**Section 1 – Definition, Investigation and Analysis**

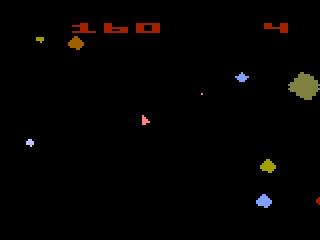
**Problem Definition**

Using the Python programming language, I will create a new and improved version of one of the original arcade games, Asteroids.

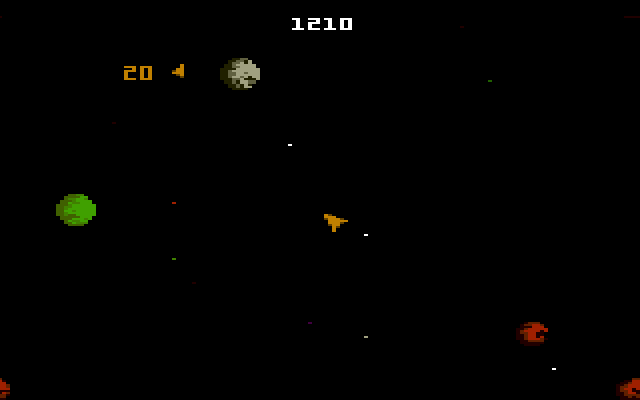
Asteroids is an arcade space shooter released in November 1979 by Atari, Inc. and designed by Lyle Rains, Ed Logg, and Dominic Walsh. Atari is a brand name that has been owned by several different companies since it was established in 1972. The original company was founded by Ted Dabney and Nolan Bushnell. Their video game consoles and arcade games helped pioneer the computer entertainment industry from the 1970s to the mid-1980s.

The objective of Asteroids is to collect as many [points](http://www.giantbomb.com/high-score/3015-260/) as possible by destroying all of the asteroids on the screen, and to avoid losing all your lives. A life will be lost if your spaceship either collides with an asteroid or is shot by a flying saucer. The space[ship](http://www.giantbomb.com/spaceship/3055-17/) appears as a small triangle, whose default position is the centre of the screen. It can rotate 360 degrees and move forward with the use of thrusters. If the spaceship goes over the edge of the screen it will appear on the opposite side of the screen. The ship possesses momentum and will drift for a time after its thrusters are deactivated.

Below are various screenshots of versions of the Asteroids game:



This is an example of one of the first ever asteroids games made. It was called 2600 Asteroids.



Another version of the Asteroids game was called [Atari 7800](https://atariage.com/7800/index.php?SystemID=7800) and was created in 1987. The difference between the two versions is that the asteroids have been made to look more like real asteroids so that the user gets more of a feel of the game.

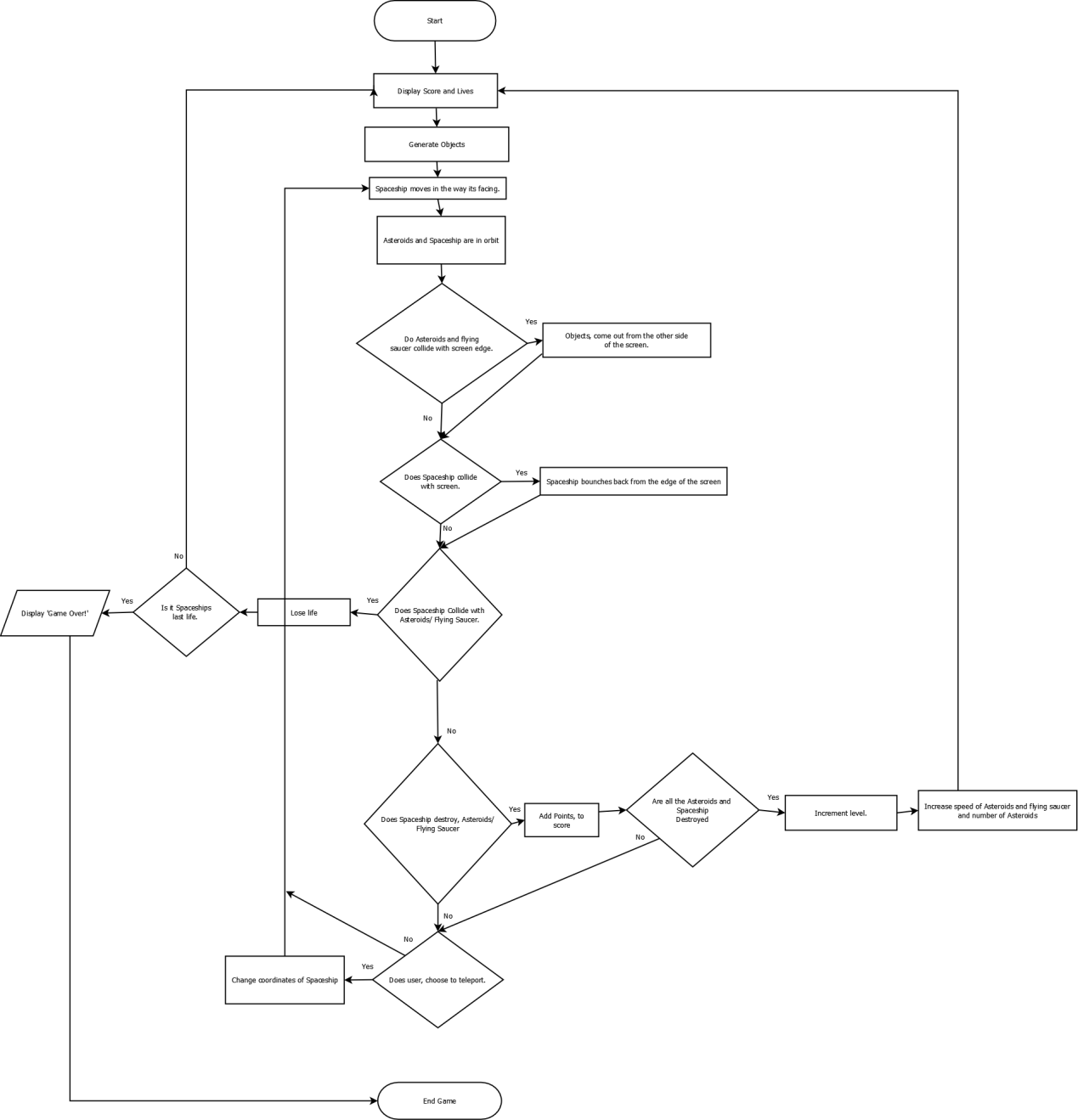


Recently Atari have taken their asteroids game and have revamped it completely. The objective of the game is still the same however they have changed the idea of being a spaceship into being a human with a weapon. The game is called Asteroids: Outpost and was released in 2015.

Even though the Asteroids game has been developed hugely, I will base my version on the original version and not the new and updated version of Asteroids.

**Current Systems**

A simple model of the basic gameplay system of my Asteroids game is below:

****

**The Client**

My client for this project is a teacher in a secondary school, who has requested the game for his new extra curriculum class, which will be studying computer games.

**Selection of User Group**

The end users for this project will be secondary school students as well as some teachers. As I cannot ask everyone in the whole school about my game, I will choose a small sample from fellow classmates, friends, family members and teachers. This sample will be asked what features that they would like to see in the game, and getting their feedback as the game is developed as their requirements may change once they have seen the game. I will also add features that I would like to see in the game.

**Investigation and Analysis**

I took 3 steps in identifying the requirements to create the game:

1. How are computer games made?
2. Internet research
3. Questionnaires

1: How are computer games made?

I found and took an online python-programming course, which built simple interactive games such as Pong, Blackjack and Asteroids. This taught me about the basics of python programming, event-driven programming and gameplay, as well as giving me ideas for programming the Asteroid game itself.

This can be found at: [**https://www.coursera.org/course/interactivepython2**](https://www.coursera.org/course/interactivepython2)

I also found a blog on writing the Asteroids game in less than 10 hours. Although this was in C++, the basic ideas surrounding the Asteroids game were all there.

This can be found at:

**https://katyscode.wordpress.com/2012/06/13/coding-challenge-write-asteroids-in-10-hours-or-less/**

2: Internet Research

Using the Internet I found different versions of the Asteroid game. From this I got ideas for different enhancements, which could be made to the different game.

One example of the sites that I looked at is:

**http://www.classicgaming.cc/classics/asteroids/game\_flash.php**

I also looked at other games such as space invaders for other ideas, which could be brought into the Asteroid game such as different levels/waves.

3: Questionnaires:

Using the knowledge I gained from the above, I produced several questionnaires about the requirements for my game. I then used these to ask my sample group a range of questions to find out what they would like to see in my game. I also asked my end client for his input.

These are the questions I asked in my first questionnaire:

* Have you ever played Asteroids before?
* What consoles have you played the Asteroids game on?
* What did you like about the game?
* What did you dislike about the game?
* Do you still play Asteroids? If not, when was the last time you played it?
* Did you like the layout of the game? If not, what did you dislike?
* Are there any new features you thought would make the game any better?
* Do you think the ideas: the ability to teleport from one place to another, the asteroids being made smaller by an attribute that flies in the air and the spaceship catches it, the spaceship being able to get a new life, are good ideas? If so what is your favourite?

**Initial Interview Results**

I have interviewed 5 students in year 13 and one teacher on their current experiences of the Asteroid game and have gathered the results. These results are below, together with the frequencies of the answers next to them.

From the results I have learnt a lot about the game from different perspectives. The interviews also help me as I can take these opinions and can use them for my own version of Asteroids. The students I asked were: Kevin Longe, Ben Gardner, Tom Gage, Gemma Yianni, Louis Cadburry Jenkins and the teacher I asked was Mr Tibble.

**Have you ever played Asteroids before?**

* Yes – (6)
* No – (0)

**What consoles have you played the Asteroids game on?**

* PC – (6)
* PS1 – (1)
* Arcade – (1)

**What did you like about the game?**

* The simplicity – (2).
* Difficulty was high – (2).
* Entertaining – (1)
* No deceleration on Asteroids – (1)

**What did you dislike about the game?**

* Very repetitive – (1)
* No end of the game –(1)
* Colour Scheme – (1)
* Nothing really changed – (1)
* Nothing – (1)
* Control System – (1)

**Do you still play Asteroids if not when was the last time you played it?**

* 5 Years Ago – (2)
* 7 Years Ago – (1)
* 8 Years Ago – (1)
* 2 Years Ago – (1)
* 3 Years Ago – (1)

**Did you like the layout of the game if not what did you dislike?**

* Liked – (2)
* Didn’t like how the Asteroids go through the sides – (1)
* More could be happening around the player – (1)
* Spaceship too small – (1)

**Are there any new features you thought would make the game any better?**

* Power ups – (3)
* More objects to shoot.
* A new object to shoot back at the spaceship.
* Ability to pick different spaceships at the start.

**Do you think the ideas: the ability to teleport from one place to another, the asteroids being made smaller by an attribute that flies in the air and the spaceship catches it, the spaceship being able to get a new life, are good ideas? If so what is your favourite?**

* Teleport – (4)
* New life – (2)
* Yes – (1)

**Analysis of Initial Interviews Results**

All 6 interviewers had played the Asteroids game and all of them had played it on the same console which is the PC which means that most of them didn’t go out of their way to actually purchase the game, they must of them just played it online for free. They all had different opinions about what they disliked about the game but mostly it came out that the interviewers thought that the game was quite repetitive and they thought that if there was something new in the game that changes the gameplay of the game it would improve it dramatically and so for this reason I have decided to concentrate on improving the game by making a difference to the gameplay. With my ideas I already had for the game I asked the interviewers and the majority of their response was towards teleportation and also the fact that the game doesn’t have different levels. I have decided to change this and include teleportation and different levels inside my new game. With the layout of the game most of them liked it but there were some new ideas about it and the best I liked was that the asteroids were going through the side.

