CSE310 Individual Project
Coursework Assignment Specification

2018/19 Semester 2
Bachelor Degree – Year 4

Module Code	Module Leader	Module Title
CSE310	**Hai-Ning Liang**	**Principles of Computer Games Design**

Coursework Assignment Number: **1 of 3**

Method of Working: **Individual**

Coursework Title: **Creating a 2D Game**

Percentage (%) Weighting: **10% of the overall module marks**

Date and time of publication: **10PM on Saturday 23 February 2019 (Week 1)**

Date and time for submission: **2PM on Monday, 15 April 2019 (Week 9)**

General Instructions

1. One copy of this assignment should be handed via the module **ICE** page at http://www.ice.xjtlu.edu.cn no later than the time and date shown above, unless an extension has been authorised by the module leader.

2. Before submission, each student must complete module coursework cover sheet obtainable from the module ICE page. This assignment is being marked by student name and id, please ensure that you complete the correct cover sheet.

3. Format of the coursework assignment submission:
   A **ZIP** file submitted via the ICE module page containing the deliverables outlined in the "**What to Submit**" section of the coursework assignment specification.

4. Use of unfair means:
   You are reminded of the University's Code of Practice on the Use of Unfair Means and that the work you submit for assignment should contain no section copied in whole or in part from any other source unless where explicitly acknowledged by means of proper citation.

5. Late penalties:
   For work submitted late the penalty is loss of **5%** marks per day. Work that is **5** or more days late will automatically be graded as **FAIL**, and no re-submission will be allowed.

The story so far…

In the year of 2040, you are in a Space Fighter, positioned in the geostationary orbit, assigned a job of collecting and grouping ionised particles into the solar power system of the international space station. Unexpectedly, space debris is mixed with ionised particles.

Scenario

You have been tasked with destroying the space debris and to group the associated ionised particles into the storage area. The complete implementation of the game should look like the following illustrative image (see Figure 1).

