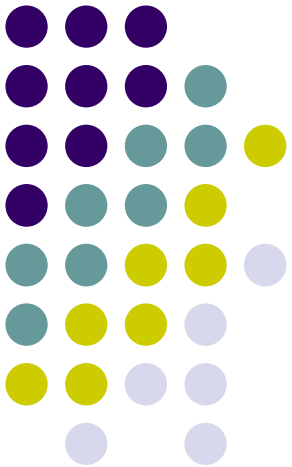
Computer Graphics
CS 543 – Lecture 1 (Part 2)

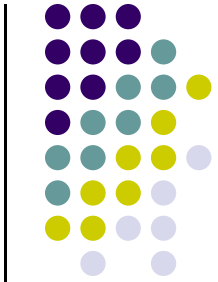
Prof Emmanuel Agu

Computer Science Dept.

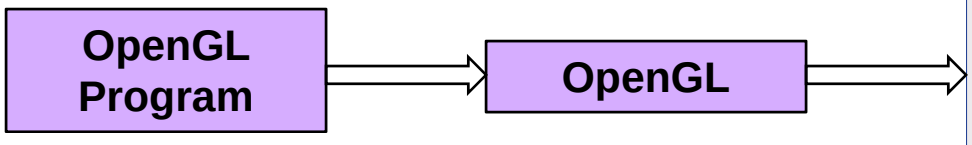
Worcester Polytechnic Institute (WPI)

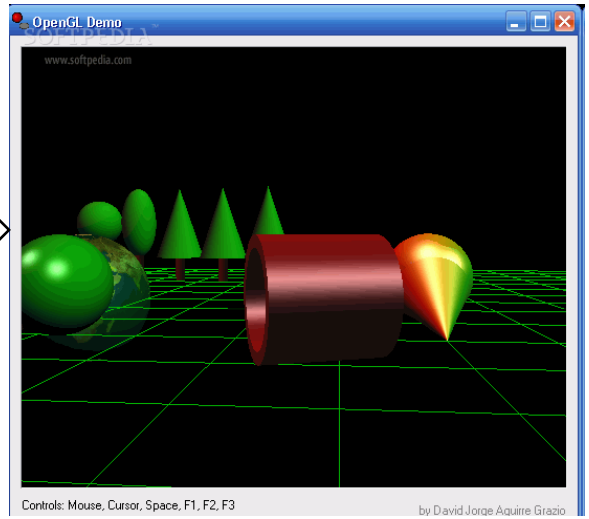


OpenGL Basics

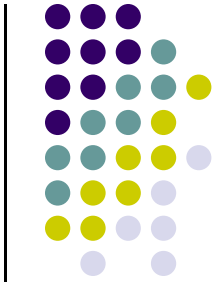


- OpenGL's function – Rendering (or drawing)
- Rendering? – Convert geometric/mathematical object descriptions into images
- OpenGL can render:
  - **Geometric primitives (lines, dots, etc)**
  - **Bitmap images (pictures, .bmp, .jpg, etc)**
- OpenGL does not manage drawing window



