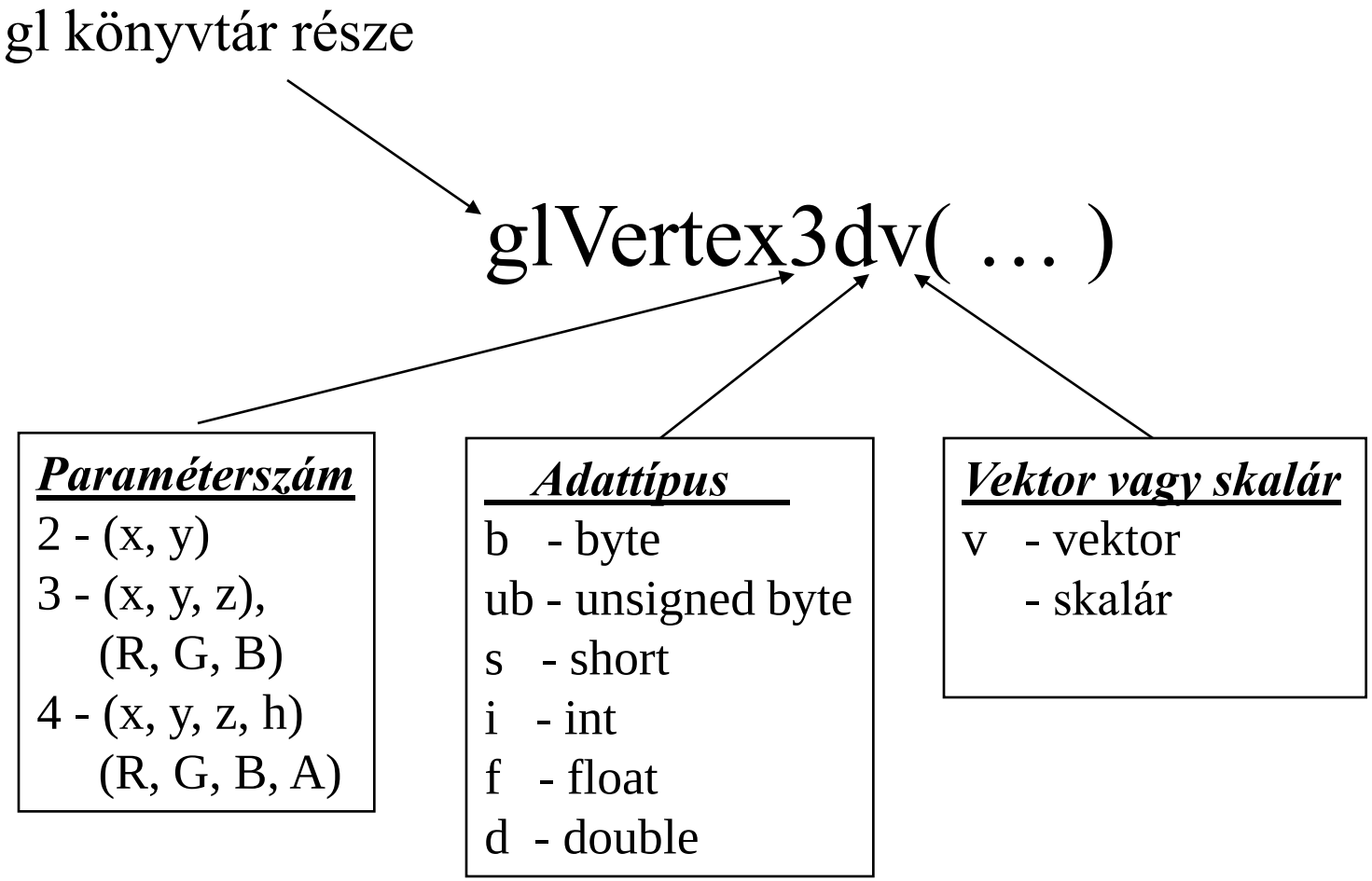
GLUT-OpenGL

- OpenGL:
  - kimeneti csővezeték
    - tesszelláció (GLU),
    - transzformáció,
    - vágás, pick!
    - raszterizáció
- GLUT
  - Op. Rendszer, Ablakozó rendszer illesztés
    - ablak létrehozás
    - bemeneti események elkapása

Ablakozó – OpenGL – alkalmazás elhelyezkedése



OpenGL szintaxis

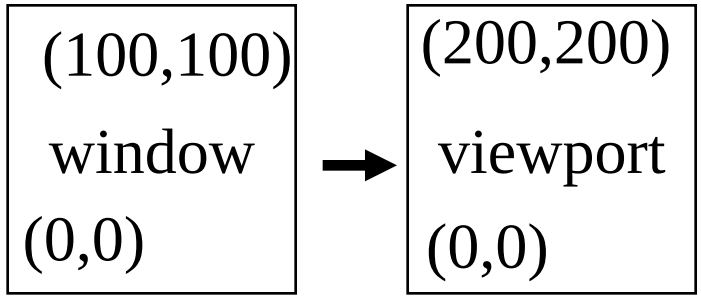


Veremkezelés

- Állapotváltozók („rajzolási állapot” elv) tárolása veremben
  - `glPushAttrib()`: kurrens állapotváltozók elmentése az aktuális veremben
  - `glPopAttrib()`: a verem tetején lévő változók visszatöltése

