5. Input Devices
Introduction to OpenGL

Dr Ann McNamara

Graphics Input Primitives

- String
- Choice/Selection
- Valuator
- Locator – coordinate pair x,y
- Pick required for interactive graphics
Input Devices

• There are hundreds of input devices
• Keyboard most familiar input device
• Early computers communicated one bit at a time
• Users had to flip switches or plug wires into switchboards to enter data

Keyboard

• Types characters on screen at point identified by cursor
• Some keys have special functions
  – Cursor (arrow) keys
  – Delete key
  – Enter key
  – Function keys (f-keys)
• QWERTY keyboard standard
Pointing devices

• Pointing devices are inadequate for input text and numbers into computers. They are usually used with keyboards.
• Mouse
  – Sends commands
  – Points around screen
    • Trackball
      – Upside down mouse

Pointing devices

• Touch pad
  – Small panel
  – Track point
  – Small joystick
  – IBM ThinkPad and other laptops
• Joystick
  – Like arcade games
• Graphics tablets
  – Used by artists and designers
• Touch Screens
  – Common in kiosks
Scanners

- A 2D scanner converts photographs, drawings, charts and other printed information into bit patterns that can be stored and manipulated in a computer's memory.
- 3D Scanners
3D Scanners

Colortron
Digital Cameras

- Convert real world to digital, like scanners
- Not limited to two dimensional objects

Motion Capture
Getting Started with OpenGL

Requirements

• An environment that lets us
  – write programs
  – execute programs
• Hardware to display (screen)
• Library of software tools to perform
drawing of graphics primitives.
Device Independent Programming

• Same program can be compiled and run on a variety of graphics environments.
• API (Application Programmer Interface)
  – a collection of routines
  – a model of how the routines work

Event Driven Programming

• Windows based systems are event driven
  – mouse click, key press, resize window
• System automatically handles an event queue
• Callback Function
• OpenGL comes with a Utility Toolkit
  – glutMouseFunc(mymouse) registers the function mymouse() as the function executed when mouse events occur.
OpenGL®

• OpenGL is a *software interface* to a graphics system implemented in hardware or software.
• It is theoretically *device independent*.
• OpenGL uses a *client-server* model, where client and server need not reside on the same machine.
• Default language is C/C++.
• The actual drawing operations are performed by the underlying windows system or *accelerated graphics hardware* (where available, e.g. 3Dfx, PowerVR, Intergraph, Silicon Graphics).

OpenGL®

• Programmer’s model:
OpenGL®

- OpenGL is interactive and dynamic and therefore we must handle interaction from the user ⇒ event driven model
- The GL library is the core OpenGL system:
  - modeling, viewing, lighting, clipping
- The GLU library (GL Utility) simplifies common tasks:
  - creation of common objects (e.g. spheres, quadrics)
  - specification of standard views (e.g. perspective, orthographic)
- The GLUT library (GL Utility Toolkit) provides the interface with the windowing system.
  - window management, menus, mouse interaction

Skeleton Program

```c
void main()
{
    initialise things
    create a screen window
    glutDisplayFunc(myDisplay);
    glutReshapeFunc(myReshape);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKeyboard);
    perhaps initialise other things
    glutMainLoop();
}
define all callback functions
```
Creating OpenGL ® Applications

OpenGL® Conventions

- all function names begin with `gl`, `glu` or `glut`
  - `glBegin(...)`
  - `gluCylinder(...)`
  - `glutInitDisplayMode(...)`
- constants begin with `GL_`, `GLU_`, or `GLUT_`
  - `GL_POLYGON`
- Function names can encode parameter types, e.g. `glVertex*`:
  - `glVertex2i(1, 3)`
  - `glVertex3f(1.0, 3.0, 2.5)`
  - `glVertex4fv(array_of_4_floats)`
Opening a Window

```c
void main(int argc, char **argv)
{
    glutInit(&argc, argv); //initialise toolkit
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); //set the display mode
    glutInitWindowSize(640,480) //set the window size
    glutInitWindowPosition(100,150); //set the window position on screen
    glutCreatWindow("My First Attempt"); //open the screen window

    //register the callback functions
    glutDisplayFunc(myDisplay);
    glutReshapeFunc(myReshape);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKeyboard);

    myInit(); //additional initialisations
    glutMainLoop(); //go into a perpetual loop
}
```

OpenGL®

- To create a red polygon with 4 vertices:
  ```c
  glColor3f(1.0, 0.0, 0.0);
  glBegin(GL_POLYGON);
      glVertex3f(0.0, 0.0, 3.0);
      glVertex3f(1.0, 0.0, 3.0);
      glVertex3f(1.0, 1.0, 3.0);
      glVertex3f(0.0, 1.0, 3.0);
  glEnd();
  ```
- `glBegin` defines a geometric primitive:
  ```c
  GL_POINTS, GL_LINES, GL_LINE_LOOP,
  GL_TRIANGLES, GL_QUADS, GL_POLYGON...
  ```
- All vertices are 3D and defined using `glVertex`*
OpenGL®

- We can use *per-vertex* information.
- To create the RG colour square:

```c
glShadeModel(GL_SMOOTH);
glBegin(GL_POLYGON);
    glColor3f(1.0, 0.0, 0.0); // Red
    glVertex3f(0.0, 0.0, 3.0);
    glColor3f(0.0, 0.0, 0.0); // Black
    glVertex3f(1.0, 0.0, 3.0);
    glColor3f(0.0, 1.0, 0.0); // Green
    glVertex3f(1.0, 1.0, 3.0);
    glColor3f(1.0, 1.0, 0.0); // Yellow
    glVertex3f(0.0, 1.0, 3.0);
    glVertex3f(1.0, 1.0, 3.0);
    glEnd();
```

OpenGL® GLUT Event Loop

- Interaction with the user is handled through an *event loop*.
- Application registers *handlers* (or *callbacks*) to be associated with particular events:
  - mouse button, mouse motion, timer, resize, redraw
- GLUT provides a wrapper on the X-Windows or Win32 core event loop.
- X-Windows or Win32 manages event creation and passing, GLUT uses them to catch events and then invokes the appropriate callback.
- GLUT is more general than X or Win32 etc.
  - more portable: user interface code need not be changed.
  - less powerful: implements a common subset
OpenGL® GLUT Event Loop

User → Windows System → GLUT Event List

Operating System
Application

GlutMainLoop() = While(TRUE)
               e = get_next_event()
               switch(e)
               case MOUSE:
                   call registered MouseFunc
               case RESIZE:
                   call registered ReshapeFunc
               ....