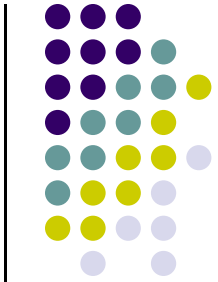


OpenGL Basics

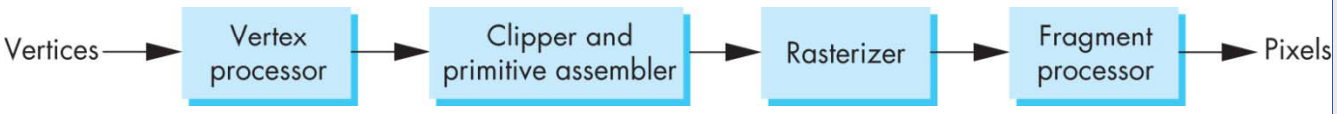


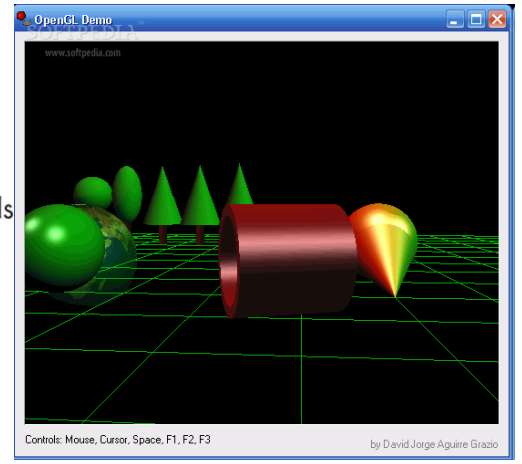
- Low-level graphics rendering API
- Maximal portability
  - **Display device independent (Monitor type, etc)**
  - **Window system independent based (Windows, X, etc)**
  - **Operating system independent (Unix, Windows, etc)**
- OpenGL programs behave same on different devices, OS
- Event-driven

Simplified OpenGL Pipeline

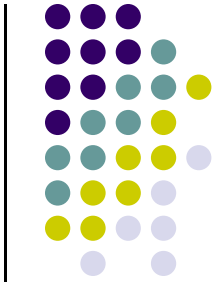


- Vertices go in, sequence of steps (vertex processor, clipper, rasterizer, fragment processor) image rendered

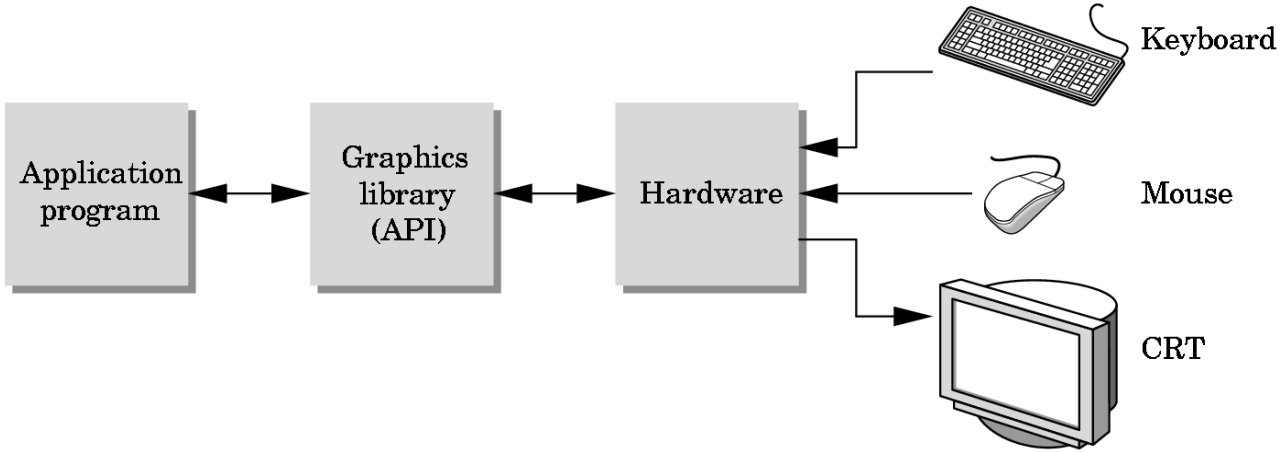




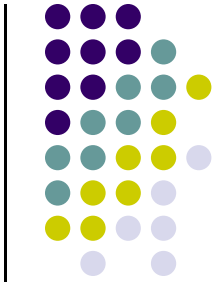
OpenGL Programming Interface



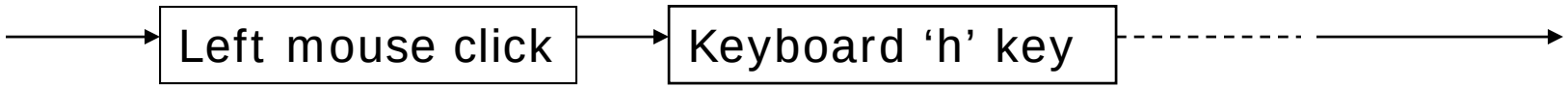
- Programmer sees the graphics system through a software interface: Application Programmer Interface (API)



