2D Graphics in OpenGL

Francesc Sebé
‘Computació gràfica i multimèdia’
Escola Politècnica Superior
Universitat de Lleida

OpenGL

- **Open G**raphics **L**ibrary
- Cross-platform API for writting applications that produce 2D and 3D computer graphics
- Developed by Silicon Graphics Inc in 1992
- Managed by Kronos Group consortium

OpenGL

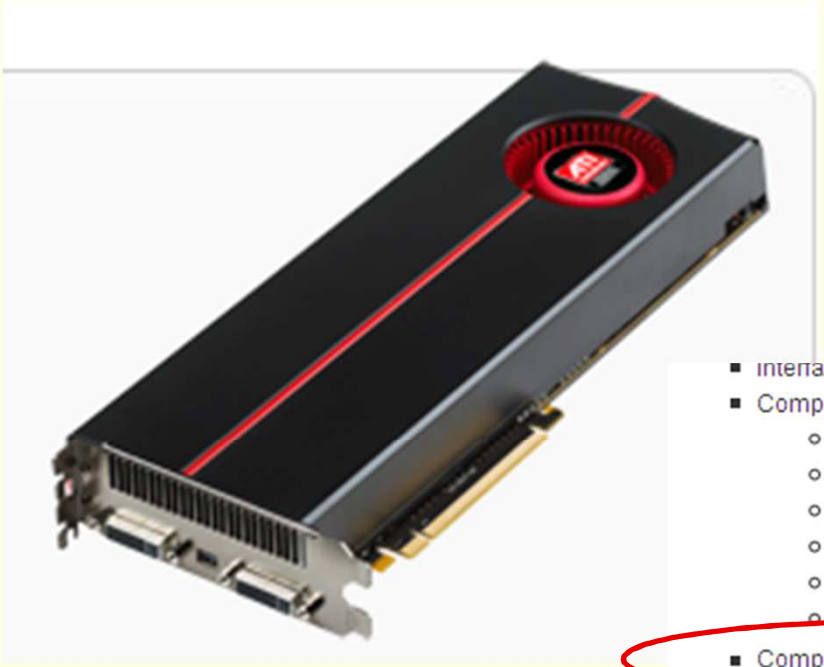
- Supported by graphics acceleration hardware

OpenGL

FUNCIONES DISPONIBLES:

NVIDIA SLI® Ready[1]	Cuádruple
NVIDIA 3D Vision Surround Ready[2]	✓
Tecnología NVIDIA PureVideo®[3]	HD
Tecnología NVIDIA PhysX™	✓
Tecnología NVIDIA CUDA™	✓
Microsoft DirectX	11
OpenGL	4.1
Soporte de bus	PCIe 2.0 x16

OpenGL



- Interfaz de bus PCI Express 2,1x16
- Compatible con DirectX 11
  - Shader Model 5.0
  - DirectCompute 11
  - Unidad de teselación de hardware program
  - Multiproceso acelerado
  - Compresión de texturas HDR
  - Transparencia independiente del orden
- Compatible con OpenGL 3.2[8]
- Tecnología para la mejora de la calidad de imagen
  - Modos antialiasing de multimuestreo y supe
  - Antialiasing adaptado
  - Super AA
  - Filtrado de texturas anisotrópico independie
  - Representación de alto rango dinámico (HD

