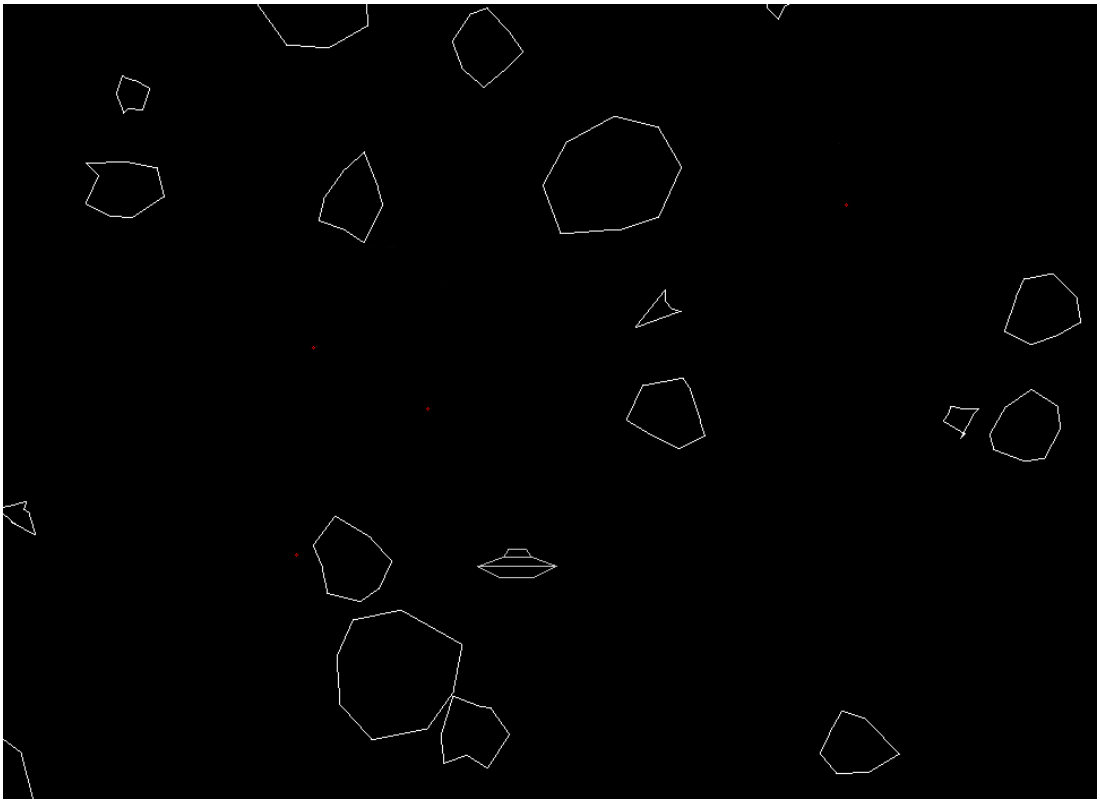
Beijing-Dublin International College

COMP2005J - Object Oriented Programming
Group Assignment: Asteroids



Assignment Details

Group size: Between 4 and 6
Due date: 15^{th} of December 2018 (No Exceptions)
Language: Solution must be completed in Java

Game Description

Asteroids is a classic arcade game, where a space ship moves through space destroying asteroids and occasional alien ships. The difficulty of the game increases as the levels progress. There are three types of asteroids:

- **Large**: These typically move very slowly and are large and therefore easier to shoot. When destroyed, two medium sized asteroids are created. These newly created asteroids will move in random directions and with random speeds but generally faster than the large asteroid was moving.

- **Medium**: These are a bit smaller than the large asteroids and a bit faster too. When destroyed, two small sized asteroids are created. These newly created asteroids will move in random directions and with random speeds but generally faster than the medium asteroid was moving.
- **Small**: These are again smaller and faster than the medium sized asteroids. When destroyed, no new asteroids are created.

Occasionally, an alien ship will appear and move through the screen from one side to the other. During this movement, it will fire at the players ship. The player can either avoid it or attempt to destroy it.

Movement

The players ship can perform 5 actions:

Rotate Right Rotate the ship in the clockwise direction

Rotate Left Rotate the ship in the anti-clockwise direction

Fire Fire a bullet in the direction the ship is currently pointing

Apply Thrust Add speed to the current motion in the direction the ship is currently pointing

Hyperspace Jump Disappear from current location and reappear in a new location on the screen (The new location should not be in contact with another object)

Momentum

When thrust is applied, the ship gains speed moving in the direction it is pointing. If we then stop thrusting, the ship will continue to move at the same speed in the same direction until we apply some thrust in the opposite direction.