With the initial interview results and my ideas I have decided to write the following requirements:

**Basic Requirements**

1. There is no ending to the game.
2. The player can rotate the spaceship.
3. The player can thrust the spaceship in different directions.
4. The player can direct shots at asteroids or flying saucers.
5. The player can activate a shield on the ship.
6. Each time an asteroid gets hit the asteroids decrease in size.

**Variable Features**

1. There are different levels of the game which you get too by destroying all the asteroids. Every level will get harder as well.
2. The Screen size is the same size as online and so quite large.
3. At each new level there will be more Asteroids to destroy.
4. The colours of the game have been determined by my interviewers.

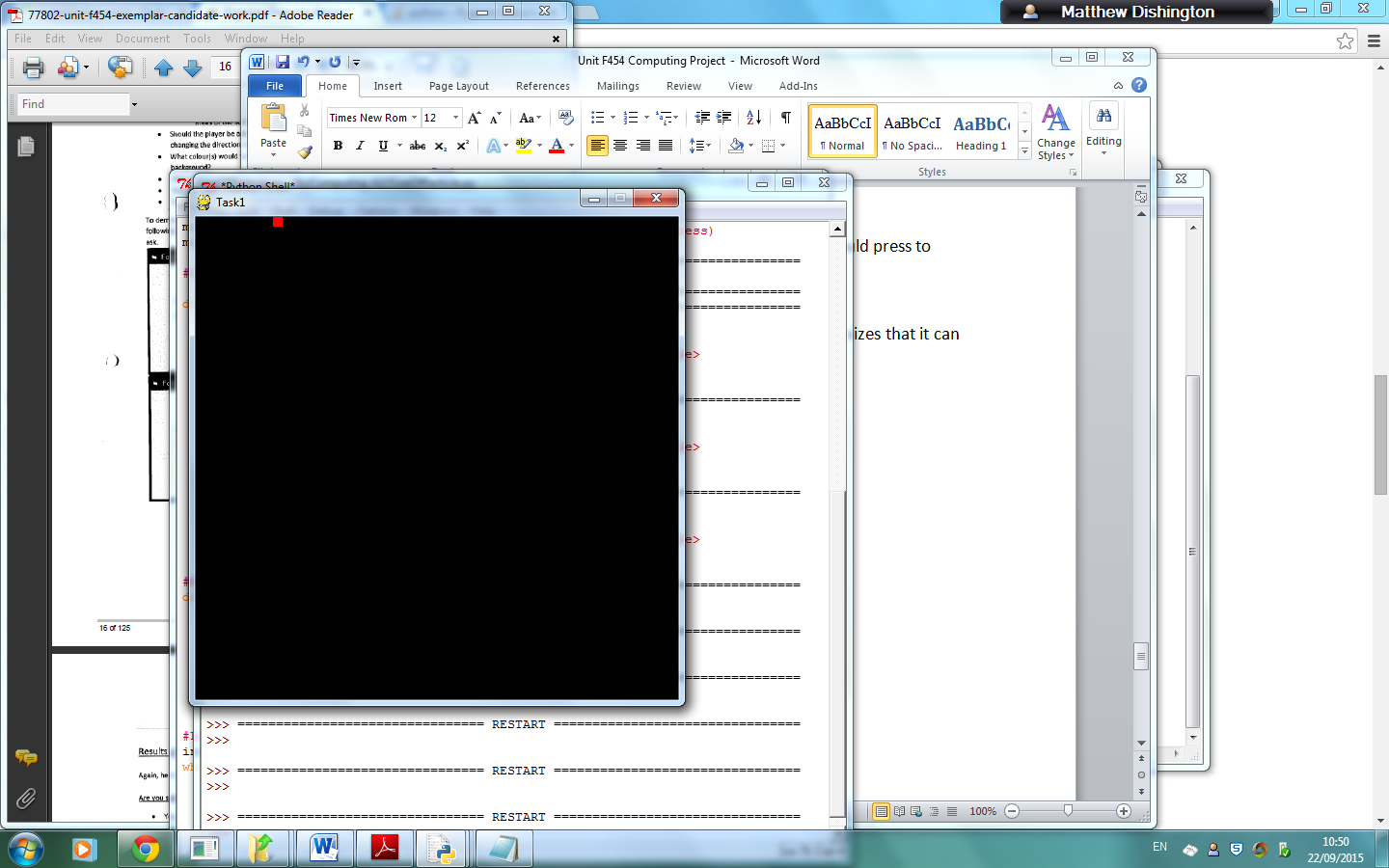
**Second Interview**

To decide whether or not I have the right variable features for my programme I will be asking the same interviewers if they agree with them and will ask their advice on some more specific topics then the first interview for example the size of the Asteroids.

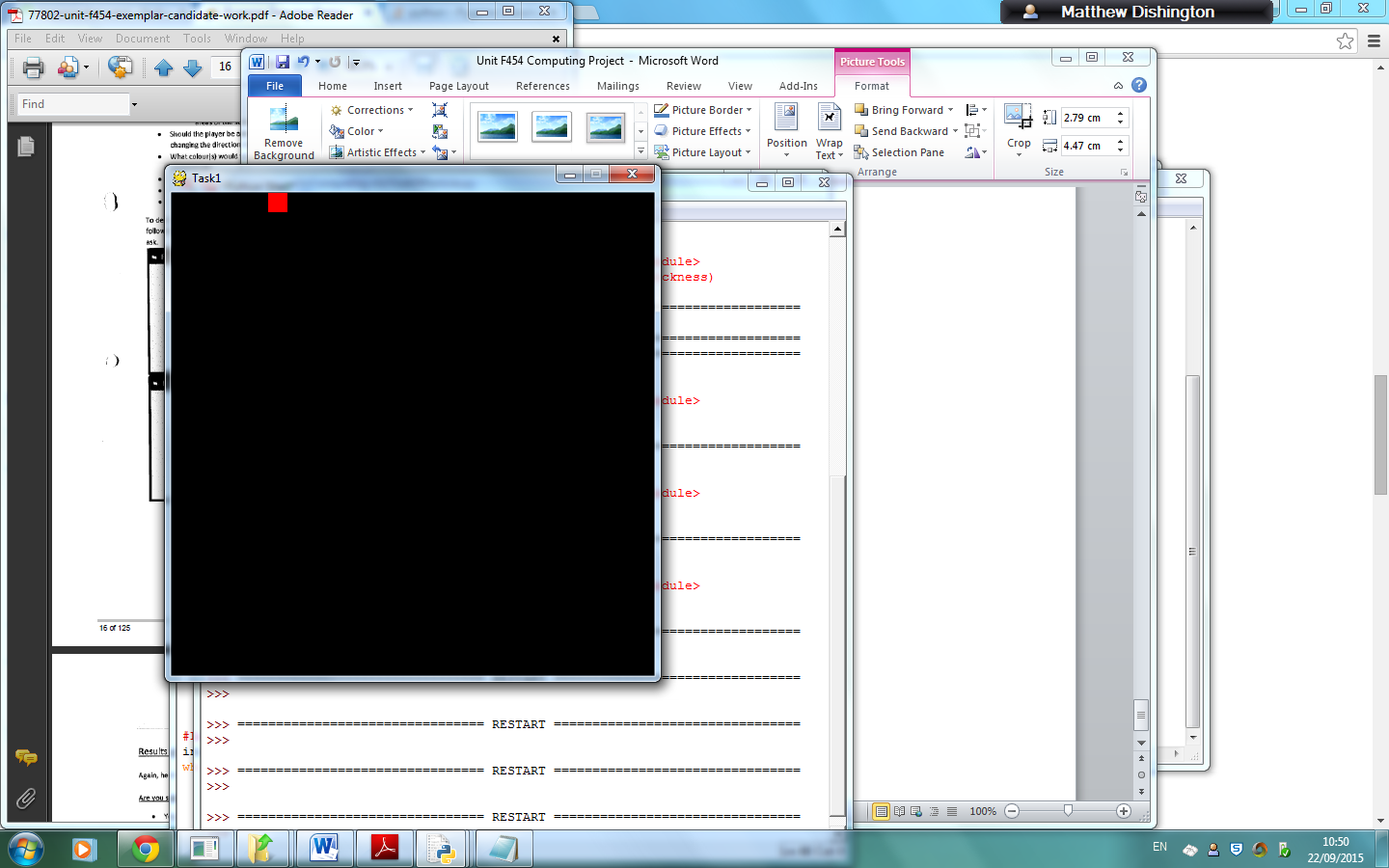
These are the questions I have decided to ask:

* Are you satisfied with all the variable features listed, if not which ones would you change or what would you add?
* How large would you recommend the screen size for the game to be?
  + - 700x400
    - 800x6 00
    - 800x800
    - Other
* As the player goes up the next level how many more Asteroids should there be?
* How large should the Asteroids be?
  + - Very Small (60x60)
    - Small (70x70)
    - Medium (80x80)
    - Large (90x90)
* How many Asteroids should there be in Level 1?
* How many Levels should there be?
* Should the player control the spaceship with the arrows or words on a keyboard?
* How large should the Spaceship be?
  + - Small 10x10
    - Medium 20x20
    - Large 30x30
* How large should the flying saucer be?
  + - Small 10x10
    - Medium 20x20
    - Large 30x30
* What colour would you like Asteroids, Spaceship and the flying saucers be?
* Should the player be able to lose the game by shooting themselves?
* How much faster should the Asteroids be?
  + - 10 milliseconds
    - 20 milliseconds
    - 30 milliseconds
    - Other
* What key on the keyboard do you think you should press to teleport?

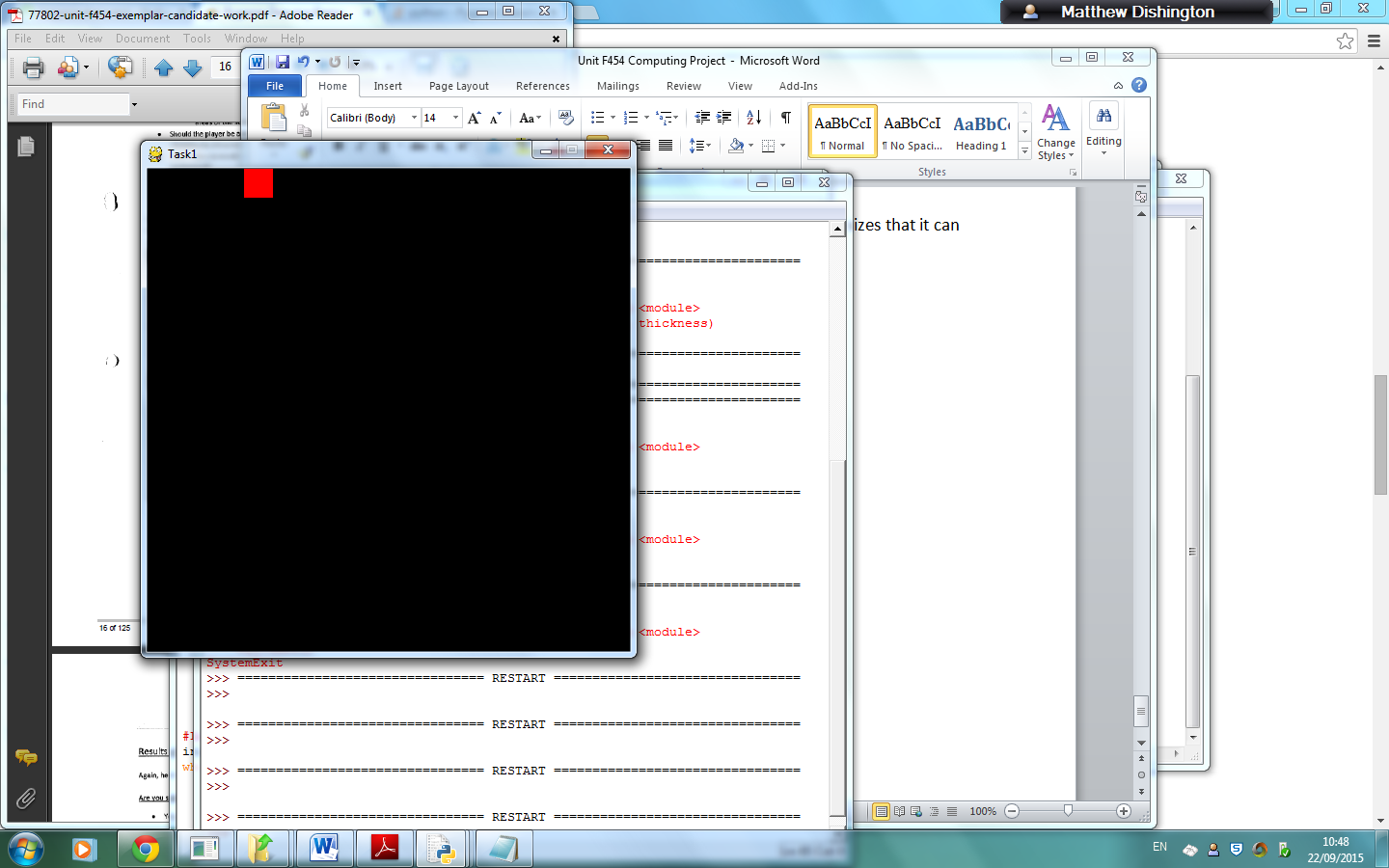
For the spaceship I will now show you all the different sizes that it can be:



Objects size: 10x10 pixels.