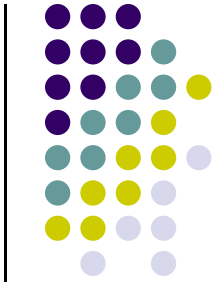
OpenGL: Event-driven



- 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 event queue
  - takes programmer-defined actions

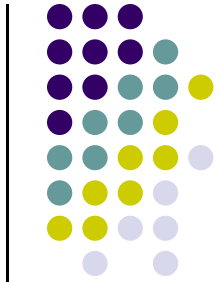


OpenGL: Event-driven

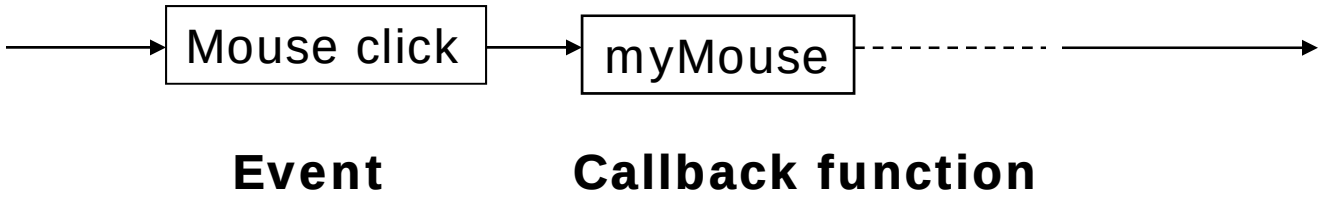


- Sequential program
  - **Start at main( )**
  - **Perform actions 1, 2, 3…. *N***
  - **End**
- Event-driven program
  - **Start at main( )**
  - **Initialize**
  - **Wait in infinite loop**
    - **Wait till defined event occurs**
    - **Event occurs => Take defined actions**
- What is World's most popular event-driven program?

OpenGL: Event-driven

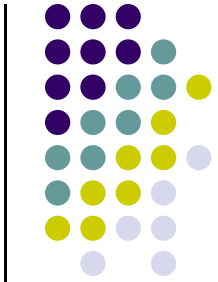


- How in OpenGL?
  - Programmer registers callback functions (event handler)
  - Callback function called when event occurs
- Example: Programmer
  1. Declare function *myMouse*, responds to mouse click
  2. Register it: glutMouseFunc(*myMouse*);

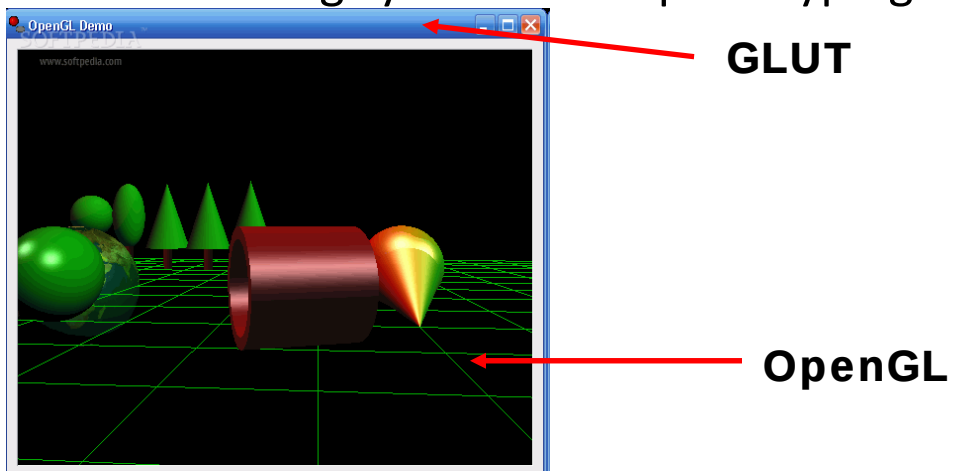


Note: OS receives mouse click, calls callback

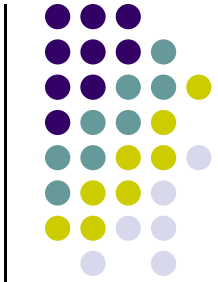
GL Utility Toolkit (GLUT)



