ARLAB

ME/CprE/ComS 557

Computer Graphics and Geometric Modeling

GLFW Windows

September 8, 2013

Rafael Radkowski

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

You should know by now

ARLAB

- Create and set up a Visual Studio project (manual and with cmake)
- Compile shader code.

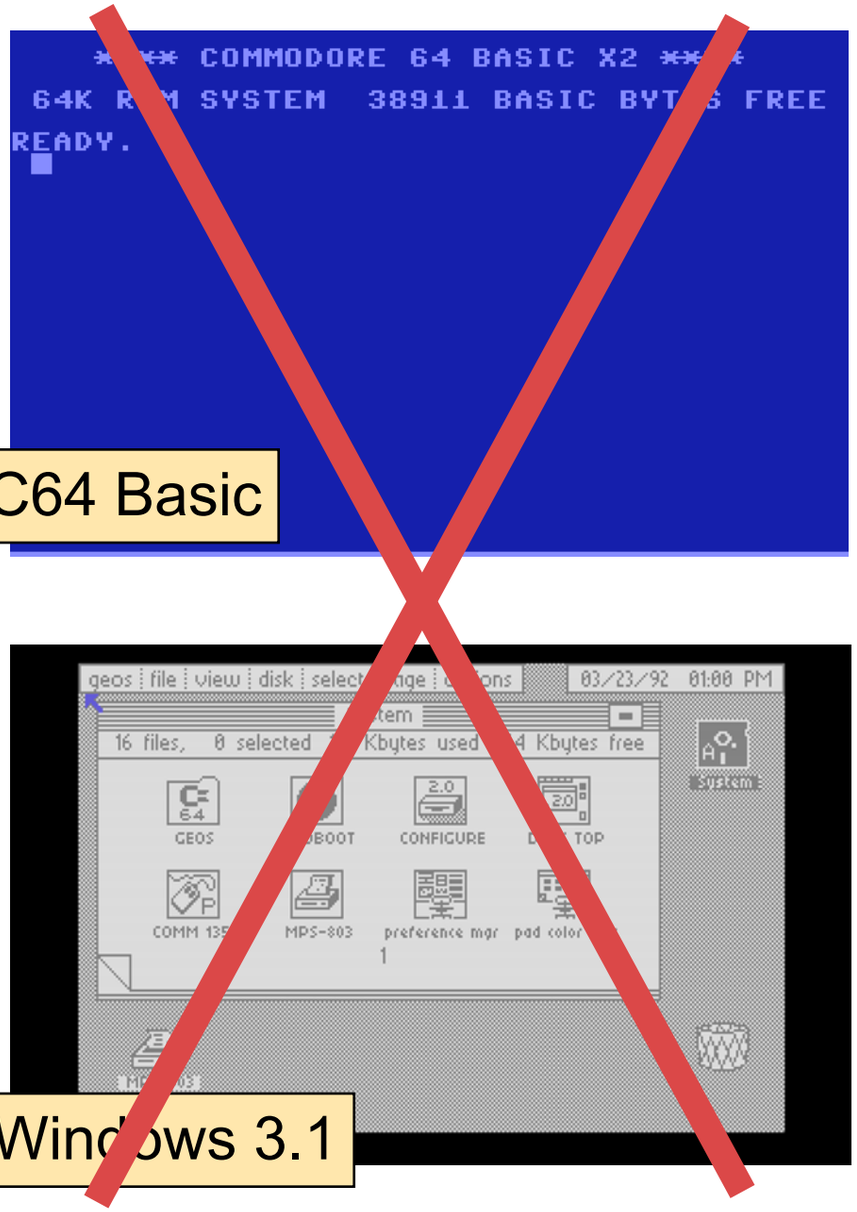
Topics

- GLFW Windows Initialization
- Rendering Loop

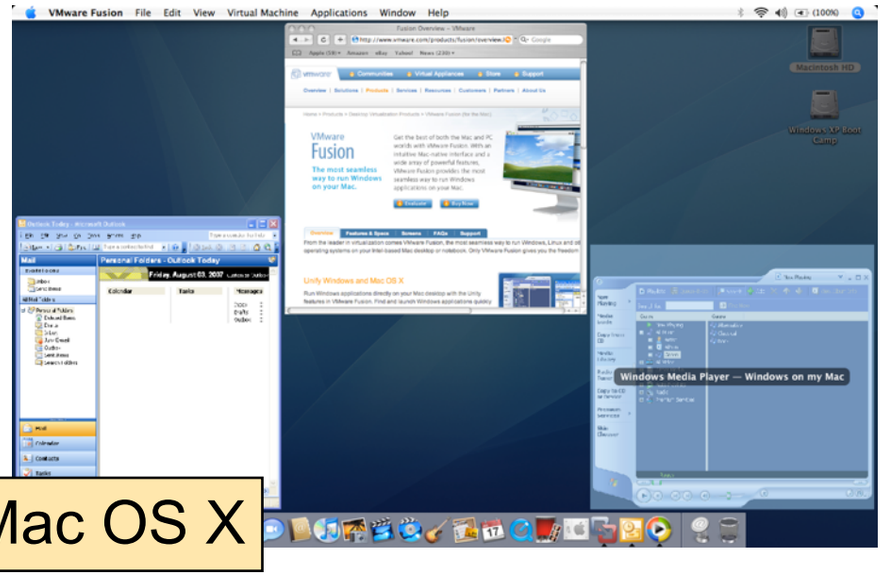
GLFW

ARLAB

GLFW is a library that creates OpenGL window context on different platforms such as MS Windows, and Apple Mac OS X with no effort. It can also manage keyboard/ mouse events.