This means that if I apply one second of thrust with the ship pointing up, it will need one second of thrust with the ship pointing down for it to stop. This is then further complicated when the ship is pointed at an angle.

When a ship, asteroid or bullet moves into the edge of the screen, it should then appear on the opposite side of the screen moving in the same direction. This means that objects will continue looping through the screen indefinitely. The only objects this is not the case for is bullets, these disappear automatically after travelling a set distance.

Levels

The game is based on a series of increasingly more difficult levels. Once each is cleared, the next begins automatically. In the first level there will be only one slow moving asteroid, in the second there will be two and so on.

While the game is being played, some info should be displayed on the screen such as the players current number of lives and their current score.

When the player is destroyed, either by hitting an asteroid or has been shot by the alien he must be placed back in the game safely. This means that it is either invincible for 2-3 seconds or is placed in a position that is calculated as safe.

After destruction one of the players lives is removed. On some versions versions of the game the player can regain lives by scoring 10000 points, but this is based on the number of points given for each destroyed asteroid.

Assessment

This section gives a breakdown of the approximate marking criteria for the assignment. The final marking scheme may vary slightly but will be relatively similar.

All submissions are to be completed by groups of at least 4 and no more than 6. Submissions by larger or smaller groups will **not be accepted** (unless I have given you permission).

The completed project should be submitted **only once** as a zip file and must contain the following:

- All of the source files (.java) associated with the project
- A PDF document listing the name and student number of everyone in the group and a short description of what each person completed in the project.

Marking Scheme

The marking scheme shown in table 1 is indicative only. This means that it may be changed at any time without notice. However, it will have approximately the same proportions and criteria. Criterion may be removed and additional criterion may be added.

It can however be used as a guide to the level required for the different parts of the assignment.

Example

To get an example of the how the asteroids game plays, go to the `http://www.freeasteroids.org/` and play the game there. There is one point that should be noted. This version of the game does not do the motion of the player correctly, as the player will slow down automatically when no thrust is applied. In your version of the game this should not be the case. The player should only slow down when thrust is applied in the opposite direction.

Item	Approx %	Fail (0 - 39)	Pass (40 - 69)	Excellent (70 – 100)
Design and Cohesion	30%	Poor use of classes such as only using a very small number of classes	OK use of classes but perhaps overusing them, such as the putting too many responsibilities into classes	Excellent use of classes, each class is used only for a sensible use and has good cohesion (right amount of sensible responsibilities)
Constants and Enumerated Types	10%	No use of constants or enumerated types in the project	Some use of constants and enumerated types, but other suitable values/variables were not encoded as constants or enums	Excellent use of enumerated types and enums to represent all values that do not change in the code.
Input	10%	User input is not completed	User input is completed but not all actions are implemented	User input is completed and all actions are implemented
Display	10%	No shapes are drawn on the game screen	Some of the shapes are drawn on the screen, or they are implemented using images loaded from files.	All required drawable objects are implemented in code by specifying the coordinates that individual pieces should be drawn at
Menu	10%	No menu or other options are shown, game moves straight into gameplay	Main menu is shown before gameplay starts	Game contains a main menu, hall of fame display (high scores) and an info screen showing the controls for the game
High Scores	5%	No High scores are recorded	Previous high scores are loaded from file, but new scores are not saved, or opposite	Previous high scores are loaded from a file and any new high scores are saved in the file when a game is completed
Motion	10%	Player motion is not implemented or not implemented well. E.g. does not continue once started or does not loop around the screen	Player motion is implemented well, but not perfectly. For example, bullet speed is not based based on the speed of the ship when they were fired	Player motion implemented perfectly
Asteroids	5%	New asteroids are not created after the destruction of a larger one	New asteroids of a smaller size are created when a medium or large asteroid is destroyed	New asteroids are created and they are given new directions and speeds based on the direction and speed of the original and a random element.
Alien	5%	The alien ship never appears	The alien ship appears, but always follows the same path	The alien ship appears and follows a (relatively) random path while shooting at the player.
Hyperspace	5%	Hyperspace not implemented	Hyperspace puts player in new random location without checking if it is currently safe	Hyperspace puts player in new location that is guaranteed to not safe (right now)

Table 1: Indicative Marking Scheme for Assignment