IN2026: Games Technology – Re-sit Coursework

Coursework

The aim of this coursework is to extend the Asteroids game you worked on in the lab sessions to include new features. The source code for you to begin working with is available on Moodle. You should be familiar with much of this code from the labs.

This coursework has been split into three parts. Each part is worth the same number of marks, although the difficulty of the task increases. For each part, you will be expected to implement one of the features described. There is nothing to stop you implementing both features but you will only get marks for the first one of these (as presented in your documentation). You should begin by spending some time designing the features you want to add. Your coursework submission should include a design document for each feature you choose to implement as well as the code itself. It is important that you document your approach to implementing a new feature. Marks will be awarded for clear explanation of how you would implement a feature even if the feature is incomplete. An implementation on its own, no matter how great, will not be awarded full marks without a clear explanation of how it was implemented.

Coursework should be submitted through Moodle by 5pm on 25th August, 2021.

You may, alternatively, design a game of your own. The same range of marks is achievable whether you decide to do this or not, but some students will prefer to expand their portfolio through this option. If you decide to do this:

(i) Create a short description of your intended game and email it to me for approval at k.o.salako@city.ac.uk. You must have a copy of my approval email for inclusion with your documentation.

(ii) Make sure the complexity of your game extension is approximately equivalent to the Asteroids game provided and that you split your code into three identifiable features (for marking purposes).

Part I

Implement either:

a) A start screen, or

b) Asteroids that split into smaller asteroids when hit

Hints

To implement a start screen you will need to look at how the Asteroids class starts a game session and add some additional code to display a message and wait for a key to be pressed. Once a key has been pressed the spaceship can be created.

To implement asteroids that split into smaller asteroids when they are hit you will need to extend the current Asteroid class to add new asteroids when it explodes.

Part II

Implement either:

a) A high score table, or

b) A bonus/power-up system

Hints

The simplest possible implementation would display the high scores for a single session (this would not require the player to input their name). A more sophisticated implementation would save and load high scores from a text file (and would require players to input their name).

To implement bonuses and/or power-ups you will need to implement new subclasses of GameObject that, when they collide with a Spaceship, alter the way the Spaceship behaves in some way. You might want to implement a new type of listener that would allow the game to change its display. Types of bonuses and power-ups might include: extra lives, bullet upgrades, ship upgrades, shields, etc.

Part III

Implement either:

a) A demo mode, or

b) An alien spaceship

Hints

A demo mode should be designed to run while a start screen or high score table is being displayed. The demo mode should display a game being "played" by a computer controller. The simplest possible demo mode would control the spaceship using a random controller. A more complete implementation would use navigation and targeting algorithms to control the spaceship. The computer player doesn't have to be great at playing the game. If you try to make the behaviours of the computer player believable, don't forget to write an explanation of how you've tried to do it.

An alien spaceship is a natural extension to the bonuses/power-ups task in Part II. To implement an alien spaceship, you will need to implement a computer control system for the spaceship. Your alien spaceship should be able to navigate itself across the game world, try to avoid the player's bullets and fire at the player's spaceship. As with the demo mode, you don't have to make the control of the alien spaceship particularly intelligent, but you might want to try to make it believable. To make the implementation a bit easier you can make the alien spaceship invulnerable to collisions with asteroids.

Further Coursework Instructions

Documentation

You are not expected to produce UML diagrams, but half the marks are assigned to the documentation and a good explanation of your design may also influence marks for the code. The document should provide a clear explanation of what you **were trying to do**, **how you did it**, **what problems you came**

across on the way, and **the solutions**. **Use diagrams and screenshots to aid explanation**. Some *small* code snippets, or preferably pseudo-code may also be useful. If you are including code, only use small sections that *you* have written and *only* if they aid the text description.

Coding

Most of the code that you write should be in Asteroids rather than in GameEngine. If you find yourself having to edit GameEngine classes, it might be worth having a re-think to see if there is any easier/better way. Don't be shy of making new classes.

Make sure you have completed *all* the tutorials before you attempt the coursework. If you attempt to hack the code without understanding the principles demonstrated in the coursework you are likely to produce badly conceived messy work which will not get full marks.

Visual Studio

Use Visual Studio (preferably 2019) for your coursework.

Sprites

There are additional asteroid and alien spaceship sprites in a zip included in the project directory. Some of these might need editing to change the colours etc. It's also a good idea to add your own sprites for things like power-ups. Don't spend a long time on these. You can download images and re-arrange them into a filmstrip using a graphics package.

NOTE: Texture sizes need to be powers of two to be displayed by most graphics cards:
2, 4, 8, 16, 32, 64, 128, 256, 512, 1024.

Submission via Moodle

Make sure you "clean" your project before sending it or the file size will be too large for the on-line submission. You can find the "clean" menu option on the build menu. Also, to significantly reduce the size of your project, **you will need to delete the (hidden) ".vs" folder**, which is contained in the Asteroids directory. This folder contains a large ".db" folder with files that speed up compilation, but may be too bulky for the Moodle submission. Once you have done all of this cleaning up, make a copy of your project, and **make sure it still builds and runs**. Your final submission should be a zip file with the "clean" code and your design document.