Object Size: 20x20 Pixels



Object Size: 30x30 Pixels

**Results of Second Interview**

**Are you satisfied with all the variable features listed, if not which ones would you change or what would you add?**

Yes – (4)

**How large would you recommend the screen size for the game to be?**

700x400 – (0)

800x600 – (1)

800x800 – (3)

**As the player goes up the next level how many more Asteroids should there be? At the first level there will be 5 asteroids to destroy.**

I think it should multiply – (1)

Increase by a 1/3 each time –(2)

1/3 or ¼ depending on how clustered it will make the screen – (1)

**How large should the Asteroids be?**

70x70 – (1)

80x80 – (2)

90x90 – (1)

**How many levels should there be?**

20 – (1)

50 – (2)

Infinity – (1)

**Should the player control the spaceship with the arrows or words on a keyboard? Or would you rather have the option for both?**

Both – (4)

**How large should the spaceship be?**

10x10 – (1)

20x20 – (1)

30x30 – (2)

**How large should the flying saucer be?**

30x30 – (4)

**What colour would you like the Asteroids, spaceship and flying saucers to be?**

Same – (2)

Prime colours – (1)

Blue, White and Yellow – (1)

**Should the player be able to lose the game by shooting themselves?**

Yes – (3)

No – (1)

**How much faster should the Asteroids be?**

20 milliseconds – (4)

**What key on the keyboard do you think you should press to teleport?**

T- (1)

P – (1)

Enter – (2)

**Analysis of Second Interview Results**

Everyone in my user group was satisfied with all the variable features that I decided and so I will keep them the same. With regards to the variable features I will go with user groups’ preferences to create a requirements specification.

**Requirements Specification**

**User Requirements**

1. **Design Requirements:**
2. The Screen Size – 700x700
3. Colour of Spaceship – White
4. Colour of Asteroids – Yellow
5. Colour of Flying Saucer – White
6. Size of Asteroids – 80x80
7. Size of Spaceship – 30x30
8. Size of Flying Saucer – 30x30
9. How many Asteroids at first level – 5
10. How many levels – 50
11. **Input Requirements:**
12. The player can rotate the spaceship using the arrow keys.
13. The player shoots bullets with the spacebar.
14. The player can thrust the spaceship by holding down the arrow keys.
15. The player can either start the game or can go to the instructions screen by clicking on either button.
16. The user can input there name for their high score.
17. The player can activate a shield on the ship.
18. The player can pause the game.
19. **Processing Requirements:**
20. The Asteroids and Spaceship will increase their speed by 20 milliseconds after every new level.
21. The number of Asteroids will increase by a 1/3 after every new level. In the first level there will 5 Asteroids.
22. The player goes onto the next level by destroying all the Asteroids.
23. The asteroids and the spaceship will be in orbit.
24. The program must know when the Asteroids have been hit.
25. When the object has reached the edge of the screen it will bounce back into the screen and so won’t go out.
26. The flying saucer will be shooting bullets at the spaceship.
27. Score will update every time an Asteroid has been hit.
28. The amount of lives the spaceship has will be updated every time it has been killed.
29. The high score will be displayed at the end of the game.
30. Spaceship will be able to teleport every time the user presses the enter button.
31. **Output Requirements:**
32. The current level that the user is on will be displayed.
33. The health of the Spaceship will be displayed.
34. The position of the Spaceship will be displayed, when in gameplay.
35. The position of the Asteroids will be displayed, when in gameplay.
36. The position of the Flying saucer will be displayed, when in gameplay.
37. The bullets will be displayed, when shot from the spaceship.
38. Instructions on how to play the game will be displayed on the start screen, if the instructions button has been pressed.
39. The score of the user.

**Hardware and Software Requirements**

The system will be using Python 3.4 and so the software and hardware requirements will be similar with Python, so that the game will function with the programme. I have researched Pythons requirements online and will list the software requirements below with a justification along with the hardware requirements. The additional hardware requirements for the game will be: A keyboard so the user can control the spaceship, a mouse to start the game and additional hard disk space to store the Asteroids game.

|  |  |
| --- | --- |
| **Software Requirements** | **Justification** |
| Python 3.4. | Software required to run the Asteroids game. |
| Microsoft Windows 7 at least. | This is the operating system that is required to run Python 3.4. |

|  |  |
| --- | --- |
| **Hardware Requirements** | **Justification** |
| PC with 1.4Ghz processor or equivalent. | Minimum processing power to run python 3.4. |
| Computer Keyboard. | Needed so the user can control the spaceship. |
| 2MB of RAM for Windows 7 however 4 recommended. | Minimum RAM needed to run Python 3.4 |
| 10MB of hard disk space. | Enough space to save the game. |
| High Resolution Monitor – Hanns.G recommended. | Needed to display the output to the user. |
| Mouse or another computer pointing device. | Needed to start the game. |

User review of requirements

All seven of my users have reviewed and agreed to the above requirements.

Signed:

**Section 2 –Design –Nature of the Solution**

**Design Objectives**

The design objectives are quite similar to the requirements specification specified before. However in the design objectives I will get more particular, for example I will talk about what’s going to be on my start up screen.

To show this is to my users and get their feedback on interface designs like these I will design the interface and then display them so that I have the opportunity to interview the users so I get there feedback on them.

I have discussed with my user group on many different considerations for my game such as Aesthetic, Input, Output and processing and have listed them below:

1. **Aesthetic considerations**
   1. The Screen Size will be large: 800x800
   2. The Colour of the Asteroids will be: Blue
   3. The Colour of the Spaceship will be: White
   4. The Colour of the Flying Saucer will be: Yellow
   5. The Size of the Asteroids will be: 80x80
   6. The Size of the Spaceship will be: 30x30
   7. The Size of the Flying Saucers will be: 30x30
   8. The Spaceship will initially face up.
   9. There are unlimited waves.
   10. The home menu will display a ‘Start Button’ and ‘Instructions’ and ‘High scores’
   11. How many Asteroids at first level – 5
2. **Input Considerations**
3. The player will be able to rotate the Spaceship and be able to thrust it.
4. The player will be able to shoot bullets at either the flying saucer or the asteroids.
5. The player will be able to view instructions for the game before they play the game.
6. The player will be able to pause the game and escape from the game.
7. The player will be able to activate a shield on the ship.
8. **Processing Considerations**
9. At each new level there will be 1/3 more Asteroids and there speed will increase by 20 milliseconds.
10. The system must record how many Asteroids there are left and increment the level once all the Asteroids have been destroyed.
11. Each movement the spaceship takes determines whether the spaceship collides with an Asteroid, Bullet from a flying saucer or its own bullets.
12. The spaceship will be able to teleport and thrust from one place on the screen to another. The spaceship will also be able to shoot bullets at the Asteroids and the flying saucer.
13. The user will be able to write their name next to their high score.
14. **Output Considerations:**
15. The current level will be displayed.
16. How many levels the spaceship has will be displayed.
17. Position of the Asteroids and the flying saucer will be displayed.
18. Instructions on how to play the game will be displayed.
19. The option for mouse or keyboard will be displayed.
20. A start up menu will be displayed.
21. There will be a word box to allow the user to enter their name for their high score and then it will output all the lists of the high scorers.

**User Review of Design Objectives**

All 5 of my users have reviewed and agreed to the above considerations. However one of my 5 users (Gemma), had asked about whether or not there should be the option for the user to enter their own name for the high score and so because of this I decided to include this consideration in my program.

Signed:

**Interface Design 1: Opening Page**

|  |  |  |
| --- | --- | --- |
| **Object** | **Property** | **Setting** |
| Title | Text | Visible=True, font=Adventurer Light SF’, size=’18’ |
| game\_button | Button | False |
| Insturctions\_button | Button | False |
| Asteroid\_image | Image | Visible=True, size= 600x600 |
| Interface | BackColour | BackColour=Black |
|  |  |  |

**ASTEROIDS!!!**

**WE NEED YOUR HELP**

Title label called Asteroids\_label



Asteroids Image

PLAY GAME

INSTRUCTIONS

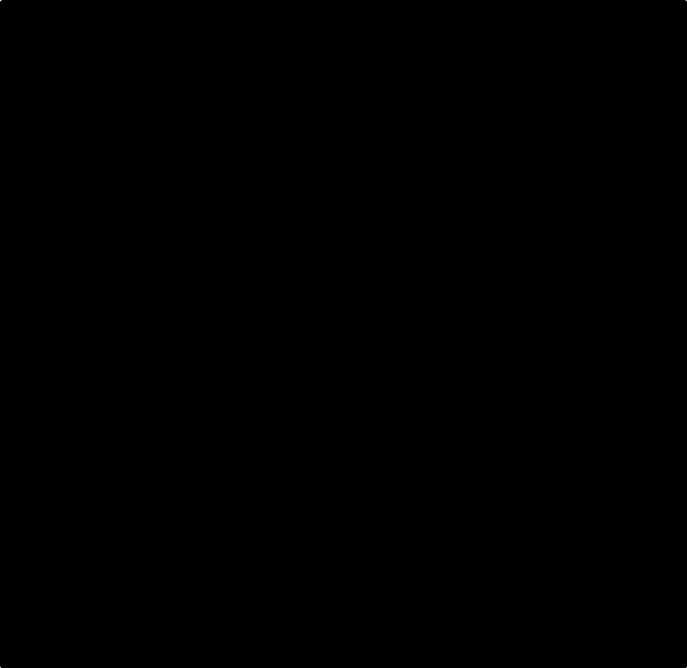
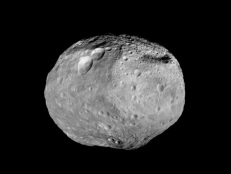
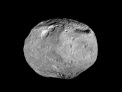
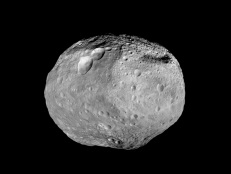
Command button named Instruction\_button – when clicked it will load the Insutrctions screen page and that will display all the insturctions to play the game.

Comman button name game\_button – when clicked it will load the option page for mouse and keyboard.

**Interface Design 2: Options Page**

|  |  |  |
| --- | --- | --- |
| **Object** | **Property** | **Setting** |
| Name\_box | Input text box | Visible=True, font=Adventurer Light SF’, size=’16’ |
| mouse\_button | Button | False |
| keyboard\_button | Button | False |
| Asteroid\_2\_image | Image | Visible=True, size= 100x100 |
| Interface | BackColour | BackColour=Black |
| Spaceship\_image | Image | Visible=True, size= 200x200 |
| Mouse\_or\_keyboard\_label | label | Visible=True, font=Adventurer Light SF’, size=’16’ |

Text Box, which allows the user to enter their name into the game, so that the game can save their high score on the system.



