**Project Proposal**

Aim:

To create a retro-like style video game inspired by Asteroids and Space Invaders able to display high score and sort them.

Features:

* User able to move the ship left, right and can fire a bullet
* All asteroids stored in an array of objects
* Sorted score system
* Collision between bullet object and class object
* Lives system

User Group:

People aged 12-18 that enjoy video games.

Advance Higher Skills:

* Array of objects
* Use of 2-dimensional array
* Sorting Algorithm e.g. bubblesort

**Feasibility Survey**

Economic Feasibility: This project can be created using various programs these being python, Microsoft Word, Microsoft Excel, paint.NET and pyscripter thus will not cost any money to create throughout the plan, analysis, implementation or testing of this project. Furthermore, all the information and skills need for this project can be obtained for free thus this project is economically feasible.

Legal Feasibility:

* No malicious content will be created during this project so no risk of breach by the Computer Misuse Act
* All images or content will be created by me with slight inspiration and guidance from sources however nothing will be coped thus Copyright, Design and Patents Act should not be breached

Technical Feasibility:Most knowledge I require to start the Project has already been established however as the Project continues more knowledge will have been gained. I will use various resources available online to help me with this such as YouTube tutorials, pygame website, w3 schools and more (highlighted in the bibliography).

Time Feasibility:This project is planned to be completed within 40 hours specified in the Test Plan if all planning and implementation goes according to plan.

**Project Plan**

My project will follow a similar structure to the waterfall methodology:

* Planning
* Analysis
* Design
* Implementation
* Testing
* Evaluation

In order to make sure this project is completed within adequate time I will set deadline for each stage to be completed. This will act as a design for the Gantt Chart as well, Identifying the main tasks but not the sub-task within them.

Outline Project Plan

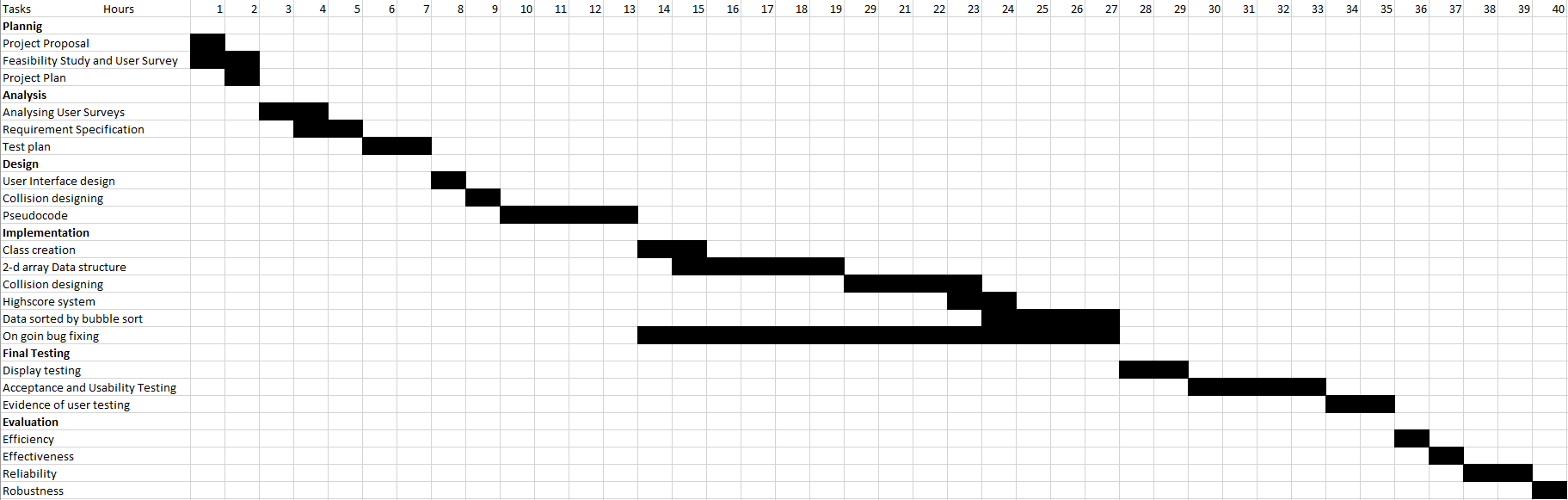
|  |  |  |
| --- | --- | --- |
| Stage | Time(approx..) | Target Date |
| Planning | 2 | 11/01/19 |
| Analysis | 5 | 21/01/19 |
| Design | 6 | 08/02/19 |
| Implementation | 14 | 28/02/19 |
| Testing | 8 | 15/03/19 |
| Evaluation | 5 | 25/03/19 |

