# Chapter 10: OpenGL

By Kabita Dhital

3rd sem

# *Introduction to OpenGL*

- Silicon Graphics Inc. (SGI) started developing OpenGL in 1991 and released it in January 1992.
- OpenGL (Open Graphics Library) is a ***cross-platform, hardware-accelerated, language-independent, industrial standard API*** for producing 3D (including 2D) graphics.
- Modern computers have dedicated GPU (Graphics Processing Unit) with its own memory to speed up graphics rendering. ***OpenGL is the software interface to graphics hardware.***
- In other words, OpenGL graphic rendering commands issued by your applications could be directed to the graphic hardware and accelerated.

# *What is OpenGL?*

- Software API to graphics hardware.
- Designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms
- No high-level commands for describing models of three-dimensional objects
- Graphics rendering API (Low Level)
- High-quality color images composed of geometric and image primitives
- Window system independent
- Operating system independent
- Display device independent
- No windowing commands!
- Close enough to the hardware to get excellent performance

# Callback Function

- A callback function is a function which the library (**GLUT**) calls when it needs to know how to proccess something.
- e.g. when glut gets a key down event it uses the glutKeybourdFunc callback routine to find out what to do with a key press.
- See glutOverlayDisplayFunc to understand how distinct callbacks for the overlay and normal plane of a window may be established.
- When a window is created, no display callback exists for the window. It is the responsibility of the programmer to install a display callback for the window before the window is shown. A display callback *must* be registered for any window that is shown. If a window becomes displayed without a display callback being registered, a fatal error occurs. Passing NULL to glutDisplayFunc is illegal as of GLUT 3.0; there is no way to ``deregister'' a display callback (though another callback routine can always be registered).
- **GLUT supports a number of callbacks** to respond to events.

# Callback Function

- **GLFW** (Graphics Library Framework) is a lightweight utility library for use with OpenGL. It provides programmers with the ability to create and manage windows and OpenGL contexts, as well handle joystick, keyboard and mouse input.
- GLFW is used in programs that require a window, due to OpenGL not providing any mechanisms for creating the necessary contexts, managing windows, user input, timing, etc

# *Callback Functions*

- Callbacks are ***user-defined functions designed to react on specific events:***
  - Whenever OpenGL decided it needs to redraw window contents
  - What to do when a user resizes a window.
  - Handle mouse motions
  - React on keyboard,
  - What to do during idle period (no input from user)
- For OpenGL to become aware of your callbacks, you need to register them within it before you start drawing things.
- **Some of the callbacks are mandatory**, such as display, so that OpenGL know how to render your graphics.

# *Callback Functions*

- Programming interface for event-driven input
- Define a callback function for each type of event the graphics system recognizes.
- This user-supplied function is executed when the event occurs. Example: **glutMouseFunc(mymouse)**
- ***glutMouseFunc(myMouse);***
  - Handles mouse button presses. Knows mouse location and nature of button (up or down and which button).
- ***glutMotionFunc(myMotionFunc);***
  - Handles case when the mouse is moved with one or more mouse buttons pressed.

# *GLUT Callback Functions*

- *Contents of window need to be refreshed*: ***glutDisplayFunc()***
- *Window is resized or moved:* ***glutReshapeFunc()***
- *Mouse button action:* ***glutMouseFunc()***
- *Mouse moves while a button is pressed:* ***glutMotionFunc()***
- *Mouse moves regardless of mouse button state:* ***glutPassiveMouseFunc()***
- *Called when nothing else is going on:* ***glutIdleFunc()***
- *glutPassiveMotionFunc(myPassiveMotionFunc):* ***Handles case where mouse enters the window with no buttons pressed.***
- *glutKeyboardFunc(myKeyboardFunc):* Handles key presses and releases. Knows which key was pressed and mouse location.
- *glutMainLoop():* Runs forever waiting for an event. When one occurs, it is handled by the appropriate callback function.

# *Events in OpenGL*

| **Event** | **Example** | **OpenGL Callback Function** |
|---|---|---|
| Keypress | KeyDown<br>KeyUp | glutKeyboardFunc |
| Mouse | leftButtonDown<br>leftButtonUp | glutMouseFunc |
| Motion | With mouse press<br>Without | glutMotionFunc<br>glutPassiveMotionFunc |
| Window | Moving<br>Resizing | glutReshapeFunc |
| System | Idle<br>Timer | glutIdleFunc<br>glutTimerFunc |
| Software | What to draw | glutDisplayFunc |

# *Color Models: RGB*

- Used in display screen. Pixels emit three kinds of light: Red, Green and Blue.
- We choose Red, Green and Blue to be our primary colors.
- ***RGB** and **RGBA modes***
  - ***"A" stands for alpha***, refers to transparency.
  - Alpha = 1.0 : Fully opaque
  - Alpha = 0.0 : Fully transparent
  - ***In RGB mode, alpha is assumed to be 1.0.***
- In OpenGL, color is specified in RGB.

