

1800242 - GAM 1b Report

by Benjamin Kiddie

FILE	12005_BENJAMIN_KIDDIE_1800242_-_GAM_1B_REPORT_254066_955270404.DOCX (341.37K)		
TIME SUBMITTED	07-MAY-2020 11:10PM (UTC+0100)	WORD COUNT	6280
SUBMISSION ID	126692832	CHARACTER COUNT	44151

Games Architecture & Mechanics - Assignment 1b

Course: Games Development

Name: Ben Kiddie

Student Number: 1800242

Uploaded: 07 / 05 / 2020

Table of Contents

1. Introduction	1
2. Task 1 – Game Loop.....	2
2.1. Solution.....	3
2.2. Testing.....	4
3. Task 2 – Object Loader	7
3.1. Solution.....	8
3.2. Testing.....	9
4. Task 3 – Update Manager	11
4.1. Solution.....	12
4.2. Testing.....	13
5. Conclusion	14
6. References.....	15

1. Introduction

This report will discuss the approach the author devised to tackle the three requirements necessary for the program. Three sections focusing on each requirement respectively and a conclusion are discussed.

2. Task 1 – Game Loop

The requirement of this task is to create a game loop that aims for an initial rate of 60 fps, with variable rate where necessary. Output of details for load testing is also useful.

2.1. Solution

In the solution, a timing loop is implemented, where the time before processes happen (initial / old time) and the time at the beginning of the processes (current time) are stored. An elapsed time is calculated by $elapsed = current - old$, which is incremented onto a lag variable during each loop.

```
dwCurrentTime = GetTickCount(); // Set current time to calculate elapsed
dwTimeElapsed = dwCurrentTime - dwOldTime; // The difference between the beginning of the loop (or last FPS output) and now

if (dwTimeElapsed > 1000) { // FPS and elapsed time output after 1s has passed
    cout << "frames per sec = " << fps << endl;
    cout << "time elapsed (ms) = " << dwTimeElapsed << endl;
    fps = 0; // As an FPS output has occurred, reset FPS
    dwOldTime = dwCurrentTime; // Update the old time so that new elapsed time calculations can begin and reupdate once 1s has passed again
}

lag += dwTimeElapsed; // Provide the system with lag time, in which operations can be done in said time
```

The lag time provides the program with time to update objects in a loop, where we want to update as many times as possible at 60fps per 1000ms (16.67 ms per update). This is done by decrementing the lag time by ms per update each loop.

```
gameObjects.UpdateLoadedObjs(); // Updates each object loaded into the container class

while (lag >= MS_PER_UPDATE) // Update the display as many times as possible until there isn't enough lag time for more updates
{
    myDisplay.update();
    lag -= MS_PER_UPDATE;
}
```

Objects are updated and the display will update as many times as it can with lag time. Following this, the program renders the updated display. To provide a consistent near 60fps, the following was implemented:

```
for (i = 0; i < gameObjects.GetNumObjsLoaded(); i++) // Go through each obj loaded into the container class and render them
{
    myDisplay.render(gameObjects.GetObj(i));
}

fps++; // Increment FPS as a loop has passed
if (fps == 60) // If the FPS has reached 60, the program sleeps for the remainder of the allotted 1s or 1000ms, providing a consistent near 60 FPS
{
    Sleep(1000 - dwTimeElapsed);
}
```

Each game loop, the fps is incremented. Once it hits 60fps, the program will sleep however much time is left before the next console output.

2.2. Testing

Testing will follow this format henceforth:

Test Case	Test Description	Prerequisites	Expected Result	Actual Result	Status (pass, partial pass, fail)
-----------	------------------	---------------	-----------------	---------------	-----------------------------------

Test Case 1	The program outputs details at 1s intervals	N/A	At 1000 ticks, the program outputs fps and elapsed time	<pre>frames per sec = 60 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1015 frames per sec = 60 time elapsed (ms) = 1016 frames per sec = 60 time elapsed (ms) = 1016</pre> The program outputs details at next to 1s intervals (1 frame greater).	Pass
-------------	---	-----	---	---	------

Test Case 2	The program is expected to update at 60 fps initially, and allow for variable rates when necessary	N/A	The program will output 60 fps initially. Variable rates are allowed.	<pre>frames per sec = 60 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1015 frames per sec = 60 time elapsed (ms) = 1016 frames per sec = 60 time elapsed (ms) = 1016</pre> The program outputs consistently at a variable rate around 60 fps.	Partial Pass
-------------	--	-----	---	---	--------------

Test Case 3	The program is expected to output the elapsed time with each update	N/A	The program outputs elapsed time at each update.	<pre>frames per sec = 60 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 61 time elapsed (ms) = 1016 frames per sec = 61 time elapsed (ms) = 1015 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1016 frames per sec = 62 time elapsed (ms) = 1015 frames per sec = 60 time elapsed (ms) = 1016 frames per sec = 60 time elapsed (ms) = 1016</pre> The program outputs an elapsed time at each update.	Pass
-------------	---	-----	--	--	------

3. Task 2 – Object Loader

This task asks that an existing object loader provided for this assignment be integrated into the program.

3.1. Solution

For this solution, the author created a container class for game objects, named `cGameOBJS`, with the aim of this taking place as an object manager. This class has functions for loading, getting and updating objects in a vector of Mesh objects also held in the class.

```
class cGameOBJS // Game object container class
{
public:
    cGameOBJS();
    ~cGameOBJS();

    void LoadNewOBJ(const char *filename); // Pass in a filename in the format "objectname.obj", which adds this mesh object to the vector of loaded objects

    int GetNumOBJsLoaded(); // Gets the number of objects in the container class
    Mesh* GetOBJ(int index); // Gets an object at a specific position in the vector

    void UpdateOBJ(int index); // Updates the mesh details for an object at the position in the vector
    void UpdateLoadedOBJs(); // Used to update all objects at once contained in the class

private:
    vector<Mesh> mLoadedOBJs; // Contains the mesh details for each object loaded into the container class
};
```

The ability to load an object is as simple as calling the `LoadNewOBJ(filename)` function. This runs the object loading function added via `objloader.cpp`. The returned item is then stored in a Mesh object and pushed into the vector.

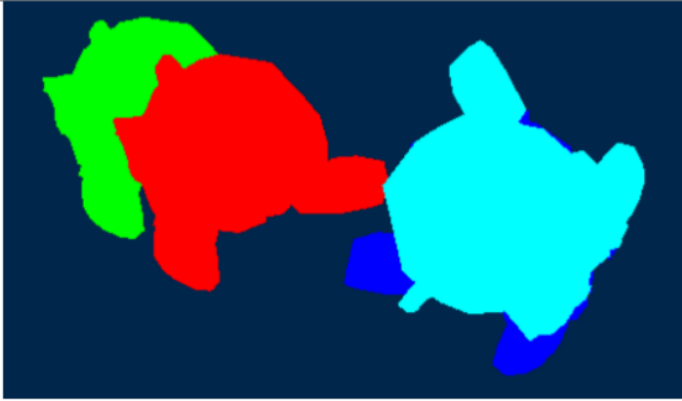
```
void cGameOBJS::LoadNewOBJ(const char *fileName) {
    mLoadedOBJs.push_back(Mesh(fileName));
}
```

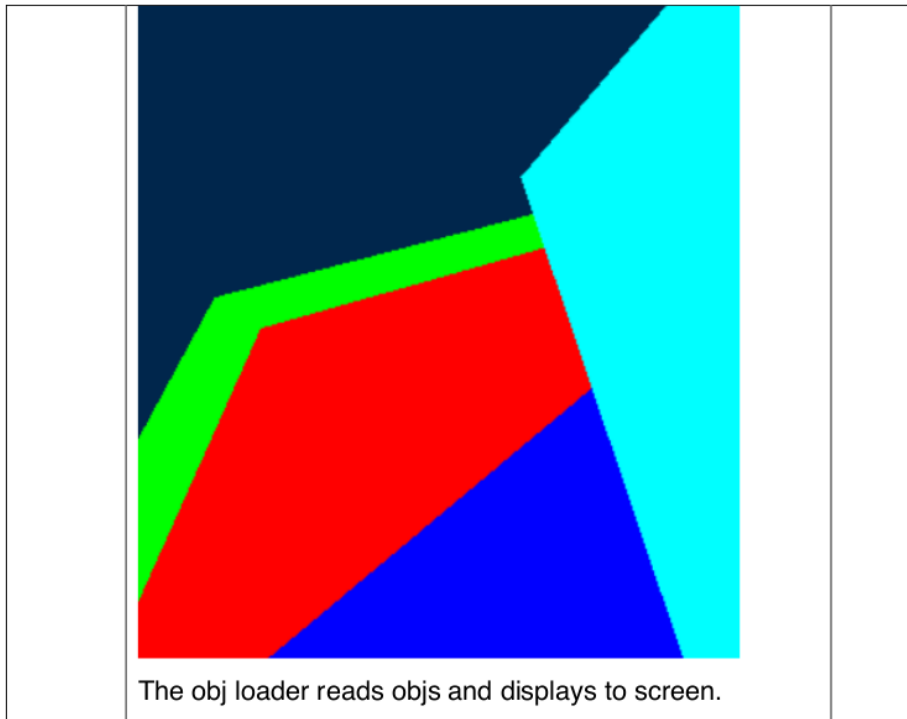
Additionally, the ability to get an object at a position in the container vector is present. This is useful for updating objects or rendering an object.

```
Mesh* cGameOBJS::GetOBJ(int index) {
    return &mLoadedOBJs[index];
}
```

3.2. Testing

Test Case 1	The object loader files are available to use in the solution.	objloader.h and objloader.c pp are included.	The functions of objloader are accessible in the main program.	<pre>class Mesh { public: Mesh(Vertex* vertices, u Mesh(const char *f); object_type myobj; void objPoly2Vertex(void void draw();</pre> Object loader utilizing type object_type is accessible in Mesh	Pass
-------------	---	--	--	--	------

Test Case 2	The integrated object loader can read object files.			N/A
The obj loader reads through the file and stores relevant data in a mesh object				Pass



4. Task 3 – Update Manager

This task asks that the ability to run an update manager in a container class be available for multiple objects.