I plan to use a Gantt Chart and a Progress Diary to help me keep track with each individual task within these steps.

Resources**:**

Throughout the development of this project, I will use a variety of different resources, these will include:

* Computer
* Python/PyScripter
* Paint.NET
* Microsoft Word and Excel
* Online resources such as scholar or the pygame website
* Modules such as pygame
* End users
* Course knowledge and skills
* ShareX for screenshots

**Gantt Chart**

|  |  |
| --- | --- |
| **Task** | **Resources Required** |
| **Planning** |  |
| Project Proposal | MS Word |
| Feasibility Study and User Survey | MS Word |
| Project Plan | MS Word, Excel for Gantt Chart |
| **Analysis** |  |
| Analysing User Surveys | MS Word, Excel for graphs |
| Requirement Specification | MS Word |
| Test plan | MS Word |
| **Design** |  |
| User Interface design/Wire frames | MS Word and Paint.NET |
| Collision designing | Paper |
| Pseudocode | MS Word |
| **Implementation** |  |
| Class creation | Python/PyScripter, School/Home computer |
| 2-d array Data structure | Python/PyScripter, School/Home computer |
| Collision designing | Python/PyScripter, School/Home computer |
| Highscore system | Python/PyScripter, School/Home computer |
| Data sorted by bubble sort | Python/PyScripter, School/Home computer |
| Ongoing bug fixing | Python/PyScripter, School/Home computer |
| **Final Testing** |  |
| Display testing | MS Word |
| Acceptance and Usability Testing | MS Word |
| Evidence of user testing | MS Word |
| **Evaluation** |  |
| Efficiency | MS Word |
| Effectiveness | MS Word |
| Reliability | MS Word |
| Robustness | MS Word |

**User Survey**

Do you like games set in space, such as Asteroids or Space Invaders?

**Yes No**

|  |
| --- |
|  |

|  |
| --- |
|  |

Do you prefer PC keyboard controls or a Console controller when playing games?

**PC Console**

|  |
| --- |
|  |

|  |
| --- |
|  |

How many hours do you spend daily playing games?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

0 1 2 3 4 5 6 7 8+

What is your favourite game genre e.g. fantasy, sci-fi, action?

|  |
| --- |
|  |

Are you interested in classic games e.g. Tetris?

**Yes No**

|  |
| --- |
|  |

|  |
| --- |
|  |

Do you prefer an unending game with a high score system or a game with no high score but a final goal?

**High score Goal**

|  |
| --- |
|  |

|  |
| --- |
|  |

**User Survey**

From these user surveys I will hopefully gain enough information to construct an accurate target audience and find out if there is really a market for this game.

Analysis

This survey was completed by 16 random second year students of a random class.

This information was analysed by me and the information I have gained from this survey has influenced this project

The answers on Q1 of the on the survey show that some 1/3 of people surveyed like games set in space. This shows that the space like retro game discussed in the aim is feasible. Furthermore, when you compared the average play time they play games to the other group of people you find the they play roughly just more that an extra hour every day.

