UG-Rehman, Abdul

IN2026

Games Technology - COURSEWORK

Contents

[Part I 2](#_Toc131637027)

[A start screen. 2](#_Toc131637028)

[Part II 5](#_Toc131637029)

[A high score table. 5](#_Toc131637030)

[Part III 16](#_Toc131637031)

[A demo mode. 16](#_Toc131637032)

[Additional Features 17](#_Toc131637033)

[Back Thrust. 17](#_Toc131637034)

# Part I

## A start screen.

To Implement start, screen I removed the code that created spaceship, asteroids and GUI containing lives and score from the Start method in the Asteroids.cpp class. This is what the method should look after removing the code:

Text

Description automatically generated

This method will start an empty game session with all the listeners needed for the game to run as well as animations.

Then I created two new GUI method in the asteroids.cpp to create the gui which will be displayed on the empty game session to provide instruction of how to start and quit the game.

Text

Description automatically generated

Then added the CreateStartScreenGUI method to the start method of the asteroids.cpp class. 

After making the empty session and adding the start screen Gui. The game window’s appearance looks like this:

Graphical user interface, text

Description automatically generated

To implement the start and exit I added another case to the OnSpecialKepPressed method of the asteroids.cpp class.

Text

Description automatically generated

I am using the down arrow key as the means to start the game. The code in the above snippet will create and add spaceship, asteroid objects which would start the game. It will also remove the start screen Gui and add the game Gui which display the lives and score of the player.

Text

Description automatically generated

The case above is to exit the game once the End Key is pressed.

There is one problem so far. When I run the game everything works fine except that if I press the down arrow key again once the game has started it will create another set of spaceship and asteroids.

To fix this issue I created a bool variable game start and set it to false. Changed the down arrow key case so it will only start one time because after the key is pressed the bool variable will be set to true meaning nothing happens if the key is pressed again. Here is the snippet of the down key pressed case:

Text

Description automatically generated

I added another break line after the if statement because if there is no break after the if statement it will close the game if the down key is pressed again.

# Part II

## A high score table.

In order to create a high score table, you must store the players name and their score in to a txt file.

Before I wrote any code I added the following libraries in the following file.

Asteroids.cpp:

#include <vector>

#include <algorithm>

#include <fstream>

Asteroids.h:

#include <map>

First, I created an empty txt file in the BIN directory.

Directory Path:



File:



Then I added Instruction for the player to type their name. so I added entername gui label in Asteroids.h



Then added the text and its properties in the CreateGUI method in Asteroids.cpp.

Text

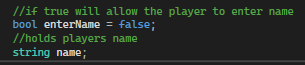
Description automatically generated

Then set visibility to true when the game is over. Add the userline code in the SHOW\_GAME\_OVER if statements in OnTimer method in Asteroids.cpp.

Text

Description automatically generated

After that I added a bool and string to store the player’s name. the bool enterName is used to allow the player to type their name.



The bool is changed to true when the game ends so the player is able to type. Add the underline code. To the On Timer method in the Asteroids.cpp.

Text

Description automatically generated

To make the enter name back to false I used the number 1 key. SO when the player is done typing their name they can press one to stop.



Once the player can type. We must register all the letter the player types. We can do that by adding cases for all the letters in the alphabet and storing them in a string name shows above. This code is added to the OnKeyPressed methods of the Asteroids.cpp.

//all the cases below will add whichever leter is pressed to a string to hold players name.

case 'a':

if (enterName) {

name += "a";

}

break;

case 'b':

if (enterName) {

name += "b";

}

break;

case 'c':

if (enterName) {

name += "c";

}

break;

case 'd':

if (enterName) {

name += "d";

}

break;

case 'e':

if (enterName) {

name += "e";

}

break;

case 'f':

if (enterName) {

name += "f";

}

break;

case 'g':

if (enterName) {

name += "g";

}

break;

case 'h':

if (enterName) {

name += "h";

}

break;

case 'i':

if (enterName) {

name += "i";

}

break;

case 'j':

if (enterName) {

name += "j";

}

break;

case 'k':

if (enterName) {

name += "k";

}

break;

case 'l':

if (enterName) {

name += "l";

}

break;

case 'm':

if (enterName) {

name += "m";

}

break;

case 'n':

if (enterName) {

name += "n";

}

break;

case 'o':

if (enterName) {

name += "o";

}

break;

case 'p':

if (enterName) {

name += "p";

}

break;

case 'q':

if (enterName) {

name += "q";

}

break;

case 'r':

if (enterName) {

name += "r";

}

break;

case 's':

if (enterName) {

name += "s";

}

break;

case 't':

if (enterName) {

name += "t";

}

break;

case 'u':

if (enterName) {

name += "u";

}

break;

case 'v':

if (enterName) {

name += "v";

}

break;

#

case 'w':

if (enterName) {

name += "w";

}

break;

case 'x':

if (enterName) {

name += "x";

}

break;

case 'y':

if (enterName) {

name += "y";

}

break;

case 'z':

if (enterName) {

name += "z";

}

break;

now the Player is able to type their name in the game. Lets write the name and score to the file.