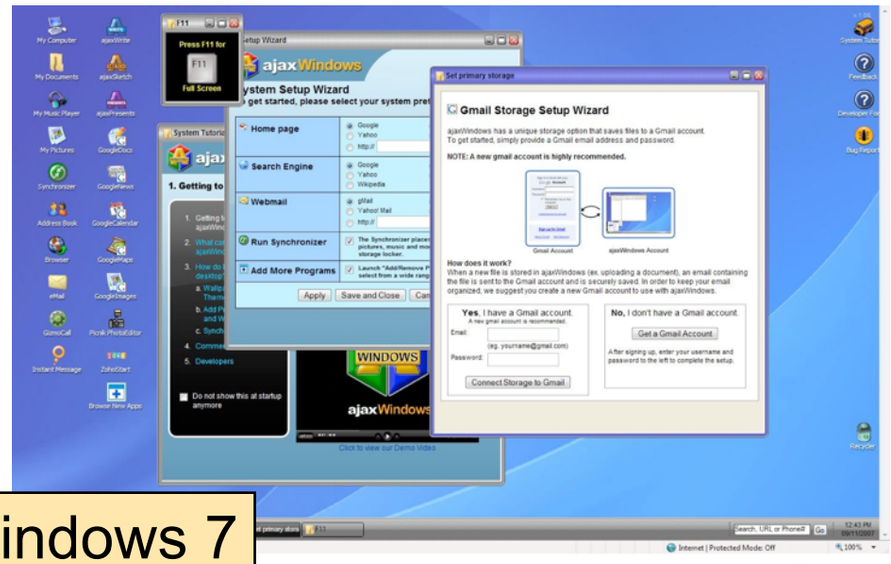


C64 Basic



Mac OS X

Windows 3.1



Windows 7

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

ARLAB

Ignore Functions you do not understand

It will take two-three weeks until you understand a complete program

Simple Window

ARLAB

Example 02_Glwf_Window available on git

Program Structure

ARLAB

Header

```
int main(int argc, const char * argv[])
{
```

Program init

[….]

Init GLFW and create a window

[….]

main loop

```
while
```

Clear the window

[….]

