CSE 333/533: Computer Graphics
Lab 0: Introduction to OpenGL and ImGUI

Instructor: Ojaswa Sharma , TAs: Vishwesh Vhavle , Aadit Kant Jha

Due date: 23:59, 11 August 2023

Introduction

OpenGL or Open-source Graphics Library was first developed by Silicon Graphics Inc. in the early 1990s has become the widely used standard for graphics programming.

ImGui or Immediate Mode Graphical User Interface is a lightweight C++ library that provides a simple way for developers to create user interfaces (UIs) for their applications.

Goal of today's lab will be to get things running on your system and to get you familiar with OpenGL and ImGui as you will need them throughout the course. Feel free to come back and refer to this document as needed later in the course.

Note: This lab is ungraded.

Installation for Linux

```
sudo apt install build-essential
sudo apt install g++
sudo apt install mesa-common-dev
sudo apt install mesa-utils
sudo apt install make cmake cmake-curses-gui
sudo apt install libglfw3 libglfw3-dev libglm-dev
sudo apt install libglew-dev
```

How to Run

Step 1 - Download the attached zip file and extract it.
Step 2 - Go into the Lab0 directory and run `cmake CMakeLists.txt`
Step 3 - Next, run `make`. You should now see an executable.
Step 4 - Run `./lab0`

ImGUI Fundamentals

Widgets are some components that allow user interaction and form the user interface's key component. Different widgets can be tied to different **events** which act basically as triggers for performing certain **actions**. ImGUI allows quick creation and interaction with windows and widgets.

Basic API Usage Steps

1. **Setup ImGui Context:** The context contains and maintains all the outline information pertaining to a window. It provides functionality such as access to I/O, theme, window construction etc. It is created with the `CreateContext` function.

2. **Get IO functionality:** `ImGuiIO` provides the main configuration and I/O between your application and ImGui.

3. **Initialize and Setup Bindings for OpenGL:** `ImGui_ImplGlfw_InitForOpenGL` performs the internal bindings and key mappings required for user interaction with GLFW which would be used for drawing. `ImGui_ImplOpenGL3_Init()` checks and binds OpenGL with ImGUI IO.

4. **Create a Frame:** `ImGui_ImplOpenGL3_NewFrame()` Calls `ImGui_ImplOpenGL3_CreateDeviceObjects()` which creates device objects for openGL.

5. **Define Windows/Widgets:** `ImGui::Begin()` and `ImGui::End()` (marks the beginning and end of window creation.)

6. **Rendering:** `ImGui::Render()` starts the rendering. `ImGui_ImplOpenGL3_RenderDrawData` starts the OpenGL rendering.

7. **Shutdown:** `ImGui_ImplOpenGL3` Shutdown shuts down OpenGL rendering. `ImGui_ImplGlfw_Shutdown` shuts down communication with GLFW. `ImGui::DestroyContext()` destroys the current context.

ImGUI Widgets

ImGui Provides several widgets for creating beautiful user interfaces. There are different elements that can be used. Some of the commonly used widgets are as follows -

- `ImGui::Text` Used for creating a text box for displaying text.
- `ImGui::CheckBox` Used for creating a checkbox
- `ImGui::SliderFloat` Creating a slider window
- `ImGui::Button` Used for Creating a button
- `ImGui::RadioButton` Used for Creating a radio button
- `ImGui::ProgressBar` Used for creating a progress bar
- `ImGui::InputText` USed for creating input text box

Managing Widgets

ImGui makes it really simple to communicate with widgets. When a widget is interacted with it triggers an event which is then captured by the event dispatcher and returns a boolean upon the trigger.

Example: To capture button even all you need to do is wrap around an `if.. else` like

```
if(ImGui::Button("Click")){
      // if clicked do something
}
```

For interacting with value based widgets such as text inputs, slider widgets, etc. take a variable to bind a particular value with which gets updated as the user interacts with the widgets.

Example:

```
ImGui::SliderFloat("float", &slider_value, min, max);
//any change is recorded by the value in variable slider_value
```

GLFW

GLFW (Graphics Library Framework) is an open-source library for creating and managing windows, handling input, and managing OpenGL contexts for graphics rendering. It provides a

simple and cross-platform interface for tasks like creating windows, managing user input, and setting up OpenGL contexts for graphics rendering, much like ImGui.

Creating Window

GLFW provides a medium to create a window and bind it to OpenGL for rendering. In order to create a window using GLFW we use the following snippet.

```
GLFWwindow* window = glfwCreateWindow(window_width, window_height, "Window Title",NULL, NULL);
```

Next, make the context of the specified window current for the calling thread by calling `glfwMakeContextCurrent(window)`.

Polling

Once the window is created you'd want to poll or run continuously to capture I/O events and perform the specific rendering changes. This is done till window is closed.

```
while (!glfwWindowShouldClose(window)){
      // do rendering and handle I/O
}
```

Key Functions

- `glfwGetFramebufferSize()` Retrieve the size, in pixels, of the framebuffer of the specified window. Framebuffer is a collection of buffers that is used as the destination for rendering.
- `glViewport()` Set the viewport, set the x,y coordinates and width height of the viewport.
- `glClearColor()` Specify clear values for the color buffers.
- `glClear()` Set the bitplane area of the window to values previously selected by `glClearColor`.
- `glfwSwapBuffers()` Swap the front and back buffers of the specified window.
- `glfwDestroyWindow()` Destroy the specified window and its context.
- `glfwTerminate()` Destroy all remaining windows.

Deliverables

None. Off you go.

References

https://www.opengl.org/documentation/
https://www.khronos.org/opengl/wiki/Rendering Pipeline Overview
https://www.khronos.org/registry/OpenGL-Refpages
https://www.glfw.org/documentation.html

Note: Your code should be written by you and be easy to read. You are NOT permitted to use any code that is not written by you. (Any code provided by the instructor/TA can be used with proper credits within your program). Theory questions need to be answered by you and not copied from other sources. Please refer to IIIT-Delhi's Policy on Academic Integrity here.

Installation for macOS

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
xcode-select --install
brew install make cmake
brew install glfw
brew install glew
```

Installation for Windows 11

You must first install **WSL2** and **Ubuntu 22.04** from Microsoft App Store. Then you can proceed with the same installation process as Linux.

Note: It will only work on WSL2, not WSL as only WSL2 runs on a Linux Kernel. If you have WSL, please update it to WSL2. You can use this link for reference.