Command\_button named keyboard\_button and when pressed the game will know that the user will use the keyboard to play his game. It will then enter the gameplay.

Label, to tell the user that they have the option of using a mouse or a keyboard to play the game.

PLEASE ENTER NAME:

Do you want to play with either Mouse or Keyboard? Click on your desired choice!!!!!

Keyboard

Mouse

Command\_button named mouse\_button and when pressed the game will know that the user will use the mouse to play his game. It will then enter the gameplay.

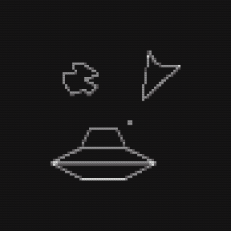
**Interface Design 3: Insturctions Page**

|  |  |  |
| --- | --- | --- |
| **Object** | **Property** | **Setting** |
| How\_to\_play\_label | Label | Visible=True, font=Adventurer Light SF’, size=’20’ |
| Asteroids\_3\_image | Image | Visible=True, size= 50x50 |
| keyboard\_label | Label | Visible=True, font=Adventurer Light SF’, size=’16’ |
| Mouse\_label | Label | Visible=True, font=Adventurer Light SF’, size=’16’ |
| Keyboard\_instructions\_label | Label | Visible=True, font= Calibri (Body), size=’12’ |
| mouse\_instructions\_label | Label | Visible=True, font= Calibri (Body), size=’12’ |
| Play\_game\_button | Button | False |
| Interface | BackColour | BackColour=Black |

Image

Text

HOW TO PLAY!!



These are the mouse\_ instructions\_text/ keyboard\_instructions\_text to inform the user on how to play the game

KEYBOARD

MOUSE

* To teleport the Spaceship you press the enter button.
* To rotate you use the arrow keys
* To shoot bullets you click on the right hand side of the mouse.
* To thrust the spaceship you hold down left hand side of the mouse.
* To quit the game you press escape.
* To teleport the Spaceship you press the enter button.
* To rotate the Spaceship you use the arrow keys.
* To shoot bullets you press the spacebar key.
* To thrust the spaceship you hold down the up arrow key.
* To quit the game you press escape.

CLICK HERE TO ENTER THE GAME

**Interface Design 4: Gameplay Screen**

|  |  |  |
| --- | --- | --- |
| **Object** | **Property** | **Setting** |
| Score\_box | Text | Visible=True, font=Adventurer Light SF’, size=’16’. Score=0 |
| Lives\_box | Image | Visible=True, font=Adventurer Light SF’, size=’16’. Lives=3 |
| Asteroid\_image | Image | Visible=True, size= 80x80 |
| Spaceship\_image | Image | Visible=True, size= 30x30 |
| Bullets | Image | Visible=True, size= 10x10 |
| Interface | BackColour | BackColour=Black,Blue |
| Flying\_saucer\_image | Image | Visible=True, size= 30x30 |
| Level\_box | Text | Visible=True, font=Adventurer Light SF’, size=’16’. Level=1 |

The score, the lives and the level display to the user, all the information they need when playing the game.

[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]

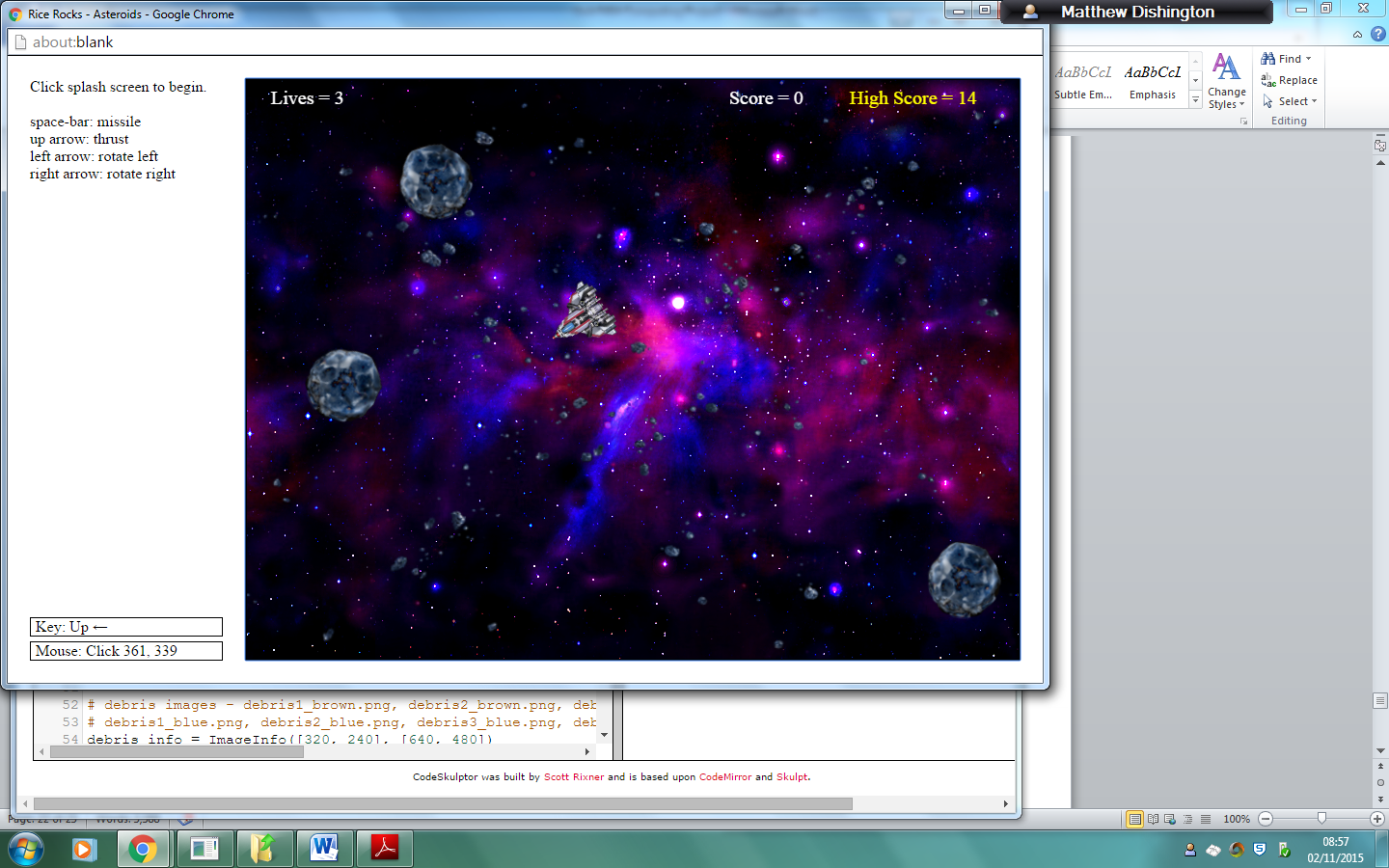
Score:

Lives: III

[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]

[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]

Level:



This is the spaceship that the gamer controls to destroy all the asteroids and the flying saucers.

These are the asteroids that are free flowing and they float about on the game screen until the gamer destroys them.

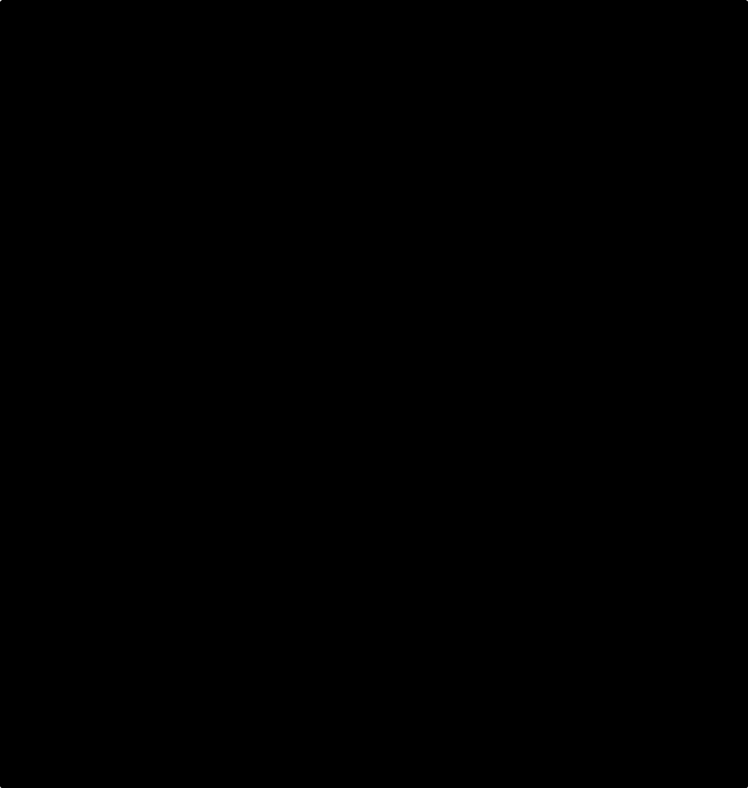
[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]

[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]

**Interface Design 5: Game Over Screen**

|  |  |  |
| --- | --- | --- |
| **Object** | **Property** | **Setting** |
| Game\_over\_text | Text | Visible=True, font=Adventurer Light SF’, size=’16’ |
| New\_game\_button | Button | False |
| Exit\_button | Button | False |
| face\_image | Image | Visible=True, size= 30x30 |
| Interface | BackColour | BackColour=Black,Blue |

This is the game\_over\_text to tell the user that they have run out of lives and the game is now over.

****

EXIT

START NEW GAME

GAME OVER!!!!

**User Group Feedback For Scenes**

Screen 1: Most of the users said that they liked the font of the words and the background colour but they said that the exclamation marks in ‘ASTEROIDS!!!’ is a bit amateurish, the image on the screen doesn’t look right in the frame and that I should vary the font size of the buttons and the title.

Screen 2: The user group said that they liked the spacing of the window and their criticism was to use a photo of the spaceship that it will actually look like.

Screen 3: All good positive feedback. They all commented on the layout of the page and the spacing.

Screen 4: The criticisms were that the whole screen should be the colour and shouldn’t have the black spot at the top for the score, the lives and the level.

Screen 5: The users said that the score should be displayed and that the screen should be more spaced theme.

**Response to Feedback:**

Screen 1: I’m going to enlarge the photo so that it fits the whole frame, also remove some of the exclamation marks and some of the text, so that it says ‘ASTEROIDS!’. As well I’m going to include a quit button and a high score button at the bottom of the frame which is sort of like a separate frame from the other frame.

Screen 2: I am going to take rid of this page altogether and say to the user that they can use the keyboard or mouse when they play the game.

Screen 3: I will not make any changes towards this screen.

Screen 4: I will remove the top blank black chunk of the screen so that the score, the lives and the level are all on the gameplay screen and not separate.

Screen 5: I am going to include the score onto the page so that the gamer knows how well they did and I am also going to make the background more space-like.

**Updated versions of the Gameplay screens**



ASTEROIDS!

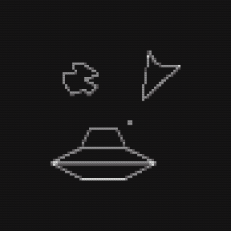
ITH SCOREONS HELPof the Gameplay screens

INSTRUCTIONS

HIGH SCORE

PLAY GAME

EXIT



HOW TO PLAY! – YOU CAN USE MOUSE OR KEYBOARD IN THE GAME

KEYBOARD

MOUSE

* To teleport the Spaceship you press the enter button.
* To rotate you use the arrow keys
* To shoot bullets you click on the right hand side of the mouse.
* To thrust the spaceship you hold down left hand side of the mouse.
* To quit the game you press escape.
* To teleport the Spaceship you press the enter button.
* To rotate the Spaceship you use the arrow keys.
* To shoot bullets you press the spacebar key.
* To thrust the spaceship you hold down the up arrow key.
* To quit the game you press escape.

