**IN2026 Coursework**

**Part 1: Start Screen**

Objectives:

* Create screen with title and key prompt using GUILabels.

An alternative would be to create a “texture”/image that is on the display at first before transitioning to the game world.

Outline of changes:

* Two new labels in Asteroids.h/.cpp
* One new Key press implementation
* Set initial visibility of lives and scores labels to false.

Implementation:

Created two new GUILabels alongside the already existing ones in Asteroids.h.

shared\_ptr<GUILabel> mStartScreenTitle;

shared\_ptr<GUILabel> mStartScreenLabel;

Next, I created the GUI.

First, I set the visibility for the score and lives labels to false, this is to make sure they are not present on the start screen.

mScoreLabel->SetVisible(false);

mLivesLabel->SetVisible(false);

Then I made the GUI components for the start screen: the Title and Key prompt in Asteroids::CreateGUI.

Text

Description automatically generated

The visibility of these is automatically true and don’t need to be set to false as they should be present on the initial display.

Finally, I made it so to start the game, the player needs to press the enter key. I did this by expanding the Asteroids::OnKeyPressed method.

switch (key)

{

case ' ':

mSpaceship->Shoot();

break;

**case 13: // Enter**

**mStartScreenTitle->SetVisible(false);**

**mStartScreenLabel->SetVisible(false);**

**mLivesLabel->SetVisible(true);**

**mScoreLabel->SetVisible(true);**

**mGameWorld->AddObject(CreateSpaceship());**

**CreateAsteroids(10);**

default:

break;

}

The tricky part here for me was seeing what could/should be in Asteroids::OnKeyPressed and what should remain in Asteroids::Start.

In the end, the code for creating the spaceship and asteroids on screen were moved to the key pressed method – this is so that nothing besides the title and key prompt are on screen for the start.

The result looks like this:

Graphical user interface

Description automatically generated with medium confidence

**Part 2: Power-Up System**

Objectives:

* Implement a power-up (Extra Lives)
* Update collisions.

Outline of changes:

* One new Header file and accompanying cpp.
* Changes to collision throughout the game. (e.g., bullets)
* New sprite
* New listener method for player listener.

Implementation:

First, I made LifeUp.h.

Text

Description automatically generated

Followed by LifeUp.cpp (Note: I did use tutorial 2 as a base for this)

Text

Description automatically generated

Next, I moved onto making changes in Asteroids.h and .cpp.

In Asteroids.h, I put a method to create the new object in game world.

void CreateLifeUp(const uint num\_LifeUp);

In Asteroids.cpp, I added: #include "LifeUp.h", so that I can define the method.

Text

Description automatically generated

I can test this by putting CreateLifeUp(1) in Asteroids::Start(). This is only for testing purposes, by putting it in start I am making it so that is it visible on the start screen.

Diagram

Description automatically generated with medium confidence

When I tried running the whole game, I noticed an issue with asteroid collisions, if an asteroid ran into the new object, it gets destroyed, but the spaceship does not.

So now I looked to make changes to other object collisions to see how other objects should interact with items.

To fix the bug with asteroid collisions, I changed this line:

if (GetType() == o->GetType()) return false;

to:

if (o->GetType() != GameObjectType("Bullet") || o->GetType() != GameObjectType("Spaceship")) return false;

However, whilst this did make it so that asteroids did no get destroyed by the new object, it made it so that previous collisions with bullets and the spaceship did not work. To fix this I swapped out the || (OR) for && (AND) which did resolve the issues.

Bullet needed to be altered so that they actually destroy the new object. (using && from before) (Bullet::CollisionTest)

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

Using the OR operator here did not create bugs, I am not sure why this is.

The, next thing to do is to give the new object an effect. To start, I went to player.h to add a new condition to void OnObjectRemoved and add a new method to send a message to all listeners.

New condition:

A screenshot of a computer

Description automatically generated with medium confidence

New method: (this is basically the equivalent of “FirePlayerKilled” for adding lives)

Text

Description automatically generated

As you can see, there is an error. This is caused by :

Text

Description automatically generated

To fix this I added: virtual void AddLives(int lives\_left) = 0; , to IPlayerListener.h.

I then had to define it in Asteroids.h and .cpp.

void AddLives(int lives\_left); -> Header file

Text

Description automatically generated

Next was to give the removal of the object an explosion. For this I Altered Asteroids::OnObjectRemoved.

Text

Description automatically generated

When I tried running it, it mostly worked – just 1 bug fix needed.

When destroying the object the player does only get 1 extra life, but the displayed life goes up by 2.

Fixed this by taking lives\_left += 1 from the top to the bottom of the AddLives() method.

Text

Description automatically generated

Two things left: sprite and implementation into game play loop.

Up until now, I have merely created the object at the start for the purpose of testing, but this isn’t how a power up would work in an actual gameplay loop. I can’t handle it the same way the game handles the actual asteroids as that would be very imbalanced (a lot of extra lives).

For this I looked into how the game handles it’s levels.

Text

Description automatically generated

The START\_NEXT\_LEVEL condition keeps track of the number of levels. So for my implementation I:

Text

Description automatically generated

Every 3 levels, a LifeUp will spawn.

