

# Lab Computación Gráfica I – CI4321

## Links importantes:

<http://nehe.gamedev.net/default.asp>

<http://www.opengl.org/resources/code/samples/s2001/>

<http://www.opengl.org>

# OpenGL

- *OpenGL* es una API grafica 3D desarrollada por SGI (*Silicon Graphics Inc.*)
- Permite:
  - dibujar primitivas de forma sencilla, rápida, y aprovechando las capacidades de la tarjeta gráfica
  - simula las capacidades gráficas de la tarjeta gráfica por Software.
  - <http://www.opengl.org>

# GLUT

- *Glut* es una libreria de ayuda de *OpenGL* escrita por *Mark Kilgard*
- provee un workframe muy comodo, sencillo y multi-plataforma.
- con *glut* podemos crear la ventanas, iniciar *OpenGL* y manejar algunas funciones basicas de E/S.
- <http://search.developer.nvidia.com/>

# Mi primera Ventana

```
int main(int argc, char** argv) {  
    glutInit(&argc, argv); // Inicializa la libreria GLUT  
    glutInitWindowSize (600, 600); // inicializa el tamaño de la ventana  
    glutInitWindowPosition (10, 50); // posición de la ventana  
    glutCreateWindow ("Mi primera ventana"); // crea la ventana  
  
    // glutDisplayFunc(display); Aqui le indicamos a GLUT que la  
    rutina utilizada como Display sera la llamada display. Este tipo de  
    rutinas se ejecuta una vez por ciclo de programa, y sera la que  
    realice todas las operaciones graficas (render).  
  
    glutDisplayFunc(display);  
  
    // ciclo de la aplicación  
  
    glutMainLoop();  
    return 0;  
}
```

# Mi primera Ventana

```
void display(){
```

```
// limpia el color del buffer en verde RGBA
```

```
glClearColor(0.0f, 1.0f, 0.0f ,1.0f);
```

```
glClear(GL_COLOR_BUFFER_BIT);
```

```
glColor3f(0.0,0.0,1.0); // asigna el color azul al objeto a dibujar
```

```
// dibuja un rectangulo en la posición 0, 0 de tamaño .8, .8 (de la ventana)
```

```
glRectf(0,0,.8,.8);
```

```
// Dependiendo del hardware, controladores, etc., ogl guarda los comandos como peticiones en pila, para optimizar el rendimiento. El comando glFlush causa la ejecución de cualquier comando que quede en espera en la pila.
```

```
glFlush();
```

```
}
```

# Mi primera Ventana

```
int main(int argc, char** argv) {  
    glutInit(&argc, argv);  
    glutInitWindowSize (600, 600);  
    glutInitWindowPosition (10, 50);  
    glutCreateWindow ("Mi primera ventana");  
    glutDisplayFunc(display);  
  
    // se llama a la función reshape siempre que se modifica el  
    // tamaño de la ventana  
  
    glutReshapeFunc(reshape);  
    glutMainLoop();  
    return 0;  
}
```

```
void display(){  
    glClearColor(0.0f, 1.0f, 0.0f ,1.0f);  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(0.0,0.0,1.0);  
    glRectf(0,0,8,8);  
    glFlush();  
}
```

# glViewport

```
void reshape(int ){
```

```
// viewport es un área rectangular de la ventana de visualización. Por defecto es la ventana entera pero podemos variarlo a gusto
```

```
glViewport(0, 0, w, h);
```

```
// activa a matriz de proyección y limpia
```

```
glMatrixMode(GL_PROJECTION);
```

```
glLoadIdentity();
```

```
glOrtho(-7, 7, -7, 7, 1.0, -1.0);
```

