# ··Windowing system

Using GLFW to make window abstraction (cross-platform support)

cocoa

MacOS

win32

Windows

Linux

X11,posix

Create a separate window class which handles window initialization and gets passed around all the other classes which need the window handle.

Some method like EventCallback

Update width and height

... (interface)

class Window

{

MouseEventCallback()

KeyBoardEventCallback()

…

UpdateWidth()

UpdateHeight()

...

//update status glfw func

Update()

InitWindow()

CloseWindow()

//maybe a struct

struct WindowData

{

title

width

height

...

}

}

# Event System

Dealing with window events

like move camera, change window size, or other complex game events such as gain resource, build, fire, .etc…

## Benefit:

If you have a simple game you could probably check a simple list of 20 things each frame refresh. If a condition is met then you run some code and repeat it.

But what if your game becomes more complicated and there are thousands of game objects that may change state at any time? Are you going to be able to check the position, orientation, energy, activation and who knows what other states for each of the thousand game objects every frame? And what about combinations of state changes? Can you handle all the special cases?

Events are the answer. Instead of checking all object states every frame you only check for new events. Events are triggered by hardware (timers or user input) and also by other events.

//event listener interface

Event System

Game itself

Rendering stuff

Windowing System