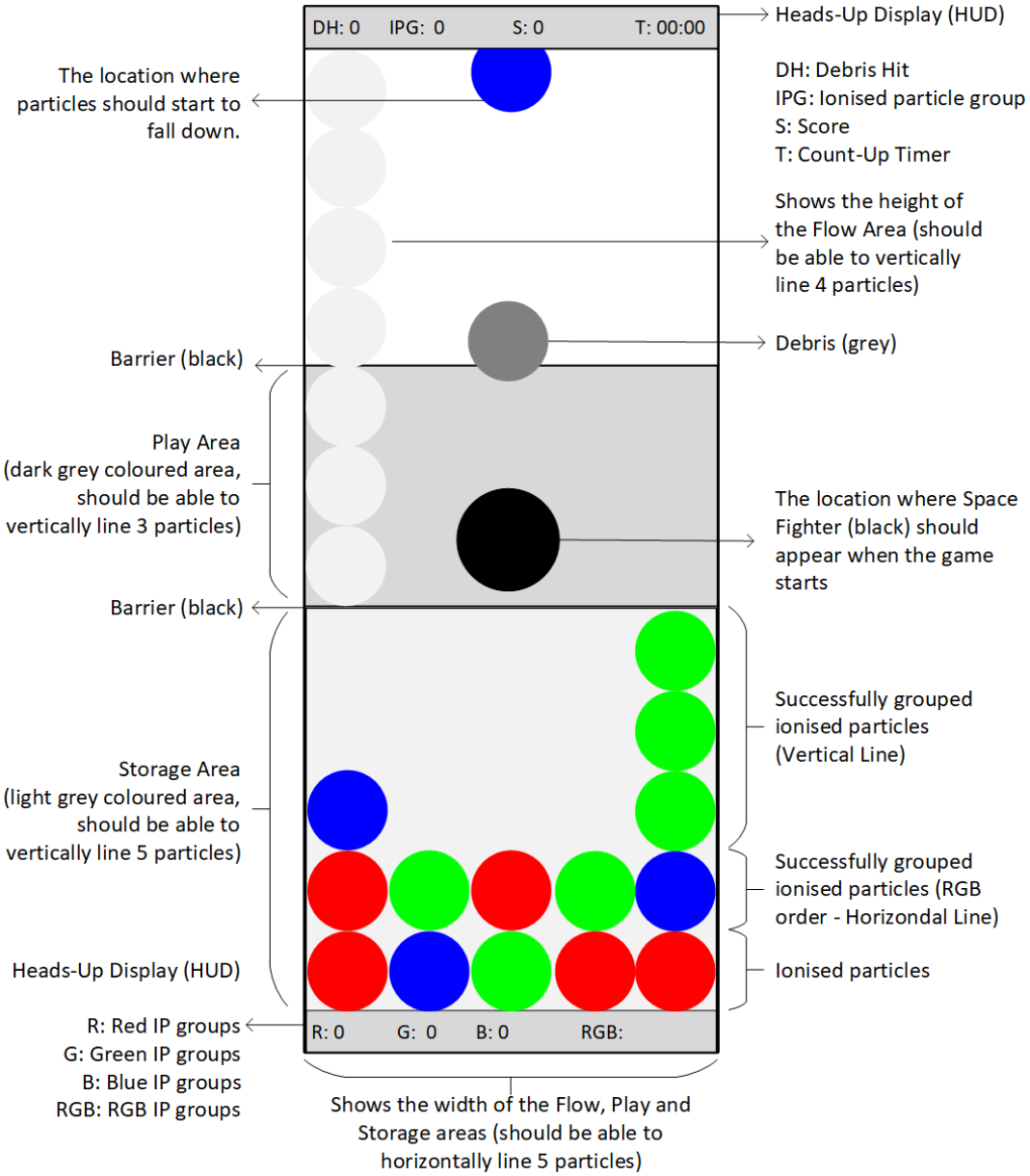

Figure 1: Illustrative image of the complete implementation of the game including details of the game elements/components

Game Play

The player should be able to direct the *"Space Fighter"* by easily moving in the dark grey coloured Play Area (See Figure 1 above). The black barriers should block the motion of the *"Space Fighter"* towards both the flow and storage areas. Thus, a collision with the black barriers will stop the motion of the *"Space Fighter"*.

There are two types of particles in the game, 1) ionised particles are coloured as *red*, *green*, and *blue* and 2) debris is coloured in *grey*. These particles should fall down towards the play area at a random sequence.

The player should be able to destroy the debris (grey coloured particles) and manipulate the ionised particles (red, green, and blue coloured) by pushing each one sideways using the space fighter, with the aim of creating vertical line of three similar coloured particles or a horizontal line of three different coloured particles (RGB order) as shown in the figure 1. When such a similar or different coloured line is created, it disappears, and any particles above the disappeared line should fall down.

The left-click on the *start* button (should be implemented by you) begins the game and a timer starts counting-up from 0 seconds.

The player will win the game when he/she successfully reaches 100 points. The player will lose when either the storage area is filled with non-lined ionised particles or filled with maximum of 5 debris.

Implementation

You have been asked to write your own code to implement the game using Unity and C#. The resources for the game should be created by you. The resources include the *particles*, *space fighter*, *user interfaces,* and *notification messages*. However, your implementation should follow the specification detailed below.

Play, Flow and Storage Areas

The following measurements should be used to implement the flow, play, and storage areas in your game (see Figure 1).

Location	**Height (Capable of storing)**	**Width (Capable of storing)**
Play Area	3 particles (vertically lined)	5 particles (horizontally lined)
Storage Area	5 particles (vertically lined)	5 particles (horizontally lined)
Flow Area	4 particles (vertically lined)	5 particles (horizontally lined)

Table 1: Specification of Flow, Play, and Storage Areas

Space Fighter

The *"Space Fighter"* (size should be 1.5 times particles) should appear at the bottom-centre of the play area, slightly above the black barrier when the game is started (see Figure 1). The black barriers must block the movement of the *"Space Fighter"*.