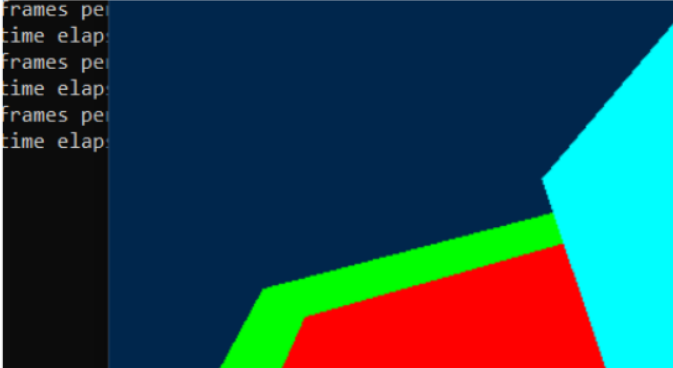
4.1. Solution

The container class `cGameOBJS` is used here. An update function implemented here can update an object by passing in an index for where in the vector it is stored. An update all function can also be used to update each object. Functionality for what the updates do is not included.

```
void cGameOBJS::UpdateOBJ(int index) {  
    float oldTime = GetTickCount(), currentTime = 0, elapsedTime = 0; // Can be used for time calculations  
  
    /*-----| Insert update code here |-----*/  
    //Sleep(1);  
  
    currentTime = GetTickCount();  
    elapsedTime = currentTime - oldTime;  
  
    //cout << "Game object updated in " << elapsedTime << "ms" << endl;  
}  
  
void cGameOBJS::UpdateLoadedOBJS() {  
    for (int i = 0; i < mLoadedOBJS.size(); i++) // Loops through each object and runs the update function for each item  
    {  
        UpdateOBJ(i);  
    }  
}
```

An object can be added to the loaded object vector by running `LoadNewOBJ(filename)`.

4.2. Testing

Test Case 1	Objects contained within the container class can be accessed.	Object files loaded into container class.
The obj files are accessible within the container class.	<pre>for (i = 0; i < gameObjects.GetNumOBJSLoaded(); i++) { myDisplay.render(gameObjects.GetOBJ(i)); }</pre>  Game objects are read through the GetOBJ function successfully.	Pass

Test Case 2	An update function will update each obj.	Objects are loaded into the container class.	<pre>void cGameOBJS::UpdateOBJ(int index) { float oldTime = GetTickCount(), currentTime = 0, elapsedTime = 0; /*----- Insert update code here -----*/ Sleep(1); currentTime = GetTickCount(); elapsedTime = currentTime - oldTime; cout << "Game object updated in " << elapsedTime << "ms" << endl; }</pre> <pre>frames per sec = 61 time elapsed (ms) = 1015 Game object updated in 0ms Game object updated in 0ms Game object updated in 0ms Game object updated in 16ms</pre>	Objects successfully run the update function.	Pass
-------------	--	--	--	---	------

5. Conclusion

All tasks were met with moderate completion. Task 1 was met by utilizing calculations of initial and current time, lag time and milliseconds per frame. The sleep pattern was chosen to be implemented as a means of getting regular 1s updates with close to 60fps.

Task 2 was integrated successfully, while task 3 was met with the means to run an update function for each object successfully.

6. References

- [1] R. Nystrom, "Game Loop," 2014. [Online]. Available: <https://gameprogrammingpatterns.com/game-loop.html>. [Accessed 5th May 2020].

Appendices

Source.cpp

```
#include "mesh.h"
#include "display.h"
#include "cGameOBJS.h"
//
#include "glut.h"
#include <Windows.h>
#include <iostream>

using namespace std;

int main(int argc, char *args[])
{
    cGameOBJS gameObjects;        // Instantiate game object container

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

    //specify a list of vertices to draw - this will be loaded into our mesh object
    //we'll replace this array with a file mesh loader later - see .obj loader
    notes.

    //Vertex vertices[48];
    //Vertex myV;
    ///a triangle
    ///*vertices[0].pos = glm::vec3(-0.5, -0.5, 0.0);
    //vertices[1].pos = glm::vec3(0.0, 0.886-0.5, 0.0);
    //vertices[2].pos = glm::vec3(0.5, -0.5, 0.0);*/
    ///A House
    //vertices[0].pos = glm::vec3(0.0, 0.0, 0.0); //front
    //vertices[1].pos = glm::vec3(0.0, -2.0, 0.0);
    //vertices[2].pos = glm::vec3(1.0, 0.0, 0.0);
    //vertices[3].pos = glm::vec3(0.0, -2.0, 0.0);
    //vertices[4].pos = glm::vec3(1.0, -2.0, 0.0);
    //vertices[5].pos = glm::vec3(1.0, 0.0, 0.0);

    //vertices[6].pos = glm::vec3(1.0, 0.0, 0.0); //right
    //vertices[7].pos = glm::vec3(1.0, -2.0, 0.0);
    //vertices[8].pos = glm::vec3(1.0, -2.0, -1.0);
    //vertices[9].pos = glm::vec3(1.0, 0.0, 0.0);
    //vertices[10].pos = glm::vec3(1.0, -2.0, -1.0);
    //vertices[11].pos = glm::vec3(1.0, 0.0, -1.0);

    //vertices[12].pos = glm::vec3(0.0, 0.0, 0.0); //left
    //vertices[13].pos = glm::vec3(0.0, 0.0, -1.0);
    //vertices[14].pos = glm::vec3(0.0, -2.0, -1.0);
    //vertices[15].pos = glm::vec3(0.0, 0.0, 0.0);
    //vertices[16].pos = glm::vec3(0.0, -2.0, -1.0);
    //vertices[17].pos = glm::vec3(0.0, -2.0, 0.0);

    //vertices[18].pos = glm::vec3(0.0, -2.0, 0.0); //bottom
    //vertices[19].pos = glm::vec3(0.0, -2.0, -1.0);
    //vertices[20].pos = glm::vec3(1.0, -2.0, -1.0);
    //vertices[21].pos = glm::vec3(0.0, -2.0, 0.0);
    //vertices[22].pos = glm::vec3(1.0, -2.0, -1.0);
```

```

//vertices[23].pos = glm::vec3(1.0, -2.0, 0.0);

//vertices[24].pos = glm::vec3(0.0, 0.0, -1.0);//back
//vertices[25].pos = glm::vec3(1.0, -2.0, -1.0);
//vertices[26].pos = glm::vec3(1.0, 0.0, -1.0);
//vertices[27].pos = glm::vec3(0.0, 0.0, -1.0);
//vertices[28].pos = glm::vec3(0.0, -2.0, -1.0);
//vertices[29].pos = glm::vec3(1.0, -2.0, -1.0);

//vertices[30].pos = glm::vec3(0.0, 0.0, 0.0);//front gable
//vertices[31].pos = glm::vec3(1.0, 0.0, 0.0);
//vertices[32].pos = glm::vec3(0.5, 1.25, 0.0);

//vertices[33].pos = glm::vec3(0.0, 0.0, -1.0);//back gable
//vertices[34].pos = glm::vec3(0.5, 1.25, -1.0);
//vertices[35].pos = glm::vec3(1.0, 0.0, -1.0);
//

//vertices[36].pos = glm::vec3(0.0, 0.0, 0.0);//left roof
//vertices[37].pos = glm::vec3(0.5, 1.25, 0.0);
//vertices[38].pos = glm::vec3(0.5, 1.25, -1.0);
//vertices[39].pos = glm::vec3(0.0, 0.0, 0.0);
//vertices[40].pos = glm::vec3(0.5, 1.25, -1.0);
//vertices[41].pos = glm::vec3(0.0, 0.0, -1.0);

//vertices[42].pos = glm::vec3(0.5, 1.25, 0.0);//right roof
//vertices[43].pos = glm::vec3(1.0, 0.0, 0.0);
//vertices[44].pos = glm::vec3(1.0, 0.0, -1.0);
//vertices[45].pos = glm::vec3(0.5, 1.25, 0.0);
//vertices[46].pos = glm::vec3(1.0, 0.0, -1.0);
//vertices[47].pos = glm::vec3(0.5, 1.25, -1.0);

//Mesh* pmesh = new Mesh(vertices, sizeof(vertices) / sizeof(vertices[0]));

gameObjects.LoadNewOBJ("house.obj");
//gameObjects.LoadNewOBJ("blob.obj");
/* gameObjects.LoadNewOBJ("blendermonkeyobj.obj");*/ // Load the
blendermonkeyobj.obj into the container class
//Mesh* pmesh = new Mesh("blendermonkeyobj.obj");

float MS_PER_UPDATE = 16.67, // 1000 ms allocated per 60 frames
lag = 0; // Amount of time the game is given to update per frame
static unsigned long dwOldTime = GetTickCount(), // Initial time, initialised
by getting ticks since device booted up
dwCurrentTime = 0, // Updated in the game loop, used to calculate
elapsed time over the while loop
dwTimeElapsed = 0; // Necessary for outputting the FPS and elapsed
time at as close to 1000ms intervals as possible
static int fps = 0, // Frames per second, tracks how many have passed
i = 0; // Used in counting loops

while (!myDisplay.isClosed()) // Application loop - it's ok as a first
attempt
{
    //myDisplay.clear();// makes the point that the frame buffer is usually
cleared before redrawing
    //glLoadIdentity();
    ///glTranslatef(-5, -2, -10);//
    //gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    /// 5,5,5 - place camera ; 0,0,0 - aim lens towards 0,0,0;
    /// 0,1,0 - the up vector defines an unique orientation
    //pmesh->draw();

