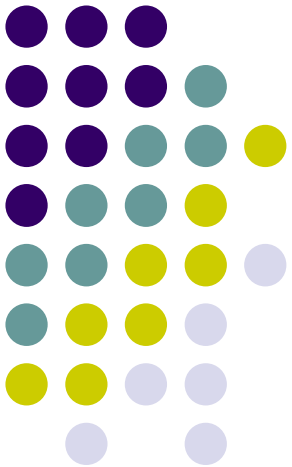
Computer Graphics
CS 543 – Lecture 2 (Part 2)
Viewports, GLUT Interaction & Menus

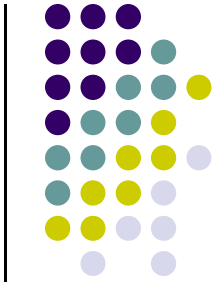
Prof Emmanuel Agu

Computer Science Dept.

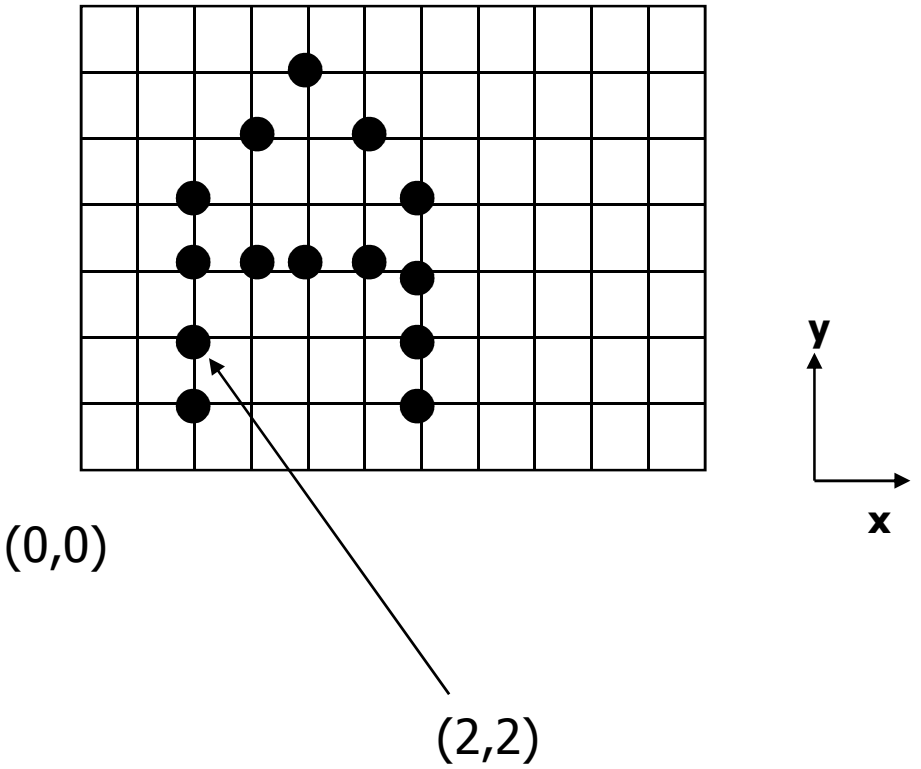
Worcester Polytechnic Institute (WPI)



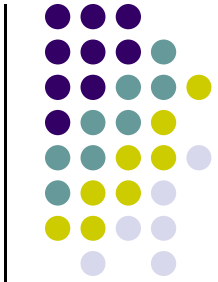
Screen Coordinate System



- Screen: 2D coordinate system (WxH)
- 2D Regular Cartesian Grid
- Origin (0,0): lower left corner (OpenGL convention)
- Horizontal axis – x
- Vertical axis – y
- Pixel positions: grid intersections