Player

The user/player should be able to play the game using the following keyboard and mouse controls:

Control	Function
Left-mouse click	Begins the game by starting the timer.
Right-Arrow	Moves *"Space Fighter"* towards the right side (move speed 3.0f)
Left-Arrow	Moves *"Space Fighter"* towards the left side (move speed 3.0f)
Up-Arrow	Moves *"Space Fighter"* upwards (move speed 3.0f).
Down-arrow	Moves *"Space Fighter"* downwards (move speed 3.0f).
Spacebar	Pause/resumes the entire game.

Table 2: Specification of Player Controls

Particles

There are four coloured particles namely *grey*, *red*, *green* and *blue* to be implemented in this game. The ionised particles are represented in *red*, *green* and *blue* colours, while the debris is characterized in *grey* colour using the following RGB codes.

Particles	Colour	RGB
Debris	Grey	128, 128, 128
Ionised Particle Red	Red	255, 0, 0
Ionised Particle Green	Green	0, 255, 0
Ionised Particle Blue	Blue	0, 0, 255

Table 3: Specification of Particles

The particles should start falling down from (at the speed of 2.0f) the top-centre (as shown in the figure 1) of the flow area in a random sequence when the game is started. One important condition is that the similar coloured particles should not be immediately following. The particles should fall down at the delay of 2 seconds. In other words, for every 2 seconds new particles should start to fall down in the mentioned location.

Debris: The debris should only be destroyed by colliding (or hitting) the space fighter at the bottom of the debris. Thus, debris cannot be destroyed by colliding from both the top and sideways.

Ionised Particles: The ionised particles should only be pushed freely using the space fighter each of the sideways. Thus, you cannot push the ionised particles from both the top and bottom sides.

Scoring

The scoring formula for the game should be built by counting the number of debris destroyed and by grouping similar and different coloured (RGB order) ionised particles together.

To destroy a single debris is worth of 5 points. The vertical line of grouping the similar coloured particles is worth of 10 points while the horizontal line of grouping the different coloured particles (RGB order) is worth 15 points.

Heads-up display (HUD)

There are two heads-up display (HUD) areas to be implemented in this game, one at the top, above the flow area and second at the bottom, below the storage area (See Figure 1). At the top, the head-up display area should display the number of *debris hit, ionised particle groups,* and *score* while the bottom HUD should show the details of successfully grouped *red, green, blue,* and *RGB particles* as specified in the figure 1.

Notification Messages

The win and lose messages should be displayed as a pop-up display message while freezing the game window (flow, play and storage areas) including head-up display components. There are 3 different messages should be implemented with the specified text colour.

Reason	Message Descriptions
To Win	Congratulations! You Won! *(Yellow text)*
To Lose – by filling with non-lined ionised particles	Opps! Ionised particles are not successfully lined in the storage! *(Orange text)*
To Lose – by filling with debris	Opps! Debris are filled in the storage! *(Black text)*

What to Submit

Your ZIP file should include these files:

- Your completed submission form (properly completed and with your signature). The submission form will be made available on ICE (submission forms with incorrect information certainly will affect your marks, so carefully complete the submission form). The submission form should be properly named as mentioned below.

- Your Unity scene file (*.unity*) and all other required source files. Make sure it can be open and runs properly on a **Windows** computer. The Unity scene file, and the project ZIP file should be properly named as mentioned below.

- Use your **Last Name**, **First Name**, and **Student Number** to name your Unity scene file (.unity), Submission Form and ZIP file—for example **LiangHaining999999** will be the name of the files module leader would be submitting, with 999999 being as his student number (any submissions with improper or incomplete file names certainly will affect your marks, so carefully name your files).

NOTE: To create ZIP files in Windows, see these two links:

- http://windows.microsoft.com/en-US/windows-8/zip-unzip-files
- https://support.microsoft.com/en-hk/help/14200/windows-compress-uncompress-zip-files

The end of the document
***