This shows this chart shows data from question 1 to question 3 and that people who play games more often are more likely to like my game. This gives me a target audience of people who play games a lot/ 4 hours or more a day.

This pie chart with data from question 6 tells me that I should make a game with a high score instead of a final goal as more people want that and it is probably easier to implement.

**Requirements Specification**

Scope:This project should be able to allow the player to destroy asteroids, gain score, write and sort these scores to a txt file and finally use a 2-d array for the asteroids. For planning I will have to create a Test plan, Gantt Chart a User interface design and Pseudocode.

Boundaries:This project will not include html or SQL, furthermore this project will not be connected to the internet, it will be a purely offline only game. Time to complete this project is of the March 18th

End Users:The end user of this project will be teenagers aged 12-18 who like video games. Ideally they play at least 1-2 hours a day of video games and however a survey will be done to

Requirements: These will be numbered so that they can be easily referenced later in the design.

User Requirements:

1. Simple controls: Standard controls that are easy to understand
2. Easy Mechanics
3. Game doesn’t crash during play

Functional Requirements:

1. Create classes, Player/ship, Bullet, Asteroid
2. Create global variables: score, asteroid 2d array, bullets array, main menu background image, game over background image, any necessary global variables needed along implementation not planned for.
3. Asteroids spawn every few cycles
4. Large asteroids split into 2 smaller asteroids when hit by bullet
5. Bullet disappears on asteroid hit
6. Small asteroids disappear when hit
7. Score increases on asteroids hit
8. When lives == 0:
   1. Display End Screen
   2. Ask for name
   3. Display top 5 best players ranked in descending order with names and score
9. Inputs:
   1. Allow user to move ship left and right
   2. ship fires bullet when space pressed
10. Output:
    1. Main menu should display; score, lives, background, player ship
    2. End screen should display; top 5 scores, game over background

**Test Plan**

This project will be tested using these types of testing:

* Iterative testing
* Component
* Acceptance(Beta)
* Final/End-user testing

Iterative/Component Testing

I will use component testing to test each and individual component highlighted in the Functional Specifications. I will test and implement each component 1 after the other. To do test this I will fill out a sheet I created only ticking it and moving on to the next step when all prior boxes have been ticked. This will make sure that the functions/procedure work before I move on to the next one. Once this sheet is filled out, I shall begin final testing.

Iterative testing is to be used to ensure each component works well with one another for example the Collison between the asteroid and the bullet. This is required because if all the components don’t work well enough together then the entire program can crash.

Component testing include (python code):

1. Display main menu (see wireframe design)
2. Create classes Player, Asteroid, Bullet
3. Asteroid classes to be stored in a 2-d array of asteroids
4. Display endgame screen when lives == 0
5. Read and bubblesort scores from a file

Iterative Testing:

1. Bullet fired from Player.
2. Collision between Asteroid and Bullet as well as player score increase as a result.
3. Bubble sort player score with txt file of other scores and display top 5 players with most score.

How Iterative/Component Testing will be recorded

|  |  |  |
| --- | --- | --- |
| **Iterative** | **Component** | **Done** |
| 4.Create Classes | Player |  |
| Bullet |  |
| Asteroid |  |
| 5.Create globals | Score |  |
| 2d Array of Asteroids |  |
| Bullets array |  |
| Main menu background image |  |
| Game-over background image |  |
| 13a.On launch display (according to user interface design) | Main menu background image |  |
| Score |  |
| Lives |  |
| Player Ship |  |
| 12.Inputs | User can move ship left and right |  |
| Ship fires bullet when space pressed |  |
| Ask for name when lives reach 0 (input validation no more or less than 5 character inputted) |  |
| 6-7.Asteroid | Asteroids spawn every few seconds |  |
| Large asteroids split into 2 smaller asteroids when hit |  |
| 8-10.Collision | Bullet disappears on asteroid hit |  |
| Small asteroids disappear when hit |  |
| Score increases on asteroids hit |  |
| 11,13b.When lives reach 0 | Display End Screen |  |
| Display top 5 best players ranked in descending order with names and score |  |

Acceptance and Usability Testing

For acceptance and usability testing I will ask four end users within my computing class doing various projects to test my solution. Two individuals who were making a game like me using computers, one individual who made a website for their project and someone who never done a project e.g. teacher. Using these end users this will allow me to test both usability and the acceptance to be tested/give-feedback on different parts of the program.