```

```

        ///glTranslatef(1.0, 0.0, 0.0);
        ///glColor3f(0.0, 1.0, 0.0);
        ///mesh.draw();
        ///glTranslatef(2.0, 0.0, -1.0);
        ///glRotatef(90.0, 0.0, 1.0, 0.0);
        ///glColor3f(0.0, 0.0, 1.0);
        ///mesh.draw();///mesh.draw(); // draw the model (s)
        ///glScalef(0.5f, 0.5f, 0.5f);
        ///glRotatef(90, 0, 0, 1);
        ///glTranslatef(0.5, 0, 0);
        ///mesh.draw();

        dwCurrentTime = GetTickCount(); // Set current time to calculate
elapsed
        dwTimeElapsed = dwCurrentTime - dwOldTime; // The difference
between the beginning of the loop (or last FPS output) and now

        if (dwTimeElapsed > 1000) { // FPS and elapsed time output after 1s has
passed

                cout << "frames per sec = " << fps << endl;
                cout << "time elapsed (ms) = " << dwTimeElapsed << endl;
                fps = 0; // As an FPS output has occurred, reset FPS
                dwOldTime = dwCurrentTime; // Update the old time so that new
elapsed time calculations can begin and reupdate once 1s has passed again
        }

        lag += dwTimeElapsed; // Provide the system with lag time, in which
operations can be done in said time

        gameObjects.UpdateLoadedOBJS(); // Updates each object loaded into the
container class

        while (lag >= MS_PER_UPDATE) // Update the display as many times as
possible until there isn't enough lag time for more updates
        {

                myDisplay.update();
                lag -= MS_PER_UPDATE;

        }

        for (i = 0; i < gameObjects.GetNumOBJSLoaded(); i++) // Go through
each obj loaded into the container class and render them
        {

                myDisplay.render(gameObjects.GetOBJ(i));

        }

        fps++; // Increment FPS as a loop has passed
        if (fps == 60) // If the FPS has reached 60, the program sleeps for the
remainder of the allotted 1s or 1000ms, providing a consistent near 60 FPS
        {

                Sleep(1000 - dwTimeElapsed);

        }

        //Sleep(dwOldTime + MS_PER_UPDATE - GetTickCount());

        //This will soon need to be revised so we can update model positions,
look after timers/ do physics/ deal with events.
    }
    return 0;
}

```


cGameOBJS.h

```
#pragma once
#include <vector>
#include <Windows.h>
#include "mesh.h"

using namespace std;

class cGameOBJS    // Game object container class
{
public:
    cGameOBJS();
    ~cGameOBJS();

    void LoadNewOBJ(const char *fileName);    // Pass in a filename in the format
    "objectname.obj", which adds this mesh object to the vector of loaded objects

    int GetNumOBJSLoaded();    // Gets the number of objects in the container
class Mesh* GetOBJ(int index);    // Gets an object at a specific position in the
vector

    void UpdateOBJ(int index);    // Updates the mesh details for an object at the
position in the vector
    void UpdateLoadedOBJS();    // Used to update all objects at once contained in
the class

private:
    vector <Mesh> mLoadedOBJS;    // Contains the mesh details for each object loaded
into the container class
};
```

cGameOBJS.cpp

```
#include "cGameOBJS.h"
#include <iostream>
#include <Windows.h>

using namespace std;

cGameOBJS::cGameOBJS() {

}

cGameOBJS::~cGameOBJS() {

}

void cGameOBJS::LoadNewOBJ(const char *fileName) {
    mLoadedOBJS.push_back(Mesh(fileName));
}

int cGameOBJS::GetNumOBJSLoaded()
{
    return mLoadedOBJS.size();
}

Mesh* cGameOBJS::GetOBJ(int index) {
    return &mLoadedOBJS[index];
}

void cGameOBJS::UpdateOBJ(int index) {
    float oldTime = GetTickCount(), currentTime = 0, elapsedTime = 0;    // Can be
    used for time calculations

    /*-----| Insert update code here |-----*/
    Sleep(1);

    currentTime = GetTickCount();
    elapsedTime = currentTime - oldTime;

    cout << "Game object updated in " << elapsedTime << "ms" << endl;
}

void cGameOBJS::UpdateLoadedOBJS() {
    for (int i = 0; i < mLoadedOBJS.size(); i++)    // Loops through each object
    and runs the update function for each item
    {
        UpdateOBJ(i);
    }
}
```

Display.h

```
#ifndef DISPLAY_H
#define DISPLAY_H
#include <SDL2/SDL.h>
#include <GL/glew.h>
#include <string>
#include <vector>
#include "cGameOBJS.h"
#include "mesh.h"

using namespace std;

//uses SDL2 to create a window to display our OGL stuff
class Display
{
public:
    Display(int width, int height, std::string title);
    virtual ~Display();
    void swapBuffers();
    bool isClosed();
    void update();
    void clear(void);
    void render(Mesh *pMesh);
private:
    bool closed;
    SDL_Window* window;    //pointer to window
    SDL_GLContext glContext; //context for drawing OGL to SDL window
};

#endif //DISPLAY_H
```

Display.cpp

```
#include "display.h"
//#include <GL/GLU.h>

#include <GL/glew.h>
#include <string>
#include <iostream>

using namespace std;

Display::Display(int width,int height, std::string title)
{
    SDL_Init(SDL_INIT_EVERYTHING); //uses SDL2 to create

    //std::cout << "hello world" << std::endl;

    SDL_GL_SetAttribute(SDL_GL_RED_SIZE, 8);
    SDL_GL_SetAttribute(SDL_GL_GREEN_SIZE, 8);
    SDL_GL_SetAttribute(SDL_GL_BLUE_SIZE, 8);
    SDL_GL_SetAttribute(SDL_GL_ALPHA_SIZE, 8);
    SDL_GL_SetAttribute(SDL_GL_BUFFER_SIZE, 32); //THESE SIZES ARE AT LEAST
    THESE VALUES.
    SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1); //1 turns double buffering on
    in SDL.

    window = SDL_CreateWindow(title.c_str(), 100,100/* SDL_WINDOWPOS_CENTERED,
    SDL_WINDOWPOS_CENTERED*/, width, height, SDL_WINDOW_OPENGL | SDL_WINDOW_RESIZABLE );

    glContext = SDL_GL_CreateContext(window); //create the context
    SDL_GL_SetSwapInterval(0);
    GLenum status = glewInit(); //glew searches to find every single OGL function
    supported.
    if (status != GLEW_OK)
    {
        cerr << "Glew failed to initialize" << endl;
    }

    closed = false;
    glShadeModel(GL_FLAT); //uses GL_SMOOTH or GL_FLAT

    //set viewport to screen size and initialize projection matrix. Leave with
    matrixmode = Modelview
    glViewport(0, 0, width, height); //make sure openGL view port is scaled to size
    of new window (if you want!) - why didn't I use freeglut?
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
    //          l,r,bot,top,near,far
    glMatrixMode(GL_MODELVIEW);
}

Display::~Display()
{
    SDL_GL_DeleteContext(glContext);

    SDL_DestroyWindow(window);
}
```

```

        SDL_Quit();
        //std::cout << "bye bye" << std::endl;
    }

void Display::update()
{
    SDL_Event e;
    while (SDL_PollEvent(&e)) //if an event occurs then process it and repeat til
event queue is empty - catches SDL_QUIT if it is in the queue
    {
        /* cout << "event number:" << e.type << endl; */
        if (e.type == SDL_QUIT)
            closed = true;

        if (e.window.event == SDL_WINDOWEVENT_RESIZED)
        {
            glViewport(0, 0, e.window.data1,e.window.data2); //make sure openGL view
port is scaled to size of new window (if you want!) - why didn't I use freeglut?
            cout << "resized to " << e.window.data1 << " , " <<
e.window.data2<<endl;
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
            //          l,r,bot,top,near,far

            glMatrixMode(GL_MODELVIEW);
        }
    }
}

void Display::swapBuffers(void)
{
    SDL_GL_SwapWindow(window);
}

void Display::render(Mesh* pMesh)
{
    clear();// makes the point that the frame buffer is usually cleared before
redrawing
    glLoadIdentity();
    //glTranslatef(-5, -2, -10);//
    gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    // 5,5,5 - place camera ; 0,0,0 - aim lens towards 0,0,0;
    // 0,1,0 - the up vector defines an unique orientation
    glColor3f(0.0, 1.0, 0.0);
    pMesh->draw();
    glTranslatef(1.0, 0.0, 0.0);
    glColor3f(1.0, 0.0, 0.0);
    pMesh->draw();
    glTranslatef(2.0, 0.0, -1.0);
    glRotatef(90.0, 0.0, 1.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    pMesh->draw();//mesh.draw(); // draw the model (s)
    //glScalef(0.5f, 0.5f, 0.5f);
    glRotatef(90, 0, 0, 1);
    glTranslatef(0.5, 0, 0);
    glColor3f(0.0, 1.0, 1.0);
    pMesh->draw();
    swapBuffers();
}

bool Display::isClosed()
{