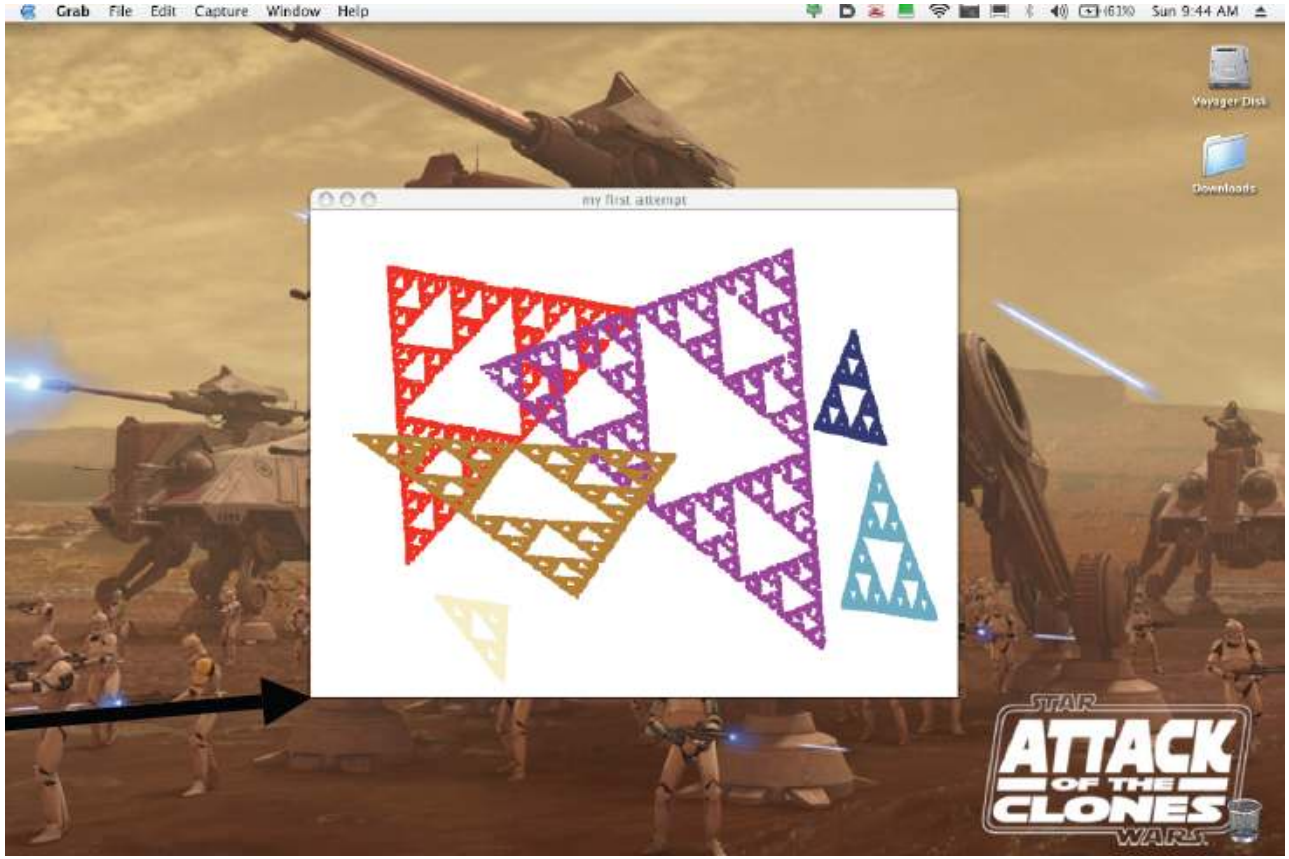


Screen Coordinate System

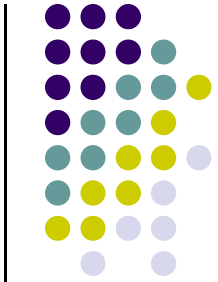


(0,0) is lower left corner of **OpenGL Window.**
NOT lower left corner of entire desktop

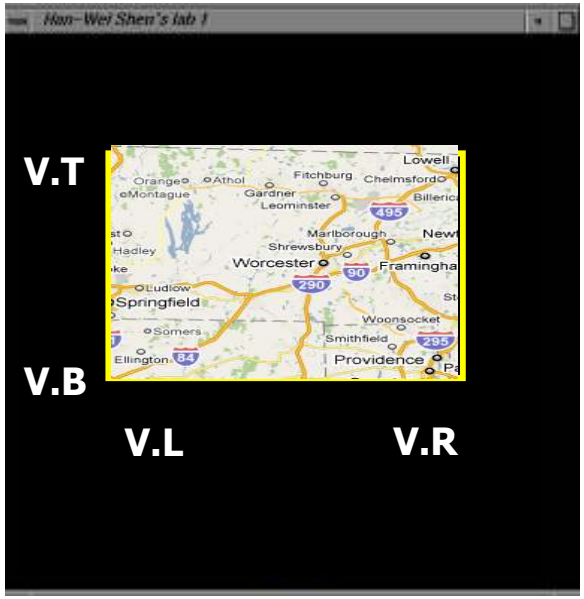
OpenGL's (0,0)



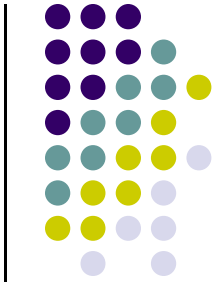
Definition: Viewport



- Rectangular region of screen used to display drawing
- Defined in screen coordinate system (pixels)
- Defined by (left, right, bottom, top) or (*V.L, V.R, V.B, V.T)*