The CreateLifeUp(1) in Asteroids::Start() is still there, comment and uncomment as needed.

For the sprite, I first looked up a free to use sprite, downloaded one and put it in the ASSET folder.

The sprite: lifeup.png (Source: https://opengameart.org/content/heart-pixel-art)

Second Power Up: Upgrade Bullet Life.

Objectives:

* Implement a power-up (Bullet life extension)

Outline of changes:

* One new Header file and accompanying cpp.
* New sprite
* New definition and method in Asteroids.h and .cpp
* Additions to Spaceship.h and .cpp.

Implementation:

First, I looked into the shoot() method in Spaceship.cpp. I confirmed that when shooting a new bullet is created. So to start, I added a new variable and method int the header and cpp file for spaceship.

Header:

int mBulletLife;

virtual void AddBulletLife(int time);

cpp:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

After that, I created the header and cpp files for the new power up. This was simple as I could just use the life up code as a template.

Text

Description automatically generated

Text

Description automatically generated

Next step is to include and implement in Asteroids.h and .cpp.

In Header file:

void CreateBulletLife(const uint num\_BulletLifeUp);

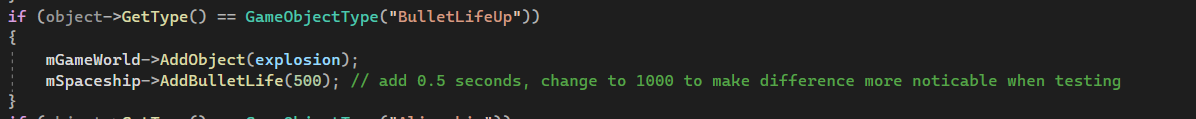
In cpp:

#include "BulletLifeUp.h"

Text

Description automatically generated

OnObjectRemoved():



For game play implementation, I thought it wouldn’t be a good idea to make this power up spawn every x level as it is a permanent statistical change. (Just like before, there is a line that spawns on session start for testing).

Text

Description automatically generated

Every time a new level is formed, there is a roll (0-100).

If the roll is greater than or equal to 80, the power up is spawned.

~20% chance.

Sprite: https://opengameart.org/content/missle-bullet

**Part 3: Alien Spaceship**

Objectives:

* Add an enemy into the game.
* Implement a computer control system.
* Sprite

Outline of Changes:

* One new header file and accompanying cpp.
* Sprite

Implementation:

Like the previous task, I started with the header and cpp file for the enemy. The behaviours is similar to asteroids (in terms of collisions), so I used asteroid.h and cpp (as well as tutorial 2, just like the previous task) as a base.

Text

Description automatically generated

Text

Description automatically generated

For collisions, the enemy will behave similarly to asteroids. Player must avoid contact whilst trying to destroy via shooting. So, the collisions for bullets and the spaceship had to be updated.

Spaceship: if (o->GetType() != GameObjectType("Asteroid") && o->GetType() != GameObjectType("Alienship")) return false;

Bullets: if (o->GetType() != GameObjectType("Asteroid") && o->GetType() != GameObjectType("LifeUp") && o->GetType() != GameObjectType("Alienship")) return false;

Next, asteroids.h and .cpp. In the header file:

void CreateAlienship(const uint num\_Alienship);

For cpp, include Alienship.h and define the method.

Text

Description automatically generated

To test I just used the method in Asteroids:Start, just like how I tested the extra lives. (the extra lives creation is commented out)

A picture containing shape

Description automatically generated

Since it’s not linked to much and can be done easily, I worked on implementing it into the game loop now. I did it the same way, by altering Asteroids::OnTimer:

Text

Description automatically generated

Every 4 levels, the enemy is spawned.

The total number is basically just (number from previous instance) + 1.

Now to make it so that levels with enemies only “end” when all asteroids AND enemies are destroyed.

Add to Asteroids.h: uint mEnemyCount;

Now all necessary changes in the .cpp file.

Text

Description automatically generated

In Asteroids::CreateAlienship: mEnemyCount = num\_Alienship;

Asteroids::OnObjectRemoved, After a lot of trial and error:

Text

Description automatically generated

Whilst I was testing I realised that I made a mistake when altering the level spawns. My intent was every 3 levels for extra lives, and every 4 levels for enemies. However, I forgot that mLevel starts at 0, so my implementation using the mod function makes it so that its: extra lives every 4 levels, enemies every 5 levels.

Alterations to fix this:

Text

Description automatically generated

Changes to Asteroids::ObjectRemoved to add explosions.

Text

Description automatically generated

For the sprite, I first looked up a free to use sprite, downloaded one and put it in the ASSET folder.

Next, to make destroying enemies adds to the score by adding to scorekeeper.h.

Text

Description automatically generated

The sprite: lifeup.png (Source: https://www.pngwing.com/en/free-png-ztrlr)

**GENERIC CHANGES**

* Added implementation for GLUT\_KEY\_DOWN for special key presses and release.

Text

Description automatically generated

* Game over when player lives reaches negative.

Text

Description automatically generated