  ***glColor3f(r, g, b);***
- Where r, g and b are floating point numbers between 0.0 and 1.0.

# Color Model

**rgb(*red, green, blue*)**

- Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
- For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.
- To display black, set all color parameters to 0, like this: rgb (0, 0, 0).
- To display white, set all color parameters to 255, like this: rgb (255, 255, 255).
- **rgba(*red, green, blue, alpha*)**
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):
- Experiment by mixing the RGBA values below:
- **Rgba (255, 99, 71, 0.5)**

# *Color Models: RGB*

- **glColor3f( red value, green value, blue value) ;**
  - Used to specify the wanted color
  - Has three float parameter;
- **glClearColor( red value, green value, blue value, alpha value) ;**
  - Used to specify the initial background color.
  - Has four float parameters
  - Alpha value is used to determine the color of two overlapped objects
- **glClear ( GL_COLOR_BUFFER_BIT);**
  - Used to set the bit value in the color buffer (refresh buffer) to the color indicated in the glClearColor function.

# *Color Models: RGB*

## *Example*

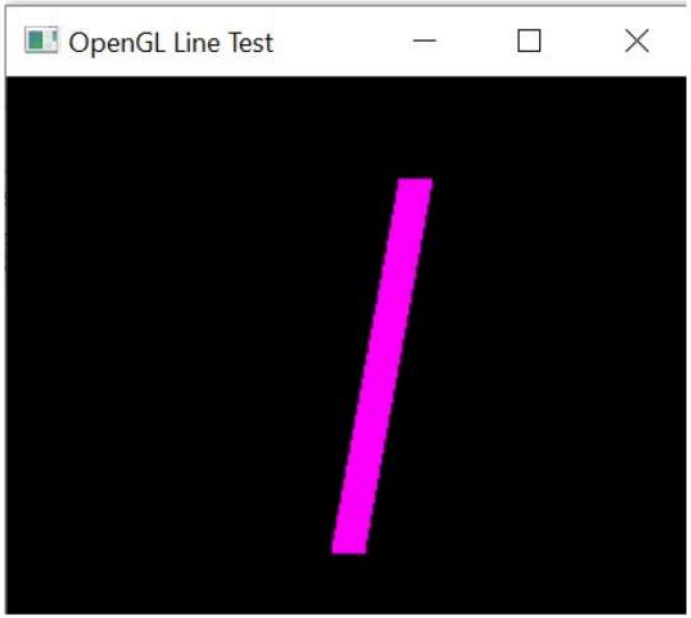
- ***glColor3f(0.5,1.0,0.9);***
  - ✓This tells display to emit 0.5 intensity red light together with 0.1 intensity green light together with 0.9 intensity blue light.
  - ✓ RGB mode, fully opaque
- ***glColor4f(0.5,1.0,0.6,0.3);*** // RGBA mode, alpha set to 0.3

| Color Component | | | Color Common Name |
|---|---|---|---|
| R | G | B | |
| 0 | 0 | 0 | Black |
| 0 | 0 | 1 | Blue |
| 0 | 1 | 0 | Green |
| 0 | 1 | 1 | Cyan |
| 1 | 0 | 0 | Red |
| 1 | 0 | 1 | Magenta |
| 1 | 1 | 0 | Yellow |
| 1 | 1 | 1 | White |