- OpenGL
  - Window system independent
  - Concerned only with drawing
  - No window management (create, resize, etc), very portable
- GLUT:
  - Minimal window management
  - Interfaces with different windowing systems
  - Easy porting between windowing systems. Fast prototyping

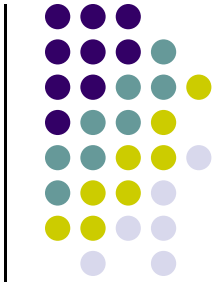


Some OpenGL Backround



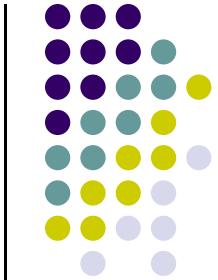
- OpenGL implemented either on graphics card or in software (e.g. Mesa)
- Software renderer actually runs on CPU and is slower
- OpenGL initially fixed function pipeline
  - Functions to generate picture fixed
  - Programmer basically invoked functions, set arguments
  - Restrictive!!
- Shaders allow programmer to write some OpenGL functions and load them
- OpenGL was fixed function up to version 1.x
- Shaders initially proposed as *extensions* to version 1.4
- Shaders became part of core in OpenGL version 2.0

Some OpenGL Backround



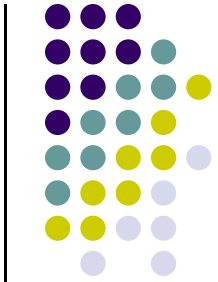
- **Extensions:** Core versions remain stable for years
- New ideas implemented as extensions that cards ***may*** choose to support
- Example: OpenGL shaders initially published as ARB extensions (ARB_vertex_shader and ARB_fragment_shader)
- Shaders part of core OpenGL from version 2.0 till date (version 4.2)
- For this class need access to either
  - Graphics card that supports OpenGL version 2.0 or later + ARB extensions (ARB_vertex_shader and ARB_fragment_shader)
  - OpenGL version 3.0 or later

glInfo: Finding out about your Graphics Card



- Gives OpenGL version and extensions information supported by your graphics card
- Homework 0!

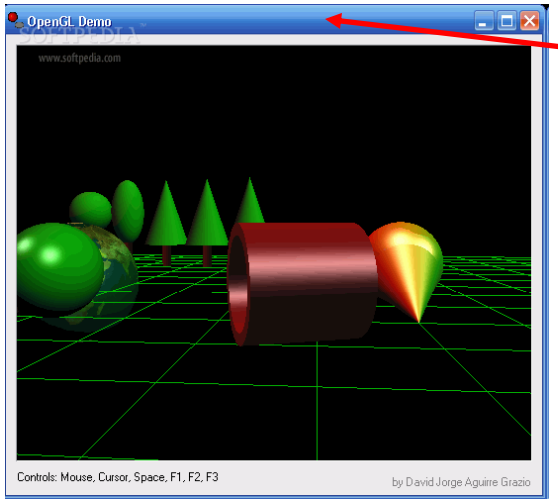
Other OpenGL Versions



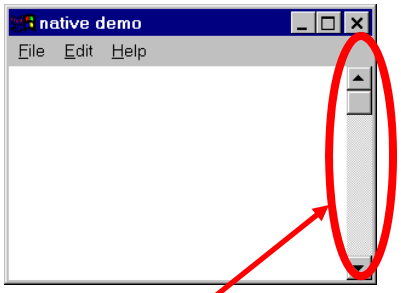
- OpenGL ES (Mobile Devices)
  - Embedded systems
  - Version 1.0 simplified OpenGL 2.1
  - Version 2.0 simplified OpenGL 3.1, shader based
- WebGL
  - Javascript implementation of ES 2.0
  - Supported on newer browsers
- OpenGL 4.1 and 4.2
  - Add geometry shaders and tessellator

GL Utility Toolkit (GLUT)