Swap buffers

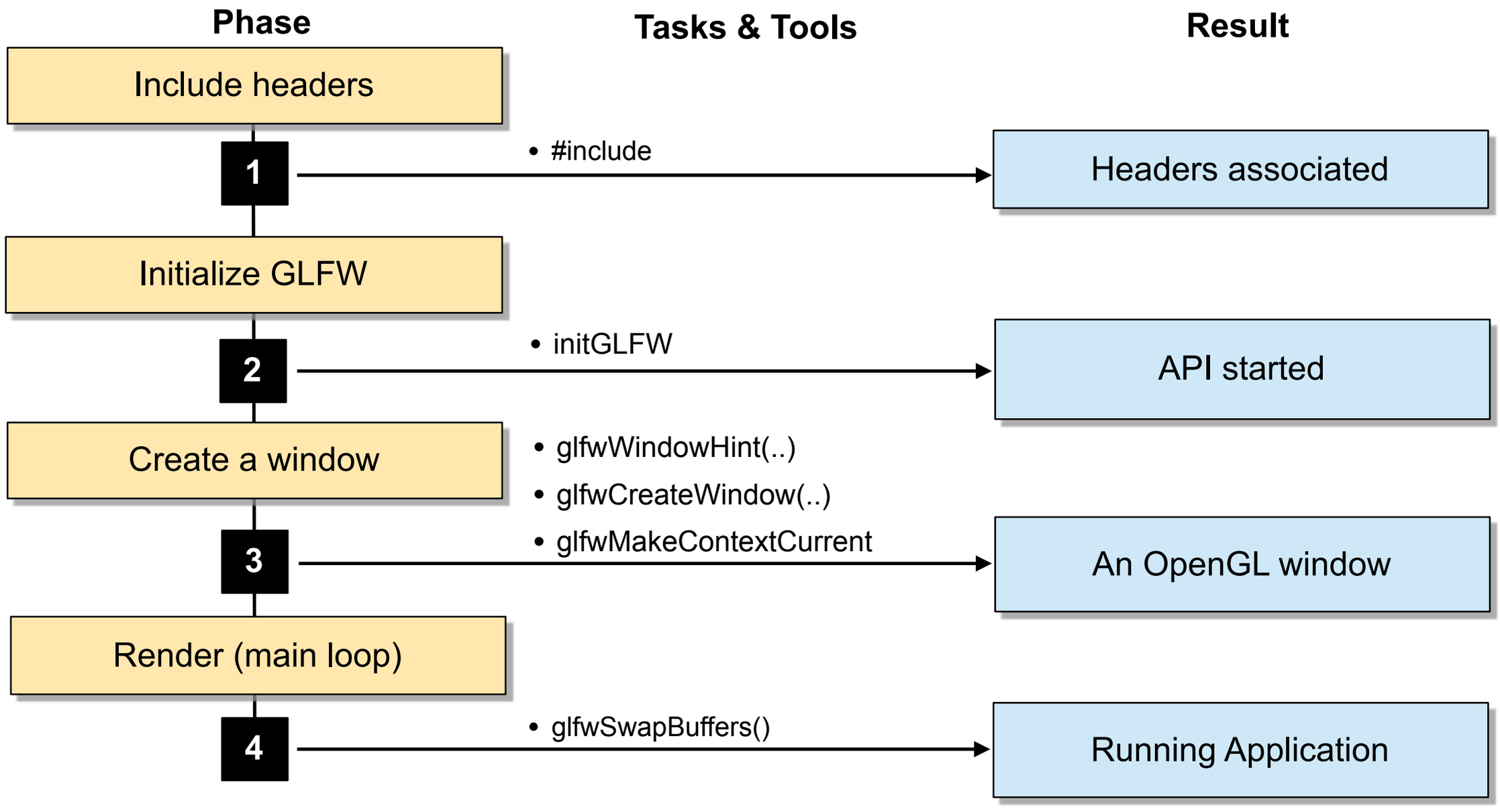
[….]

```
}
```

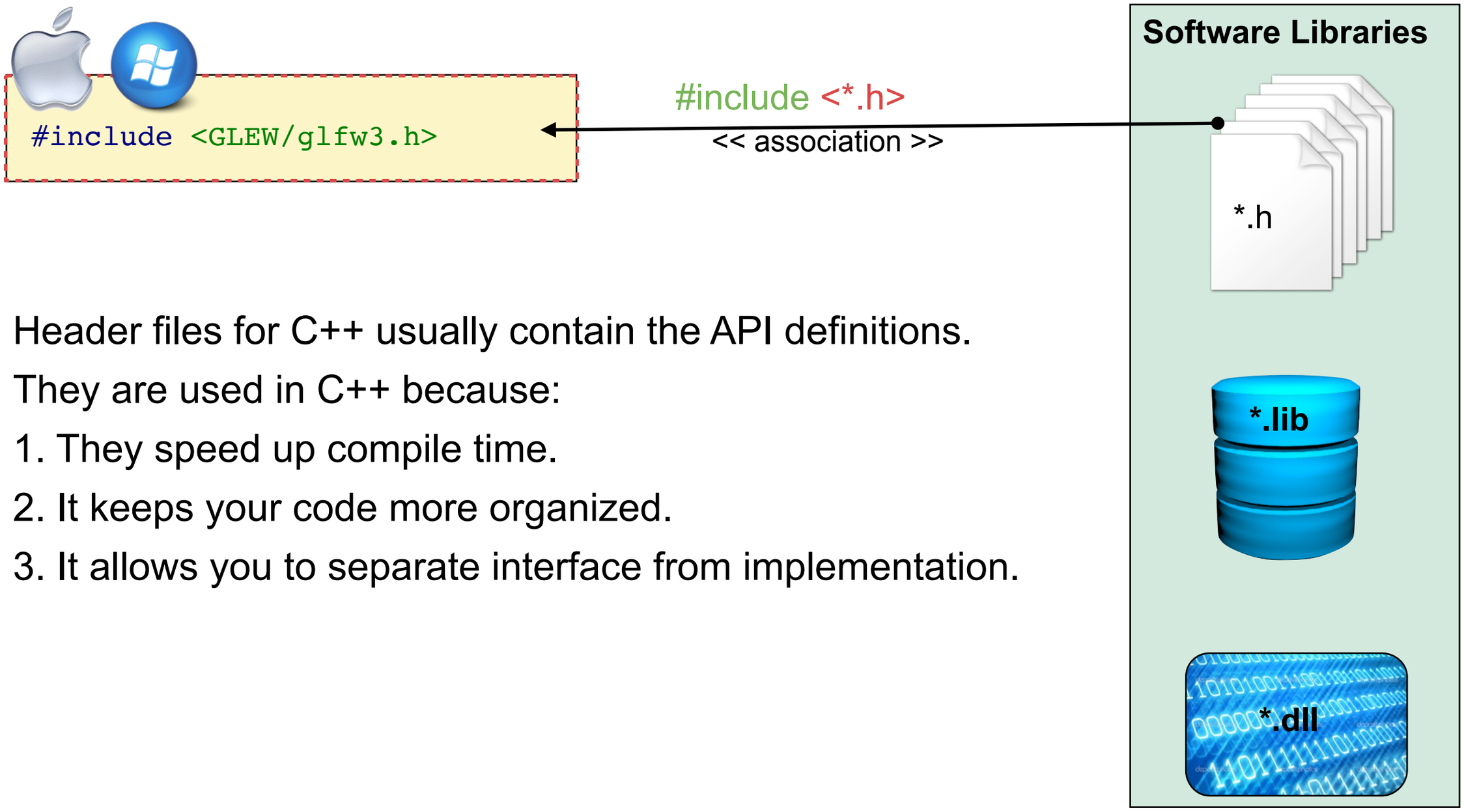
Process

This diagram shows the steps to initialize a GLFW and to create a window.

It is always the same process. Just follow the steps.



Header for OpenGL and GLUT



i.e., GLUT and OpenGL

Header files for C++ usually contain the API definitions.

They are used in C++ because:

1. They speed up compile time.
2. It keeps your code more organized.
3. It allows you to separate interface from implementation.

You will sometimes see those whenever we need to change the process for Mac and Windows

Init the API

ARLAB

```
int glfwInit (void)
```

This function initializes the GLFW library. Before most GLFW functions can be used, GLFW must be initialized, and before an application terminates GLFW should be terminated in order to free any resources allocated during or after initialization.

Example:

```
// Initialize GLFW
if(!glfwInit() )
{
        cout <<  "Failed to initialize GLFW\n" << endl;
        system("pause");
        return -1;
}
```

Prepare the Window

ARLAB

```
void glfwWindowHint (int target,int hint )
```