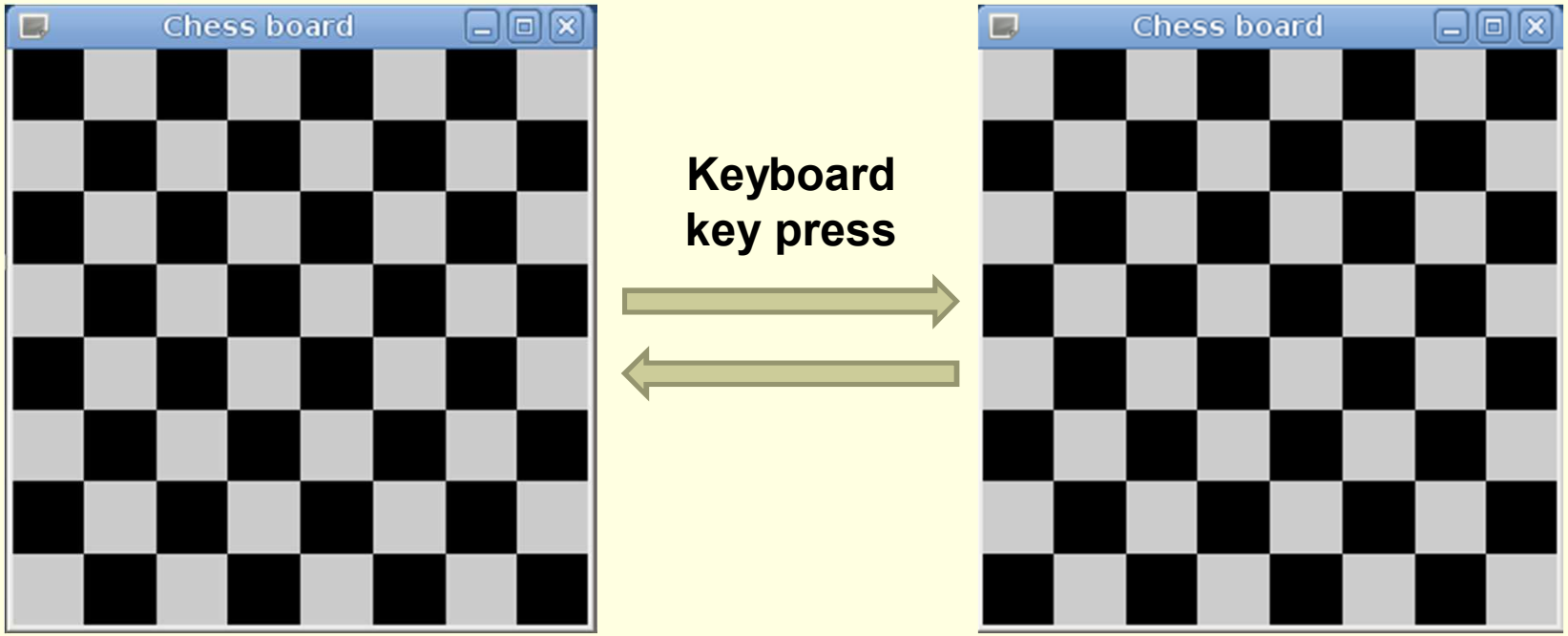
Use of OpenGL in C

- In the source code:
  - #include <GL/glut.h>

- Compile:
  - gcc **-lglut -lGLU -lGL -lm** prog.c -o prog

A simple 2D OpenGL example

- Chess board with square color flipping



A simple 2D OpenGL example

- Use constant definition to set values that may require to be tuned

```
#include <GL/glut.h>

#define COLUMNS 8
#define ROWS 8
#define WIDTH 300
#define HEIGHT 300
```

A simple 2D OpenGL example

- Main procedure

```
int main(int argc,char *argv[])
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowPosition(50, 50);
  glutInitWindowSize(WIDTH, HEIGHT);
  glutCreateWindow("Chess board");

  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);

  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0,WIDTH-1,0,HEIGHT-1);

  glutMainLoop();
  return 0;
}
```

Programming in OpenGL

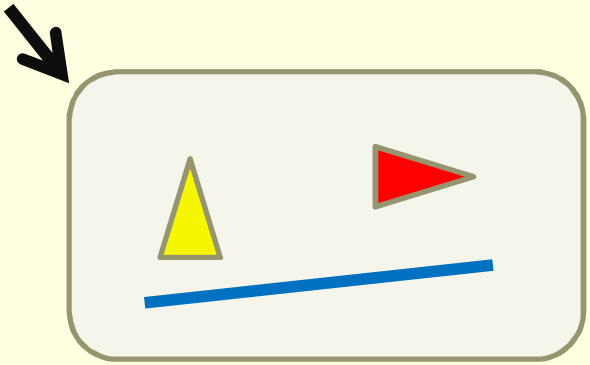
- **glutInit(&argc,argv);**
  - Init GLUT (OpenGL Utility Toolkit)
  - GLUT provides procedures for:
    - Window management
    - Keyboard and mouse I/O
    - Geometric figures drawing
      - Cube, Cone, Sphere, etc

