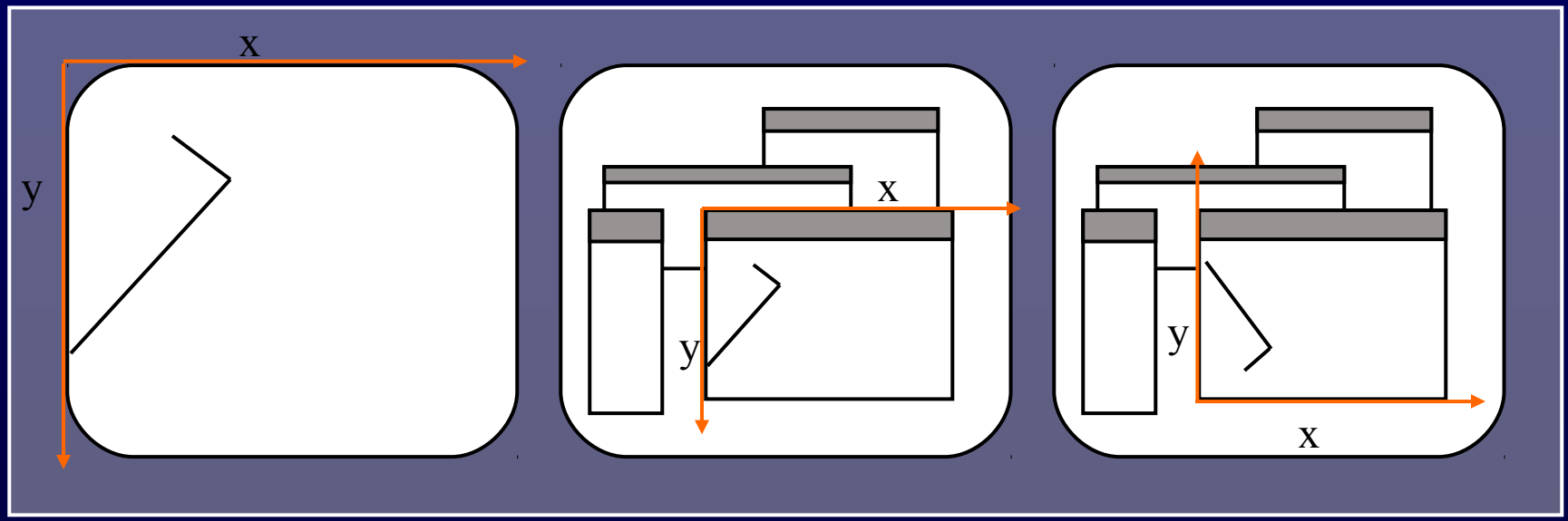
Introducción a la Computación Gráfica

Ayuda OpenGL

Roberto Guerrero
Univ. Nac. de San Luis
Marzo 2013

Independencia de Dispositivos

- **Ambiente y Herramientas**



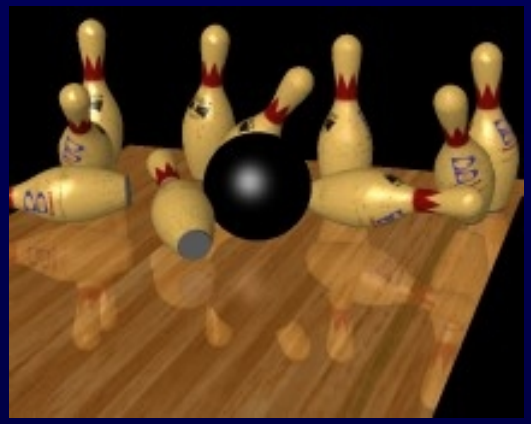
  - **Ejemplo:**

```
line(100, 50, 150, 80);
line(150, 80, 0, 290);
```

- **Cada ambiente posee sus propias herramientas para producir sus primitivas de dibujo.**

Comienzo…

- **OpenGL es una biblioteca gráfica que facilita la interacción con hardware gráfico especializado y habilita el desarrollo de aplicaciones independientes del hardware.**



Dos escenas generadas con OpenGL....