OpenGL: GLUT inicializálás

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h> // download!!!
main( int argc, char *argv[] ) {
    glutInitWindowSize(200, 200);
    glutInitWindowPosition(100, 100);
    glutInit(&argc, argv);
    glutInitDisplayMode( GLUT_RGB );
    glutCreateWindow("Sample Window");
            // callback függvények
    glutKeyboardFunc( Keyboard );
    glutDisplayFunc( ReDraw );
            // transzformáció
    glViewport(0, 0, 200, 200);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity( );
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity( );
    gluOrtho2D(0., 100., 0., 100.);
            // fő hurok
    glutMainLoop();
}
```



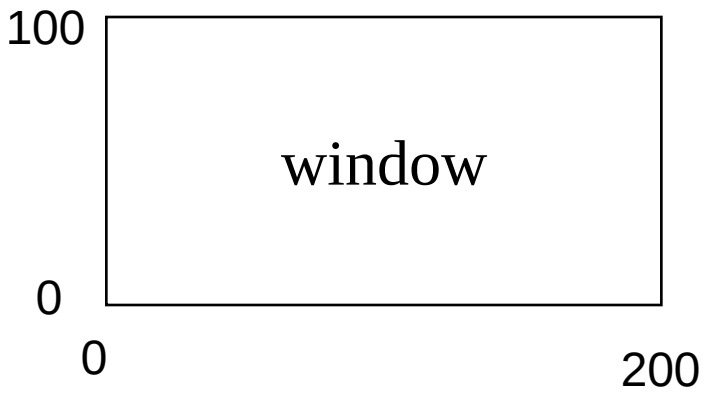
GLUT: eseménykezelés

```
main( int argc, char *argv[] ) {
…    glutKeyboardFunc( Keyboard );
     glutDisplayFunc( ReDraw );
     glutMouseFunc( MousePress );
     glutMotionFunc( MouseMotion );        ……
}
void Keyboard(unsigned char key, int x, int y) {
  if (key == 'd')  myPressedKey_d(x, y);
}
void ReDraw( ) {
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   myRender( );
}
void MousePress( int button, int state, int x, int y ) {
   if (button == GLUT_LEFT && state == GLUT_DOWN)
       myMouseLD(x, y);
}
void MouseMotion( int x, int y ) {
     myMouseMov(x, y);
}
```

Vetítés (párhuzamos)

- `void` **`gluOrtho2D(`**`GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);`
  - Koordinátarendzser definíciója:
    - left, right: bal- és jobboldali vágási sík (az ablak pixel koordinátáiban, ami a bal felső sarokhoz viszonyít)
    - bottom, top: alsó és felső vágási sík

`gluOrtho2D(.0,200.0,100.0,.0)`



Rajzolás az OpenGL-ben

- ablak törlése
- geometriai objektumok rajzolása
- raszteres objektumok rajzolása

Ablak törlése

- `void` **`glClearColor`**`(GLclampf` *`red`*`, GLclampf` *`green`*`, GLclampf` *`blue`*`, GLclampf` *`alpha`*`);`
  - háttérszín beállítása
  - színkomponensek: [0 1] intervallumon belül
- `void` **`glClear`**`(GLbitfield` *`mask`*`);`
  - GL_COLOR_BUFFER_BIT színpuffer
  - GL_DEPTH_BUFFER_BIT mélységpuffer (lásd később...)
  - GL_ACCUM_BUFFER_BIT gyűjtőpuffer (lk...)
  - GL_STENCIL_BUFFER_BIT stencilpuffer (lk...)

A rajzolás befejezésének kikényszerítése

- Teljes megjelenítési lánc:
  - a rajzolási parancs és az objektum megjelenés között
  - a kirajzolás késhet...
- `void` **`glFlush`**`()`
  - kikényszeríti a korábban kiadott OpenGL parancsok azonnali végrehajtását

Geometriai alapelemek megadása

- Csúcspont = vertex
  - `void` **`glVertex`**`[234]{sifd}(`*`TYPE coords`*`);`
  - `void` **`glVertex`**`[234]{sifd}v(const` *`TYPE* coords`*`);`
  - Példák:
    - glVertex2s(2, 3);
    - glVertex3d(0.0, 0.0, 3.1415926535898);
    - glVertex4f(2.3, 1.0, -2.2, 2.0);
    - GLdouble dvect[3] = {5.0, 9.0, 1992.0};
      glVertex3dv(dvect);
- glBegin() és glEnd() között kell kiadni

Normálvektor

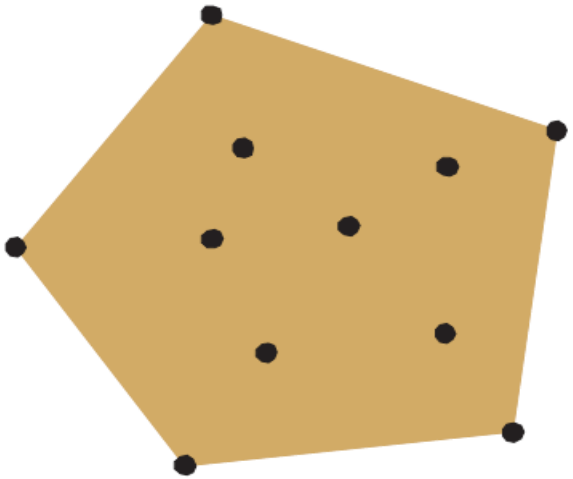
- `void` **`glNormal3`**`{bsidf}(`*`TYPE nx, TYPE ny, TYPE nz`*`);`
- `void` **`glNormal3`**`{bsidf}`**`v`**`(const` *`TYPE`* `*v);`
  - Adott (3D) pontban megjelenítendő felület normálvektora
  - Árnyalási, megvilágítási feladatokhoz szükséges
  - Felületek közelítése poligonokkal: csúcsponthoz a pontban találkozó lapok normálisának átlagát rendeljük – így az élek nem fognak látszani

Geometriai alapelemek

- Csúcspontok -> pont, szakasz, poligon
- Téglalap: külön függvény (mert gyalkori)
  - `void glRect{sifd}(TYPE x1, TYPE y1, TYPE x2, TYPE y2);`
  - `void glRect{sifd}v(const TYPE *v1, const TYPE *v2);`

Geometriai alapelemek

- Poligon: zárt törött vonallal határolt **KONVEX** terület
  - csak egyszerű (nem lyukas) alakzatok
  - mindig síkbeli
  - tetszőleges felület = háromszögek egyesítése
  - Nem egyszerű poligon helyett általában a konvex burok jelenik meg:



Geometriai alapelemek felsorolása

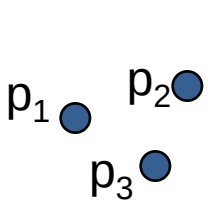
- Geometriai elemek felsorolása: glBegin() és glEnd() parancsok között