```

```
        return closed;
    }

    void Display::clear(void)
    {
        glClearColor(0.0f, 0.15f, 0.3f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT);
    }
```

objloader.h

```
#ifndef OBJLOAD
#define OBJLOAD
/*****
****
*Loads obj file - limited to vertices, faces, normals, texture maps
*loads to object structure as defined in .h file
****/

#define MAX_VERTICES 8000 // Max number of vertices (for each object)
#define MAX_POLYGONS 8000 // Max number of polygons (for each object)
#define MAX_NORMALS 8000 // Max number of polygons (for each object)

// Our vertex type
typedef struct {
    float x, y, z;
}vertex_type;

// Our normal type
typedef struct {
    float i, j, k;
}normcoord_type;

// The polygon (triangle), 3 numbers that aim 3 vertices
typedef struct {
    int v[3], t[3], n[3];
}polygon_type;

// The mapcoord type, 2 texture coordinates for each vertex
typedef struct {
    float u, v;
}mapcoord_type;

// The object type
class object_type {
public:
    int id_texture;
    object_type() {}
    ~object_type() {}
    int objloader(const char *p_filename);
    int objdatadisplay();
    int getNumPolys(void) {return polygons_qty;}
    polygon_type getPolygon(int i) { return polygon[i]; }
    vertex_type getVertex(int i) { return vertex[i]; }

private:
    char name[20];
    int vertices_qty;
    int polygons_qty;
    int mapcoord_qty;
    int normcoord_qty;

    vertex_type vertex[MAX_VERTICES];
    mapcoord_type mapcoord[MAX_VERTICES];
    normcoord_type normcoord[MAX_NORMALS];
    polygon_type polygon[MAX_POLYGONS];
};
```

```
#endif
```

```
/****** /
```

objloader.cpp

```
#include <iostream>
#include <fstream>
#include "objloader.h"

using namespace std;
int object_type::objloader(const char * f)
{
    FILE *l_file; //File pointer

    int ipolygon = 0, imap = 0, ivertex = 0, inormal = 0;
    char l_char; //Char variable
    char myString[256];
    // int vertex;
    unsigned short l_face_flags; //Flag that stores some face information

    if ((l_file=fopen(f, "rt")) == NULL) return 0; //Open the file

    while (!feof(l_file)) //Loop to scan the whole file
    {

        fscanf_s(l_file, "%c", &l_char);
        if (l_char == '\n')//read char if '\n' -skip to next and read
            fscanf_s(l_file, "%c", &l_char);
        switch (l_char) //parse
        {
            default: fgets(myString, 256, l_file);
                    break;
            case 'v': //a vertex or a normal or a text co-ord
                    fscanf_s(l_file, "%c", &l_char);
                    switch (l_char)
                    {
                        case ' ': //a vertex -expect and so read 3 floats next
                                fscanf_s(l_file, "%f %f %f", &vertex[ivertex].x,
                                &vertex[ivertex].y, &vertex[ivertex].z);

                                ivertex++;
                                break;
                        case 'n': //a normal -expect and so read 3 floats next
                                fscanf_s(l_file, "%f %f %f", &normcoord[inormal].i,
                                &normcoord[inormal].j, &normcoord[inormal].k);
                                inormal++;
                                break;
                        case 't': //a texture map coord-expect and so read 2 floats next
                                fscanf_s(l_file, "%f %f", &mapcoord[imap].u,
                                &mapcoord[imap].v);
                                imap++;
                                break;
                    } //end switch
                    break;
            case 'f': //a face read next assume format is -> f 1/1/1 2/2/2 3/3/3
                    for (int i = 0; i < 3; i++) //for each vertex of polygon
                    {
                        fscanf_s(l_file, "%c", &l_char); //read space char -
                        fscanf_s(l_file, "%d", &polygon[ipolygon].v[i]); //read
                        ignore this vertex.
                    }
                }
            (triangle)
        }
    }
}
```

```

                                fscanf_s(l_file, "%c", &l_char); //read space char -
ignore this                                fscanf_s(l_file, "%d", &polygon[ipolygon].t[i]); //read
text coord.                                fscanf_s(l_file, "%c", &l_char); //read space char -
ignore this                                fscanf_s(l_file, "%d", &polygon[ipolygon].n[i]); //read
normal.

                                }
                                ipolygon++;
                                break;

                                } //end switch
                                }
                                fclose(l_file); // Closes the file stream
                                vertices_qty = ivertex;
                                polygons_qty = ipolygon;
                                mapcoord_qty = imap;
                                normcoord_qty = inormal;
                                return 0;
}

```

mesh.h

```
#ifndef MESH_H
#define MESH_H

#include <glm/glm.hpp>
#include <gl/glew.h>
#include <SOIL/SOIL.h>
#include "objloader.h"

#define NUM_BUFFERS 1
typedef struct {
    glm::vec3 pos;
}Vertex;

class Mesh
{
public:
    Mesh(Vertex* vertices, unsigned int numVertices);
    Mesh(const char *f);
    object_type myobj;
    void objPoly2Vertex(void);
    void draw();
    virtual ~Mesh();
private:
    Vertex *pVertices;
    GLuint vertexArrayObject; //for use with Modern OGL
    GLuint vertexArrayBuffers[NUM_BUFFERS]; //for use with Modern OGL
    GLuint vertexCount;

};

#endif
```

mesh.cpp

```
#include "mesh.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

Mesh::Mesh(const char* f)
{
    myobj.objloader(f);
    objPoly2Vertex();
    //std::cout << "hi" << endl;
    //vertexCount = numVertices;
    //pVertices = vertices;
    //glGenVertexArrays(1, &vertexArrayObject);
    //glBindVertexArray(vertexArrayObject);
    //glGenBuffers(NUM_BUFFERS, vertexArrayBuffers);
    //glBindBuffer(GL_ARRAY_BUFFER, vertexArrayBuffers[0]);
    //glBufferData(GL_ARRAY_BUFFER, numVertices * sizeof(vertices[0]), vertices,
GL_STATIC_DRAW);
    //
    //glEnableVertexAttribArray(0);
    //glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    //glBindVertexArray(0);
}

Mesh::Mesh(Vertex* vertices, unsigned int numVertices)
{
    vertexCount = numVertices;
    pVertices = vertices;
    //glGenVertexArrays(1, &vertexArrayObject);
    //glBindVertexArray(vertexArrayObject);
    //glGenBuffers(NUM_BUFFERS, vertexArrayBuffers);
    //glBindBuffer(GL_ARRAY_BUFFER, vertexArrayBuffers[0]);
    //glBufferData(GL_ARRAY_BUFFER, numVertices * sizeof(vertices[0]), vertices,
GL_STATIC_DRAW);
    //
    //glEnableVertexAttribArray(0);
    //glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
    //glBindVertexArray(0);
}

Mesh::~Mesh()
{
    //glDeleteVertexArrays(1, &vertexArrayObject);
}

void Mesh::draw()
{
    //glBindVertexArray(vertexArrayObject);
    //glDrawArrays(GL_TRIANGLES, 0, vertexCount);
    for (int i = 0; i < vertexCount; i+=3)
    {
        glBegin(GL_TRIANGLES);

        glVertex3f(pVertices[i].pos.x, pVertices[i].pos.y, pVertices[i].pos.z);

        glVertex3f(pVertices[i+1].pos.x, pVertices[i+1].pos.y,
pVertices[i+1].pos.z);
```

```

        glVertex3f(pVertices[i+2].pos.x, pVertices[i+2].pos.y,
pVertices[i+2].pos.z);

        glEnd();
    }
}

void Mesh::objPoly2Vertex(void)
{
    int stride = 3; //poly is a triangle;
    vertexCount = stride * myobj.getNumPolys(); //3 points per polygon
    int polyCount = myobj.getNumPolys(); //number of polygons
    pVertices = new Vertex[vertexCount]; //number of vertices

    for (int i = 0; i < polyCount; i++) //for each polygon
    {
        polygon_type poly= myobj.getPolygon(i);
        pVertices[stride*i + 0].pos.x = myobj.getVertex(poly.v[0] - 1).x;
        pVertices[stride*i + 0].pos.y = myobj.getVertex(poly.v[0] - 1).y;
        pVertices[stride*i + 0].pos.z = myobj.getVertex(poly.v[0] - 1).z;

        pVertices[stride*i + 1].pos.x = myobj.getVertex(poly.v[1] - 1).x;
        pVertices[stride*i + 1].pos.y = myobj.getVertex(poly.v[1] - 1).y;
        pVertices[stride*i + 1].pos.z = myobj.getVertex(poly.v[1] - 1).z;

        pVertices[stride*i + 2].pos.x = myobj.getVertex(poly.v[2] - 1).x;
        pVertices[stride*i + 2].pos.y = myobj.getVertex(poly.v[2] - 1).y;
        pVertices[stride*i + 2].pos.z = myobj.getVertex(poly.v[2] - 1).z;

        //copy 3 vertices of a polygon to 3 vertices - each vertex has x,y,z
    }
}

```

transform.h

```
#ifndef TRANSFORM_H
#define TRANSFORM_H
#include <glm/glm.hpp>
#include <GL/glew.h>
#include <string>

//Holds a mesh's transform
class Transform
{
public:
    Transform();
    Transform(glm::mat4x4* m);
    virtual ~Transform();

    void UpdateMV(glm::mat4x4* m);

private:
    glm::mat4x4 mv;
};

#endif
```

transform.cpp

```
#include "transform.h"
Transform::Transform()
{
}
Transform::Transform(glm::mat4x4* m)
{
}
Transform::~~Transform()
{
}
void UpdateMV(glm::mat4x4* m)
{
}
```

```

glut.h

#ifndef __glut_h__

#define __glut_h__

/* Copyright (c) Mark J. Kilgard, 1994, 1995, 1996, 1998. */

/* This program is freely distributable without licensing fees and is
   provided without guarantee or warrantee expressed or implied. This
   program is -not- in the public domain. */

#if defined(_WIN32)

/* GLUT 3.7 now tries to avoid including <windows.h>
   to avoid name space pollution, but Win32's <GL/gl.h>
   needs APIENTRY and WINGDIAPI defined properly. */
# if 0

/* This would put tons of macros and crap in our clean name space. */

# define WIN32_LEAN_AND_MEAN

# include <windows.h>

# else

/* XXX This is from Win32's <windef.h> */

# ifndef APIENTRY

# define GLUT_APIENTRY_DEFINED

# if (_MSC_VER >= 800) || defined(_STDCALL_SUPPORTED) ||
defined(__BORLANDC__) || defined(__LCC__)