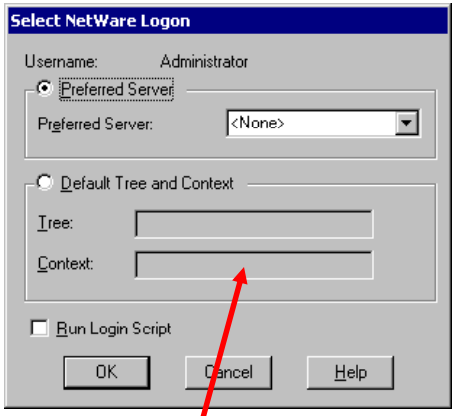
- No bells and whistles
  - No sliders
  - No dialog boxes
  - No elaborate menus, etc
- To add bells and whistles, use system's API or GLUI:
  - X window system
  - Apple: AGL
  - Microsoft :WGL, etc



GLUT (minimal)

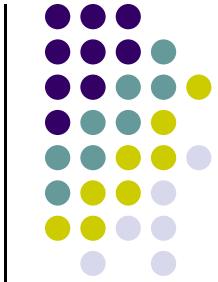


Slider

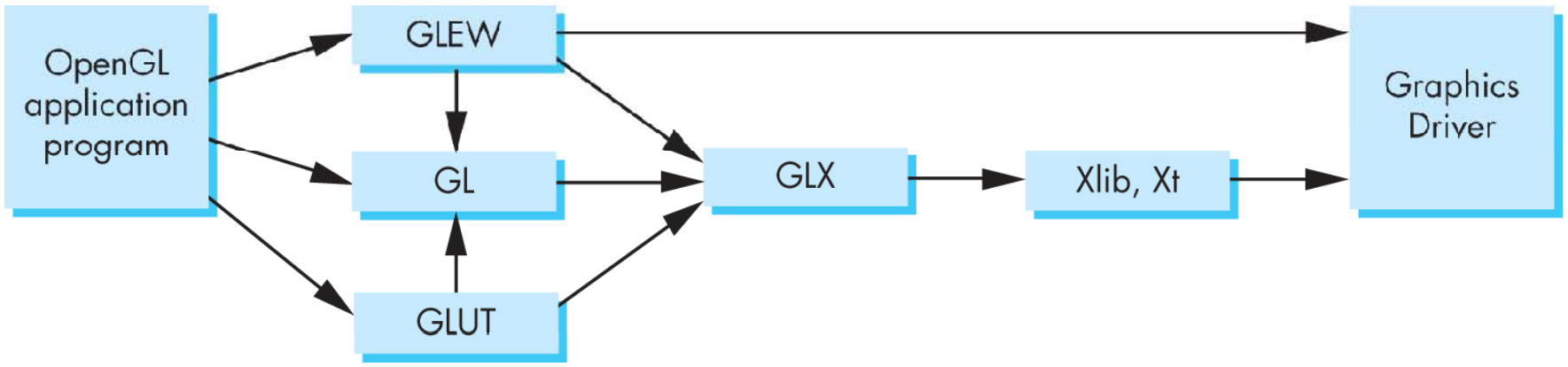


Dialog box

GLEW

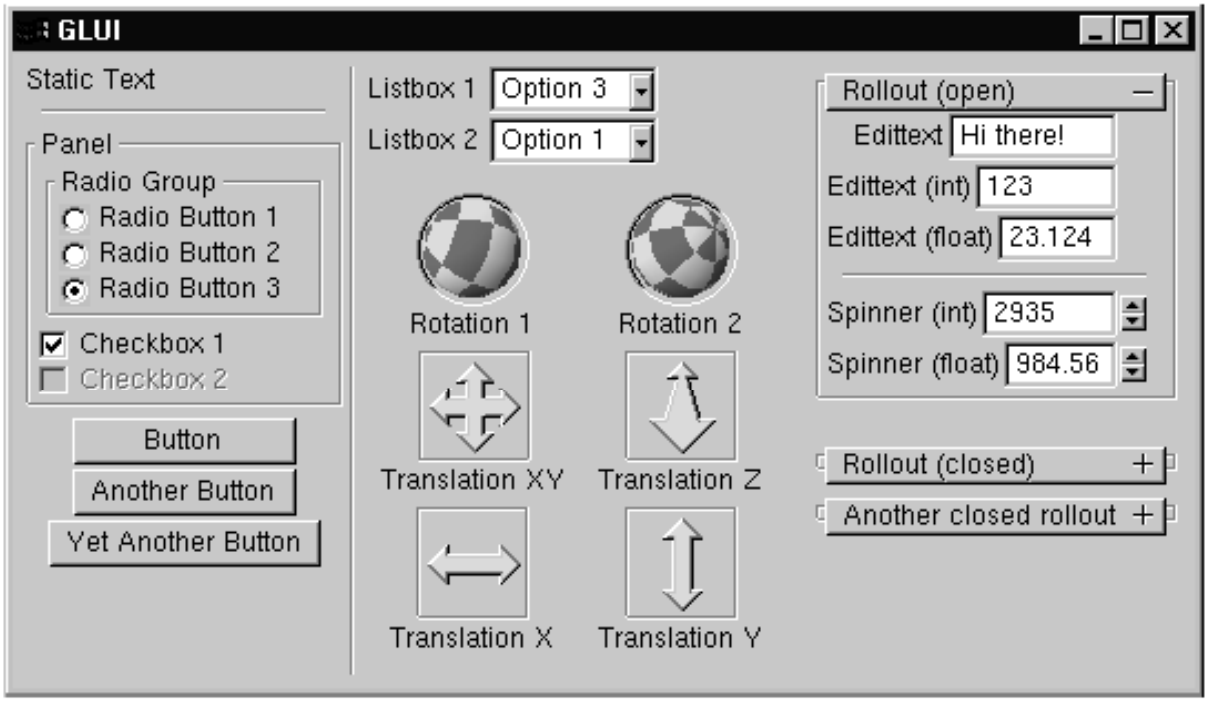


- OpenGL Extension Wrangler Library
- Makes it easy to access OpenGL extensions available on a particular system
- More no this later

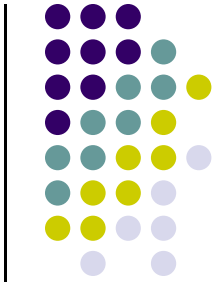


GLUI

- User Interface Library
- Provides sophisticated controls and menus
- Not used in this class/optional



Getting Started: First OpenGL program (Visual studio instructions)



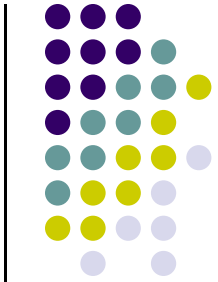
1. Create empty project
2. Create blank console application (C program)
3. Add console application to project
4. Include **`glew.h`** and **`glut.h`** at top of your program

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