```
void glBegin(GLenum mode);
void glEnd(void);
```

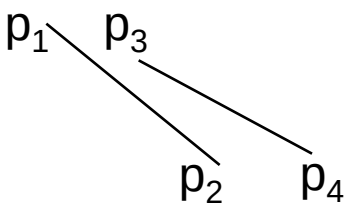
- „mode” adja meg az alapelem típusát, pl háromszög rajzolás példa:

```
glBegin(GL_TRIANGLES);
        glVertex2d(10.0, 10.0);
        glVertex2d(20.0, 100.0);
        glVertex2d(90.0, 30.0);
 glEnd( );
```

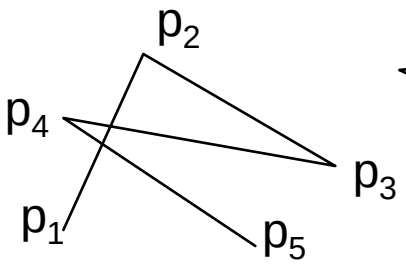
OpenGL: primitívek



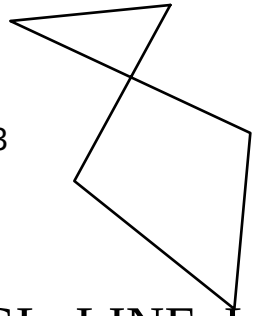
GL_POINTS



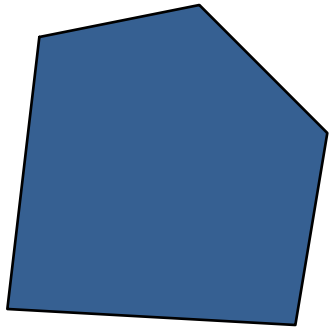
GL_LINES



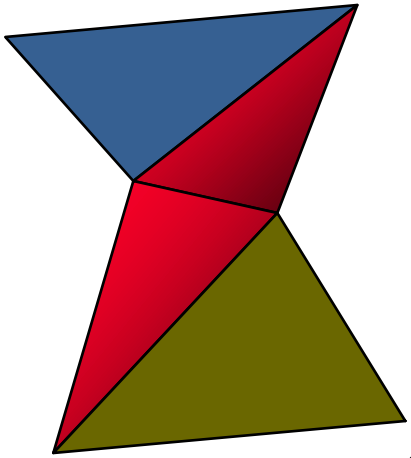
GL_LINE_STRIP

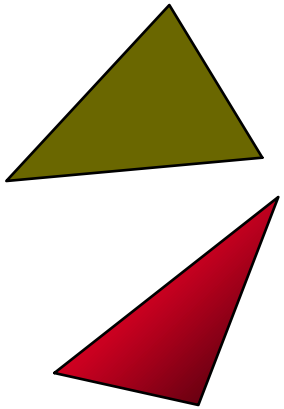


GL_LINE_LOOP

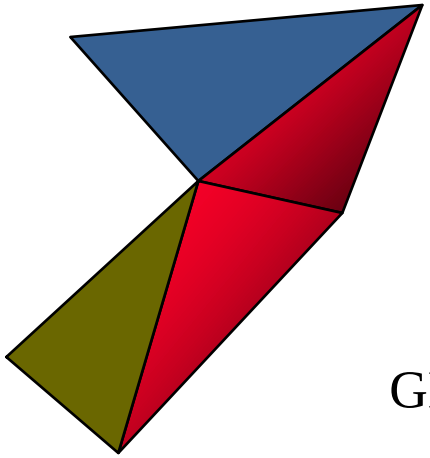


GL_POLYGON



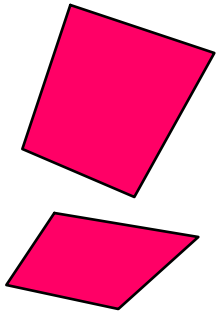


GL_TRIANGLES

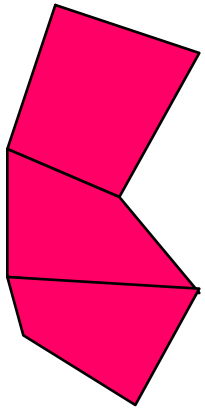


GL_TRIANGLE_STRIP

GL_TRIANGLE_FAN



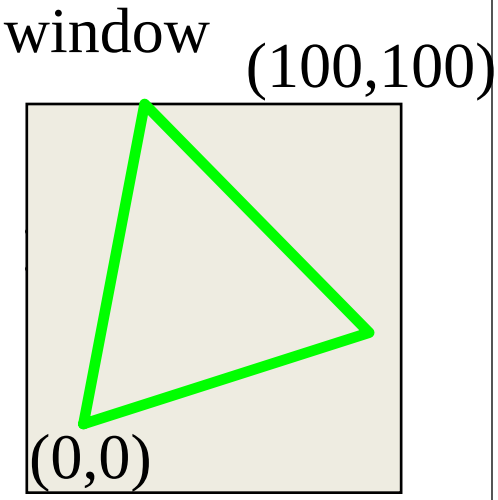
GL_QUADS

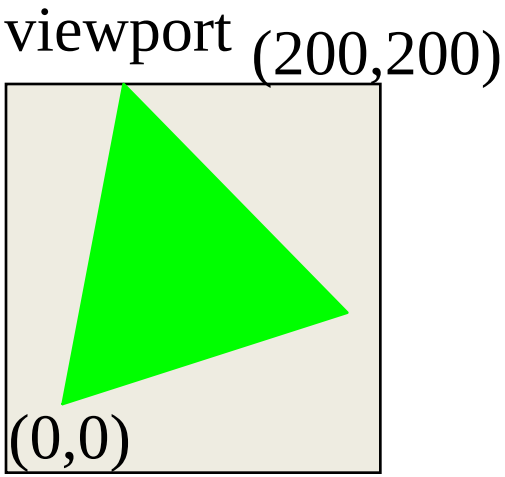


GL_QUAD_STRIP

Eseménykezelés és rajzolás