The function allows us to set window parameters which change the properties of the window we create.

Note, all those properties must be set before you create the window.

Parameters:

- target - The window hint to set.
- hint - The new value of the window hint.

Example:

```
// Set the openGL version which we expect, here, 3.3
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
```

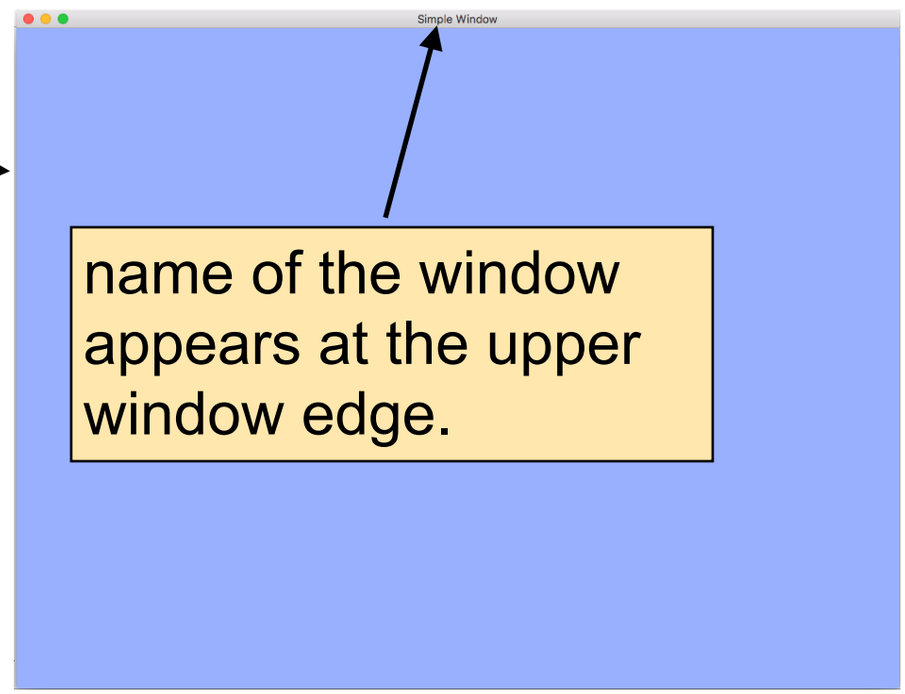
http://www.glfw.org/docs/latest/window.html#window_hints

The Window

The window object

```
GLFWwindow* window;
```

```
GLFWwindow* glfwCreateWindow (
int width, int height,
const char *title,
GLFWmonitor *monitor,
GLFWwindow * share )
```



This function creates a window and its associated OpenGL context.

Parameters:

- width - The desired width, in screen coordinates, of the window. This must be greater than zero.
- height - The desired height, in screen coordinates, of the window. This must be greater than zero.
- title - The initial, UTF-8 encoded window title.
- monitor - The monitor to use for full screen mode, or NULL to use windowed mode.
- share - The window whose context to share resources with, or NULL to not share resources.