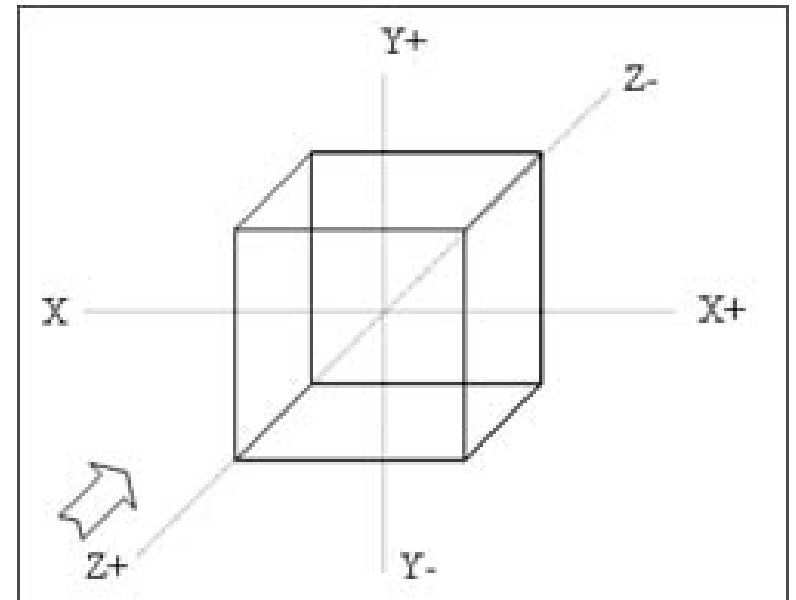
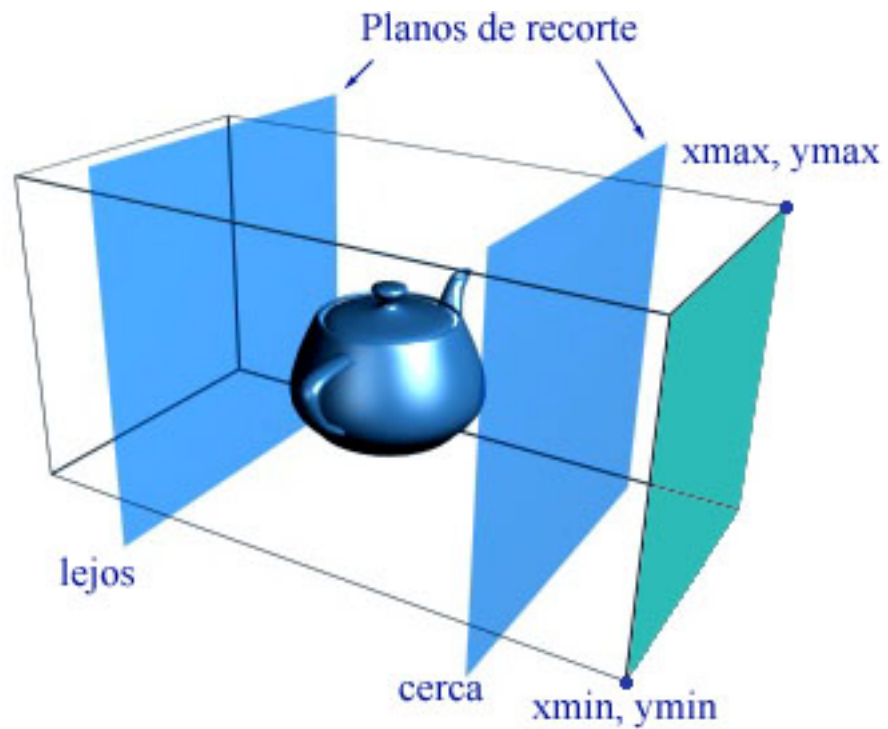
```
// activa a matriz de modelado / visualizado (modelview)
```

```
glMatrixMode(GL_MODELVIEW);
```

```
glLoadIdentity();
```

```
}
```

# glOrtho



`glOrtho(-X, X, -Y, Y, -Z, Z);`

# glOrtho

```
void reshape(int w, int h){  
    glViewport(0, 0, w, h);  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    glOrtho(-7, 7, -7, 7, 1.0, -1.0);  
    glMatrixMode(GL_MODELVIEW);  
    glLoadIdentity();  
}
```

# glOrtho

...

```
glLoadIdentity();
```

```
    aspectratio = (float) w / (float) h;
```

```
if (w<=h) {
```

```
    glOrtho(-7, 7, -7/aspectratio, 7/aspectratio, 1.0, -1.0);
```

```
}
```

```
else{
```

```
    glOrtho(-7*aspectratio, 7*aspectratio, -7, 7, 1.0, -1.0);
```

```
}
```

```
glMatrixMode(GL_MODELVIEW);
```