I created to methods in the Asteroids.h file. WriteScore and ReadScore.

Text

Description automatically generated

In write score method we will be using fstream to write to the file. I added the name then space and then score to separate them. Here is the code for it.

Text

Description automatically generated

In Read Score method I first added the Title for the Score table to gui. Then to read the name and the score I added two variable int score and string pName to store the score and their respective player name.

Then I used fstream to open the file and added the first word to pNAme and second word to score. Now that the score and name are stored. I added the score to the vector and then added score and name to hashmap to keep track of the player name and their score. Then I sort the nums vector from the high to low score and then create another vector to store top 5 from the nums vector.

Then using the GUI label I displayed the score.

To display the name I used the map to find the player name using the score.

Before this code I initialised the maps and the Gui labels in the Asteroids.h.

Text

Description automatically generated



Here is the code.

//Reads the Top 5 Scores and Display Once the game ends;

void Asteroids::ReadScore() {

//create vector to store scores

vector<int> nums;

//cretae string to stores name

string pName;

//displays High score Table Title

mHighScoreTable = shared\_ptr<GUILabel>(new GUILabel("Top 5 Highest Scores"));

mHighScoreTable->SetHorizontalAlignment(GUIComponent::GUI\_HALIGN\_CENTER);

mHighScoreTable->SetVerticalAlignment(GUIComponent::GUI\_VALIGN\_MIDDLE);

shared\_ptr<GUIComponent> high\_score\_table\_component = static\_pointer\_cast<GUIComponent>(mHighScoreTable);

mGameDisplay->GetContainer()->AddComponent(high\_score\_table\_component, GLVector2f(0.5f, 0.9f));

try {

fstream file;

file.open("RecordScore.txt");

int num;

//first word in pName and second in num

while (file >> pName >> num) {

//adds the score to the nums vector

nums.push\_back(num);

//adds the name and the score

playerRecords[num] = pName;

}

//closes the file

file.close();

cout << "Success";

}

//catches exception

catch(int e){

cout << "Error No.: " << e << endl;

}

//sorts the vector high to low score

sort(nums.begin(), nums.end(), std::greater<int>());

//adds top 5 scores from the nums

vector<int> top\_five(nums.begin(), nums.begin() + min(5, (int)nums.size()));

//y position of the gui

float yPos = 0.8f;

//displays top 5 score

for (int i = 0; i < top\_five.size(); ++i) {

//displays score

mPrintScore = shared\_ptr<GUILabel>(new GUILabel(to\_string(top\_five[i])));

mPrintScore->SetHorizontalAlignment(GUIComponent::GUI\_HALIGN\_CENTER);

shared\_ptr<GUIComponent> print\_score\_component = static\_pointer\_cast<GUIComponent>(mPrintScore);

mGameDisplay->GetContainer()->AddComponent(print\_score\_component, GLVector2f(0.7, yPos));

//displays the player name by using map to find the name by score

mPrintName = shared\_ptr<GUILabel>(new GUILabel(playerRecords[top\_five[i]]));

mPrintName->SetHorizontalAlignment(GUIComponent::GUI\_HALIGN\_CENTER);

shared\_ptr<GUIComponent> print\_name\_component = static\_pointer\_cast<GUIComponent>(mPrintName);

mGameDisplay->GetContainer()->AddComponent(print\_name\_component, GLVector2f(0.3, yPos));

//decreases the yPos by 0.1f

yPos = yPos - 0.1f;

}

}

At last, I added another Timer variable in Asteroids.h which will read and write score table.



Then I added another if statement in the OnTimer method in Asteroids.cpp to write and read. And also make the enter name gui disappear.

Text

Description automatically generated

Then this is called when the player presses 1 after typing their name.

A screenshot of a computer

Description automatically generated with medium confidence

That’s it the High score table should be working!