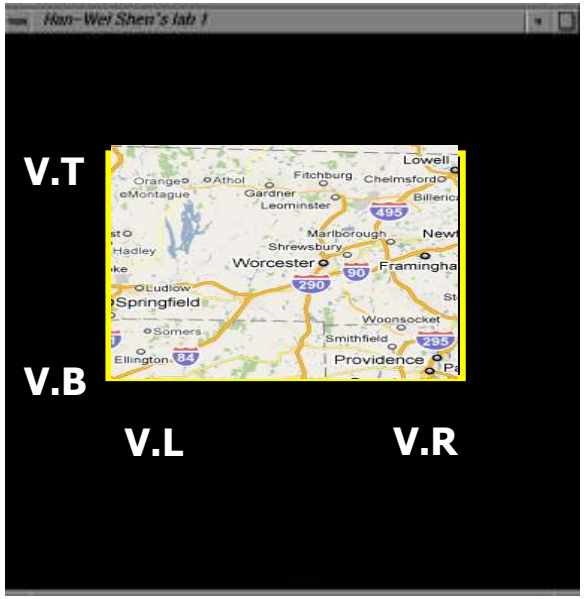
Defining a Viewport

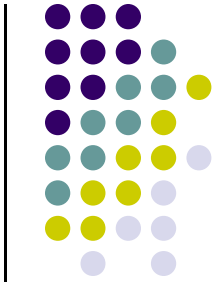


- To define viewport

  `glViewport(left, bottom, width, height)`

  `or glViewport(V.L, V.B, V.R – V.L, V.T – V.B)`

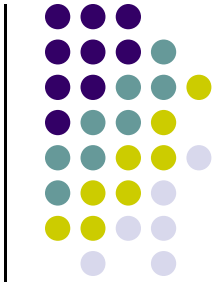




Keyboard Interaction

- Declare prototype
  - myKeyboard(unsigned int key, int x, int y)
- Register callback:
  - glutKeyboardFunc(myKeyboard): when keyboard is pressed
- Key values:
  - ASCII value of key pressed
- X,Y values:
  - Coordinates of mouse location
- Large **`switch`** statement to check which key

Example: Keyboard Callback



- Using keyboard to control program?
- 1. register callback in main( ) function

```
glutKeyboardFunc( myKeyboard );
```

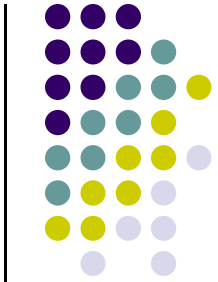
- 2. implement keyboard function

```
void myKeyboard(char key, int x, int y )
{    // put keyboard stuff here
   …….
    switch(key){    // check which key
       case 'f':
         // do stuff
       break;

        case 'k':
           // do other stuff
       break;

    }
……..
}
```

Note: Backspace, delete, escape keys checked using their ASCII codes

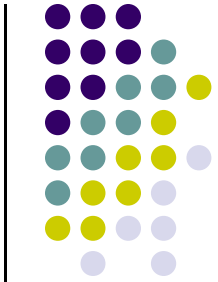


Keyboard Interaction

- For function, arrow and other special-purpose keys, use

```
glutSpecialFunc (specialKeyFcn);
…
Void specialKeyFcn (Glint specialKey, GLint, xMouse,
                                          Glint yMouse)
```

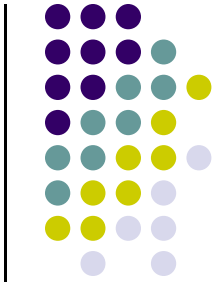
- Example: if (**`specialKey == GLUT_KEY_F1)`** // F1 key pressed
  - **`GLUT_KEY_F1, GLUT_KEY_F12,`** …. for function keys
  - **`GLUT_KEY_UP, GLUT_KEY_RIGHT,`** …. for arrow keys keys
  - **`GLUT_KEY_PAGE_DOWN, GLUT_KEY_HOME,`** …. for page up, home keys
- Complete list of special keys designated in **`glut.h`**



Mouse Interaction

- Declare prototype
  - `myMouse(int button, int state, int x, int y)`
  - `myMovedMouse`
- Register callbacks:
  - `glutMouseFunc(myMouse)`: mouse button pressed
  - `glutMotionFunc(myMovedMouse)`: mouse moves with button pressed
  - `glutPassiveMotionFunc(myMovedMouse)`: mouse moves with no buttons pressed
- Button returned values:
  - GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
- State returned values:
  - GLUT_UP, GLUT_DOWN
- X,Y returned values:
  - x,y coordinates of mouse location