Nota: referencias según el libro *OpenGL Programming Guide, 3rd Edition ver. 1.2* ("The Red Book").

Reseña

- Primitivas Básicas en OpenGL
- GLUT y ciclo visualización / interacción
- Transformaciones y Vistas
- Otras primitivas de Desarrollo

Especificar Vértices de Objetos (C2 p42)

- **Cada objeto es especificado por vértices**

```
glVertex3f (2.0, 4.1, 6.0);   // especifica el vértice en la coord.  x, y, z  (2.0, 4.1, 6.0).
                              // El "3f" significa 3 coord. de punto flotante.
```

  - **Otro ejemplo:**

```
glVertex2i (4, 5);          // 2 enteros para x e y, z = 0.
glVertex3fv (vector);       // float vector[3] = {5.0, 3.2, 5.0};
```

- **Especificación de Color, afecta a cualquier vértice**
  - `glColor3f (0.0, 0.5, 1.0);` // no Rojo, Verde media-intensidad, Azul color total.

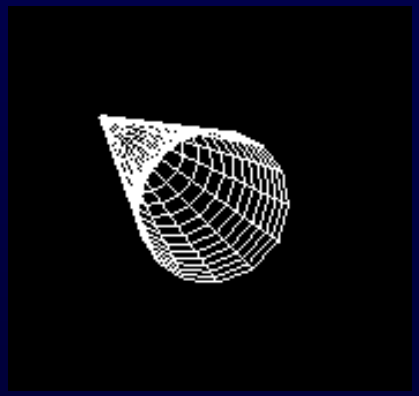
- **Los Vértices se especifican sólo entre `glBegin(`*`mode`*`)` y `glEnd()`, Polígonos: en orden de las agujas del reloj.**

```
glBegin (GL_TRIANGLES);
  glVertex2i (0, 0);
  glVertex2i (2, 0);
  glVertex2i (1, 1);
glEnd();
```

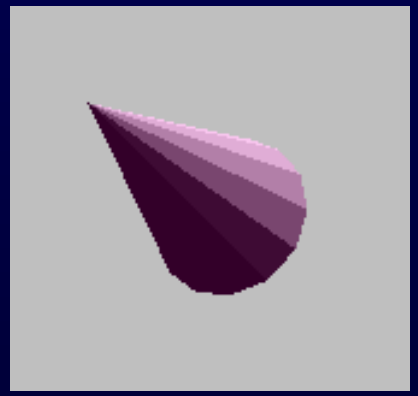
Tipos Primitivos de glBegin (C2, p44)

- Puntos GL_POINTS
- Líneas GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
- Triángulos GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
- Cuádruplas GL_QUADS, GL_QUAD_STRIP
- Polígonos GL_POLYGON

```
glBegin(GL_LINES);
  [conj. de glVertex];
glEnd();
```



```
glBegin(GL_QUADS);
  [conj. de glVertex];
glEnd();
```



GLUT – OpenGL Utility Toolkit (Apéndice D)

- **GLUT es una biblioteca que permite la administración de ventanas y eventos del sistema, en forma independiente de la plataforma.**

Comienzo:

```
int main (int argc, char *argv[])
{
   glutInit(&argc, argv);    // Pasa argumentos originales a GLUT
   glutInitDisplayMode (GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
   glutInitWindowSize (windowWidth, windowHeight);
   glutInitWindowPosition (0, 0);
   glutCreateWindow ("Programa de Prueba");

   SetStates();            // Inicializa estados (codigo propio).
   RegisterCallbacks();    // Seteo rut. de atención de eventos (cod. propio).

   glutMainLoop();         // Transfiere el control a GLUT. No Retorna.
   return 0;
}
```

GLUT: Atención de Eventos

- **Registro de funciones que son llamadas cuando un evento ocurre.**

Ejemplos:

```
glutDisplayFunc( Display );              // cuando hay que dibujar
glutKeyboardFunc( Keyboard );            // detecta tecla apretada
glutReshapeFunc( Reshape );              // modificación de ventana
glutMouseFunc( Mouse );                  // cambios en botón de mouse
glutMotionFunc( MouseDraggedFunc );      // movimiento de mouse
glutIdleFunc( Idle );                    // eventos en estado ocioso
```

OpenGL – Buffers de Rendering

- **Los Buffers se utilizan para almacenar Color y Profundidad.**
  - El buffer de profundidad, permite realizar la eliminación de caras ocultas, de manera que se puedan ordenar los objetos en el espacio 3D.