```
bool drawtriangle=false;

void ReDraw( ) {
      glClearColor(0, 0, 0, 0);
      glClear(GL_COLOR_BUFFER_BIT);
      if (drawtriangle){
            glBegin(GL_TRIANGLES);
                  glColor3d( 0.0, 1.0, 0.0
                  glVertex2d(10.0, 10.0);
                  glVertex2d(20.0, 100.0);
                  glVertex2d(90.0, 30.0);
            glEnd( );
      }
}
void Keyboard(unsigned char key,
                  int x, int y) {
      if (key == 't') {
            drawtriangle=true;
    }
      glutPostRedisplay();
}
```





Primitívek rajzolása

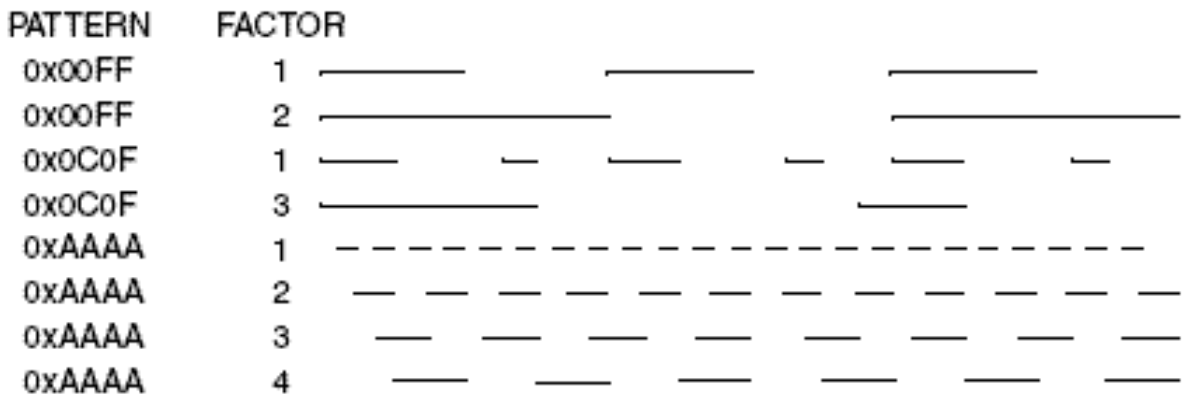
- A **`glBegin`**`()` és **`glEnd`**`()` között kiadható még:
  - szín **`glColor*`**`()`
  - normális: **`glNormal*`**`()`
  - textúra: **`glTexCoord*`**`()` lásd sokkal később...
  - stb

Geometriai alapelemek megjelenését befolyásoló tényezők

- `void` **`glPointSize`**`(GLfloat` *`size`*`);`
  - pontot reprezentáló pixeltömb mérete
- `void` **`glLineWidth`**`(GLfloat` *`width`*`);`
  - szakasz szélessége

Szaggatott szakaszok

- Szaggatottság be-kikapcsolása