```

```

# define APIENTRY __stdcall

# else

# define APIENTRY

# endif

# endif

/* XXX This is from Win32's <winnt.h> */

# ifndef CALLBACK

# if (defined(_M_MRX000) || defined(_M_IX86) || defined(_M_ALPHA) ||
defined(_M_PPC)) && !defined(MIDL_PASS) || defined(__LCC__)

# define CALLBACK __stdcall

# else

# define CALLBACK

# endif

# endif

/* XXX Hack for lcc compiler. It doesn't support __declspec(dllimport), just
__stdcall. */

# if defined( __LCC__ )

# undef WINGDIAPI

# define WINGDIAPI __stdcall

# else

/* XXX This is from Win32's <wingdi.h> and <winnt.h> */

# ifndef WINGDIAPI

# define GLUT_WINGDIAPI_DEFINED

# define WINGDIAPI __declspec(dllimport)

# endif

```

```

# endif

/* XXX This is from Win32's <ctype.h> */

# ifndef _WCHAR_T_DEFINED

typedef unsigned short wchar_t;

# define _WCHAR_T_DEFINED

# endif

# endif

/* To disable automatic library usage for GLUT, define GLUT_NO_LIB_PRAGMA
   in your compile preprocessor options. */

# if !defined(GLUT_BUILDING_LIB) && !defined(GLUT_NO_LIB_PRAGMA)

# pragma comment (lib, "winmm.lib") /* link with Windows MultiMedia lib */

/* To enable automatic SGI OpenGL for Windows library usage for GLUT,
   define GLUT_USE_SGI_OPENGL in your compile preprocessor options. */

# ifdef GLUT_USE_SGI_OPENGL

# pragma comment (lib, "opengl.lib") /* link with SGI OpenGL for Windows lib */

# pragma comment (lib, "glu.lib") /* link with SGI OpenGL Utility lib */

# pragma comment (lib, "glut.lib") /* link with Win32 GLUT for SGI OpenGL lib */

# else

# pragma comment (lib, "opengl32.lib") /* link with Microsoft OpenGL lib */

# pragma comment (lib, "glu32.lib") /* link with Microsoft OpenGL Utility lib */

# pragma comment (lib, "glut32.lib") /* link with Win32 GLUT lib */

# endif

# endif

```

```

/* To disable supression of annoying warnings about floats being promoted
to doubles, define GLUT_NO_WARNING_DISABLE in your compile preprocessor
options. */

#ifndef GLUT_NO_WARNING_DISABLE

# pragma warning (disable:4244) /* Disable bogus VC++ 4.2 conversion warnings.
*/

# pragma warning (disable:4305) /* VC++ 5.0 version of above warning. */

#endif

```

/* Win32 has an annoying issue where there are multiple C run-time
 libraries (CRTs). If the executable is linked with a different CRT
 from the GLUT DLL, the GLUT DLL will not share the same CRT static
 data seen by the executable. In particular, atexit callbacks registered
 in the executable will not be called if GLUT calls its (different)
 exit routine). GLUT is typically built with the
 "/MD" option (the CRT with multithreading DLL support), but the Visual
 C++ linker default is "/ML" (the single threaded CRT).

One workaround to this issue is requiring users to always link with
 the same CRT as GLUT is compiled with. That requires users supply a
 non-standard option. GLUT 3.7 has its own built-in workaround where
 the executable's "exit" function pointer is covertly passed to GLUT.
 GLUT then calls the executable's exit function pointer to ensure that
 any "atexit" calls registered by the application are called if GLUT

needs to exit.

Note that the `__glut*WithExit` routines should NEVER be called directly.

To avoid the `atexit` workaround, `#define GLUT_DISABLE_ATEXIT_HACK. */`

```
/* XXX This is from Win32's <process.h> */  
  
# if !defined(_MSC_VER) && !defined(__cdecl)  
    /* Define __cdecl for non-Microsoft compilers. */  
  
    # define __cdecl  
  
    # define GLUT_DEFINED__CDECL  
  
    # endif  
  
    # ifndef _CRTIMP  
  
    #   ifdef _NTSDK  
  
        /* Definition compatible with NT SDK */  
  
        #   define _CRTIMP  
  
        #   else  
  
            /* Current definition */  
  
            #   ifdef _DLL  
  
            #   define _CRTIMP __declspec(dllimport)  
  
            #   else  
  
            #   define _CRTIMP  
  
            #   endif  
  
            #   endif  
  
            #   define GLUT_DEFINED__CRTIMP
```

```
# endif
```

```
/* GLUT API entry point declarations for Win32. */
```

```
# ifdef GLUT_BUILDING_LIB
```

```
#  define GLUTAPI __declspec(dllexport)
```

```
# else
```

```
#  ifdef _DLL
```

```
#   define GLUTAPI __declspec(dllimport)
```

```
#  else
```

```
#   define GLUTAPI extern
```

```
#  endif
```

```
# endif
```

```
/* GLUT callback calling convention for Win32. */
```

```
# define GLUTCALLBACK __cdecl
```

```
#endif /* _WIN32 */
```

```
#include <GL/gl.h>
```

```
#include <GL/glu.h>
```

```
#ifdef __cplusplus
```

```
extern "C" {
```

```
#endif
```

```

#if defined(_WIN32)

# ifndef GLUT_BUILDING_LIB

extern _CRTIMP void __cdecl exit(int);

# endif

#else

/* non-Win32 case. */

/* Define APIENTRY and CALLBACK to nothing if we aren't on Win32. */

# define APIENTRY

# define GLUT_APIENTRY_DEFINED

# define CALLBACK

/* Define GLUTAPI and GLUTCALLBACK as below if we aren't on Win32. */

# define GLUTAPI extern

# define GLUTCALLBACK

/* Prototype exit for the non-Win32 case (see above). */

extern void exit(int);

#endif

```

/**

GLUT API revision history:

GLUT_API_VERSION is updated to reflect incompatible GLUT
API changes (interface changes, semantic changes, deletions,
or additions).

GLUT_API_VERSION=1 First public release of GLUT. 11/29/94

GLUT_API_VERSION=2 Added support for OpenGL/GLX multisampling, extension. Supports new input devices like tablet, dial and button box, and Spaceball. Easy to query OpenGL extensions.

GLUT_API_VERSION=3 glutMenuStatus added.

GLUT_API_VERSION=4 glutInitDisplayString, glutWarpPointer, glutBitmapLength, glutStrokeLength, glutWindowStatusFunc, dynamic video resize subAPI, glutPostWindowRedisplay, glutKeyboardUpFunc, glutSpecialUpFunc, glutIgnoreKeyRepeat, glutSetKeyRepeat, glutJoystickFunc, glutForceJoystickFunc (NOT FINALIZED!).

***/**

```
#ifndef GLUT_API_VERSION /* allow this to be overridden */
```

```
#define GLUT_API_VERSION      3
```

```
#endif
```

/*

GLUT implementation revision history:

GLUT_XLIB_IMPLEMENTATION is updated to reflect both GLUT API revisions and implementation revisions (ie, bug fixes).

GLUT_XLIB_IMPLEMENTATION=1 mjk's first public release of
GLUT Xlib-based implementation. 11/29/94

GLUT_XLIB_IMPLEMENTATION=2 mjk's second public release of
GLUT Xlib-based implementation providing GLUT version 2
interfaces.

GLUT_XLIB_IMPLEMENTATION=3 mjk's GLUT 2.2 images. 4/17/95

GLUT_XLIB_IMPLEMENTATION=4 mjk's GLUT 2.3 images. 6/?/95

GLUT_XLIB_IMPLEMENTATION=5 mjk's GLUT 3.0 images. 10/?/95

GLUT_XLIB_IMPLEMENTATION=7 mjk's GLUT 3.1+ with glutWarpPoiner.
7/24/96

GLUT_XLIB_IMPLEMENTATION=8 mjk's GLUT 3.1+ with glutWarpPoiner
and video resize. 1/3/97

GLUT_XLIB_IMPLEMENTATION=9 mjk's GLUT 3.4 release with early GLUT 4
routines.

GLUT_XLIB_IMPLEMENTATION=11 Mesa 2.5's GLUT 3.6 release.

GLUT_XLIB_IMPLEMENTATION=12 mjk's GLUT 3.6 release with early GLUT 4 routines + signal handling.

GLUT_XLIB_IMPLEMENTATION=13 mjk's GLUT 3.7 beta with GameGLUT support.

GLUT_XLIB_IMPLEMENTATION=14 mjk's GLUT 3.7 beta with f90gl friend interface.