- **Doble buffering:**
  - Se dibuja en el *back* buffer mientras el *front* buffer esta siendo mostrado.
  - Cuando finaliza el dibujado, se intercambian, y se comienza a trabajar nuevamente en el nuevo *back* buffer.
  - `glutSwapBuffers();  // se llama al finalizar el dibujado`

- **Limpieza de buffers:**

```
// Limpia la pantalla y el buffer al color establecido.
glClearColor (0.0, 0.0, 0.0, 0.0);

// Limpia ambos buffers.
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

OpenGL – Transformaciones y Vistas (C3)

OpenGL tiene 3 modos diferentes de matrices de Rendering:

- GL_MODELVIEW
- GL_PROJECTION
- GL_TEXTURE
- Ej: cuando se desea trabajar en modo matríz de proyección:

```
glMatrixMode(GL_PROJECTION);
```

- **La matríz *Modelview* se utiliza para ejecutar transformaciones de objetos.**
- **La matríz *Projection* se utiliza para realizar transformaciones de Perspectiva. Usualmente se realiza una sola vez al comienzo.**
- **La matríz *Texture* se utiliza para elaborar Texturas.**

OpenGL: Matríz Modelview

- Transforma el Punto de Vista y los Objetos dentro de una escena.
- Ejemplo:

```
glMatrixMode(GL_MODELVIEW);// setea la matríz
glLoadIdentity();          // carga la matríz identidad
glTranslatef(10.5, 0, 0);  // traslada 10.5 unidades en eje x
glRotatef(45, 0, 0, 1);    // rota 45 grados sent. reloj sobre eje z
DrawCube();                // func. que dibuja un cubo en el origen
```

- Donde queda el objeto?
- Resp.: sobre el eje x, rotado 45 grados SR. Imágen en página 107, fig. 3-4.

  Nota: las operaciones se multiplican a derecha, la transformación justo antes de `DrawCube()` es la primera que se ejecuta.

- Se puede utilizar `gluLookAt(...)` (pág. 119) junto a las rotaciones y traslaciones para modificar el punto de vista.

OpenGL: Matríz de Projection

- **Habilita una Proyección de Perspective. Como los objetos son proyectados en pantalla.** (pág. 123)
- **Opciones:**
  - `glFrustrum (...);` // crea el cono de proyección definido por el usuario
  - `gluPerspective (fovy, aspect, near, far);`
    // calcula el cono de proyección, dado el ángulo del punto de vista, relación de aspecto y planos de corte de frente y fondo.
  - `gluOrtho2D (...);` // crea una proyección ortográfica paralela para 2D.
- **Ejemplo:**

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(64, (float)windowWidth / (float)windowHeight, 4, 4096);
```

OpenGL – Seteo de Estados

- OpenGL es una máquina de *estado*: los polígonos son afectados por el color corriente, la transformación, el modo de dibujo, etc..
- Habilitar y deshabilitar características tales como, iluminación, texturado y matizado Alfa (alpha blending).
  - `glEnable (GL_LIGHTING);  // habilita iluminación (NO default)`
- Olvidar habilitar algo es un error de codificación muy común.