One last thing I added was displaying the name when typing. I did that by displaying string name where the name is stored after every key is pressed.

add this underlined code to every letter case in the asteroids.cpp.

Text

Description automatically generated

Then add the code for create the pointers gui label to display the name. 

Then added the code to display mName where the player will be typing their name: this should be in the CreateGUI method in the Asteroids.cpp

Text

Description automatically generated

And set it to true when the game ends . underlined code. In asteroids.cpp in the OnTimer method.

Text

Description automatically generated

The implementation for the CreateNameGUI will be:

Text

Description automatically generated

Also make sure to add the method in the Asteroids.h



Here is what the game should look like when the game is over. You should be able to see the name you are typing.

A picture containing text

Description automatically generated

Then after you type your name and press the key 1. The high score table to appear on the screen.

A picture containing text

Description automatically generated

# Part III

## A demo mode.

In demo mode I wanted to have another type of spaceship that will play the game in the background once the game ends. So, I decided to choose the enemy\_fs from the filmstrip folder as my demo spaceship.

Then I added the enemy-fs from the filmstrip folder to the assets directory in the project and renamed it demospaceship\_fs.

The I created the classes for the demospaceship which are similar to spaceship classes. I added 2 files “DemoSpaceship.cpp” and “DemoSpaceship.h” in the Src directory of the project.

**Code for DemoSpaceship.h:**

#ifndef \_\_DEMOSPACESHIP\_H\_\_

#define \_\_DEMOSPACESHIP\_H\_\_

#include "GameUtil.h"

#include "GameObject.h"

#include "Shape.h"

class DemoSpaceship : public GameObject

{

public:

DemoSpaceship();

DemoSpaceship(GLVector3f p, GLVector3f v, GLVector3f a, GLfloat h, GLfloat r);

DemoSpaceship(const DemoSpaceship& s);

virtual ~DemoSpaceship(void);

virtual void Update(int t);

virtual void Render(void);

virtual void Thrust(float t);

virtual void Rotate(float r);

virtual void Shoot(void);

void SetSpaceshipShape(shared\_ptr<Shape> demo\_spaceship\_shape) { mDSpaceshipShape = demo\_spaceship\_shape; }

void SetThrusterShape(shared\_ptr<Shape> thruster\_shape) { mThrusterShape = thruster\_shape; }

void SetBulletShape(shared\_ptr<Shape> bullet\_shape) { mBulletShape = bullet\_shape; }

bool CollisionTest(shared\_ptr<GameObject> o);

void OnCollision(const GameObjectList& objects);

private:

float mThrust;

shared\_ptr<Shape> mDSpaceshipShape;

shared\_ptr<Shape> mThrusterShape;

shared\_ptr<Shape> mBulletShape;

};

#endif#pragma once

**Code for the DemoSpaceship.cpp:**

#include "GameUtil.h"

#include "GameWorld.h"

#include "Bullet.h"

#include "DemoSpaceship.h"

#include "BoundingSphere.h"

using namespace std;

// PUBLIC INSTANCE CONSTRUCTORS ///////////////////////////////////////////////

/\*\* Default constructor. \*/

DemoSpaceship::DemoSpaceship()

: GameObject("DemoSpaceship"), mThrust(0)

{

}

/\*\* Construct a spaceship with given position, velocity, acceleration, angle, and rotation. \*/

DemoSpaceship::DemoSpaceship(GLVector3f p, GLVector3f v, GLVector3f a, GLfloat h, GLfloat r)

: GameObject("DemoSpaceship", p, v, a, h, r), mThrust(0)

{

}

/\*\* Copy constructor. \*/

DemoSpaceship::DemoSpaceship(const DemoSpaceship& s)

: GameObject(s), mThrust(0)

{

}

/\*\* Destructor. \*/

DemoSpaceship::~DemoSpaceship(void)

{

}

// PUBLIC INSTANCE METHODS ////////////////////////////////////////////////////

/\*\* Update this spaceship. \*/

void DemoSpaceship::Update(int t)

{

// Call parent update function

GameObject::Update(t);

}

/\*\* Render this spaceship. \*/

void DemoSpaceship::Render(void)

{

if (mDSpaceshipShape.get() != NULL) mDSpaceshipShape->Render();

// If ship is thrusting

if ((mThrust > 0) && (mThrusterShape.get() != NULL)) {

mThrusterShape->Render();

}

GameObject::Render();

}

/\*\* Fire the rockets. \*/

void DemoSpaceship::Thrust(float t)

{

mThrust = t;