  - **glEnable**(GL_LINE_STIPPLE);
  - **glDisable**(GL_LINE_STIPPLE);
- void **glLineStipple**(GLint *factor*, GLushort *pattern*);
  - szaggatottság egységének (factor) és mintájának (pattern) beállítása



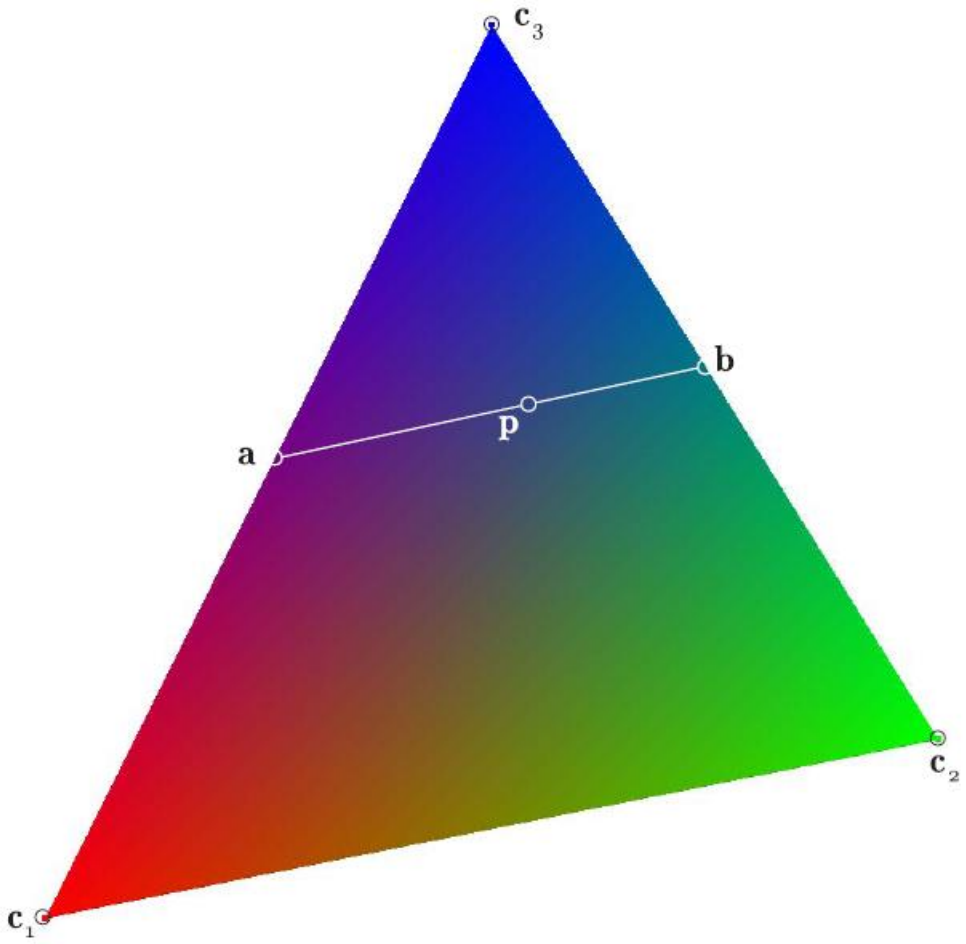
Szín megadása

- `void` **`glColor3{bsifdu}`**`(TYPE r, TYPE g, TYPE b)`
- `void` **`glColor3{bsifdu}`** `(TYPE r, TYPE g, TYPE b, TYPE a)`
- `void` **`glColor{3,4}{bsifdu}v`**`(const TYPE *v)`

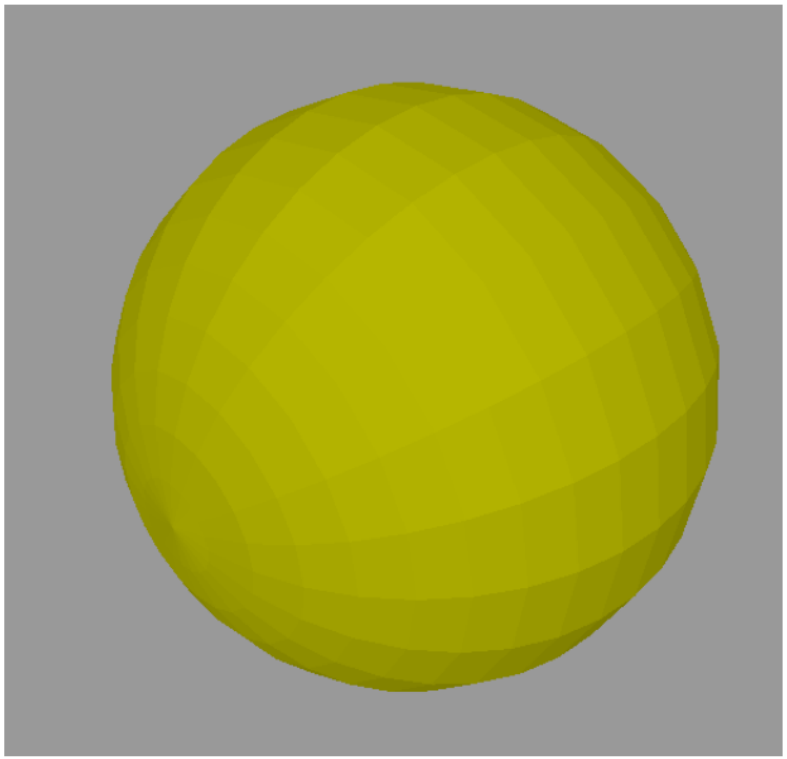
Árnyalás

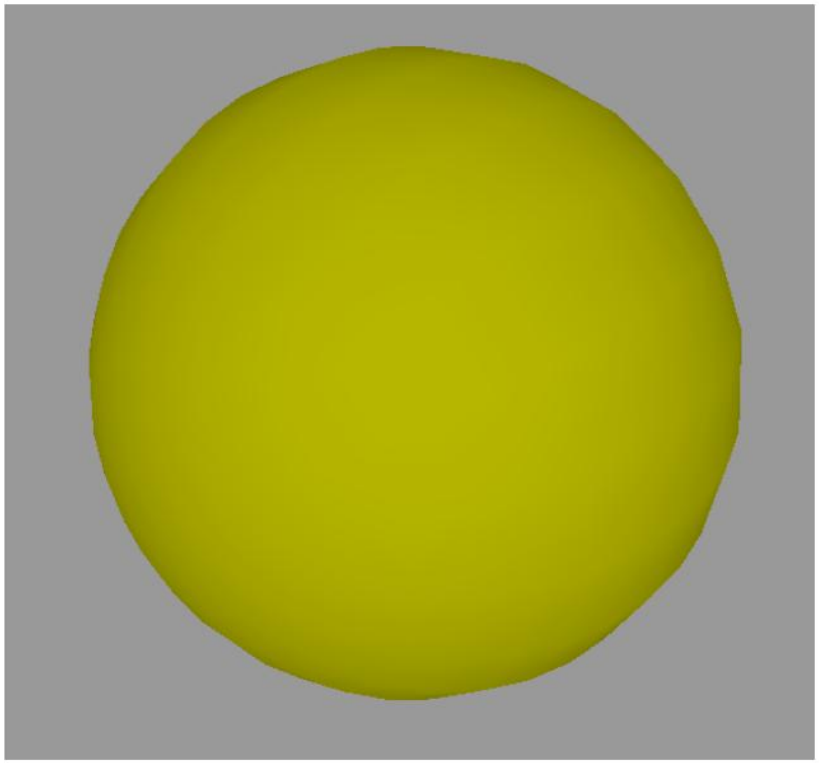
- `void` **`glShadeModel`**`(GLenum` *`mode`*`);`
  - mode
    - GL_FLAT: a teljes objektumot egy meghatározott csúcspont színével színezi ki. A kiválasztott csúcspont objektumtípusonként változik, pl törött vonal esetén a végpont, GL_POLYGON esetén az első csúcspont
    - GL_SMOOTH a csúcspontok színéből lineárisan interpolálja a belső pontok színét

Gouroud árnyalás



Flat vs. smooth árnyalás

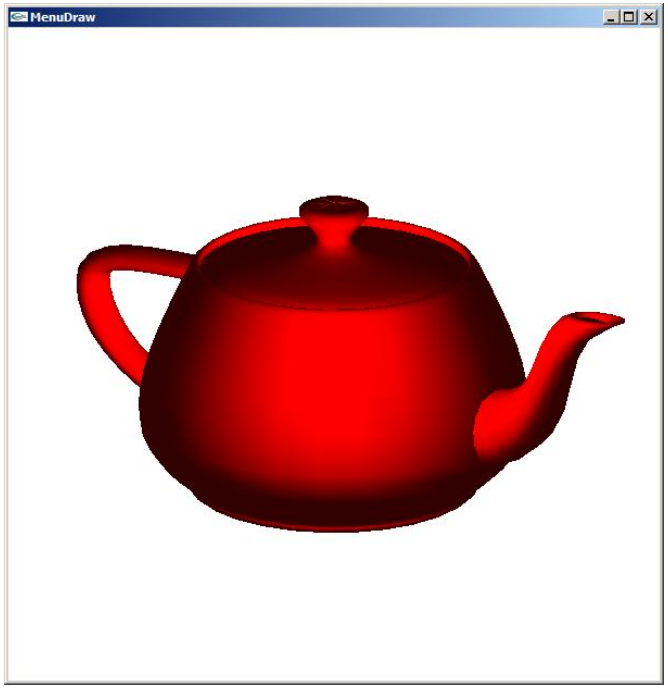


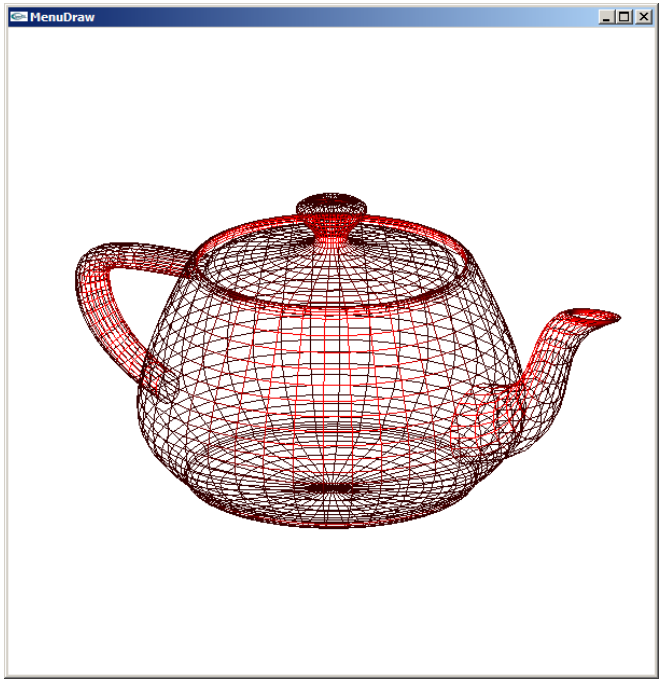


Egyéb alapvető GLUT funciók:

- Mintaobjektumok rajzolása teszteléshez:

