***COM1074 - In Course Assessment (Component 2 Resit)***

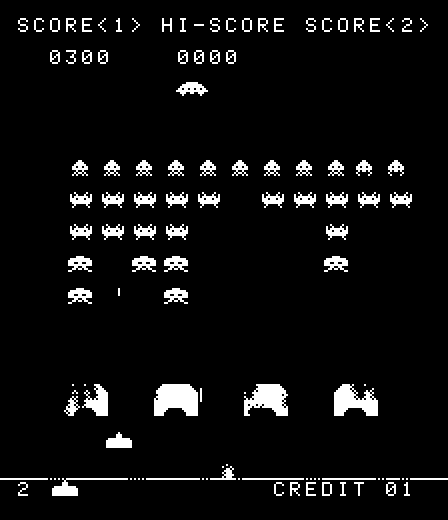
***“Space Invaders”***

***Submission deadline: 15th August 2014***

*Please read this ICA brief carefully so that you know exactly what is required of you*

***Background***

“Space invaders” is a classic computer game from 1978 developed by the Japanese company Taito and published by Midway. The concept is simple: There is a swarm of aliens attacking and you’re the only thing that stands in their way!



*Figure 1: "Space Invaders" (1978) by Taito.*

Fortunately the swarm of aliens move in a predictable pattern. Each row moves one step horizontally, for instance left, then when the left most alien reaches the left edge of the screen the swarm move down one step, then start moving to the right. Annoyingly the aliens will also drop missiles on to the player, losing the player one life if hit. If all the players’ lives are lost, then the game is over.

As you can see from *figure 1* above, there are four different types of aliens. Three of alien types are part of the swarm, and are of different sizes (the smallest requiring the most accuracy to shoot). Additionally, the forth alien type is a flying saucer that appears from time to time (randomly) and moves at speed across the top of the screen (at a fixed height), shooting it will give you a bonus score.

***For the purposes of this ICA, you will need to know that:***

1. The 3 different aliens have different point values:
   1. 5 points for the largest.
   2. 10 points for the medium sized alien.
   3. 15 points for the largest sized alien.
   4. 50 points for shooting the flying saucer.
2. If the bottom row of aliens reach the ground (defined by the Y axis position of the player), then the game is over and the player loses.
3. If all the aliens have been shot (including flying saucers), then the game is over the player has won.
4. A maximum of five missiles fired by the player can be on screen at once.
5. A maximum of two missiles dropped by the alien swarm can be on screen at once.
6. ***You’re not required to:***
   1. A full implementation of the space invaders game (see section “Requirements to pass the ICA”)
   2. Implement game credits.
   3. Implement protective shields.

***You must use the PRG library (PRG API) provided for the module, you will develop a basic version of the space invaders game.***

**PRG API Download from here:**

<https://scm-intranet.tees.ac.uk/users/u0012604/C++/>

**PRG API Documentation:**

<https://scm-intranet.tees.ac.uk/users/u0012604/C++/resources/api_doc/interactive/html/>

Please note that you are not being assessed on your graphic design skills, so please do not spend too much time producing images. To assist in this, you’ve been provided with a “sprite sheet (a bitmap image) containing the main game “sprites” [[1]](#footnote-1), you can download it from here:

<https://scm-intranet.tees.ac.uk/users/u0012604/C++/images/space_invaders_sprites.zip>

space_invader_sprites.bmp

*Figure 2: Space invader "sprite sheet”.*

*Table 1* (next page) provides you with the position of each game sprite in the “sprite sheet” image from *figure 2* (above), which you will need when blitting the sprite images onto the game canvas.

|  |  |  |
| --- | --- | --- |
|  | Position in source image (in pixels) | |
| Image | Bottom,Left | Top, Right |
| alien_1_pose_1.bmp | 1, 137 | 24, 154 |
| alien_1_pose_2.bmp | 33, 138 | 56, 154 |
| alien_2_pose_1.bmp | 4, 106 | 20, 122 |
| alien_2_pose_2.bmp | 36, 106 | 52, 122 |
| alien_3_pose_1.bmp | 1, 74 | 23, 90 |
| alien_3_pose_2.bmp | 33, 74 | 55, 90 |
| player.bmp | 1, 31 | 33, 59 |
| saucer.bmp | 1, 1 | 29, 15 |
| player_missile.bmp | 37, 31 | 50, 54 |
| alien_missile.bmp | 36, 1 | 51, 24 |
| Transparent colour: RGB(255,0,255) | | |

*Table 1: Sprite positions in the "sprite sheet"*

***Requirements to pass the ICA***

***Game Demo (85%)***

1. You need to identify suitable classes and structures. Appendix A provides a very incomplete UML diagram for the “Spaceship” class.
   * Demonstrate the use of polymorphic classes for the aliens
   * Note, not everything needs to be a class, for instance a 2D position is probably better as a struct as it has public access by default, making it a full blown class with “getters” and “setters” is a unnecessary for such a simple data type.
2. The swarm of aliens must move as described above.
3. Using the keyboard, the player must be able to move left and right and they must not be able to go beyond the edge of the screen.
   * The keys ‘z’ and ‘x’ will move the player’s ship left and right respectively.
   * Alternatively you may use the mouse to control the ship.
4. Using the keyboard (or mouse), the player should be able to fire a missile, which will move vertically until it either hits an alien or reaches the top of the screen.
   * The space bar (or mouse button) will be used to fire the player’s missiles.
5. Only very simple collision between missiles and targets needs to be implemented, the shape of target (player or alien) is irrelevant.
   * Perform a simple “box” test to check if the missile (for instance) has hit the bounding box (as indicated by the red square in *figure 3* below).

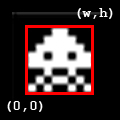


Figure 3: Bounding box indicating the simple collision

1. Use a vector or list container from the Standard Template Library (STL) to keep track of the position of each missile fired by the player.
   * You are required to identify additional uses of either the STL vector and/or list containers in the game.
2. Display the score on screen
   * Use either the function sprintf()[[2]](#footnote-2) or the ostringstream[[3]](#footnote-3) class to create a string representing the score.

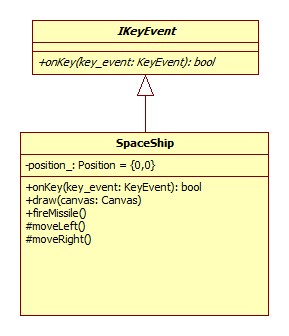
***Report (15%)***

Produce a brief report (maximum of 2000 words) that:

1. Includes a UML class diagram that details only the most important/relevant member functions (methods) and data members (attributes).
2. A brief discussion of the purpose of the class(es) presented in [1].
3. A discussion of any problems encountered during the development of the solution.
   * Where applicable, discuss how you overcame these problems.

***Appendix A – Spaceship class hierarchy***

The class diagram below offers a starting point for one of the classes you should implement. It is, however, incomplete so you may need to identify other data members and member functions appropriate to the needs of the object.



1. This is adapted from the sprites sheet created by “~SP534”, originally available from here: <http://sp534.deviantart.com/art/Space-Invaders-Sprite-Pack-151022197> (Account deactivated now) [↑](#footnote-ref-1)
2. Reference and example of using sprintf() can be found here: <http://www.cplusplus.com/reference/clibrary/cstdio/sprintf/> [↑](#footnote-ref-2)
3. A relevant example of the use of ostringstream can be found here: <http://net.pku.edu.cn/~course/cs101/resource/www.cppreference.com/cppsstream/all.html> [↑](#footnote-ref-3)