EXIT

HIGH SCORE

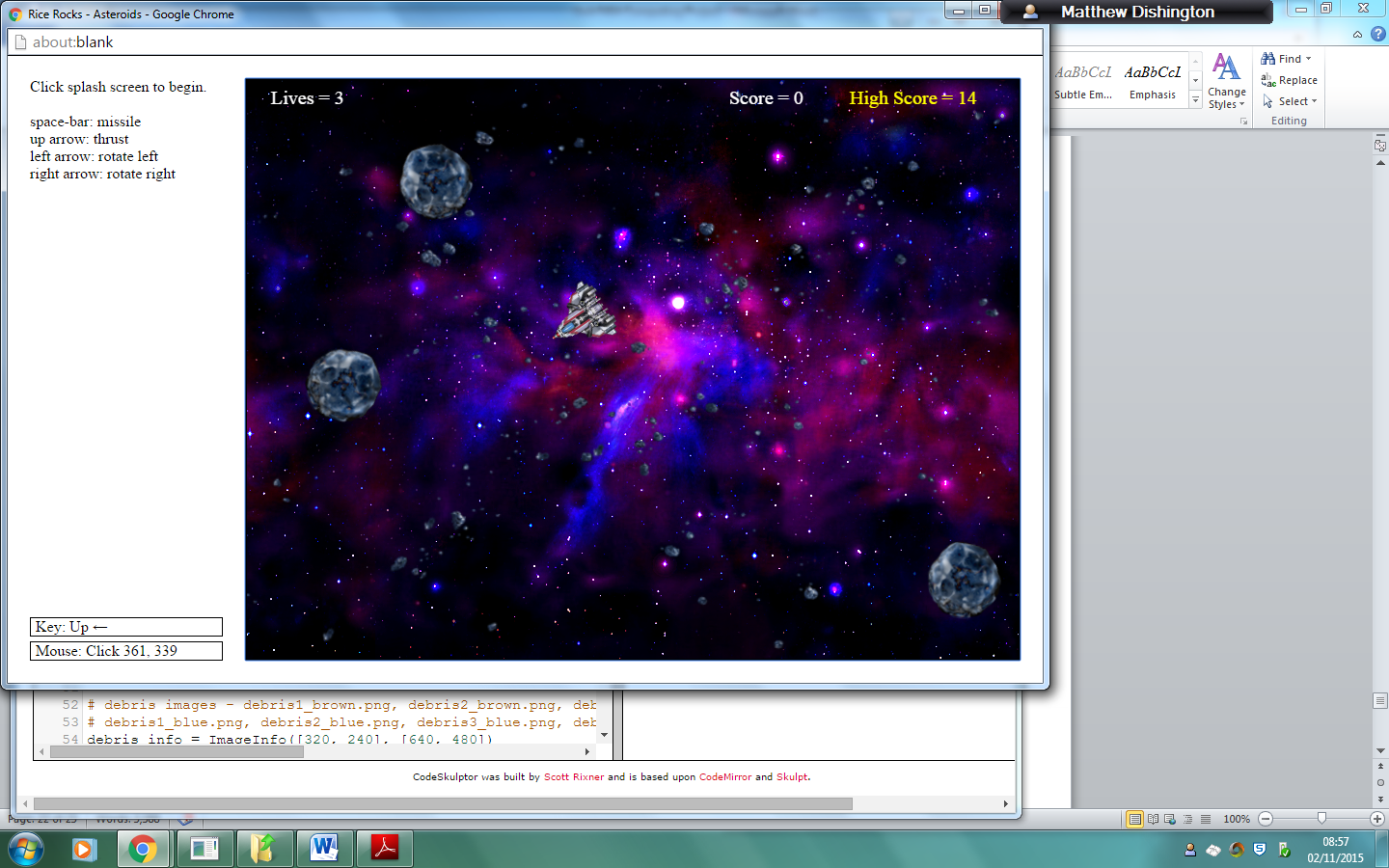
INSTRUCTIONS

PLAY GAME

The score, the lives and the level display to the user, all the information they need when playing the game.

Score:

Level:



Score:

Lives: III

Lives: III

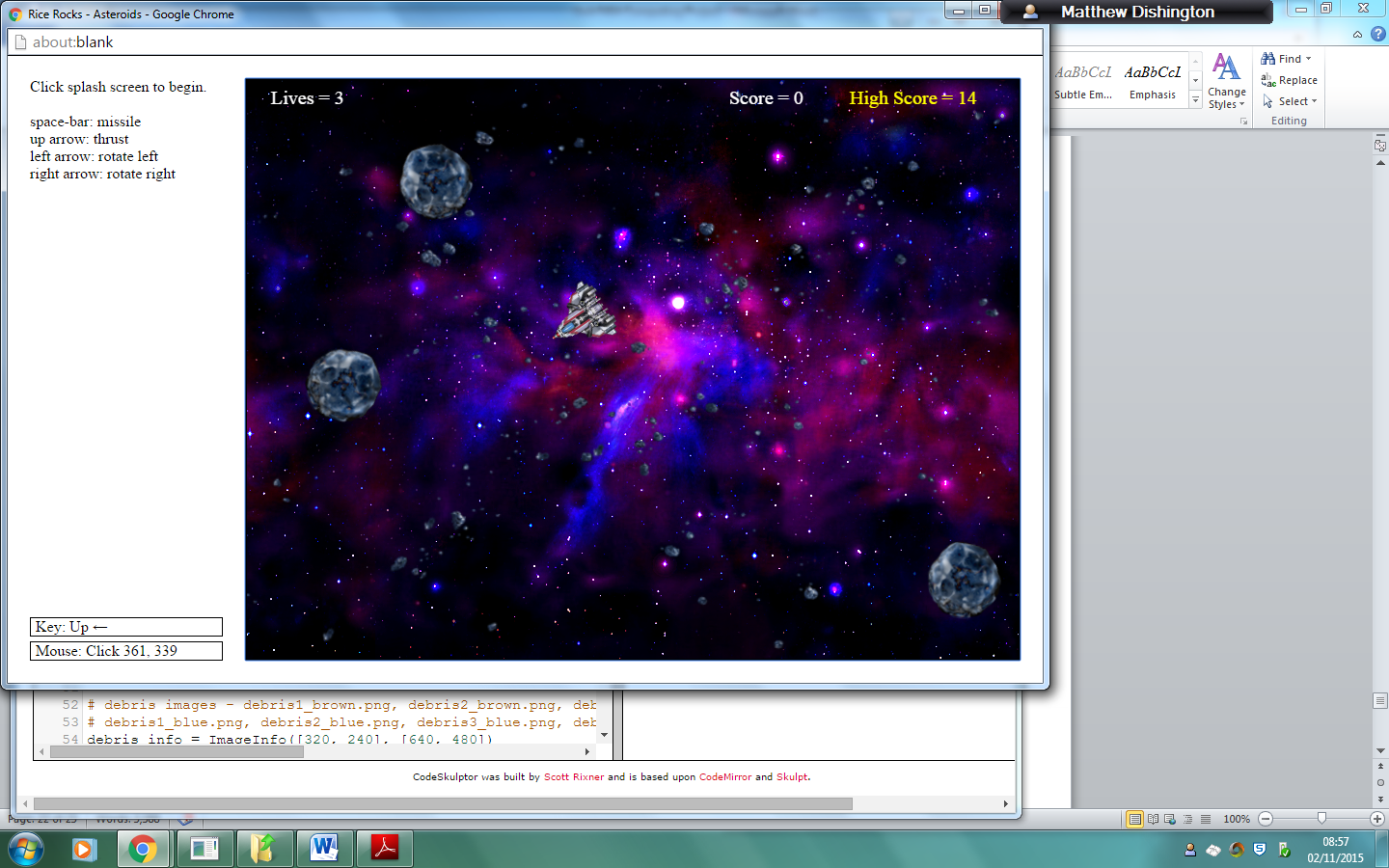
Level:

EXIT

HIGH SCORE

INSTRUCTIONS

PLAY GAME



GAME OVER!!

Your Score:

Enter Your Name:

EXIT

HIGH SCORE

INSTRUCTIONS

PLAY GAME

|  |  |  |
| --- | --- | --- |
|  | Name | Score |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

HIGH SCORES

HIGH SCORE

INSTRUCTIONS

PLAY GAME

EXIT

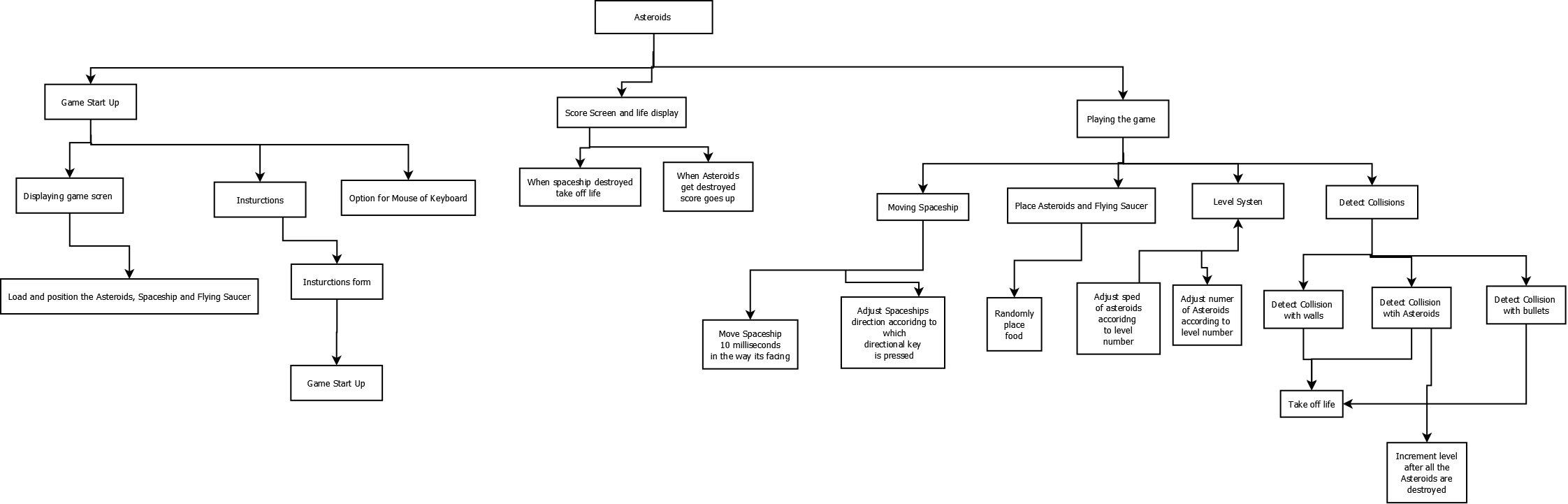
**Data Structure Design**

No files will be created or used by the Asteroids program. The program will use the built in functions, in Python: Tkinter, random and pygame. These modulars will be used to create the frame and outline of the game and also the positions of the Asteroids.