```
void glutSolidTeapot(GLdouble size);
void glutWireTeapot(GLdouble size);
```





Mintaobjektumok rajzolása teszteléshez

```
void glutSolidCube(GLdouble size);
void glutWireCube(GLdouble size);
void glutSolidIcosahedron(void);
void glutWireIcosahedron(void);
void glutSolidTetrahedron(void);
void glutWireTetrahedron(void);
```

Stb…

Egyéb alapvető GLUT funciók: Menükezelés

```
int glutCreateMenu(void (*func)(int value));
```

- Létrehoz egy új menüt (vagy almenüt)
- Callback függvényt regisztrál a menühöz
- Visszatérési érték: egyedi azonosító
- A program aktuális menüjének az új menüt állítja

```
void glutAddMenuEntry(char *name, int value);
```

- Új menübejegyzést ad az aktuális menühöz.
- Ha majd ráklikkelünk a bejegyzésre az aktuális menü Callback-je „value” paraméterrel hívódik meg

Menükezelés

```
void glutAddSubMenu(char *name, int
  submenuID);
```

- Almenüt ad az aktuális menühöz. Az almenüt korábban létre kellet hozni, és az azonosítóját eltárolni

```
void glutAttachMenu(int button);
```

- Az aktuális menüt a „button” egérgombhoz rendeli, azaz azzal lehet majd előhívni

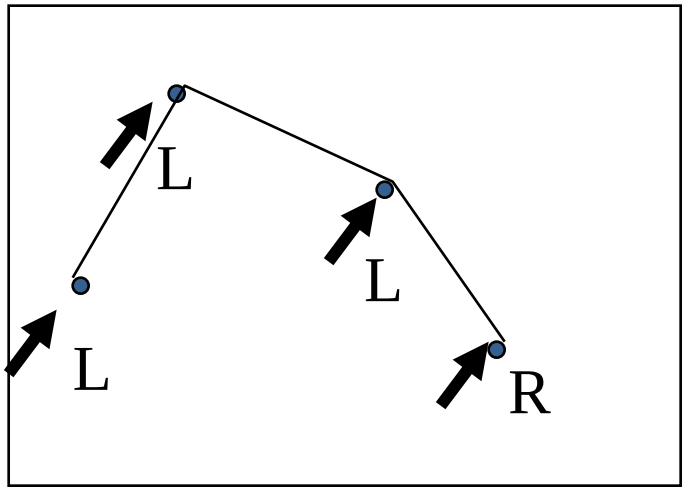
Minta: menürendszer létrehozása

```
int submenu=glutCreateMenu(MenuFunc);
  glutAddMenuEntry("ElsoFunkcio",0);
  glutAddMenuEntry("MasodikFunkcio",1);
  glutAddMenuEntry("HarmadikFunkcio",2);
  glutAddMenuEntry("NegyedikFuncio",3);
  glutCreateMenu(MenuFunc);
  glutAddSubMenu("Type",submenu);
  glutAddMenuEntry("Exit",4);
  glutAddMenuEntry("About",5);
  glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Minta: menücallback implementációja