Note: `GL/` is sub-directory of compiler **`include/`** directory

- **`glut.h`** contains GLUT functions, also includes **`gl.h`**
- OpenGL drawing functions in **`gl.h`**

Getting Started…



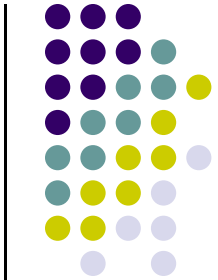
- On windows, add **windows.h** for more elaborate windowing functions (sliders, dialog boxes, etc)

```
#include <windows.h>  // add this before gl.h, glu.h
```

- Most OpenGL applications use standard C library (e.g for **printf),** so

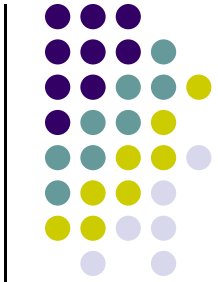
```
#include <stdlib.h>
#include <stdio.h>
```

Program Structure



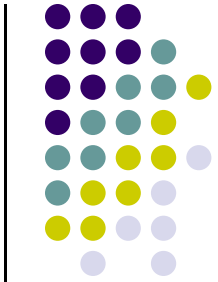
- Configure and open window (GLUT)
  - Configure Display mode, Window position, window size
- Initialize GLEW
- Initialize OpenGL state
- Register input callback functions (GLUT)
  - Render, resize, input: keyboard, mouse, etc
- My initialization
  - Set background color, clear color, drawing color, point size, establish coordinate system, etc.
  - Generate points to be drawn
  - Initialize shader stuff
- glutMainLoop( )
  - Waits here infinitely till action is selected