Mouse Interaction Example

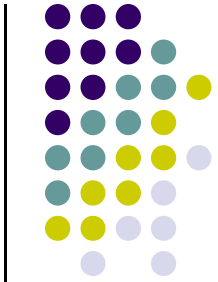


- Each mouse click generates separate events
- Store click points in **global** or **static** variable in mouse function
- **Example:** draw (or select) rectangle on screen
- Mouse y returned assumes y=0 at top of window
- OpenGL assumes y=0 at bottom of window. Solution? Flip mouse y

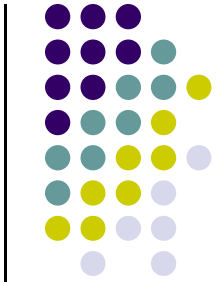
```
void myMouse(int button, int state, int x, int y)
{
   static GLintPoint corner[2];
   static int numCorners = 0;   // initial value is 0
   if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
   {
        corner[numCorners].x = x;
        corner[numCorners].y = screenHeight - y; //flip y coord
        numCorners++;
```

Screenheight is height of drawing window

Mouse Interaction Example (continued)



```
    if(numCorners == 2)
    {
         // draw rectangle or do whatever you planned to do
         glRecti(corner[0].x, corner[0].y,
                                corner[1].x, corner[1].y);
         numCorners == 0;
    }
    else if(button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
         glClear(GL_COLOR_BUFFER_BIT); // clear the window
    glFlush( );
}
```



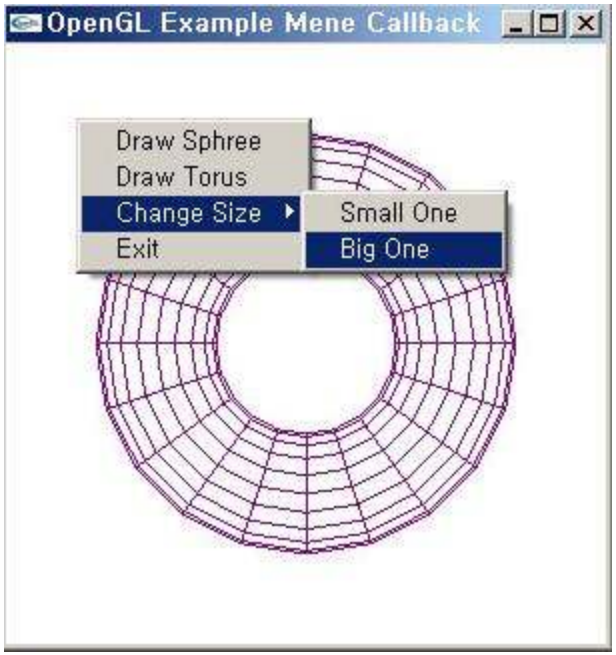
Menus

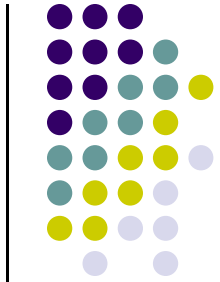
- Adding menu that pops up on mouse click

1. Create menu using **`glutCreateMenu(myMenu);`**

2. Use **`glutAddMenuEntry`** adds entries to menu

3. Attach menu to mouse button (left, right, middle) using **`glutAttachMenu`**





Menus

- Example:

Shows on menu

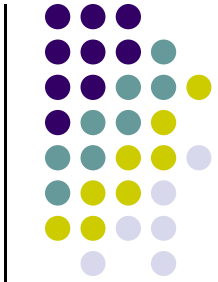
Checked in mymenu

```
glutCreateMenu(myMenu);
 glutAddMenuEntry("Clear Screen", 1);
 glutAddMenuEntry("Exit", 2);
 glutAttachMenu(GLUT_RIGHT_BUTTON);

 ….

 void mymenu(int value){
    if(value == 1){
        glClear(GL_COLOR_BUFFER_BIT);
        glFlush( );
    }
    if (value == 2) exit(0);
}
```

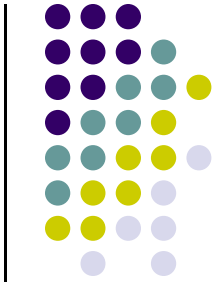
GLUT Interaction using other input devices



- Tablet functions (mouse cursor must be in display window)

```
glutTabletButton (tabletFcn);
…..
void tabletFcn(Glint tabletButton, Glint action, Glint
   xTablet, Glint yTablet)
```

- Spaceball functions
- Dial functions
- Picking functions: use your finger
- Menu functions: minimal pop-up windows within your drawing window
- Reference: *Hearn and Baker, 3rd edition (section 20-6)*



References

- Angel and Shreiner, Chapter 2
- Hill, chapter 2