OpenGL: Normales e Iluminación

- **OpenGL puede simular la iluminación a partir de información de geometría. Se deben especificar vértices normales en la geometría.**
- **Los vectores Normales deben ser de longitud unitaria (normalizados).**

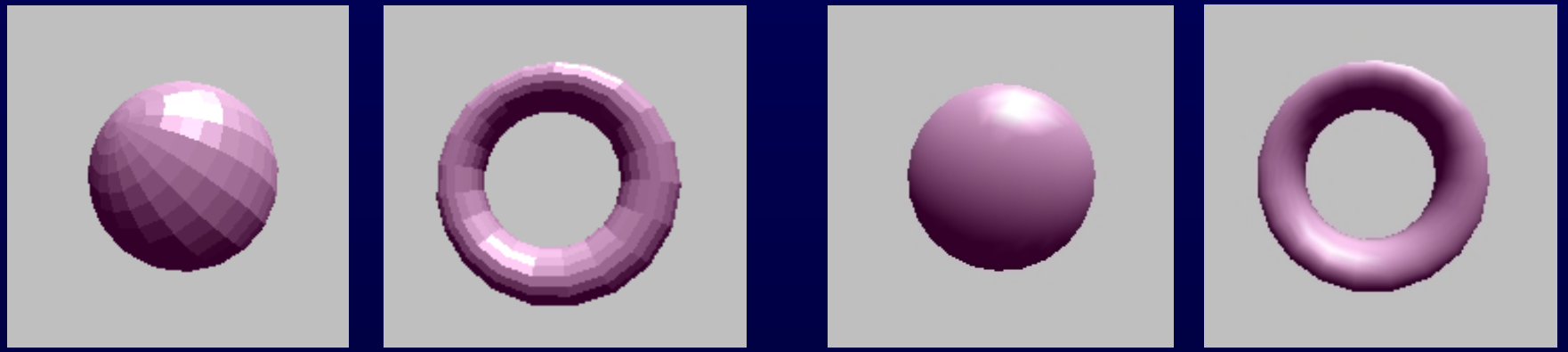
```
// cada vértice tiene una normal diferente
glColor3f (0.8, 1.0, 0.5);
glBegin(GL_TRIANGLES);
     glNormal3fv (n0);
     glVertex3fv (v0);
     glNormal3fv (n1);
     glVertex3fv (v1);
     glNormal3fv (n2);
     glVertex3fv (v2);
glEnd();

// todos los vértices tienen la misma normal
glBegin(GL_TRIANGLES);
     glNormal3fv (n0);
     glVertex3fv (v0);
     glVertex3fv (v1);
     glVertex3fv (v2);
glEnd();
```

OpenGL: Iluminación (C5 p173)

- `glEnable (GL_LIGHTING);`
- **OpenGL soporta un mínimo de 8 luces.**
  - ```
    glEnable (GL_LIGHT0);
    ...
    glEnable (GL_LIGHT7);
    ```
- Las luces tienen características tales como: posición, tipo, color, etc..
- Posición:
  - ```
    float light0Position[4] = {1.0, 0.0, 4.0, 1.0};
    glLightfv (GL_LIGHT0, GL_POSITION, light0Position);
    ```
- Los Tipos de luz son: puntual, direccional y spot. El último parámetro de la posición determina el tipo de luz.
- El Color puede ser uno de tres: Ambiental, Difuso, Especular.

OpenGL: Iluminación (cont.)

- **OpenGL suporta 2 modelos basicos de sombreado: flat and smooth.**
  - `glShadeModel(GL_FLAT);`
  - `glShadeModel(GL_SMOOTH);`

