

# GAMES TECHNOLOGY - COURSEWORK

UG-Rehman, Abdul


IN2026

## Contents

Part I .....	2
Part II .....	5
Part III .....	14

## Part I

Implement either:

- a) A start screen 
- b) Asteroids that split into smaller asteroids when hit

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:

```

/** Start an asteroids game. */
void Asteroids::Start()
{
    // Create a shared pointer for the Asteroids game object - DO NOT REMOVE
    shared_ptr<Asteroids> thisPtr = shared_ptr<Asteroids>(this);

    // Add this class as a listener of the game world
    mGameWorld->AddListener(thisPtr.get());

    // Add this as a listener to the world and the keyboard
    mGameWindow->AddKeyListener(thisPtr);

    // Add a score keeper to the game world
    mGameWorld->AddListener(mScoreKeeper);

    // Add this class as a listener of the score keeper
    mScoreKeeper.AddListener(thisPtr);

    // Create an ambient light to show sprite textures
    GLfloat ambient_light[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat diffuse_light[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient_light);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse_light);
    glEnable(GL_LIGHT0);

    Animation *explosion_anim = AnimationManager::GetInstance().CreateAnimationFromFile("explosion", 64, 1024, 64, 64, "explosion.fs.png");
    Animation *asteroid_anim = AnimationManager::GetInstance().CreateAnimationFromFile("asteroid1", 128, 8192, 128, 128, "asteroid1.fs.png");
    Animation *spaceship_anim = AnimationManager::GetInstance().CreateAnimationFromFile("spaceship", 128, 128, 128, 128, "spaceship.fs.png");

    // Add a player (watcher) to the game world
    mGameWorld->AddListener(mPlayer);

    // Add this class as a listener of the player
    mPlayer.AddListener(thisPtr);

    // Start the game
    GameSession::Start();
}

```

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.

```

void Asteroids::CreateStartScreenGUI() {
    // Create a new UILabel and wrap it up in a shared_ptr
    mStartGame = make_shared<UILabel>("Press \"DOWN ARROW KEY\" To Start");
    // Set the vertical alignment of the label to GUI_VALIGN_BOTTOM
    mStartGame->SetVerticalAlignment(GUIComponent::GUI_VALIGN_BOTTOM);
    // Add the UILabel to the GUIComponent
    shared_ptr<GUIComponent> start_game_component = static_pointer_cast<GUIComponent>(mStartGame);
    mGameDisplay->GetContainer()->AddComponent(start_game_component, GLVector2f(0.18f, 0.6f));

    // Create a new UILabel and wrap it up in a shared_ptr
    mExitGame = make_shared<UILabel>("Press \"End KEY\" To Exit");
    // Set the vertical alignment of the label to GUI_VALIGN_BOTTOM
    mExitGame->SetVerticalAlignment(GUIComponent::GUI_VALIGN_BOTTOM);
    // Add the UILabel to the GUIComponent
    shared_ptr<GUIComponent> exit_game_component = static_pointer_cast<GUIComponent>(mExitGame);
    mGameDisplay->GetContainer()->AddComponent(exit_game_component, GLVector2f(0.25f, 0.2f));
}

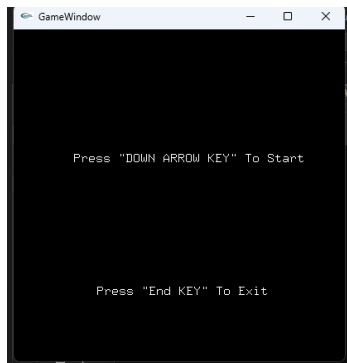
void Asteroids::RemoveStartScreenGUI() {
    mStartGame->SetVisible(false);
    mExitGame->SetVisible(false);
}

```

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

```
// Creates a start screen gui which displays option to start and exit the game
CreateStartScreenGUI();
```

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



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

```
case GLUT_KEY_DOWN:
    // Create a spaceship and add it to the world
    mGameWorld->AddObject(CreateSpaceship());
    // Create some asteroids and add them to the world
    CreateAsteroids(10);
    //removes start screen gui
    RemoveStartScreenGUI();
    //Create the GUI
    CreateGUI();
    break;
```

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.

```
case GLUT_KEY_END:
    Stop();
    break;
```

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:

```
case GLUT_KEY_DOWN:
    if (!gamestart) {
        // Create a spaceship and add it to the world
        mGameWorld->AddObject(CreateSpaceship());
        // Create some asteroids and add them to the world
        CreateAsteroids(10);
        //removes start screen gui
        RemoveStartScreenGUI();
        //Create the GUI
        CreateGUI();
        //set the game start bool to true
        gamestart = true;
        break;
    }
    break;
```

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

Implement either:

- a) A high score table 
- b) A bonus/power-up system

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:

IN2026 Coursework Code > BIN

File:

 RecordScore	29/03/2023 14:58	Text Document	1 KB
---	------------------	---------------	------

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

```
shared_ptr<GUILabel> mEnterName;
```

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

```
// Create a new UILabel and wrap it up in a shared_ptr
mEnterName = shared_ptr<UILabel>(new UILabel("Please Type Your Name - Press 1 When Done"));
// Set the horizontal alignment of the label to GUI_HALIGN_CENTER
mEnterName->SetHorizontalAlignment(GUIComponent::GUI_HALIGN_CENTER);
// Set the vertical alignment of the label to GUI_VALIGN_MIDDLE
mEnterName->SetVerticalAlignment(GUIComponent::GUI_VALIGN_BOTTOM);
// Set the visibility of the label to false (hidden)
mEnterName->SetVisible(false);
// Add the UILabel to the GUIContainer
shared_ptr<GUIComponent> play_again_component
    = static_pointer_cast<GUIComponent>(mEnterName);
mGameDisplay->GetContainer()->AddComponent(play_again_component, GLVector2f(0.5f, 0.1f));
```

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.

```
if (value == SHOW_GAME_OVER)
{
    spaceshipAlive = false;
    mGameOverLabel->SetVisible(true);
    mScoreLabel->SetVisible(false);
    mLivesLabel->SetVisible(false);
    mEnterName->SetVisible(true);
}
```

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.

```
//if true will allow the player to enter name
bool enterName = false;
//holds players name
string 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.