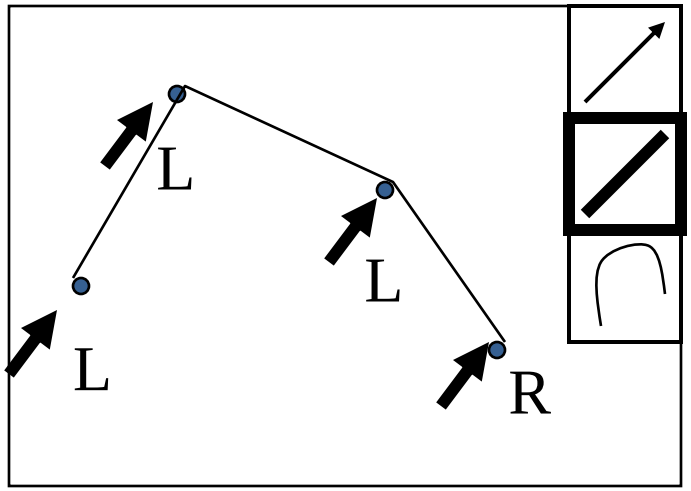
```
void MenuFunc(int menuItemIndex) {
  switch(menuItemIndex) {
       case 0: …  ; break;
       case 1: … ; break;
       case 2: … ; break;
       case 3: … ; break;
       case 4: exit(0);
       case 5: MessageBox(NULL,"Hello Glut","About",MB_OK);
  break;
  }

  glutPostRedisplay();
}
```