GLUT_XLIB_IMPLEMENTATION=15 mjk's GLUT 3.7 beta sync'ed with Mesa
<GL/glut.h>

*/

#ifndef GLUT_XLIB_IMPLEMENTATION /* Allow this to be overridden. */

#define GLUT_XLIB_IMPLEMENTATION 15

#endif

/* Display mode bit masks. */

#define GLUT_RGB 0

#define GLUT_RGBA GLUT_RGB

#define GLUT_INDEX 1

#define GLUT_SINGLE 0

#define GLUT_DOUBLE 2

#define GLUT_ACCUM 4

#define GLUT_ALPHA 8

#define GLUT_DEPTH 16

```

#define GLUT_STENCIL          32

#if (GLUT_API_VERSION >= 2)

#define GLUT_MULTISAMPLE     128

#define GLUT_STEREO          256

#endif

#if (GLUT_API_VERSION >= 3)

#define GLUT_LUMINANCE        512

#endif

/* Mouse buttons. */

#define GLUT_LEFT_BUTTON      0

#define GLUT_MIDDLE_BUTTON    1

#define GLUT_RIGHT_BUTTON     2

/* Mouse button state. */

#define GLUT_DOWN             0

#define GLUT_UP               1

#if (GLUT_API_VERSION >= 2)

/* function keys */

#define GLUT_KEY_F1           1

#define GLUT_KEY_F2           2

#define GLUT_KEY_F3           3

#define GLUT_KEY_F4           4

45

```

```

#define GLUT_KEY_F5          5
#define GLUT_KEY_F6          6
#define GLUT_KEY_F7          7
#define GLUT_KEY_F8          8
#define GLUT_KEY_F9          9
#define GLUT_KEY_F10         10
#define GLUT_KEY_F11         11
#define GLUT_KEY_F12         12

/* directional keys */

#define GLUT_KEY_LEFT         100
#define GLUT_KEY_UP           101
#define GLUT_KEY_RIGHT        102
#define GLUT_KEY_DOWN         103
#define GLUT_KEY_PAGE_UP      104
#define GLUT_KEY_PAGE_DOWN    105
#define GLUT_KEY_HOME         106
#define GLUT_KEY_END          107
#define GLUT_KEY_INSERT       108

#endif

/* Entry/exit state. */

#define GLUT_LEFT              0
#define GLUT_ENTERED          1

```

```

/* Menu usage state. */

#define GLUT_MENU_NOT_IN_USE      0

#define GLUT_MENU_IN_USE         1


/* Visibility state. */

#define GLUT_NOT_VISIBLE          0

#define GLUT_VISIBLE              1


/* Window status state. */

#define GLUT_HIDDEN                0

#define GLUT_FULLY_RETAINED        1

#define GLUT_PARTIALLY_RETAINED    2

#define GLUT_FULLY_COVERED         3


/* Color index component selection values. */

#define GLUT_RED                   0

#define GLUT_GREEN                 1

#define GLUT_BLUE                  2


#if defined(_WIN32)

/* Stroke font constants (use these in GLUT program). */

#define GLUT_STROKE_ROMAN          ((void*)0)

#define GLUT_STROKE_MONO_ROMAN     ((void*)1)

```

```

/* Bitmap font constants (use these in GLUT program). */

#define GLUT_BITMAP_9_BY_15          ((void*)2)

#define GLUT_BITMAP_8_BY_13          ((void*)3)

#define GLUT_BITMAP_TIMES_ROMAN_10   ((void*)4)

#define GLUT_BITMAP_TIMES_ROMAN_24   ((void*)5)

#if (GLUT_API_VERSION >= 3)

#define GLUT_BITMAP_HELVETICA_10     ((void*)6)

#define GLUT_BITMAP_HELVETICA_12     ((void*)7)

#define GLUT_BITMAP_HELVETICA_18     ((void*)8)

#endif

#else

/* Stroke font opaque addresses (use constants instead in source code). */

GLUTAPI void *glutStrokeRoman;

GLUTAPI void *glutStrokeMonoRoman;

/* Stroke font constants (use these in GLUT program). */

#define GLUT_STROKE_ROMAN             (&glutStrokeRoman)

#define GLUT_STROKE_MONO_ROMAN        (&glutStrokeMonoRoman)

/* Bitmap font opaque addresses (use constants instead in source code). */

GLUTAPI void *glutBitmap9By15;

GLUTAPI void *glutBitmap8By13;

GLUTAPI void *glutBitmapTimesRoman10;

GLUTAPI void *glutBitmapTimesRoman24;

```

```
GLUTAPI void *glutBitmapHelvetica10;
```

```
GLUTAPI void *glutBitmapHelvetica12;
```

```
GLUTAPI void *glutBitmapHelvetica18;
```

```
/* Bitmap font constants (use these in GLUT program). */
```

```
#define GLUT_BITMAP_9_BY_15          (&glutBitmap9By15)
```

```
#define GLUT_BITMAP_8_BY_13          (&glutBitmap8By13)
```

```
#define GLUT_BITMAP_TIMES_ROMAN_10   (&glutBitmapTimesRoman10)
```

```
#define GLUT_BITMAP_TIMES_ROMAN_24   (&glutBitmapTimesRoman24)
```

```
#if (GLUT_API_VERSION >= 3)
```

```
#define GLUT_BITMAP_HELVETICA_10     (&glutBitmapHelvetica10)
```

```
#define GLUT_BITMAP_HELVETICA_12     (&glutBitmapHelvetica12)
```

```
#define GLUT_BITMAP_HELVETICA_18     (&glutBitmapHelvetica18)
```

```
#endif
```

```
#endif
```

```
/* glutGet parameters. */
```

```
#define GLUT_WINDOW_X                ((GLenum) 100)
```

```
#define GLUT_WINDOW_Y                ((GLenum) 101)
```

```
#define GLUT_WINDOW_WIDTH            ((GLenum) 102)
```

```
#define GLUT_WINDOW_HEIGHT           ((GLenum) 103)
```

```
#define GLUT_WINDOW_BUFFER_SIZE      ((GLenum) 104)
```

```
#define GLUT_WINDOW_STENCIL_SIZE     ((GLenum) 105)
```

```
#define GLUT_WINDOW_DEPTH_SIZE       ((GLenum) 106)
```

```

#define GLUT_WINDOW_RED_SIZE      ((GLenum) 107)

#define GLUT_WINDOW_GREEN_SIZE    ((GLenum) 108)

#define GLUT_WINDOW_BLUE_SIZE     ((GLenum) 109)

#define GLUT_WINDOW_ALPHA_SIZE    ((GLenum) 110)

#define GLUT_WINDOW_ACCUM_RED_SIZE ((GLenum) 111)

#define GLUT_WINDOW_ACCUM_GREEN_SIZE ((GLenum) 112)

#define GLUT_WINDOW_ACCUM_BLUE_SIZE ((GLenum) 113)

#define GLUT_WINDOW_ACCUM_ALPHA_SIZE ((GLenum) 114)

#define GLUT_WINDOW_DOUBLEBUFFER ((GLenum) 115)

#define GLUT_WINDOW_RGBA          ((GLenum) 116)

#define GLUT_WINDOW_PARENT        ((GLenum) 117)

#define GLUT_WINDOW_NUM_CHILDREN  ((GLenum) 118)

#define GLUT_WINDOW_COLORMAP_SIZE ((GLenum) 119)

#if (GLUT_API_VERSION >= 2)

#define GLUT_WINDOW_NUM_SAMPLES    ((GLenum) 120)

#define GLUT_WINDOW_STEREO        ((GLenum) 121)

#endif

#if (GLUT_API_VERSION >= 3)

#define GLUT_WINDOW_CURSOR         ((GLenum) 122)

#endif

#define GLUT_SCREEN_WIDTH          ((GLenum) 200)

#define GLUT_SCREEN_HEIGHT         ((GLenum) 201)

#define GLUT_SCREEN_WIDTH_MM       ((GLenum) 202)

#define GLUT_SCREEN_HEIGHT_MM      ((GLenum) 203)

```

```

#define GLUT_MENU_NUM_ITEMS      ((GLenum) 300)

#define GLUT_DISPLAY_MODE_POSSIBLE ((GLenum) 400)

#define GLUT_INIT_WINDOW_X      ((GLenum) 500)

#define GLUT_INIT_WINDOW_Y      ((GLenum) 501)

#define GLUT_INIT_WINDOW_WIDTH  ((GLenum) 502)

#define GLUT_INIT_WINDOW_HEIGHT ((GLenum) 503)

#define GLUT_INIT_DISPLAY_MODE  ((GLenum) 504)

#if (GLUT_API_VERSION >= 2)

#define GLUT_ELAPSED_TIME      ((GLenum) 700)

#endif

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 13)

#define GLUT_WINDOW_FORMAT_ID  ((GLenum) 123)

#endif


#if (GLUT_API_VERSION >= 2)

/* glutDeviceGet parameters. */

#define GLUT_HAS_KEYBOARD      ((GLenum) 600)

#define GLUT_HAS_MOUSE         ((GLenum) 601)

#define GLUT_HAS_SPACEBALL     ((GLenum) 602)

#define GLUT_HAS_DIAL_AND_BUTTON_BOX ((GLenum) 603)

#define GLUT_HAS_TABLET        ((GLenum) 604)

#define GLUT_NUM_MOUSE_BUTTONS ((GLenum) 605)

#define GLUT_NUM_SPACEBALL_BUTTONS ((GLenum) 606)

#define GLUT_NUM_BUTTON_BOX_BUTTONS ((GLenum) 607)

```

```

#define GLUT_NUM_DIALS                ((GLenum) 608)

#define GLUT_NUM_TABLET_BUTTONS       ((GLenum) 609)

#endif

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 13)

#define GLUT_DEVICE_IGNORE_KEY_REPEAT ((GLenum) 610)

#define GLUT_DEVICE_KEY_REPEAT        ((GLenum) 611)

#define GLUT_HAS_JOYSTICK              ((GLenum) 612)

#define GLUT_OWNS_JOYSTICK             ((GLenum) 613)

#define GLUT_JOYSTICK_BUTTONS         ((GLenum) 614)

#define GLUT_JOYSTICK_AXES            ((GLenum) 615)

#define GLUT_JOYSTICK_POLL_RATE       ((GLenum) 616)

#endif


#if (GLUT_API_VERSION >= 3)

/* glutLayerGet parameters. */

#define GLUT_OVERLAY_POSSIBLE         ((GLenum) 800)

#define GLUT_LAYER_IN_USE             ((GLenum) 801)

#define GLUT_HAS_OVERLAY              ((GLenum) 802)

#define GLUT_TRANSPARENT_INDEX        ((GLenum) 803)

#define GLUT_NORMAL_DAMAGED           ((GLenum) 804)

#define GLUT_OVERLAY_DAMAGED          ((GLenum) 805)


#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

/* glutVideoResizeGet parameters. */