For the testing, the end user will be given a series of tasks and be asked to comment on the usability of the program. This task will highlight the robustness, functionality and efficiency of the program.

Final/End-user Testing

For the final testing I will carry it out myself make sure no bugs were able to go un-noticed with inspiration taken from acceptance testing comments. Furthermore, I will take screenshots for evidence of each iterative/component working

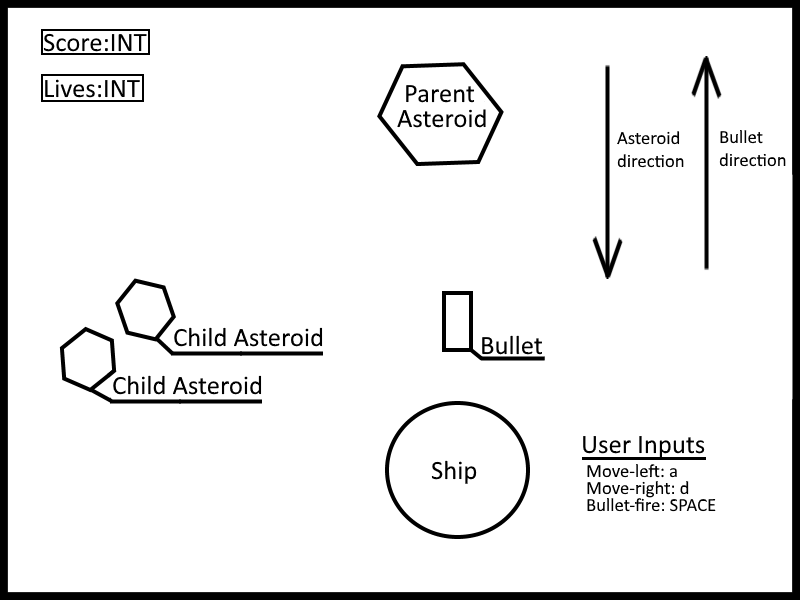
|  |
| --- |
| Final Testing Table |
| 13a.Main menu displays on launch  Screenshot Evidence of working: |
| 12.Inputs work  Screenshot Evidence of working: |
| 6-7.Asteroid functions  Screenshot Evidence of working: |
| 8-10.Collision  Screenshot Evidence of working: |
| 11,13b.Display End Screen and Display top 5 best players ranked in descending order with names and score when lives reach 0  Screenshot Evidence of working: |
| Performance Testing  Screenshot Evidence of working:  The program never goes bellow 29.00fps out of 30.00 fps |

Expected Outputs

|  |
| --- |
| Final Testing Plan |
| 13a.Main menu displays on launch  Final Product User-Interface Design |
| 12.Inputs work  Screenshot Evidence of working:  Player should move left(a)  On start When “a” was pressed    Player should move right(d) when pressed  On start When “d” was pressed  When the player presses space and asteroid should spawn on the characters cutting position.  Before space pressed After space pressed  Asks for name when lives reach 0  Just before reaching 0 After reaching 0  Input Validation no more or less than 4 characters for name  Before wrong input After wrong input |
| 8-10.Collision  Screenshot Evidence of working:  Child asteroids should completely disappear when hit. After some testing it does  Before collision After collision  Score increases on asteroid hit. After testing it, it does  Before collision After collision  Wen the bullet hits the asteroid, the bullet should be deleted. After some testing it does  Before collision After collision |
| 6-7.Asteroid functions  Screenshot Evidence of working:  An asteroid should spawn every second or so at the start.  On launch After a few seconds    When a bullet hits a parent asteroid the asteroid should split into 2 smaller child asteroids.  Before collision After collision |
| 11,13b.Display End Screen, ask for name and Display top 5 best players ranked in descending order with names and score when lives reach 0  Screenshot Evidence of working:  I tested my game to see if the end game screen matches the criteria set by the requirements specifications. It follows the requirements meeting all criteria set in 11b.  Except instead of top 5 players it top 4 as that’s what the end users suggested in the Acceptance and Usability testing.  Final Product User-Interface Design |
| Performance Testing  Screenshot Evidence of working:  The program never goes bellow 29.00fps out of 30.00 fps  Game running normally |