```
if (value == SHOW_GAME_OVER)
{
    spaceshipAlive = false;
    mGameOverLabel->SetVisible(true);
    mScoreLabel->SetVisible(false);
    mLivesLabel->SetVisible(false);
    mEnterName->SetVisible(true);
    enterName = true;
}
```

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.

```
//stop player from typing their name
case '1':
    enterName = false;
```

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 letter 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 two methods in the Asteroids.h file. WriteScore and ReadScore.

```
//writes the players name and their score to the file
void WriteScore();

//reads the players name and add it to the score.
void ReadScore();
```

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.

```
//Writes the Current Score to the File to keep record of scores.
void Asteroids::WriteScore() {
    int score = mScoreKeeper.getmScore();
    cout << "WriteScore" << endl;
    try {
        //opens the file
        fstream file("RecordScore.txt", ios_base::out | ios_base::app);
        //write name score to the file
        file << name << " " << score << endl;
        //close the file
        file.close();
    }
    catch (int e)
    {
        cout << "Error number " << e << endl;
    }
}
```

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.

```
//High Score Table Labels
shared_ptr<GUILabel> mHighScoreTable;
shared_ptr<GUILabel> mPrintScore;
shared_ptr<GUILabel> mPrintName;
```

```
//keep the players name and their score
map<int,string > playerRecords;
```

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;

    //create string to stores name
    string pName;

    //displays High score Table Title
```

```
mHighScoreTable = shared_ptr<UILabel>(new UILabel("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<UILabel>(new
UILabel(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<UILabel>(new
UILabel(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.

```
const static uint SHOW_HIGH_SCORE_TABLE = 3;
```

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.

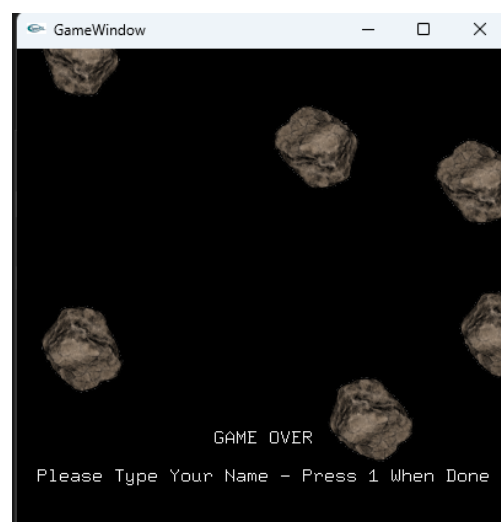
```
if (value == SHOW_HIGH_SCORE_TABLE) {
    WriteScore();
    ReadScore();
    mEnterName->SetVisible(false);
}
```

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

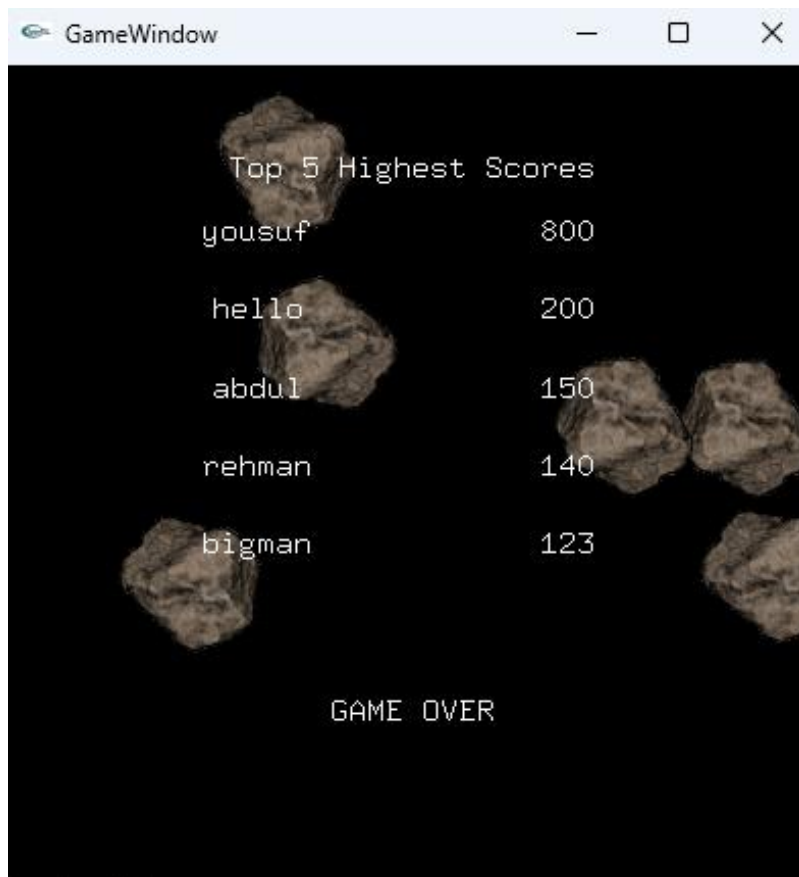
```
//stop player from typing their name
case '1':
    enterName = false;
    SetTimer(500, SHOW_HIGH_SCORE_TABLE);
    break;
```

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

Here is what the game should look like when the game is over.



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



## Part III

Implement either:

- a) A demo mode, or
- b) An alien spaceship