

# report

*by* Luke Webb

---

|                |                                                      |                 |      |
|----------------|------------------------------------------------------|-----------------|------|
| FILE           | 12132_LUKE_WEBB_REPORT_254066_191378842.ODT (80.53K) |                 |      |
| TIME SUBMITTED | 11-MAY-2020 04:58PM (UTC+0100)                       | WORD COUNT      | 776  |
| SUBMISSION ID  | 127100575                                            | CHARACTER COUNT | 3649 |

```
const int MAX_GAMEOBS = 5;
GameObj gameObjList[MAX_GAMEOBS];
```

On these lines I create a const int for the max size elements in the array below where I am storing 5 "Game Objects"

```
int MS_FRAME_TIME = 2000; // 2000ms
float last = GetTickCount(); // get current tick
float timePassed; // created to calculate timePassed
```

On these lines I create an int with the value of 2000 that's going to act as the time I want to pass before the game loop need to pass before it can update my "Game Objects" from the array above.

I also make a variable for tracking the last time a frame started and another to track the time passed in total. Using both variables I'll be able to keep track of the time since the loop started and know when to run the update loop.

```
while (!myDisplay.isClosed()) // Application loop
{
    static int fps = 0; //frame count;
    timePassed = GetTickCount() - last; // calculate delta time
    fps++; // every time the loops passes increase frame counter
```

In this screenshot the application loop is started using a while loop to check that the window is open and then on the line below I create a static int called fps that will be used to track how many frames have passed, it is a static int so that it doesn't get redefined every time the loop loops.

After that the variable "TimePassed" gets set to the current time tick minus the last so we know how long its been since the last loop and we increment the fps counter by one.

```
if (timePassed >= MS_FRAME_TIME) { // if delta time is same as or larger then time we want to pass before an update or in this case (MS_FRAME_TIME)
    std::cout << "frames per sec = " << (float)fps / 5.0 << std::endl; //print fps
    for (size_t i = 0; i < MAX_GAMEOBS; i++) { //loop game objects and take in the time it took.
        std::cout << "GameObj " << i << ": "; gameObjList[i].update(timePassed);
    }
    fps = 0; //reset counter
    last = GetTickCount(); //get new tick time
}
```

In the screenshot above starts with an if statement where if the condition of the "timePassed" is the same or larger than the "MS\_FRAME\_TIME" then the rest of the statement will execute, first the program will print out the frame rate of the program followed by the next line where you can see a for loop where each of the elements in the gameObjects array will get its "update" function will be called taking in the "timePassed" variable so it can print the time its been since it was last updated.

After all the elements of the array have been looped the next two lines reset the fps counter to 0 and then after that the last variable get set to the current time.

```
myDisplay.update(); //checks mouse state and displays this to screen
myDisplay.render(pmesh); // renders loaded mesh
```

Then after the program exits the if statement the display/window gets updated and gets told to render the mesh.

Using the code I have implemented the program can be told how long it should take before it can update game objects using variables this can be changed to accommodate a number of factors.

### Testing Plan

| Tested               | Test Description                                              | Expected Result                                                                  | Actual Result                                      | Test Number |
|----------------------|---------------------------------------------------------------|----------------------------------------------------------------------------------|----------------------------------------------------|-------------|
| FPS variable         | Testing that the variable increments each loop                | FPS should increase +1 each loop                                                 | FPS does increase by 1 each loop                   | 1           |
| timePassed           | TimePassed is accurately calculated                           | timePassed should reach a value of 2000 after 2 seconds                          | timePassed hits 2000 and triggers the if statement | 2           |
| Game Object For loop | Every element in Game Object for loop should be accounted for | For the size of the array all elements should be called                          | Game object array gets every member called         | 3           |
| FPS                  | Fps should be reset when exiting if statement                 | Fps should be set to 0                                                           | Fps was set to 0                                   | 4           |
| Last                 | Last variable should be set to current tick                   | Last should get set to current tick allowing loop to trigger at the correct time | Last was set to current time                       | 5           |

**Code:**

**MAIN.CPP**

```
#include "mesh.h"

#include "display.h"

#include "GameObj.h"

//

#include "glut.h"

#include <Windows.h>

#include <iostream>

int main(int argc, char *args[])

{

    Display myDisplay(800, 800, "OGL Window"); //create and open a window

    Mesh* pmesh = new Mesh("house.obj"); // load obj file

    const int MAX_GAMEOBS = 5; // Max array size

    GameObj gameObjList[MAX_GAMEOBS]; // array of "Game Objects"

    int MS_FRAME_TIME = 2000; // 2000ms

    float last = GetTickCount(); // get current tick

    float timePassed; // created to calculate timePassed

    while (!myDisplay.isClosed()) // Application loop

    {

        static int fps = 0; //frame count;

        timePassed = GetTickCount() - last; // calculate delta time

        fps++; // every time the loops passes increase frame counter

        if (timePassed >= MS_FRAME_TIME) { // if delta time is same as or larger then time

            we want to pass before an update or in this case (MS_FRAME_TIME)
```

```

        std::cout << "frames per sec = " << (float)fps / 5.0 << std::endl; //print fps

        for (size_t i = 0; i < MAX_GAMEOBS; i++) { //loop game objects and take in
the time it took.
            std::cout << "[GameObj " << i << "]: ";
gameObjList[i].update(timePassed);

        }

        fps = 0; //reset counter
        last = GetTickCount(); //get new tick time

    }

    myDisplay.update(); //checks mouse state and displays this to screen
    myDisplay.render(pmesh); // renders loaded mesh

}

return 0;
}

```

#### **GAMEOBJ.H**

```

#pragma once
#include <iostream>

class GameObj
{
public:
    GameObj();
    ~GameObj();

    void update(float dt);
}

```

```
private:
```

```
};
```

### **GAMEOBJ.CPP**

```
#include "GameObj.h"
```

```
GameObj::GameObj()
```

```
{
```

```
}
```

```
GameObj::~~GameObj()
```

```
{
```

```
}
```

```
void GameObj::update(float dt)
```

```
{
```

```
    std::cout << "update time: [" << dt/1000 << "s]\n";
```

```
}
```

FINAL GRADE

GENERAL COMMENTS

Instructor

48/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

GRADING FORM: SMU GENERIC\_GJ

LUKE  
WEBB

48.25

RESEARCH-INFORMED LITERATURE



No references used here. Please use standard referencing that uses the school's standard (IEEE). Report very brief.

3.5

KNOWLEDGE AND UNDERSTANDING



A very brief report. A basic timer discussed. No screen shots of this running. Test schedule focused on each line of code. This is too fine grained. One test checking that the fps is 60fps under varying loads would be better here. Object updating attempted. Again, no screen shots provided. No obj loader in evidence.

**8.5**

## ANALYSIS



Basic integration of game loop and update outlined in the code presented in the report. Testing poor.

**8.5**

## PRACTICAL APPLICATION AND DEP.



No evidence in report of screen shots of working code. No obj loader. Update and game loop patterns attempted here. Code executes - please make sure you clean your build before submitting.

**26**

## SKILLS FOR PROFESSIONAL PRAC.



Report very brief and could be better structured. No refs.

**1.75**