Programming in OpenGL

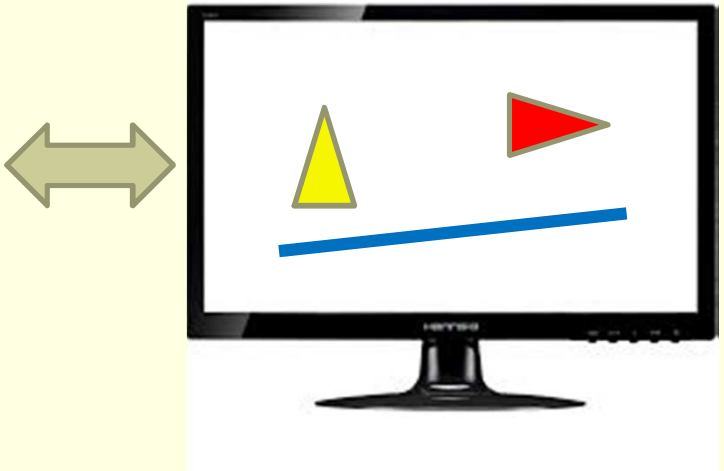
- **glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);**
  - Our program will use double buffering
  - Color specified in RGB components

Double buffering

- Objects are drawn on a secondary buffer



- Buffer swap



Programming in OpenGL

- **glutInitWindowPosition(50,100);**
  - Place the window, to be created next, 50 pixels to the right of the left edge of the screen and 100 pixels down from the top

Programming in OpenGL

- **glutInitWindowSize(width,height)**
  - In pixels
- **glutCreateWindow("Window title");**
  - Create a graphics window with the provided caption and the previously specified location and size

Programming in OpenGL

- **glutDisplayFunc(procedure);**
  - Whenever OpenGL determines the content of the window has to be plotted, **procedure** will be called.

  - **void procedure(void)**
    - **{**
    - **/* drawing code */**
    - **}**

Programming in OpenGL

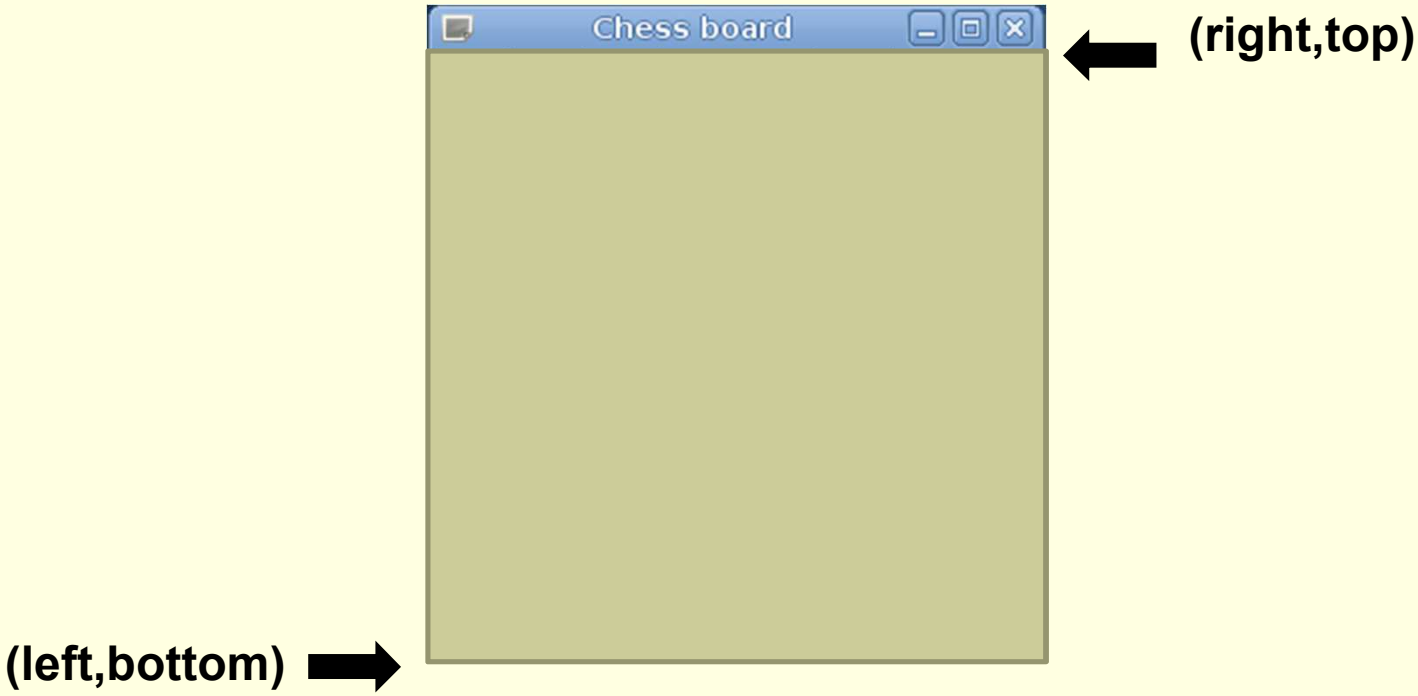
- **glutKeyboardFunc(procedure);**
  - Whenever a keyboard key is pressed, **procedure** will be called.
  - **void procedure(unsigned char c, int x, int y)**
  - ‘c’ is the key that has been pressed
  - (x,y) are the coordinates of mouse pointer location