Window Example

ARLAB

```
GLFWwindow* window;

    [....]


/ Open a window and create its OpenGL context
window = glfwCreateWindow( 1024, 768, "Simple Window", NULL, NULL);
if( window == NULL ){
  cout <<  "Failed to open GLFW window. If you have an Intel GPU, they
  are not 3.3 compatible." << endl;
  glfwTerminate();
  system("pause");
  return -1;
}
```

Terminating a Window

ARLAB

```
void glfwTerminate(void )
```

This function destroys all remaining windows and cursors.

Call it at the end of your program, when you want to quit your program OR when and error occurs somewhere.

Make Current Context

ARLAB

```
void glfwMakeContextCurrent(GLFWwindow * window)
```

This function makes the OpenGL context of the specified window current.

We have to tell our function in which window we want to render our graphics content.

This happens here!

The term **context** refers to specific resources that the graphics program allocates and that you want to switch as active resources. Literally, the context are the resources you work with.

Program Structure

ARLAB

Header

```
int main(int argc, const char * argv[])
{
```

Program init

[….]

Init GLFW and create a window

[….]

main loop

```
while
```

Clear the window

[….]

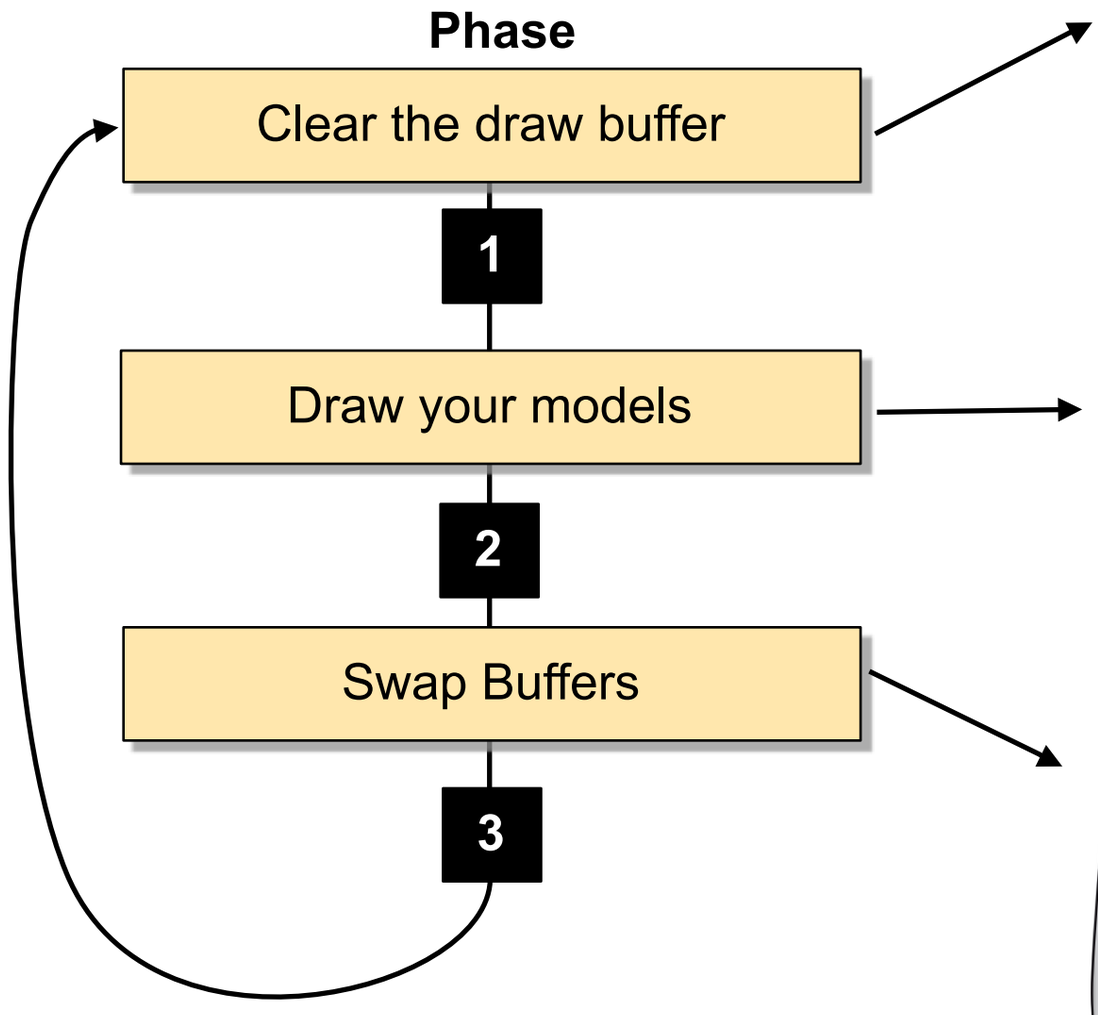
Swap buffers

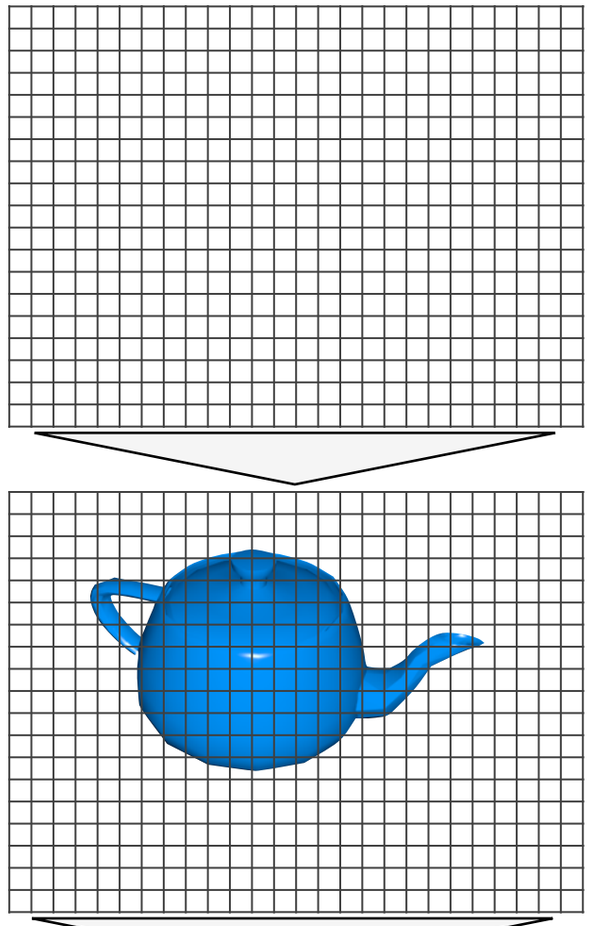
[….]

```
}
```

OpenGL Draw Loop

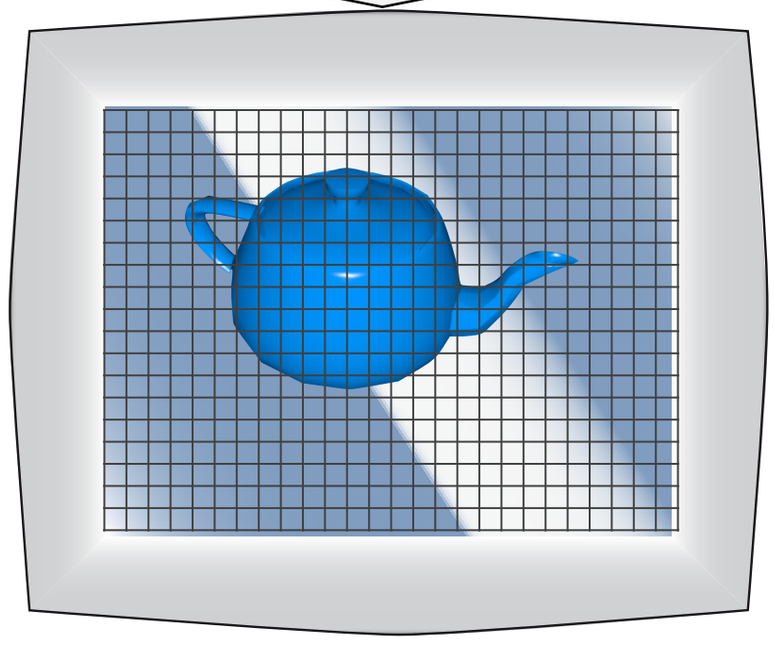
ARLAB





Empty draw buffer

3D model in draw buffer



draw buffer is shown on screen.

Example

ARLAB

```
while( glfwGetKey(window, GLFW_KEY_ESCAPE ) != GLFW_PRESS &&
       glfwWindowShouldClose(window) == 0 )
{
```

Clear the window

```
// Clear the entire buffer with our blue color (sets the
// background to be blue).
glClearBufferfv(GL_COLOR , 0, clear_color);
glClearBufferfv(GL_DEPTH , 0, clear_depth);
```

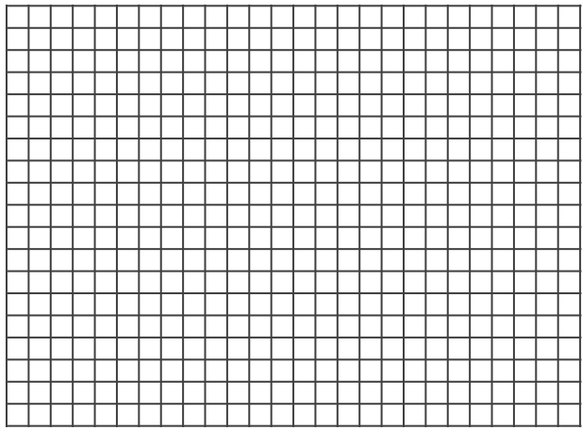
```
    // [this is the place where we need to draw some objects]
```

Swap buffers

```
// Swap buffers
glfwSwapBuffers(window);
glfwPollEvents();
```

```
} // Check if the ESC key was pressed or the window was closed
```

Clear Buffers



Empty draw buffer

We have to clear our buffer = computer memory.