Below are the examples and the descriptions of some of the variables that I will use in my main program:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable Name** | **Type** | **Size** | **Descriptions** | **Sample Value** | **Validation** |
| Asteroids\_size | Integer | Different every time the Asteroids get hit with a bullet but they will start off as 80x80. | Contains the size of the Asteroid. | 80x80 | The asteroids decrease in size every time been hit. |
| score | Byte | Changes everytime, you destroy an asteroid or gain a level but starts off at 0. | Stores the value of the score that the player is on. | 500 | N/A |
| lives | Integer | 3-0 | Stores the value of the number of lives that the player is on. | 1 | N/A |
| Level | Byte | 1-50 | Stores the value of the level that the player is on. | 35 | N/A |
| Number\_of\_Asteroids | Integer | Will start off as 5 and will go up in numbers by a 1/3 everytime | Contains the number of Asteroids the user has to destroy to complete the level. | 15 | N/A |
| Asteroids\_speed | Real | Starts off at 30 milliseconds but as the levels increment so do the Asteroids by 2 milliseconds. | Contains the speed of the Asteroids and has a local variable in determining on whether or not the speed should increase. | 40 | N/A |
| Bullet\_size | Integer | 5x5 | Contains the size of the bullet. | 5x5 | N/A |
| Bullet\_speed | Real | 20 milliseconds | Contains the speed of the bullets. | 20 milliseconds | N/A |
| Spaceship\_speed | Real | 30 milliseconds | Contains the speed of the spaceship. | 30 milliseconds | N/A |
| Flying\_saucer\_speed | Real | Starts off at 30 milliseconds but as the levels increment so do the flying saucer by 2 milliseconds. | Contains the speed of the flying saucer and has a local variable in determining on whether or not the speed should increase. | 32 milliseconds | N/A |
| Canvas\_height | Integer | 800 | Is a global variable is used to determine whether or not the Asteroids or Spaceship goes through it. | 800 | If asteroids\_position[1]>Canvas\_height:  The asteroid has gone through the height of the canvas |
| Canvas\_width | Integer | 800 | Is a global variable is used to determine whether or not the Asteroids or Spaceship goes through it. | 800 | If asteroids\_position[0]>Canvas\_height:  The asteroid has gone through the height of the canvas |
| Key\_pressed | Single | 1-145 | Determines which key the user has pressed on, and uses it to apply to the spaceship. | 100 | N/A |
| facing | Byte | 1-4 | Determines which way the spaceship is facing. | 2 | N/A |
| Hit | Boolean | True or False | Contains the Boolean value if the spaceship has been hit or not. | True | The Spaceship loses a life. |
| Flying\_saucer\_size | Integer | 30x30 | Contains the size of the Flying saucer. | 30x30 | N/A |
| Asteroids\_position | Integer | [Canvas\_width/2], [Canvas\_height/2] | Contains the starting position of the Asteroid and determined where the asteroids move. | [50,50] | If the asteroids position is bigger than the canvas\_height or width then the asteroid has hit the side of the canvas. |

**Modular Design of Asteroids – Structure Design**



**Updated Design Objectives**

* + 1. **Aesthetic considerations**
  1. The Screen Size will be large: 800x800
  2. The Colour of the Asteroids will be: Blue
  3. The Colour of the Spaceship will be: White
  4. The Colour of the Flying Saucer will be: Yellow
  5. The Size of the Asteroids will be: 80x80
  6. The Size of the Spaceship will be: 30x30
  7. The Size of the Flying Saucers will be: 30x30
  8. The Spaceship will initially face up.
  9. There are 50 levels.
  10. The user will have the option for using the keyboard or a mouse.
  11. Every screen will show a panel at the bottom with the buttons ‘Start Game’, ‘Instructions’, ‘High scores’ and ‘Exit’.
  12. How many Asteroids at first level – 5

1. **Input Considerations**
2. The player will be able to pick if they would rather pick playing with a mouse or the keyboard.
3. The player will be able to rotate the Spaceship and be able to thrust it.
4. The player will be able to shoot bullets at either the flying saucer or the asteroids.
5. The player will be able to view instructions at any time by pressing the ‘Instructions’ button on the control panel.
6. The player will be able to pause the game and escape from the game at any time.
7. The player will be able to check all the high scores at any time expect for wen there in the middle of a game.
8. **Processing Considerations**
9. At each new level there will be 1/3 more Asteroids and there speed will increase by 2 milliseconds.
10. The system must record how many Asteroids there are left and increment the level once all the Asteroids have been destroyed.
11. Each movement the spaceship takes determines whether the spaceship collides with an Asteroid, Bullet from a flying saucer or its own bullets.
12. The spaceship will be able to teleport and thrust from one place on the screen to another. The spaceship will also be able to shoot bullets at the Asteroids and the flying saucer.
13. The user will be able to write their name next to their high score at the end of the game.
14. The user will be able to use the mouse or the keyboard in the game and the game will outline this.
15. **Output Considerations:**
16. The current level will be displayed.
17. How many levels the spaceship has will be displayed.
18. Position of the Asteroids and the flying saucer will be displayed.
19. The buttons on the control will always be displayed.
20. A start up menu will be displayed at the start up game.
21. There will be an instructions page which the user can access using the instructions button on the bottom control panel.
22. There will be a word box to allow the user to enter their name for their high score and then it will output all the lists of the high scorers.

Section B – Part 2

**Test Strategy – Test Plan**

I will firstly execute white-box testing in the form of dry-runs to be carried out on the design algorithms. Then I will perform black-box testing on the finished program, using the test plan outlined below, and respond accordingly to any issues that present themselves. Finally I shall get my user group to test the program and give me feedback, which may result in any changes to my program. Each test will be evidenced with screenshots.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function being tested** | **Method of input** | **Input** | **Expected result** |
| Game Start-Up |  |  |  |
| Placing Ship | **Test1:** Open up the game and see if the ship is in exactly the middle of the screen.  **Test2**: When the spaceship has lost a life, the ship will the ship be in the centre of the screen. | There is no direct input, expect when the user has pressed the ‘Play’ button to start the game. The other input is when the spaceship has lost its life. | The spaceship will appear in the middle of the screen with a shield around it, facing upwards. |
| Placing Asteroids | **Test1:** Open up the game and see if the Asteroids are all in a random different position  **Test2**: When the user has advanced onto the next wave, will the Asteroids be placed again in a random position. | There is no direct input, expect when the user has pressed the ‘Play’ button to start the game. | The result will be the Asteroids spawning in a random position on the gameplay screen. |
| Movement of Ship | When in game, pressing a directional button on the keyboard. | Up (when ship is facing right or left). | The ship will accelerate forward in the right or left direction. |
| Right (when ship is facing up or down). | The ship will rotate to the right to the amount that the right arrow button is pressed. |
| Down (when ship is facing right or left). | No change |
| Left (when ship is facing up or down). | The ship will rotate to the left to the amount that the left arrow button is pressed. |
| Up (when ship is facing up down). | The ship will accelerate forward in the up or down direction direction. |
| Right (when ship is facing right or left). | The ship will rotate to the right to the amount that the right arrow button is pressed. |
| Down (when ship is facing up or down). | No change |
| Left (when ship is facing right or left). | The ship will rotate to the left to the amount that the left arrow button is pressed. |
| None | The Ship will stand still facing the direction that it was left at. |
| Collision with walls | When in a game, directing the ship forward into any side of the wall using the forward arrow key | The up arrow key to move to the direction of the wall. | The ship will appear on the opposite side of the wall that it went through. |
| Collison with Asteroid (without bullets) | When in a game, directing the ship into an Asteroid. | The up arrow key to direct the ship into an Asteroid | The ship will lose a life and re spawn in the middle of the gameplay screen with a shield around it. |
| Collision with Asteroids (with bullets) | When in a game, directing bullets at Asteroids | The spacebar key will execute this transaction and will fire bullets in the direction that the ship is facing. | The Asteroid will end up splitting up into smaller and smaller Asteroids until they have been destroyed. When hitting each sized Asteroid the score will be updated according to its size. Once all the Asteroids have been destroyed the next wave will start. |
| Display instructions | Clicking the instructions button on the bottom control panel. | N/A | Displays the instructions screen |
| Display high scores | Clicking the high scores button on the bottom control panel. | N/A | Will display the list of high scores that has happened on the game. |
| Going up the next wave | Destroying all the Asteroids and seeing if the wave number changes and more Asteroids appear on the screen. | Destroying the Asteroids with bullets | More Asteroids will appear on the screen and the next wave will appear. |
| Shield appearing | Test1: When the game is loaded and a next wave happens the shield will surround the ship. | The user destroys all the Asteroids. | The shield will come up for a short period of time. |
| Test2: When the user presses the ‘s’ key on the keyboard the shield should come up. However you cannot press it all the time and it only lasts for a couple of seconds. | The user presses the ‘s’ key | The shield will come up for a short period of time. |

All seven members of my user group have seen and agreed to the test strategy and the design structure diagram.

Signed:

Algorithms and Testing

The asteroids program will require some algorithms within various modules. The modular names are written below and can also be found in the modular design above. I will describe each algorithm inside of the modulars.

* + - 1. Game Start Up
* Make the control panel clear at the bottom of the screen.
* Load and position the spaceship
  + - 1. Placing Asteroids
* Randomly place the Asteroids and ensure that none of them overlap each other and also don’t overlap the spaceship.
  + - 1. Detecting key presses
* Detect collisions with canvas
* Detect collisions with Asteroid
* Detect collision with bullet
  + - 1. Moving Spaceship
* Move the whole of the spaceship so it follows the same direction.
* Be able to thrust if key is held down.
* Move the spaceship by 20 milliseconds if the user has chosen to thrust.
* Spaceship can rotate in full 360 degrees.
  + - 1. Detecting key presses
* Detect if user has pressed T to teleport.
* Detect if user has chosen to press the spacebar to shoot bullets.

**Game Start Up:**

**Make the control panel visible:**

Control\_panel()

**Load and position the spaceship:**

CLASS Ship:

FUNCTION \_\_init\_\_( (self): VOID

Initialise self.acceleration\_factor to 0.35

Initialise self.drag\_factor = 0.02

Initialise self.rotation\_factor = 10

Initialise xTops coordinate to 0

Initialise xSize to 25

Initialise ySize to 30

Initialise yTops coorduianate = -ySize/2

Initialise yIndent = 10

Initialise self.shape = ( (xTop, yTop),

(xTop + xSize/2.0, yTop + ySize),

(xTop, yTop + ySize - yIndent),

(xTop - xSize/2.0, yTop + ySize) )

self.reset()

END FUNCTION

FUNCTION reset(self): VOID

Initialise self.x = CANVAS\_WIDTH/2

Initialise self.y = CANVAS\_HEIGHT / 2

Initialise self.angle = 0 # orientation

Initialise self.velocity\_x = 0

Initialise self.velocity\_y = 0

END FUNCTION

**Summary:**

This code will initialise all of the ships variables in the program and will initialise the first position for ship will be in the middle of the canvas when the game first begins.

**Placing Asteroids**

**if** x **is** None:

Initialise self.x = random coordinate between(0,CANVAS\_WIDTH)  
**else**:  
 Initialise self.x = x  
  
**if** y **is** None:  
 Initialise self.y = random coordinate between (0,CANVAS\_HEIGHT)  
**else**:  
 Initialise self.y = y  
  
Initialise self.angle= random angle between (0,360)  
Initialise self.rotation\_factor= random factor that the asteroid is rotating (0,100) / 100.0 - 0.5

**if** velocity\_x **is** None:  
 Initialise self.velocity\_x = random x velocity (0,100) / 100.0 - 0.5  
**else**:  
 Initialise self.velocity\_x = velocity\_x  
  
**if** velocity\_y **is** None:  
 Initialise self.velocity\_y = random y velocity (0,100) / 100.0 - 0.5  
**else**:  
 Initialise self.velocity\_y = velocity\_y  
  
FUNCTION self.update()

**Summary:**

This section of code is similar to placing the ship however, the Asteroids will not be instantly put into the middle of the room the game will randomly place all the Asteroids on the screen. The user will have a shield up when the Asteroids are placed so that the user will not lose all their lives instantly.