Programming in OpenGL

- Also:
  - **glutMouseFunc**
  - **glutReshapeFunc**
  - **glutIdleFunc**

Programming in OpenGL

- **gluOrtho2D(left,right,bottom,top);**
  - Indicate 2D clipping coordinates (in world units)



Programming in OpenGL

- **glutMainLoop();**
  - Display initial graphics and put the program into an infinite loop waiting for events.

Display procedure

```
void display()
{
  int i,j;

  glClearColor(0.0,0.0,0.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);

  for(i=0;i<WIDTH;i++)
    for(j=0;j<HEIGHT;j++)
      if( (keyflag==0 && (i+j)%2==0) || (keyflag==1 && (i+j)%2==1) )
      {
        glColor3f(0.8,0.8,0.8);
        glBegin(GL_QUADS);

        glVertex2i(i*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);
        glVertex2i(i*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);

        glEnd();
      }

  glutSwapBuffers();

}
```

Clear the window. It is filled in black color.

Display procedure

```
void display()
{
  int i,j;

  glClearColor(0.0,0.0,0.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);

  for(i=0;i<WIDTH;i++)
    for(j=0;j<HEIGHT;j++)
      if( (keyflag==0 && (i+j)%2==0) || (keyflag==1 && (i+j)%2==1) )
      {
        glColor3f(0.8,0.8,0.8);
        glBegin(GL_QUADS);

        glVertex2i(i*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);
        glVertex2i(i*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);

        glEnd();
      }

  glutSwapBuffers();

}
```

Draw some light grey squares.

Display procedure

```
void display()
{
  int i,j;

  glClearColor(0.0,0.0,0.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);

  for(i=0;i<WIDTH;i++)
    for(j=0;j<HEIGHT;j++)
      if( (keyflag==0 && (i+j)%2==0) || (keyflag==1 && (i+j)%2==1) )
      {
        glColor3f(0.8,0.8,0.8);
        glBegin(GL_QUADS);

        glVertex2i(i*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,j*HEIGHT/ROWS);
        glVertex2i((i+1)*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);
        glVertex2i(i*WIDTH/COLUMNS,(j+1)*HEIGHT/ROWS);

        glEnd();
      }

  glutSwapBuffers();

}
```

Move the drawn scene to the front buffer (screen).

Handling keyboard events

- ‘keyflag’ is a global variable

```
void keyboard(unsigned char c,int x,int y)
{
  if(keyflag==0)
    keyflag=1;
  else
    keyflag=0;

  glutPostRedisplay();

};
```

- A call to ‘**glutPostRedisplay’** generates a display event.