Computer memory cannot be "empty" there are always values inside.

Empty is just our interpretation of the stored values.

To empty the draw buffer, we overwrite it with values that we consider as "empty", called, the **clear color.**

```
void glClearBufferfv( GLenum buffer, GLint drawbuffer, const GLfloat *value);
```

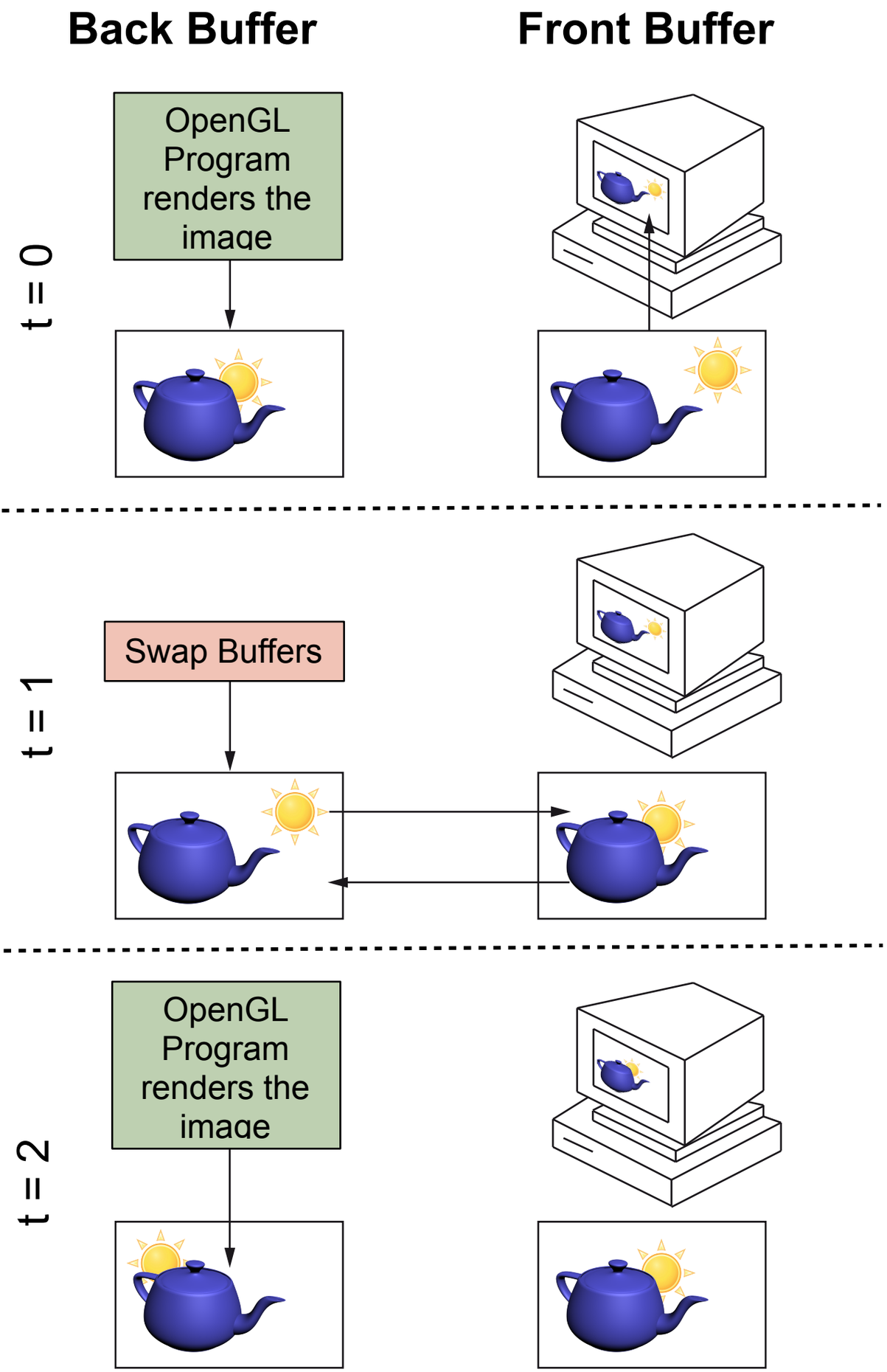
Parameters:

- buffer - Specify the buffer to clear.
  - GL_COLOR: the color buffer
  - GL_DEPTH: the depth buffer
- drawbuffer - Specify a particular draw buffer to clear.
- value - A pointer to the value or values to clear the buffer to.