```

```

#define GLUT_VIDEO_RESIZE_POSSIBLE    ((GLenum) 900)

#define GLUT_VIDEO_RESIZE_IN_USE      ((GLenum) 901)

#define GLUT_VIDEO_RESIZE_X_DELTA     ((GLenum) 902)

#define GLUT_VIDEO_RESIZE_Y_DELTA     ((GLenum) 903)

#define GLUT_VIDEO_RESIZE_WIDTH_DELTA ((GLenum) 904)

#define GLUT_VIDEO_RESIZE_HEIGHT_DELTA ((GLenum) 905)

#define GLUT_VIDEO_RESIZE_X           ((GLenum) 906)

#define GLUT_VIDEO_RESIZE_Y           ((GLenum) 907)

#define GLUT_VIDEO_RESIZE_WIDTH       ((GLenum) 908)

#define GLUT_VIDEO_RESIZE_HEIGHT      ((GLenum) 909)

#endif

/* glutUseLayer parameters. */

#define GLUT_NORMAL                    ((GLenum) 0)

#define GLUT_OVERLAY                  ((GLenum) 1)

/* glutGetModifiers return mask. */

#define GLUT_ACTIVE_SHIFT             1

#define GLUT_ACTIVE_CTRL              2

#define GLUT_ACTIVE_ALT               4

/* glutSetCursor parameters. */

/* Basic arrows. */

#define GLUT_CURSOR_RIGHT_ARROW       0

```

```

#define GLUT_CURSOR_LEFT_ARROW          1

/* Symbolic cursor shapes. */

#define GLUT_CURSOR_INFO                 2

#define GLUT_CURSOR_DESTROY             3

#define GLUT_CURSOR_HELP                 4

#define GLUT_CURSOR_CYCLE                5

#define GLUT_CURSOR_SPRAY                6

#define GLUT_CURSOR_WAIT                 7

#define GLUT_CURSOR_TEXT                 8

#define GLUT_CURSOR_CROSSHAIR           9

/* Directional cursors. */

#define GLUT_CURSOR_UP_DOWN             10

#define GLUT_CURSOR_LEFT_RIGHT          11

/* Sizing cursors. */

#define GLUT_CURSOR_TOP_SIDE             12

#define GLUT_CURSOR_BOTTOM_SIDE          13

#define GLUT_CURSOR_LEFT_SIDE            14

#define GLUT_CURSOR_RIGHT_SIDE           15

#define GLUT_CURSOR_TOP_LEFT_CORNER     16

#define GLUT_CURSOR_TOP_RIGHT_CORNER    17

#define GLUT_CURSOR_BOTTOM_RIGHT_CORNER 18

#define GLUT_CURSOR_BOTTOM_LEFT_CORNER 19

/* Inherit from parent window. */

#define GLUT_CURSOR_INHERIT              100

```

```

/* Blank cursor. */

#define GLUT_CURSOR_NONE      101

/* Fullscreen crosshair (if available). */

#define GLUT_CURSOR_FULL_CROSSHAIR 102

#endif


/* GLUT initialization sub-API. */

GLUTAPI void APIENTRY glutInit(int *argcp, char **argv);

#if defined(_WIN32) && !defined(GLUT_DISABLE_ATEXIT_HACK)

GLUTAPI void APIENTRY __glutInitWithExit(int *argcp, char **argv, void (__cdecl
*exitfunc)(int));

#endif

GLUTAPI void APIENTRY glutInit_ATEXIT_HACK(int *argcp, char **argv) {
__glutInitWithExit(argcp, argv, exit); }

#define glutInit glutInit_ATEXIT_HACK

#endif

#endif

GLUTAPI void APIENTRY glutInitDisplayMode(unsigned int mode);

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

GLUTAPI void APIENTRY glutInitDisplayString(const char *string);

#endif

GLUTAPI void APIENTRY glutInitWindowPosition(int x, int y);

GLUTAPI void APIENTRY glutInitWindowSize(int width, int height);

GLUTAPI void APIENTRY glutMainLoop(void);

```

```

/* GLUT window sub-API. */

GLUTAPI int APIENTRY glutCreateWindow(const char *title);

#if defined(_WIN32) && !defined(GLUT_DISABLE_ATEXIT_HACK)

GLUTAPI int APIENTRY __glutCreateWindowWithExit(const char *title, void
(__cdecl *exitfunc)(int));

#endif

#ifdef GLUT_BUILDING_LIB

static int APIENTRY glutCreateWindow_ATEXIT_HACK(const char *title) { return
__glutCreateWindowWithExit(title, exit); }

#define glutCreateWindow glutCreateWindow_ATEXIT_HACK

#endif

#endif

GLUTAPI int APIENTRY glutCreateSubWindow(int win, int x, int y, int width, int
height);

GLUTAPI void APIENTRY glutDestroyWindow(int win);

GLUTAPI void APIENTRY glutPostRedisplay(void);

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 11)

GLUTAPI void APIENTRY glutPostWindowRedisplay(int win);

#endif

GLUTAPI void APIENTRY glutSwapBuffers(void);

GLUTAPI int APIENTRY glutGetWindow(void);

GLUTAPI void APIENTRY glutSetWindow(int win);

GLUTAPI void APIENTRY glutSetWindowTitle(const char *title);

GLUTAPI void APIENTRY glutSetIconTitle(const char *title);

GLUTAPI void APIENTRY glutPositionWindow(int x, int y);

GLUTAPI void APIENTRY glutReshapeWindow(int width, int height);

```

```

GLUTAPI void APIENTRY glutPopWindow(void);

GLUTAPI void APIENTRY glutPushWindow(void);

GLUTAPI void APIENTRY glutIconifyWindow(void);

GLUTAPI void APIENTRY glutShowWindow(void);

GLUTAPI void APIENTRY glutHideWindow(void);

#if (GLUT_API_VERSION >= 3)

GLUTAPI void APIENTRY glutFullScreen(void);

GLUTAPI void APIENTRY glutSetCursor(int cursor);

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

GLUTAPI void APIENTRY glutWarpPointer(int x, int y);

#endif

/* GLUT overlay sub-API. */

GLUTAPI void APIENTRY glutEstablishOverlay(void);

GLUTAPI void APIENTRY glutRemoveOverlay(void);

GLUTAPI void APIENTRY glutUseLayer(GLenum layer);

GLUTAPI void APIENTRY glutPostOverlayRedisplay(void);

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 11)

GLUTAPI void APIENTRY glutPostWindowOverlayRedisplay(int win);

#endif

GLUTAPI void APIENTRY glutShowOverlay(void);

GLUTAPI void APIENTRY glutHideOverlay(void);

#endif

```

```

/* GLUT menu sub-API. */

GLUTAPI int APIENTRY glutCreateMenu(void (GLUTCALLBACK *func)(int));

#if defined(_WIN32) && !defined(GLUT_DISABLE_ATEXIT_HACK)

GLUTAPI int APIENTRY __glutCreateMenuWithExit(void (GLUTCALLBACK
*func)(int), void (__cdecl *exitfunc)(int));

#ifdef GLUT_BUILDING_LIB

static int APIENTRY glutCreateMenu_ATEXIT_HACK(void (GLUTCALLBACK
*func)(int)) { return __glutCreateMenuWithExit(func, exit); }

#define glutCreateMenu glutCreateMenu_ATEXIT_HACK

#endif

#endif

GLUTAPI void APIENTRY glutDestroyMenu(int menu);

GLUTAPI int APIENTRY glutGetMenu(void);

GLUTAPI void APIENTRY glutSetMenu(int menu);

GLUTAPI void APIENTRY glutAddMenuEntry(const char *label, int value);

GLUTAPI void APIENTRY glutAddSubMenu(const char *label, int submenu);

GLUTAPI void APIENTRY glutChangeToMenuEntry(int item, const char *label, int
value);

GLUTAPI void APIENTRY glutChangeToSubMenu(int item, const char *label, int
submenu);

GLUTAPI void APIENTRY glutRemoveMenuItem(int item);

GLUTAPI void APIENTRY glutAttachMenu(int button);

GLUTAPI void APIENTRY glutDetachMenu(int button);

/* GLUT window callback sub-API. */

```

```

GLUTAPI void APIENTRY glutDisplayFunc(void (GLUTCALLBACK *func)(void));

GLUTAPI void APIENTRY glutReshapeFunc(void (GLUTCALLBACK *func)(int
width, int height));

GLUTAPI void APIENTRY glutKeyboardFunc(void (GLUTCALLBACK
*func)(unsigned char key, int x, int y));

GLUTAPI void APIENTRY glutMouseFunc(void (GLUTCALLBACK *func)(int button,
int state, int x, int y));

GLUTAPI void APIENTRY glutMotionFunc(void (GLUTCALLBACK *func)(int x, int
y));

GLUTAPI void APIENTRY glutPassiveMotionFunc(void (GLUTCALLBACK *func)(int
x, int y));

GLUTAPI void APIENTRY glutEntryFunc(void (GLUTCALLBACK *func)(int state));

GLUTAPI void APIENTRY glutVisibilityFunc(void (GLUTCALLBACK *func)(int
state));

GLUTAPI void APIENTRY glutIdleFunc(void (GLUTCALLBACK *func)(void));

GLUTAPI void APIENTRY glutTimerFunc(unsigned int millis, void (GLUTCALLBACK
*func)(int value), int value);

GLUTAPI void APIENTRY glutMenuStateFunc(void (GLUTCALLBACK *func)(int
state));

#if (GLUT_API_VERSION >= 2)

GLUTAPI void APIENTRY glutSpecialFunc(void (GLUTCALLBACK *func)(int key, int
x, int y));

GLUTAPI void APIENTRY glutSpaceballMotionFunc(void (GLUTCALLBACK
*func)(int x, int y, int z));

GLUTAPI void APIENTRY glutSpaceballRotateFunc(void (GLUTCALLBACK
*func)(int x, int y, int z));

GLUTAPI void APIENTRY glutSpaceballButtonFunc(void (GLUTCALLBACK
*func)(int button, int state));