# *OpenGL display function to draw Line*

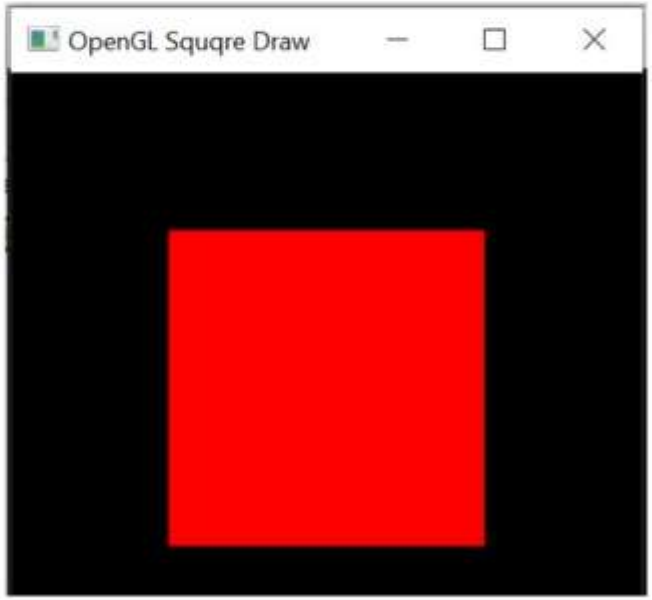
```
void display() {
   // Set background color to black and opaque
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
   // Clear the color buffer (background)
   glClear(GL_COLOR_BUFFER_BIT);

    glLineWidth(15);
    glBegin(GL_LINES);
      glColor3f(1.0,0.0,1.0);
      glVertex2f(0.2,0.7);
      glVertex2f(0.0,-0.4);
    glEnd();
    glFlush();  // Render now
}
```



# OpenGL display function to draw Square

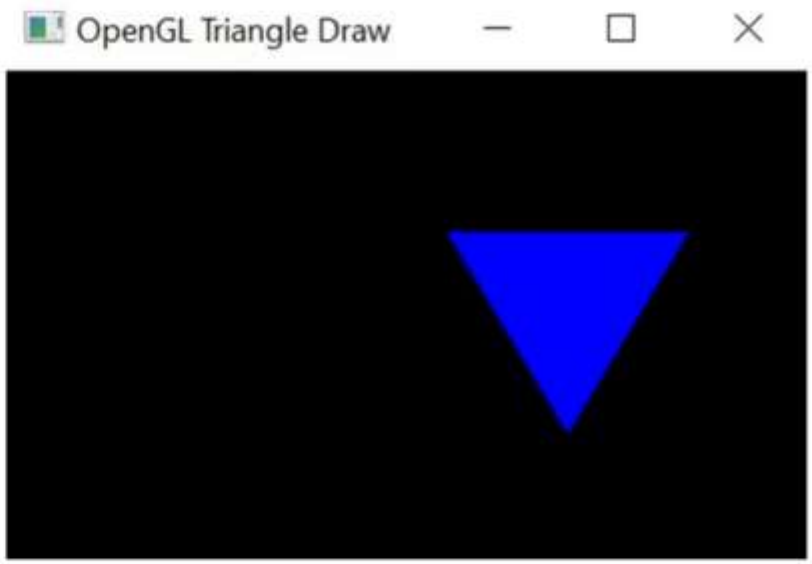
```
void display() {
   // Set background color to black and opaque
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
   // Clear the color buffer (background)
   glClear(GL_COLOR_BUFFER_BIT);
   // Draw a Red 1x1 Square centered at origin
   glBegin(GL_QUADS); // Each set of 4 vertices form a quad
      glColor3f(1.0f, 0.0f, 0.0f); // Red
      glVertex2f(-0.5f, -0.5f);    // x, y
      glVertex2f( 0.5f, -0.5f);
      glVertex2f( 0.5f,  0.5f);
      glVertex2f(-0.5f,  0.5f);
   glEnd();
  glFlush();  // Render now
}
```



# *OpenGL display function to draw Triangle*

```
void display() {
   // Set background color to black and opaque
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
   // Clear the color buffer (background)
   glClear(GL_COLOR_BUFFER_BIT);

   glBegin(GL_TRIANGLES);
      glColor3f(0.0f, 0.0f, 1.0f); // Blue
      glVertex2f(0.1f, 0.6f);
      glVertex2f(0.7f, 0.6f);
      glVertex2f(0.4f, 0.1f);
   glEnd();
  glFlush();  // Render now
}
```



# OpenGL display function to draw Polygon

```
void display() {
   // Set background color to black and opaque
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
   // Clear the color buffer (background)
   glClear(GL_COLOR_BUFFER_BIT);
    // These vertices form a closed polygon
    glBegin(GL_POLYGON);
      glColor3f(1.0f, 1.0f, 0.0f); // Yellow
      glVertex2f(0.4f, 0.2f);
      glVertex2f(0.6f, 0.2f);
      glVertex2f(0.7f, 0.4f);
      glVertex2f(0.6f, 0.6f);
      glVertex2f(0.4f, 0.6f);
      glVertex2f(0.3f, 0.4f);
   glEnd();
   glFlush();  // Render now
}
```