GLUT: Opening a window

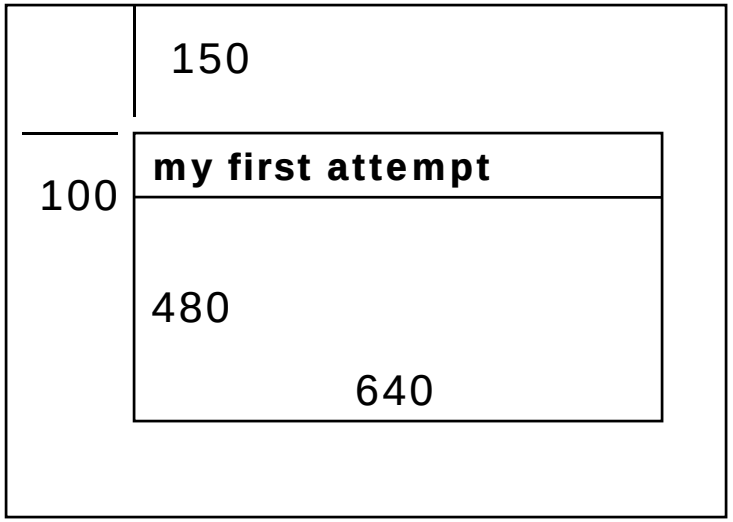


- GLUT used to create and open window
  - **`glutInit(&argc, argv);`**
    - Initializes GLUT
  - **`glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);`**
    - sets display mode (e.g. single buffer with RGB colors)
  - **`glutInitWindowSize(640,480);`**
    - sets window size (WxH) in pixels
  - **`glutInitPosition(100,150);`**
    - sets location of upper left corner of window
  - **`glutCreateWindow("my first attempt");`**
    - open window with title "my first attempt"
- Then also initialize GLEW
  - **`glewInit( );`**

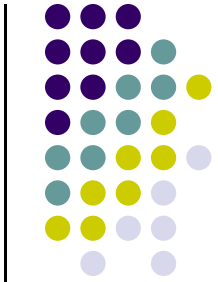
OpenGL Skeleton



```
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");
    glewInit( );

    // … then register callback functions,
    // … do my initialization
    // .. wait in glutMainLoop for events

}
```

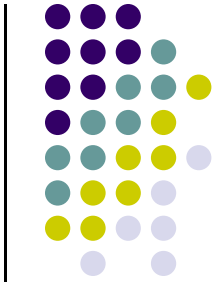


GLUT Callback Functions

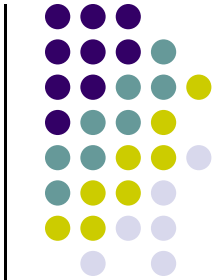


- Register all events your program will react to
- Callback: a function system calls when event occurs
- Event occurs => system callback
- No registered callback = no action
- Example: if no keyboard callback function, banging on keyboard generates NO RESPONSE!!

GLUT Callback Functions



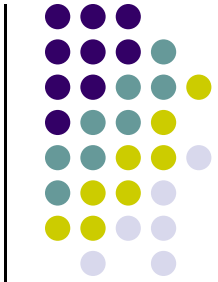
- GLUT Callback functions in skeleton
  - **`glutDisplayFunc(myDisplay)`:** Image to be drawn initially
  - **`glutReshapeFunc(myReshape)`:** called when window is reshaped
  - **`glutMouseFunc(myMouse)`:** called when mouse button is pressed
  - **`glutKeyboardFunc(mykeyboard)`:** called when keyboard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



OpenGL Skeleton

```
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");
    glewInit( );

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

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

Example of Rendering Callback



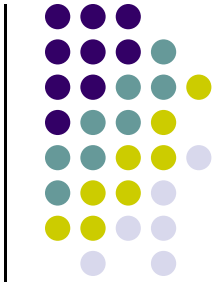
- Do all drawing code in display function
- Called initially and when picture changes (e.g.resize)
- First, register callback in main( ) function

```
glutDisplayFunc( display );
```

- Then, implement display function

```
void display( void )
{
    // put drawing commands here

}
```

References



- Angel and Shreiner Chapter 2
- Hill, chapter 2