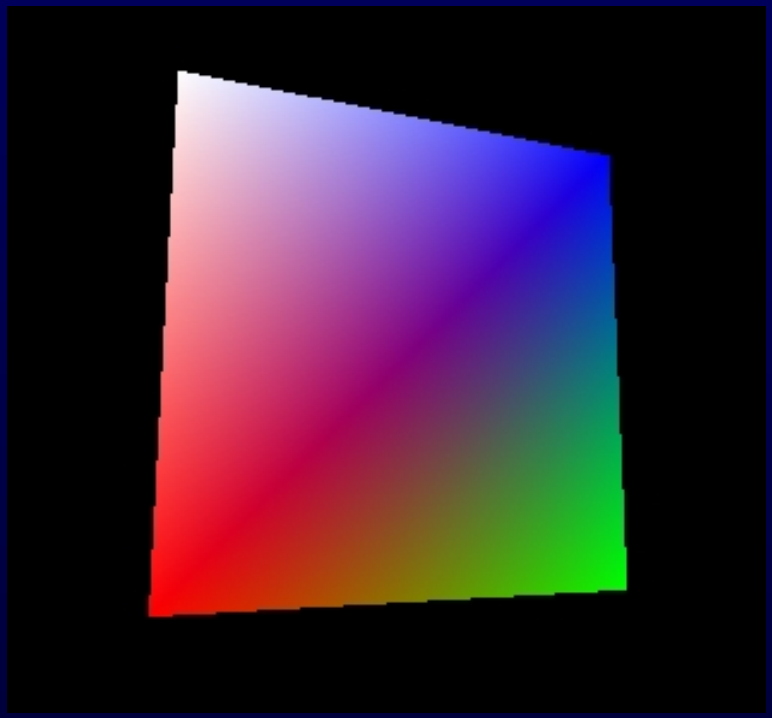


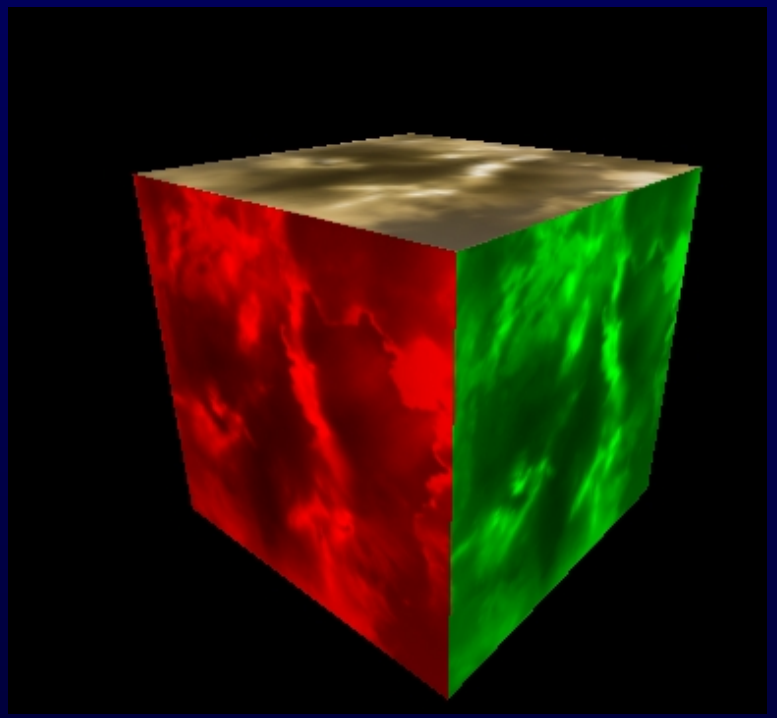
- Los cálculos de Luz pueden ser extensos.

OpenGL: Propiedades de Material (C5)

- **Se pueden especificar diferentes propiedades de material para diferentes polígonos, cambiando los efectos de luz.**
  - `glMaterial*(GLenum face, GLenum pname, TYPE param);`
- **Algunas propiedades (`pname`), pag. 202:**
  - GL_AMBIENT: color Ambiente del material
  - GL_DIFFUSE: color Difuso (Opaco)
  - GL_SPECULAR: componente Especular (Reflejo)
  - GL_SHININESS: exponente Especular (Brillo)

Demo





Desarrollo

- **Links de interés:**
  - www.opengl.org.
  - www.cs.brown.edu/courses/cs224/links.html
  - www-imagis.imag.fr/Members/Fabrice.Neyret/doc/opengl.html.
- **En Linux:**
  - GLUT está instalado.
  - En su Makefile de compilar con flags: -L/usr/lib -lGL -lGLU –lglut (chequear en su configuración!).
- Para depuración, se debe llamar a glutReportErrors() en cada loop de muestra (display).
  - Esto mostrará un reporte de errores que han ocurrido durante el rendering, taled como operaciones ilegales en un glBegin/glEnd pair.