**Detecting Collisions**

**Detect Collision with wall:**

FUCNTION \_update\_position(self):INTEGER

IF self.x\_coordinate < 0:

self.x\_coordinate += CANVAS\_WIDTH

ELIF self.x\_coordinate > CANVAS\_WIDTH:

self.x\_coordinate -= CANVAS\_WIDTH

END IF

IF self.y\_coordinate < 0:

self.y\_coordinate += CANVAS\_HEIGHT

ELIF self.y\_coordinate > CANVAS\_HEIGHT:

self.y\_coordinate -= CANVAS\_HEIGHT

END IF

END FUNCTION

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lines | Self.x\_coordinate | Self.y\_coordinate | CANVAS\_WIDTH | CANVAS\_HEIGHT | Output | Comment |
| 01 |  |  |  |  | Enter Function | TRUE |
| 02 | 10 |  | 800 | 800 | Stays inside the canvas at the same point. | The if statement checks to see if the x coordinate of the spaceship then the spaceship has collided with the left hand side of the canvas. |
| 03 |  |  |  |  | Void, go to line 03 |  |
| 04 | 10 |  | 800 | 800 | The program checks to see if the x\_coordinate has collided with the right hand side of the wall and in this case the coniditon is false, so the spaceship stays in the same x\_ Coordinate. | Condition is false. |
| 05 |  |  |  |  | Void, go to line 05 |  |
| 06 |  |  |  |  | Void, go to line 06 |  |
| 07 |  | -1 | 800 | 800 | This if statement is TRUE. | The if statement checks to see if the y coordinate of the spaceship has collided with the top of the canvas. |
| 08 |  | 799 | 800 | 800 | The coordinate of the y has been added by 800 to come out on the bottom of the screen. | The coordinate gets added by the canvas\_height so that the coordinate is changed to come out on the other side of the canvas. |
| 09 |  | 799 | 800 | 800 | The spaceship will stay in the same new position. | This condition is false and so the if statement will not be run. |
| 10 |  |  |  |  | Void, go to line 09 |  |
| 11 |  |  |  |  | Return the x and y coordinates of the spaceship. | END FUCNTION |

**Summary:**

What this code does is checks to see if the x and the y coordinate of the spaceship, have hit any side of the canvas. There are 2 separate IF statements to check if the x\_coordinate hits the left or the right hand side of the canvas, and if it has hit the side the X\_coordinate will either be added or subtracted by the canvas\_width. The same will happen also for the y\_coordinate to check if it has hit the top or the bottom of the screen and will add or subtract by the canvas\_height.

In this testing I have checked to see if the x coordinate has hit the side of the wall, which it hasn’t and so it will keep the same X coordinate however, the y coordinate has hit the top of the wall and so the program will add the canvas height to the coordinate which means that it will come back in at the bottom of the screen.

**Detect Collision with Asteroids with bullet/ Asteroid with ship:**

**FUNCTION** is\_collision(self, gameObject):  
 Initialise collision = False  
  
 Initialise x = gameObject.get\_x() - self.get\_x()  
 Initialise y = gameObject.get\_y() - self.get\_y()  
 Initialise distance = math.sqrt(x\*\*2 + y\*\*2)  
 **if** gameObject.get\_radius() + self.get\_radius() >= distance:  
 Initialise collision = True

**return** collision

END FUNCTION

**for** asteroid **in** set(self.asteroids):  
 *#Compare each bullet against Asteroid* Initialise bullets = self.ship.get\_bullets()  
 **for** bullet **in** bullets:  
 **if** bullet.is\_collision(asteroid):  
 Initialise explosion\_sound.play()  
 Initialise size = asteroid.get\_size()  
 **if** size < 3:  
 Initialise a1 = Asteroid(size + 1, asteroid.get\_x(), asteroid.get\_y(), asteroid.get\_velocity\_y() \* -2, asteroid.get\_velocity\_x() \* 2) self.asteroids.add(a1)  
  
 Initialise a2 = Asteroid(size + 1, asteroid.get\_x(), asteroid.get\_y(), asteroid.get\_velocity\_y() \* 2, asteroid.get\_velocity\_x() \* -2) self.asteroids.add(a2)  
  
  
 *#Compare ship against Asteroid* **if not** self.ship.is\_invincible():  
 **if** self.ship.is\_collision(asteroid):  
 play explosion\_sound  
 self.ship.remove\_live()  
 **if** self.ship.get\_lives() <= 0:  
 self.ship.set\_active(False)  
 Initialise self.game\_over=True  
 **else**:  
 self.ship.make\_inactive\_for(350)

**Summary:**

This portion of code detects whether the ship that the user controls has crashed with the Asteroid and if so the user will lose a life, will play an explosion sound and the ship will get reborn back into the centre of the screen. The code also detects if the bullet fired from the ship has hit the Asteorid and if it has, the Asteroid will then split up into 2 smaller bits and then 3 even smaller and then the Asteroid will be destroyed. Once all the Asteroids have been destroyed, the user will go onto the new wave and the same number of Asteroids from the last wave +2 will appear on the screen.

**Moving Spaceship**

FUNCTION \_set\_points(self):VOID

Initialise radians to self.angle \* math.pi /180.0

Initialise sine to math.sin(radians)

Initialise cosine to math.cos(radians)

Initialise self.points to list()

FOR LOOP p in self.shape:

new\_x\_coordiante = position [0] \* cosine - position[1] \* sine

new\_y\_coordiante = position [0] \* sine + position[1] \* cosine

self.points.append((self.x + new\_x , self.y + new\_y))

END LOOP

END FUNCTION

FUNCTION rotate\_left(self, event):INTEGER

Initialise self.angle -= self.rotation\_factor

def rotate\_\_ship right(self, event):

Initialise self.angle += self.rotation\_factor

END FUNCTION

FUNCTION draw(self, canvas): VOID

self.\_update\_position()

self.\_set\_points()

canvas.delete('all')

canvas.create\_polygon(self.points, fill='', outline='White')

END FUNCTION

IF key press is ('<Left>',)

ship.rotate\_left

END IF

IF key press is ('<Right>')

ship.rotate\_right)

END IF

IF key press is ('<Up>)'

ship.accelerate)

END IF

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Lines | Self.angle | new\_x\_coordinate | new\_y\_coordiante | self.rotation\_factor | Key\_pressed | Output | Comment |
| 01 |  |  |  |  |  | Void, go to line 02 | Enter function |
| 02 | 30 |  |  |  |  | 1/6 \* pi | Converting an angle into radians. |
| 03 |  |  |  |  |  |  | Converts the sin rule in python to radians. |
| 04 |  |  |  |  |  |  | Converts the cosine rule in python to radians. |
| 05 |  |  |  |  |  |  | Creates a new list of points |
| 06 |  |  |  |  |  |  | Enter loop |
| 07 |  | 50 |  |  |  |  | Changing the x\_coordinate so that when the user can rotate the spaceship. |
| 08 |  |  | 70 |  |  |  | Changing the y\_coordinate so that when the user can rotate the spaceship. |
| 09 |  |  |  |  |  |  | Add the points to the new list. |
| 10 |  |  |  |  |  |  | End Loop |
| 11 |  |  |  |  |  | Void, go to line 12 |  |
| 12 |  |  |  |  |  | Void, go to line 13 | Enter function |
| 13 | 30 |  |  | 0.25 |  | New angle = 22.5 | Calculates the angle that the spaceship will have after the user has decided to turn left. It will minus the self.angle as its going anti-clockwise and in the negative direction. |
| 14 |  |  |  |  |  | Void, go to line 15 | End function |
| 15 |  |  |  |  |  | Void, go to line 16 | Enter function |
| 16 | 30 |  |  | 0.25 |  | New angle = 70 | Calculates the angle that the spaceship will have after the user has decided to turn right. It will be added to the self.angle as its going clockwise and in the positive direction. |
| 17 |  |  |  |  |  | Void, go to line 18 | End function |
| 18 |  |  |  |  |  | Void, go to line 19 | Enter Function |
| 19 |  | 50 | 70 |  |  | [50,70] | Updates the new points of the spaceship. |
| 20 |  |  |  |  |  | Void, go to line 21 | Sets the points of the spaceship |
| 21 |  |  |  |  |  | Void, go to line 22 | Deletes everything on the canvas |
| 22 |  | 50 | 70 |  |  | Spaceship, but with its changes position. | Draws the new spaceship with its news coordinates but does this so quickly the user can’t see. |
| 23 |  |  |  |  |  | Void, go to line 24 | End function |
| 24 |  |  |  |  | Right | Void, go to line 25 | Checks which key the user has pressed to move the spaceship. |
| 25 |  |  |  |  |  | Void, go to line 26 |  |
| 26 |  |  |  |  | Right | True | Checks which key the user has pressed to move the spaceship. |
| 27 |  |  |  |  |  | Rotates the ship right using the rotational factor and adds it on to the self.angle. | Rotates the ship right using the rotational factor and adds it on to the self.angle. |
| 28 |  |  |  |  |  | Void, go to line 29 |  |
| 29 |  |  |  |  | Right | False | Checks which key the user has pressed to move the spaceship. |
| 30 |  |  |  |  |  | Void, go to line 29 |  |
| 31 |  |  |  |  |  | Finish |  |

**Summary:**

This portion of code checks whether or not the user has decided to either rotate the ship that they control, left or right and also to check if they have decided to thrust the spaceship. The program gets the angle at which the spaceship is at, and then calculates what it is in radians so that they can multiple the angle by the x and y position of the

***Testing:***

*What this code does is that it checks to see if the x and the y coordinate of the spaceship, have hit any side of the canvas. There are 2 separate IF statements to check if the x\_coordinate hits the left or the right hand side of the canvas, and if it has hit the side the X\_coordinate will either be added or subtracted by the canvas\_width. The same will happen also for the y\_coordinate to check if it has hit the top or the bottom of the screen and will add or subtract by the canvas\_height.*

*In this testing I have checked to see if the x coordinate has hit the side of the wall, which it hasn’t and so it will keep the same X coordinate however, the y coordinate has hit the top of the wall and so the program will add the canvas height to the coordinate which means that it will come back in at the bottom of the screen.*

**Detecting Keypresses:**

CASE [key code for bullets]

Parent.bind(<Space>, ship.fire)

CASE [key code for shield]

Parent.bind(<s>, ship.handle\_activate\_shield\_event)

CASE [key code for rotating left]

Parent.bind(<Left>, ship.rotate\_left)

CASE [key code for rotating right]

Parent.bind(<Right>, ship.rotate\_right)

CASE [key code for accelerating]

Parent.bind(<Up>, ship.accelerate)

**Summary:**

This section of code, detects which key the user has pressed and so whatever key the user has pressed the game will detect this and will then call the function to which that key is assigned to and will carry out what the user had wanted the game to do.

Section 3 – Software Development and Testing

Part 1 – Software Development

The development can be broken up into steps which I will describe in the order in which I carried out. I will highlight the changes I had to make during the process.