More OpenGL commands

- The following code draws a point

```
glBegin(GL_POINTS);
    glVertex2i(100,100);
glEnd();
```

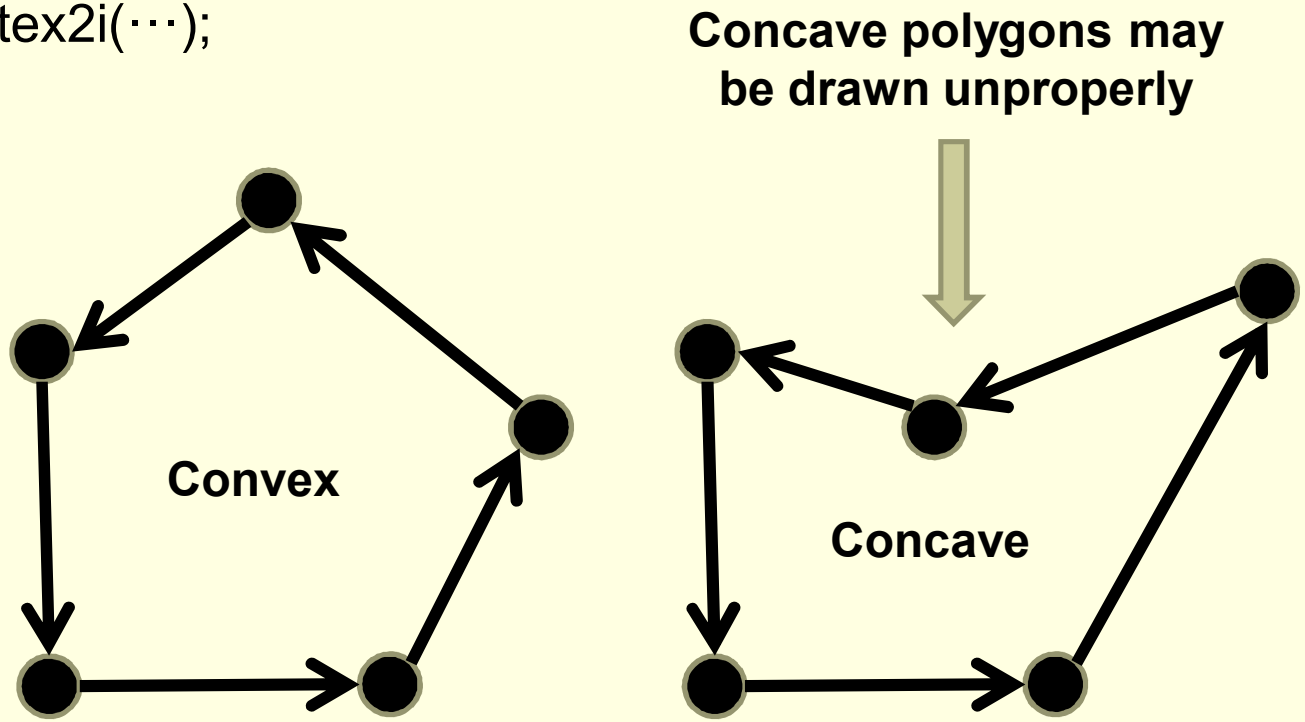
- The following code draws a straight line

```
glBegin(GL_LINES);
    glVertex2i(100,100);
    glVertex2i(200,200);
glEnd();
```

More OpenGL commands

- The following code draws a "**convex**" polygon.
  - Vertices given "**counterclockwisely**".

```
glBegin(GL_POLYGON);
    glVertex2i(···);
      (···)
glEnd();
```



More OpenGL commands

- Also

```
glBegin(GL_TRIANGLES);
    glColor3f(1.0,  0.0,  0.0);
    glVertex2i(100,100);

    glColor3f(0.0,  1.0,  0.0);
    glVertex2i(200,100);

    glColor3f(0.0,  0.0,  1.0);
    glVertex2i(150,250);
glEnd();
```