Clear Buffer Example

ARLAB

```
// Set up our blue background color
static const GLfloat clear_color[] = { 0.6f, 0.7f, 1.0f, 1.0f };
static const GLfloat clear_depth[] = { 1.0f, 1.0f, 1.0f, 1.0f };

glClearBufferfv(GL_COLOR , 0, clear_color);
glClearBufferfv(GL_DEPTH , 0, clear_depth);
```



Swap Buffers

We distinguish between a front buffer and a back buffer (double buffer)

- Front Buffer is always visible on screen.
- The Back Buffer is the render target for the current render loop.

Swap Buffers

ARLAB

`void` **`glfwSwapBuffers`**`(GLFWwindow *window)`

This function swaps the front and back buffers of the specified window.

Parameters:

- window - The window whose buffers to swap.

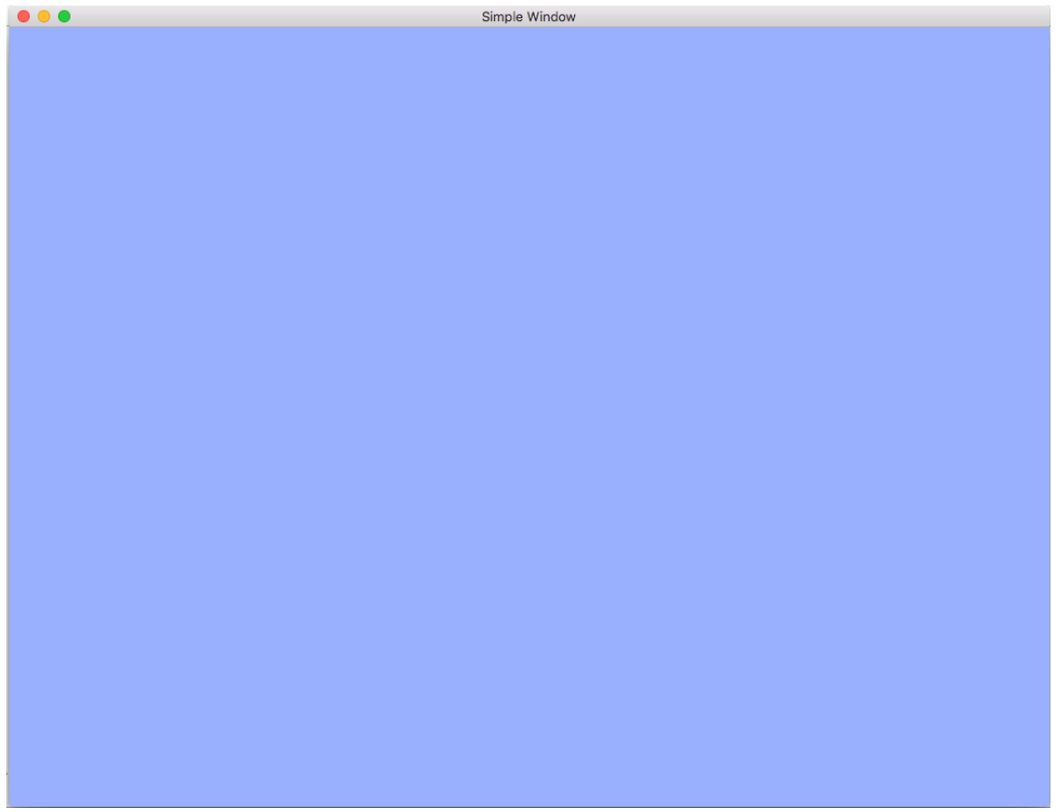
Example Code

ARLAB

02_Glwf_Window available on git

main.cpp

```cpp
#include <iostream>
#include <string>

// GLEW include
#include <GL/glew.h>

// GLM include files
#define GLM_FORCE_INLINE
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>

// GLFW include
#include <GLFW/glfw3.h>


using namespace std;
using namespace glm;



// The main window
GLFWwindow* window;


int main( void )
{
    // Initialise GLFW
    if( !glfwInit() )
    {
        cout <<  "Failed to initialize GLFW\n" << endl;
        system("pause");
        return -1;
    }


    // Set the version which we expect, here, 3.3
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);


    // Open a window and create its OpenGL context
    window = glfwCreateWindow( 1024, 768, "Simple Window", NULL, NULL);
    if( window == NULL ){
        cout <<  "Failed to open GLFW window. If you have an Intel GPU, they are not 3.3 compatible." << endl;
        glfwTerminate();
        system("pause");
        return -1;
    }

    // Switch the window as the object that we work with.
    glfwMakeContextCurrent(window);


    // Initialize GLEW
    if (glewInit() != GLEW_OK) {
        cout <<  "Failed to initialize GLEW\n" << endl;
        system("pause");
        return -1;
    }

    // Ensure that we can notice the escape key being pressed
    glfwSetInputMode(window, GLFW_STICKY_KEYS, GL_TRUE);
```



Ignore the functions you do not understand. We did not talk about all of them today!