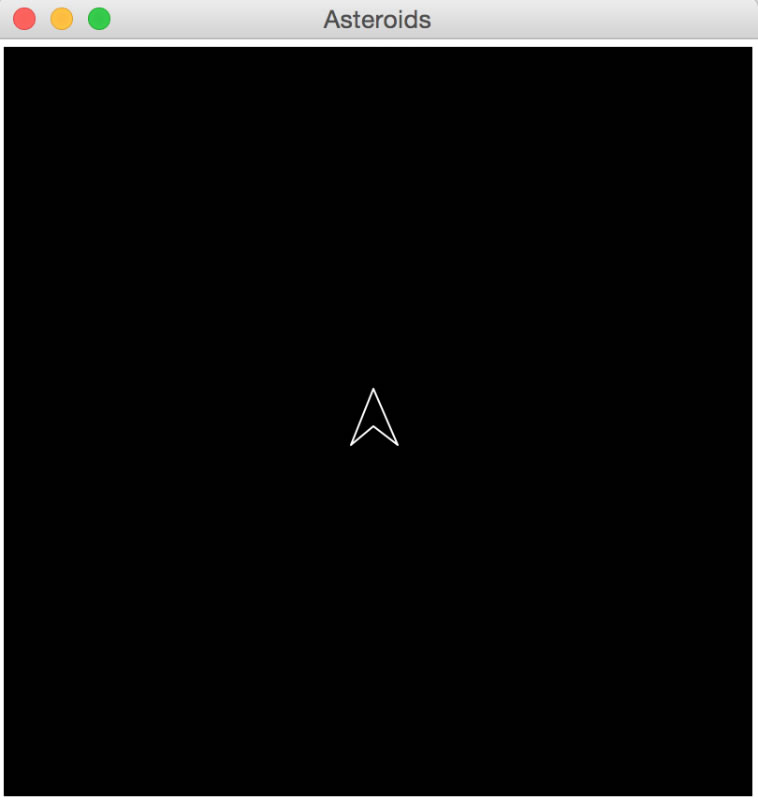
These are the steps that I took to produce my game:

Steps:

1. Ship and basic interface
2. Asteroids
3. Add Bullets
4. Asteroid bullet collision
5. Making a super class
6. Tweaking
7. Score and Lives
8. Waves and Shields
9. Background Image
10. Sounds
11. Begin Writing the Ship class and make the basic interface

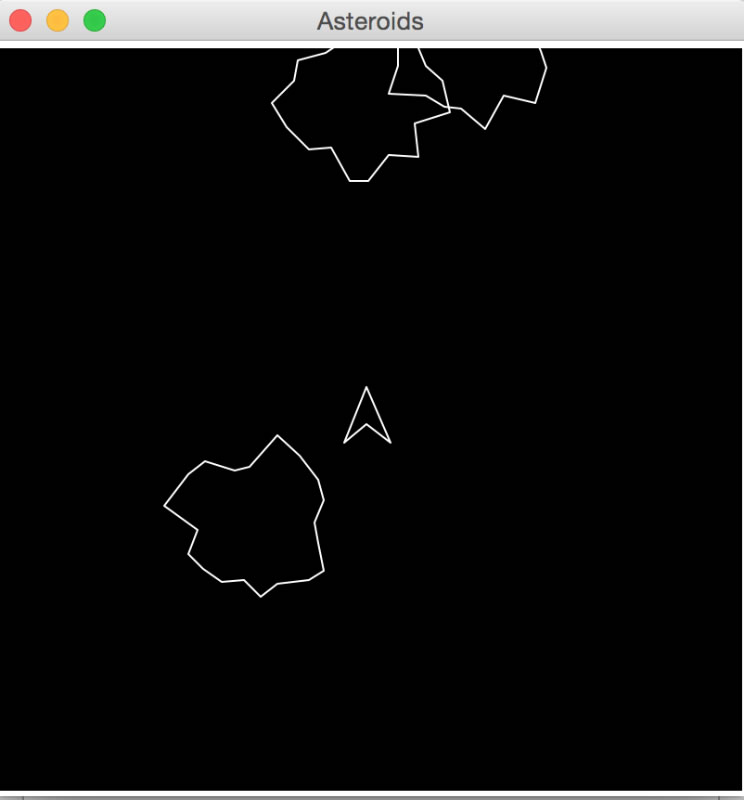
Firstly I made a basic gameplay screen with a black background. Then I began writing the class for the ship that is executed when the game has started and when the user has decided to either move or shoot bullets. The code was not as advanced for the

bullets yet, so it was just able to move on the screen. The ship was able to rotate 360 degrees and move with a forward force. I decided to add a factor for the ship to hold up when it moves on the screen, so that it looks more like its flying in space. The code for rotating the ship took me time to understand and to create on my game, and so I had to look up how to do it. The ship was also able to go through the sides of the walls and come out the other opposite side.



1. Began writing the Asteroids Class

In this class I decided to draw one Asteroid and used a random module for the way that they all look. They all had a similar shape but then the random modular would take in turn how to dent the circle to make it look like an Asteroid. I then added another random modular, onto the velocity for the Asteroids so that all of them are moving in different velocities. After I made a list of Asteroids, so that there was more than one on the screen at a time.



1. Add Bullets

After creating the Asteroids, I decided I need to create the bullets from the ship and so I added a couple of functions to the ship class to execute this. The bullets were created so that there are only 5 of them at a time when you press the spacebar and so you can’t just keep your finger on the spacebar down. As well they do not travel the whole distance on the canvas, so that it makes it harder to shot the Asteroids. The bullets function took some time as I needed to draw the bullets coming out right on the point of the ship.

1. Collisions with bullets and Asteroids

This module took me the longest of them all. I needed to calculate the distance between the two objects and then see if the distance is bigger than the two radiuses combined. If the distance was smaller it means that the two objects have collided and so if the Asteroid was hit by a bullet, it would get split down into a smaller Asteroid and then another, then vanish. If the ship was hit by the Asteroid it will spawn back in the middle of the screen.

1. Making a Super Class

I think decided that many of the classes such as the Asteroid and the ship had many of the same properties and so I decided to put everything that was similar into one big class called an Application Class so that the lines of code would be reduced and I wouldn’t be repeating code in my programme. It would also make the code more efficient.

1. Tweaking

Before I added in the score and the waves into my programme. I decided to make a few tweaks to my programme so that it runs better and looks better. I decided to change the colour of the Asteroids to yellow, so that they stand out more than the ship to make it easier for the end user. I also played around with some of the variables on the ship so that it was easier to move it around and fire bullets. I then realised that if (SHIELD AROUND SHIP) I then made a list of the advancements I can make towards my game and prioritised them all and saw which ones were going to take less time than other ones.

1. Score and Lives

The programme will now have a more of a game feeling towards it. I decided to put in score and lives to the spaceship so that the user can know how well there doing in the game. All I had to do with the score and lives was to create the global variable to 0 for the score and 3 for the lives. Then whenever there was a collision I would either lose a live or will add to the score. Finally the score and lives got drew on the top left and right on the gameplay screen.

1. Waves and Shields

After programming the score and lives, I wanted to include some more extra features to the game. I put in levels so that when all the Asteroids are destroyed the next wave will happen and more Asteroids will appear. With the shield I thought about adding an extra attribute to the ship and this was a shield which lasts up to 10 seconds and allows the ship to hit the Asteroids without losing a life. I also got in a bit of trouble because I realised that if the ship appeared in the middle of the screen and there was an Asteroid there, it would instantly lose all 3 lives and so this for this reason I decided to add the shield when the waves begin.

1. Background Image

When I was playing the game and was looking at different types of the game online, I thought they have a black background and so I decided to mix it up so that my game is different from every other one. I then added in a ‘space’ background to the game so that it looks more like that the ship is in space with the Asteroids around it. This didn’t take a long time as luckily Tkinter has a background image built in function for it.

1. Sounds

Finally for my first version of my programme I added sound to my programme. There was now sound for when the bullets are activated, there is a collision and whenever the ship is moving. I imported the sound from an online version of the game Asteroids and I added it onto mine.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Function being tested** | **Method of input** | **Input** | **Expected result** | Actual result | | | Screenshot reference | | |
| Game Start-Up |  |  |  |  | | | 1 | | |
| Placing Ship | **Test1:** Open up the game and see if the ship is in exactly the middle of the screen.  **Test2**: When the spaceship has lost a life, the ship will the ship be in the centre of the screen. | There is no direct input, expect when the user has pressed the ‘Play’ button to start the game. The other input is when the spaceship has lost its life. | The spaceship will appear in the middle of the screen with a shield around it, facing upwards. | Expected result | | | 2 | | |
| Expected result | | | 3 | | |
| Placing Asteroids | **Test1:** Open up the game and see if the Asteroids are all in a random different position  **Test2**: When the user has advanced onto the next wave, will the Asteroids be placed again in a random position. | There is no direct input, expect when the user has pressed the ‘Play’ button to start the game. | The result will be the Asteroids spawning in a random position on the gameplay screen. | Expected result | | | 4,5 | | |
| Expected result | | | 6,7 | | |
|  | | |  | | |
| Movement of Ship | When in game, pressing a directional button on the keyboard. | Up (when ship is facing right or left). | The ship will accelerate forward in the right or left direction. | Expected result | | | 8,9 | | |
| Right (when ship is facing up or down). | The ship will rotate to the right to the amount that the right arrow button is pressed. | Expected result | | | 9,10 | | |
| Down (when ship is facing right or left). | No change | Expected result | | 10,11 | | |
| Left (when ship is facing up or down). | The ship will rotate to the left to the amount that the left arrow button is pressed. | Expected result | | 12, | | |
| Up (when ship is facing up down). | The ship will accelerate forward in the up or down direction direction. | Expected result | | 12 | | |
| Right (when ship is facing right or left). | The ship will rotate to the right to the amount that the right arrow button is pressed. | Expected result | | 13 | | |
| Down (when ship is facing up or down). | No change | Expected result | | 14 | | |
| Left (when ship is facing right or left). | The ship will rotate to the left to the amount that the left arrow button is pressed. | Expected result | |  | | |
| None | The Ship will stand still facing the direction that it was left at. | Expected result | |  | | |
| Collision with walls | When in a game, directing the ship forward into any side of the wall using the forward arrow key | The up arrow key to move to the direction of the wall. | The ship will appear on the opposite side of the wall that it went through. | Expected result. | |  | | |
| Collison with Asteroid (without bullets) | When in a game, directing the ship into an Asteroid. | The up arrow key to direct the ship into an Asteroid | The ship will lose a life and re spawn in the middle of the gameplay screen with a shield around it. | Expected result | |  | | |
| Collision with Asteroids (with bullets) | When in a game, directing bullets at Asteroids | The spacebar key will execute this transaction and will fire bullets in the direction that the ship is facing. | The Asteroid will end up splitting up into smaller and smaller Asteroids until they have been destroyed. When hitting each sized Asteroid the score will be updated according to its size. Once all the Asteroids have been destroyed the next wave will start. | Expected result but the bullets are moving slowly and don’t travel that far so will need to change some variables. Also when the Asteroids have been shot close by they slowly move off together. Also the ship is not able to shoot and move at the same time. |  | | |
| Display instructions | Clicking the instructions button on the main screen. | N/A | Displays the instructions screen | Expected result |  | | |
| Display high scores | Clicking the high scores button on the main screen. | N/A | Will display the list of high scores that has happened on the game. | Expected Result |  | | |
| Going up the next wave | Destroying all the Asteroids and seeing if the wave number changes and more Asteroids appear on the screen. | Destroying the Asteroids with bullets | More Asteroids will appear on the screen and the next wave will appear. | Expected result |  | | |
| Shield appearing | Test1: When the game is loaded and a next wave happens the shield will surround the ship.  Test2: When the user presses the ‘s’ key on the keyboard the shield should come up. However you cannot press it all the time and it only lasts for a couple of seconds. | The user destroys all the Asteroids and starts the game. | The shield will come up for a short period of time. | Expected result |  | | |
| The user presses the ‘s’ key | The shield will come up for a short period of time. |

Screenshots:

|  |  |  |  |
| --- | --- | --- | --- |
| Screenshot reference | | | Screenshot |
|  | | | N:\Computing A2\Coursework\ScreenShots\Main_menu.jpg |
|  | N:\Computing A2\Coursework\ScreenShots\Start Game.jpg | |
|  | N:\Computing A2\Coursework\ScreenShots\Spaceship lost life.jpg | |
|  | N:\Computing A2\Coursework\ScreenShots\Asteroid position when loaded.jpg | |
|  | N:\Computing A2\Coursework\ScreenShots\Asteroid_position_loaded2.jpg | |
|  | N:\Computing A2\Coursework\ScreenShots\Asteroids_position_Wave_2.jpg | |
|  | N:\Computing A2\Coursework\ScreenShots\Next Wave.jpg | |
|  |  | |
|  | N:\Computing A2\Coursework\ScreenShots\Facing Right.jpg | |
| 9. |  | |
| 10. |  | |
| 11. |  | |
| 12. |  | |
| 13. |  | |
| 14. |  | |
| 15. |  | |
| 16. |  | |
| 17. |  | |
| 18. |  | |

Results of testing

By testing according to my test plan, I have made known that all the features in the game are functional. From the alpha testing I realised that I need to tweak a couple of variables in the game to make it function better. One of these variables is the distance and the speed of the bullets. With the code now, the bullets travel quite slowly in a pack and don’t go that far and the bullets are meant to travel signally and reasonably far. One thing I also saw from the alpha testing that when the big asteroid is shot, it takes a long time for the smaller Asteroids to split.

I have now fixed this problem by…….

Original separate code: