2D Vs. 3D

- 2D:
  - Flat
  - \((x,y)\) color values on screen
  - Objects no depth or distance from viewer

- 3D:
  - \((x,y,z)\) values on screen
  - Perspective: objects have distances from viewer

Creating 3D

- Start with 3D shapes (modeling)
  - Basic shapes (cube, sphere, etc), meshes, etc
  - Scale them (may also stretch them)
  - Position them (rotate them, translate, etc)
- Then, render scene (realism)
  - Perspective
  - Color and shading
  - Shadows
  - Texture mapping
  - Fog
  - Transparency and blending
  - Anti-aliasing
- Practical note: modeling and rendering packages being sold (Maya, 3D studio max, etc)

3D Modeling example: Robot Hammer
3D Modeling example: Polygonal Mesh

Original: 424,000 triangles
60,000 triangles (14%) 1000 triangles (0.2%)
(courtesy of Michael Garland and Data courtesy of Iris Development.)

3D Effects example: Texturing

OpenGL Basics
- OpenGL’s primary function – rendering
- Rendering? – Convert geometric/mathematical object descriptions into images
- OpenGL can render:
  - Geometric primitives (lines, dots, etc)
  - Bitmap images (.bmp, .jpg, etc)
OpenGL Basics
- Application Programming Interface (API)
- Low-level graphics rendering API
- Widely used – will be used in this class
- Maximal portability
  - Display device independent
  - Window system independent (Windows, X, etc)
  - Operating system independent (Unix, Windows, etc)
- Event-driven

OpenGL: Event-driven
- Program only responds to events
- Do nothing until event occurs
- Example Events:
  - mouse clicks
  - keyboard stroke
  - window resize
- Programmer:
  - defines events
  - actions to be taken
- System:
  - maintains an event queue
  - takes programmer-defined actions

OpenGL: Event-driven
- Sequential program
  - Start at main( )
  - Perform actions 1, 2, 3….N
  - End
- Event-driven program
  - Initialize
  - Wait in infinite loop
    - Wait till defined event occurs
    - Take defined actions
- World’s most popular event-driven program?

OpenGL: Event-driven
- How in OpenGL?
  - Programmer registers callback functions
  - Callback function called when event occurs
- Example:
  - Declare a function myMouse to respond to mouse click
  - Register it: Tell OpenGL to call it when mouse clicked
  - Code? glutMouseFunc(myMouse);
**GL Utility Toolkit (GLUT)**

- OpenGL
  - is window system independent
  - Concerned only with drawing
  - No window management functions (create, resize, etc)
  - Very portable
- GLUT:
  - Minimal window management: fast prototyping
  - Interfaces with different windowing systems
  - Allows easy porting between windowing systems

**GL Utility Toolkit (GLUT)**

- No bells and whistles
  - No sliders
  - No dialog boxes
  - No menu bar, etc
- To add bells and whistles, need other API:
  - X window system
  - Apple: AGL
  - Microsoft :WGL, etc

**Program Structure**

- Configure and open window (GLUT)
- Initialize OpenGL state
- Register input callback functions (GLUT)
  - Render
  - Resize
  - Input: keyboard, mouse, etc
- My initialization
  - Set background color, clear color, drawing color, point size, establish coordinate system, etc.
- glutMainLoop( )
  - Waits here infinitely till action is selected

**GLUT: Opening a window**

- GLUT used to open window
  - glutInit(&argc, argv);
    - initializes
  - glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    - sets display mode (e.g. single buffer with RGB)
  - glutInitWindowSize(640,480);
    - sets window size (WxH)
  - glutInitPosition(100,150);
    - sets upper left corner of window
  - glutCreateWindow("my first attempt");
    - open window with title "my first attempt"
OpenGL Skeleton

```c
void main(int argc, char** argv){
  // First initialize toolkit, set display mode and create window
  glutInit(&argc, argv); // initialize toolkit
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(640, 480);
  glutInitWindowPosition(100, 150);
  glutCreateWindow("my first attempt");
  // … then register callback functions,
  // … do my initialization
  // … wait in glutMainLoop for events
}
```

GLUT Callback Functions

- Register all events your program will react to
- Event occurs => system generates callback
- Callback: routine system calls when event occurs
- No registered callback = no action

GLUT Callback Functions

- GLUT Callback functions in skeleton
  - glutDisplayFunc(myDisplay): window contents need to be redrawn
  - glutReshapeFunc(myReshape): called when window is reshaped
  - glutMouseFunc(myMouse): called when mouse button is pressed
  - glutKeyboardFunc(mykeyboard): called when keyboard is pressed or released
- glutMainLoop( ): program draws initial picture and enters infinite loop till event

Example: Rendering Callback

- Do all your drawing in the display function
- Called initially and when picture changes (e.g. resize)
- First, register callback in main( ) function
- glutDisplayFunc( display );
- Then, implement display function

```c
void display( void )
{
  // put drawing stuff here
  glBegin( GL_LINES );
  glVertex3fv( v[0] );
  glVertex3fv( v[1] );
  glEnd();
}
```
OpenGL Skeleton

```c
void main(int argc, char** argv){
    // First initialize toolkit, set display mode and create window
    glutInit(&argc, argv);  // initialize toolkit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("my first attempt");
    // ... now register callback functions
    glutDisplayFunc(myDisplay);
    glutReshapeFunc(myReshape);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKeyboard);

    myInit();
    glutMainLoop();
}
```

References

- Hill, chapter 2