2D grafikus editor

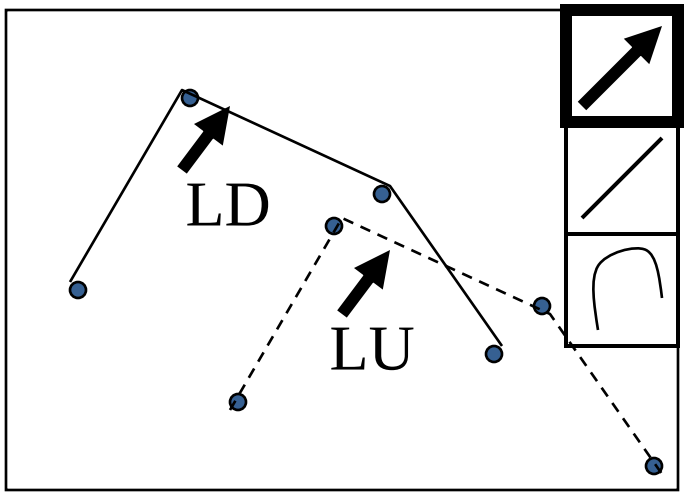


MouseLDown	első pont
MouseLDown	második
...	
MouseLDown	n.
MouseRDown	utolsó

2D grafikus editor:
GUI, use-case, dinamikus modell

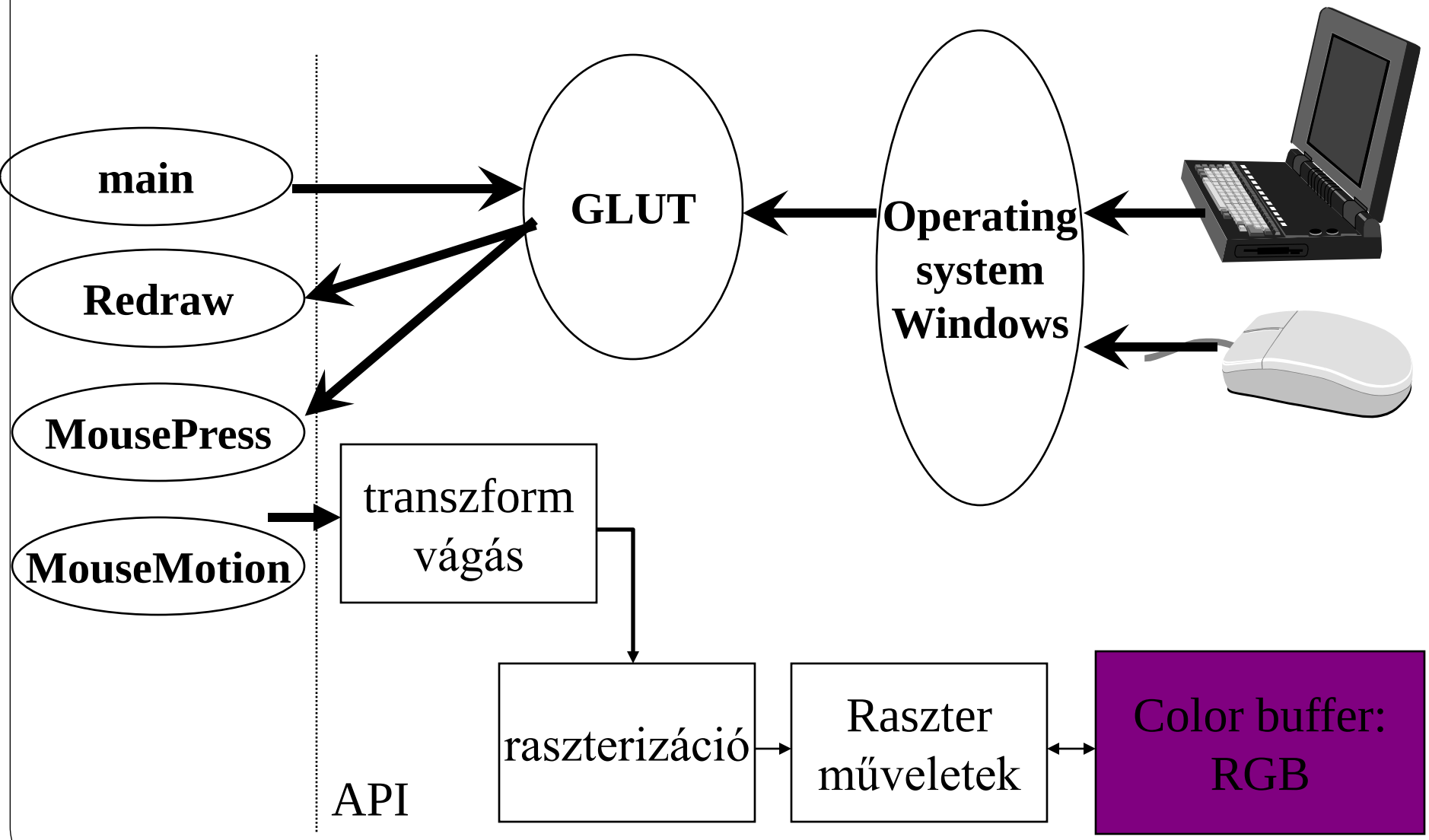


MouseLDown	első pont
MouseLDown	második
...	
MouseLDown	n.
MouseRDown	utolsó

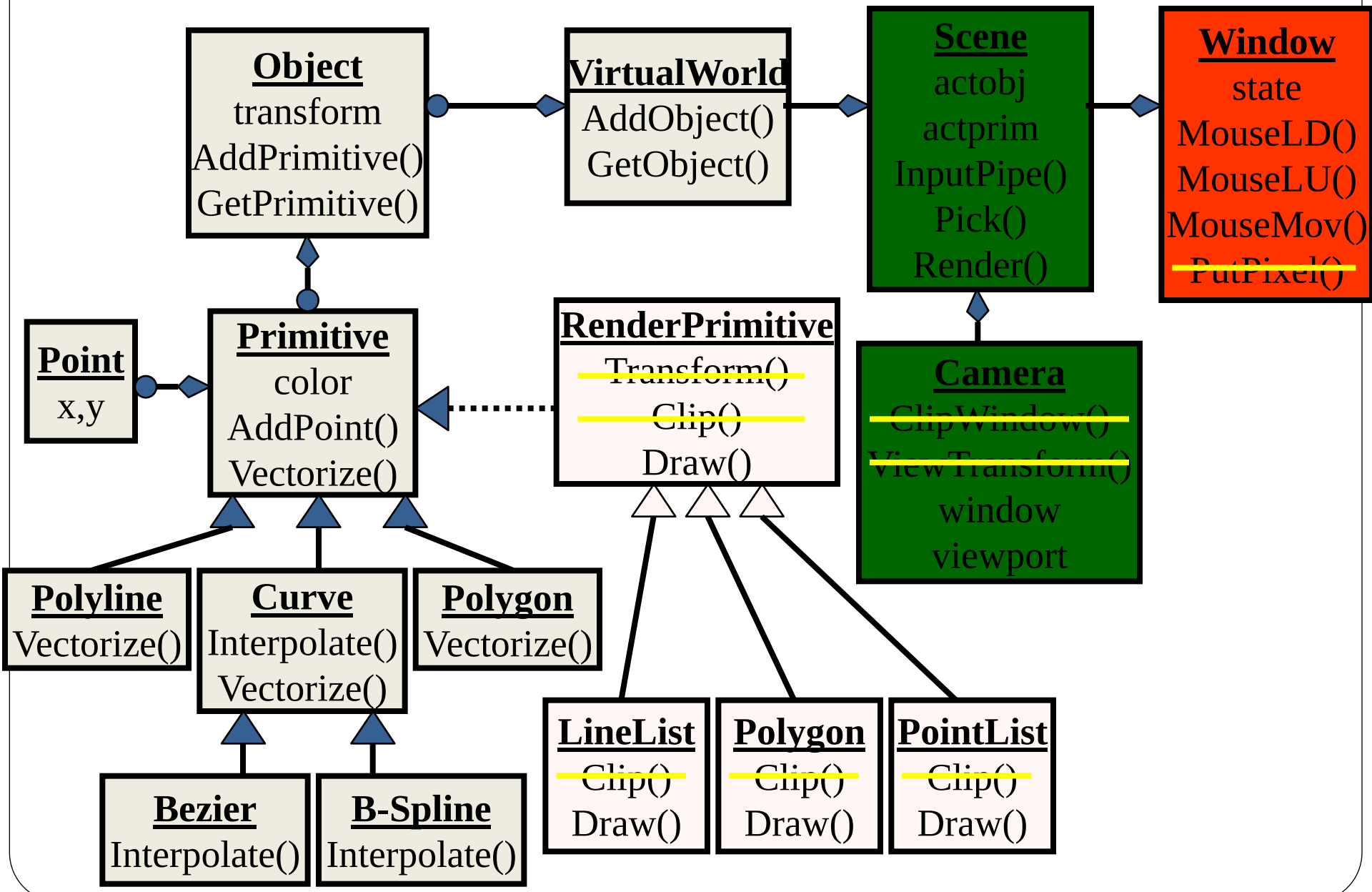


MouseLDown	pick?
MouseMove	mozgat
MouseLUp	letesz

Glut/OpenGL program architektúra



Osztálydiagram



GLUT: inicializálás

```
main(argc, argv) {
    glutInitWindowSize(200, 200);
    glutInitWindowPosition(100, 100);
    glutInit(&argc, argv);
    glutInitDisplayMode( GLUT_RGB );

    glutCreateWindow("2D graphics editor");

    glutMouseFunc(MousePress);  // callback
    glutMotionFunc(MouseMotion);
    glutDisplayFunc(ReDraw);

    glutMainLoop();             // fő hurok
}
```

OpenGL LineList

```
void Draw( ) {
   glColor3d( color.R, color.G, color.B );

   glBegin( GL_LINE_STRIP );
   for( i = 0; i < npoints; i++ )
      glVertex2d(points[i].x, points[i].y);
   glEnd( );

   glFlush( );
}
```