```

```

GLUTAPI void APIENTRY glutButtonBoxFunc(void (GLUTCALLBACK *func)(int
button, int state));

GLUTAPI void APIENTRY glutDialsFunc(void (GLUTCALLBACK *func)(int dial, int
value));

GLUTAPI void APIENTRY glutTabletMotionFunc(void (GLUTCALLBACK *func)(int x,
int y));

GLUTAPI void APIENTRY glutTabletButtonFunc(void (GLUTCALLBACK *func)(int
button, int state, int x, int y));

#if (GLUT_API_VERSION >= 3)

GLUTAPI void APIENTRY glutMenuStatusFunc(void (GLUTCALLBACK *func)(int
status, int x, int y));

GLUTAPI void APIENTRY glutOverlayDisplayFunc(void (GLUTCALLBACK
*func)(void));

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

GLUTAPI void APIENTRY glutWindowStatusFunc(void (GLUTCALLBACK *func)(int
state));

#endif

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 13)

GLUTAPI void APIENTRY glutKeyboardUpFunc(void (GLUTCALLBACK
*func)(unsigned char key, int x, int y));

GLUTAPI void APIENTRY glutSpecialUpFunc(void (GLUTCALLBACK *func)(int key,
int x, int y));

GLUTAPI void APIENTRY glutJoystickFunc(void (GLUTCALLBACK *func)(unsigned
int buttonMask, int x, int y, int z), int pollInterval);

#endif

#endif

#endif

```

```

/* GLUT color index sub-API. */

GLUTAPI void APIENTRY glutSetColor(int, GLfloat red, GLfloat green, GLfloat blue);

GLUTAPI GLfloat APIENTRY glutGetColor(int ndx, int component);

GLUTAPI void APIENTRY glutCopyColormap(int win);


/* GLUT state retrieval sub-API. */

GLUTAPI int APIENTRY glutGet(GLenum type);

GLUTAPI int APIENTRY glutDeviceGet(GLenum type);

#if (GLUT_API_VERSION >= 2)

/* GLUT extension support sub-API */

GLUTAPI int APIENTRY glutExtensionSupported(const char *name);

#endif

#if (GLUT_API_VERSION >= 3)

GLUTAPI int APIENTRY glutGetModifiers(void);

GLUTAPI int APIENTRY glutLayerGet(GLenum type);

#endif


/* GLUT font sub-API */

GLUTAPI void APIENTRY glutBitmapCharacter(void *font, int character);

GLUTAPI int APIENTRY glutBitmapWidth(void *font, int character);

GLUTAPI void APIENTRY glutStrokeCharacter(void *font, int character);

GLUTAPI int APIENTRY glutStrokeWidth(void *font, int character);

#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

```

```

GLUTAPI int APIENTRY glutBitmapLength(void *font, const unsigned char *string);

GLUTAPI int APIENTRY glutStrokeLength(void *font, const unsigned char *string);

#endif


/* GLUT pre-built models sub-API */

GLUTAPI void APIENTRY glutWireSphere(GLdouble radius, GLint slices, GLint
stacks);

GLUTAPI void APIENTRY glutSolidSphere(GLdouble radius, GLint slices, GLint
stacks);

GLUTAPI void APIENTRY glutWireCone(GLdouble base, GLdouble height, GLint
slices, GLint stacks);

GLUTAPI void APIENTRY glutSolidCone(GLdouble base, GLdouble height, GLint
slices, GLint stacks);

GLUTAPI void APIENTRY glutWireCube(GLdouble size);

GLUTAPI void APIENTRY glutSolidCube(GLdouble size);

GLUTAPI void APIENTRY glutWireTorus(GLdouble innerRadius, GLdouble
outerRadius, GLint sides, GLint rings);

GLUTAPI void APIENTRY glutSolidTorus(GLdouble innerRadius, GLdouble
outerRadius, GLint sides, GLint rings);

GLUTAPI void APIENTRY glutWireDodecahedron(void);

GLUTAPI void APIENTRY glutSolidDodecahedron(void);

GLUTAPI void APIENTRY glutWireTeapot(GLdouble size);

GLUTAPI void APIENTRY glutSolidTeapot(GLdouble size);

GLUTAPI void APIENTRY glutWireOctahedron(void);

GLUTAPI void APIENTRY glutSolidOctahedron(void);

GLUTAPI void APIENTRY glutWireTetrahedron(void);

```

```

GLUTAPI void APIENTRY glutSolidTetrahedron(void);

GLUTAPI void APIENTRY glutWireIcosahedron(void);

GLUTAPI void APIENTRY glutSolidIcosahedron(void);


#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 9)

/* GLUT video resize sub-API. */

GLUTAPI int APIENTRY glutVideoResizeGet(GLenum param);

GLUTAPI void APIENTRY glutSetupVideoResizing(void);

GLUTAPI void APIENTRY glutStopVideoResizing(void);

GLUTAPI void APIENTRY glutVideoResize(int x, int y, int width, int height);

GLUTAPI void APIENTRY glutVideoPan(int x, int y, int width, int height);


/* GLUT debugging sub-API. */

GLUTAPI void APIENTRY glutReportErrors(void);

#endif


#if (GLUT_API_VERSION >= 4 || GLUT_XLIB_IMPLEMENTATION >= 13)

/* GLUT device control sub-API. */

/* glutSetKeyRepeat modes. */

#define GLUT_KEY_REPEAT_OFF          0

#define GLUT_KEY_REPEAT_ON           1

#define GLUT_KEY_REPEAT_DEFAULT      2


/* Joystick button masks. */

```

```

#define GLUT_JOYSTICK_BUTTON_A      1

#define GLUT_JOYSTICK_BUTTON_B      2

#define GLUT_JOYSTICK_BUTTON_C      4

#define GLUT_JOYSTICK_BUTTON_D      8


GLUTAPI void APIENTRY glutIgnoreKeyRepeat(int ignore);

GLUTAPI void APIENTRY glutSetKeyRepeat(int repeatMode);

GLUTAPI void APIENTRY glutForceJoystickFunc(void);


/* GLUT game mode sub-API. */

/* glutGameModeGet. */

#define GLUT_GAME_MODE_ACTIVE        ((GLenum) 0)

#define GLUT_GAME_MODE_POSSIBLE     ((GLenum) 1)

#define GLUT_GAME_MODE_WIDTH        ((GLenum) 2)

#define GLUT_GAME_MODE_HEIGHT       ((GLenum) 3)

#define GLUT_GAME_MODE_PIXEL_DEPTH  ((GLenum) 4)

#define GLUT_GAME_MODE_REFRESH_RATE ((GLenum) 5)

#define GLUT_GAME_MODE_DISPLAY_CHANGED ((GLenum) 6)


GLUTAPI void APIENTRY glutGameModeString(const char *string);

GLUTAPI int APIENTRY glutEnterGameMode(void);

GLUTAPI void APIENTRY glutLeaveGameMode(void);

GLUTAPI int APIENTRY glutGameModeGet(GLenum mode);

#endif

```

```
#ifdef __cplusplus
```

```
}
```

```
#endif
```

```
#ifdef GLUT_APIENTRY_DEFINED
```

```
# undef GLUT_APIENTRY_DEFINED
```

```
# undef APIENTRY
```

```
#endif
```

```
#ifdef GLUT_WINGDIAPI_DEFINED
```

```
# undef GLUT_WINGDIAPI_DEFINED
```

```
# undef WINGDIAPI
```

```
#endif
```

```
#ifdef GLUT_DEFINED__CDECL
```

```
# undef GLUT_DEFINED__CDECL
```

```
# undef __cdecl
```

```
#endif
```

```
#ifdef GLUT_DEFINED__CRTIMP
```

```
# undef GLUT_DEFINED__CRTIMP
```

```
# undef _CRTIMP
```

```
#endif
```

```
#endif      /* __glut_h__ */
```

1800242 - GAM 1b Report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

61 / 100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

ok but oyu should allow other processes to make use of this time.

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

GRADING FORM: SMU GENERIC_GJ

**BENJAMIN
KIDDIE****60.50****RESEARCH-INFORMED LITERATURE**

Only one ref used but it is apposite - it is the games design patterns text we used in class. You could have worked more on the update task - check out the purpose of the update pattern and how it is integrated into the game loop.

5.5**KNOWLEDGE AND UNDERSTANDING**

The fps loop timer works but it could be more sophisticated . It should allow the update method to execute when there is time available. No real update process attempted as described in the assignment brief. Obj loader included- good but report doesn't explain the container data structures well.

12.75**ANALYSIS**



The code works and you have successfully designed and integrated the majority of patterns into the framework. Basic testing included. obj loader included - good. Tested with more than one obj file.

12.75

PRACTICAL APPLICATION AND DEP.



Basic 60fps Game loop created No update pattern incorporated into this. objloader integrated - good.

27

SKILLS FOR PROFESSIONAL PRAC.



Report structured. The code you placed in the appendix as requested - good but alot of this is library code you could have left out. Only one ref.

2.5