// Increase acceleration in the direction of ship

mAcceleration.x = mThrust \* cos(DEG2RAD \* mAngle);

mAcceleration.y = mThrust \* sin(DEG2RAD \* mAngle);

}

/\*\* Set the rotation. \*/

void DemoSpaceship::Rotate(float r)

{

mRotation = r;

}

/\*\* Shoot a bullet. \*/

void DemoSpaceship::Shoot(void)

{

// Check the world exists

if (!mWorld) return;

// Construct a unit length vector in the direction the spaceship is headed

GLVector3f spaceship\_heading(cos(DEG2RAD \* mAngle), sin(DEG2RAD \* mAngle), 0);

spaceship\_heading.normalize();

// Calculate the point at the node of the spaceship from position and heading

GLVector3f bullet\_position = mPosition + (spaceship\_heading \* 4);

// Calculate how fast the bullet should travel

float bullet\_speed = 30;

// Construct a vector for the bullet's velocity

GLVector3f bullet\_velocity = mVelocity + spaceship\_heading \* bullet\_speed;

// Construct a new bullet

shared\_ptr<GameObject> bullet

(new Bullet(bullet\_position, bullet\_velocity, mAcceleration, mAngle, 0, 2000));

bullet->SetBoundingShape(make\_shared<BoundingSphere>(bullet->GetThisPtr(), 2.0f));

bullet->SetShape(mBulletShape);

// Add the new bullet to the game world

mWorld->AddObject(bullet);

}

bool DemoSpaceship::CollisionTest(shared\_ptr<GameObject> o)

{

if (o->GetType() != GameObjectType("Asteroid")) return false;

if (mBoundingShape.get() == NULL) return false;

if (o->GetBoundingShape().get() == NULL) return false;

return mBoundingShape->CollisionTest(o->GetBoundingShape());

}

void DemoSpaceship::OnCollision(const GameObjectList& objects)

{

mWorld->FlagForRemoval(GetThisPtr());

}

After adding this code which is basically the same code as the spaceship class because it will perform the same actions but, in this instance, it will act like an ai and perform action automatically.

To make it act like an ai we will have to compare the positions of the demospaceship and asteroids on the map. If an asteroid is in range of x the demospaceship will shoot.

To get the asteroids positions I created a vector to hold shared\_ptr<GameObject> in Asteroids.h



Then I cleared the vector at the start of the createasteroids method and pushback the asteroid pointer to the vector. Createdasteroids is in the asteroids.cpp.

Text

Description automatically generated

Then I created a method for creating the demospaceship.

Code for Asteorids.h:

Graphical user interface, text

Description automatically generated



Code for Asteorids.cpp;

Text

Description automatically generated

Then I added the createdemospaceship to the SHOW\_GAME\_OVER if statement in the OnTimer method of the asteroids.cpp:

Text

Description automatically generated

Then I created to OnTimer if statements to reset the demospaceship when destroyed and another if statement for starting the demo mode and implement the ai behaviour.

First I added two uint to store the values for the ontimer method:

A screenshot of a computer

Description automatically generated with medium confidence

Code for the Start\_Demo\_Mode:

Text, chat or text message

Description automatically generated

Explanation: First I set the thrust and the rotation of the demospaceship and the store the position of the demo spacespace in the GLfloat array where I am store the x, y and z pos of the demo spaceship. Then I set the distance in which the demospaceship will shoot if the asteoird enters that parameter.

Then I loop through the vector which contains all the pointers to the asteoid gameobject. Then create glfloat array and store the x,y and z. then calculated the distance between the ship and the asteroid by subtracting the x,y and z pos of both object and store it in the dx,dy and dz.

Then I perform a sqrt(dx\*dx + dy\*dy + dz\*dz) to calculate the distance.

Then finally checking if the distance is less than or equal to the range the demospaceship will shoot.

Then I call this if statement on a recursive call every 8 millisecond.

Code for Reset\_Demo\_Spaceship:

Text

Description automatically generated

Explanation: This just resets the demo spaceship and adds to the game world.

One last thing I added was to call the RESET\_DEMO\_SPACESHIP once its destroyed. I did this by added the following code in the OnObjectRemoved method of the asteroids.cpp:

Text

Description automatically generated

# Additional Features

## Back Thrust.

Code for OnSpecialKeyPressed method in asteroids.cpp:

Text

Description automatically generated

Code for OnSpecialKeyReleased method of the asteroids.cpp:

A screenshot of a computer

Description automatically generated with medium confidence