**Interface Design**

Start playing Screen



User Inputs

When user presser “d” ships x will increase gradually, moving the ship right. When user presser “a” ships x will decrease gradually, moving the ship left. When “space” is pressed the Bullet class will be fired from the x and y of the Ship class. If “w” is pressed the power up class is activated

Score

Keeps track of the users score and displays it. Score increased when bullet hits an Asteroid class

Lives

Starts at 3 and decreases for every asteroid that passes of the bottom of the screen.

Ship

When user inputs are met will move left or right or will spawn bullet at the position of its self.

Parent Asteroid

If bullet hits the Parent Asteroid it will be deleted, and two Child Asteroids should replace it and player Score will be increased.

Child Asteroid

Just before Parent Asteroid is deleted the Child Asteroids will move to its position. Once the Child Asteroid it hit by a bullet it will be deleted and score will be increased.

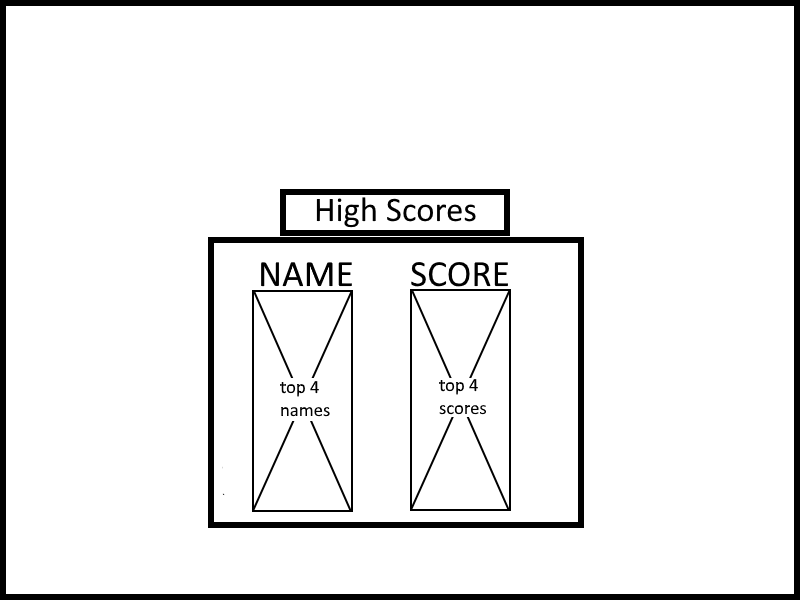
Bullet

If bullet hits any asteroid the Asteroid will be deleted as well as the bullet.

Input Validation

When the user presses space the bullet should fire, when the user presses “a” he should move left when the player presses “d” he should move right.

End Game Screen



Input Validation

At the end of the game the player should be asked to input a name, this name should be a maximum and minimum of 4 characters long. If the name entered is to long or to short the user should be asked to re-enter a name within 4 characters, the name will only be stored and will stopped being asked once the name is 4 characters long.

**Data Structure**

Pseudocode

1. Global Variables
   1. screen\_width = 800
   2. screen\_bottom = 600
   3. astEntire\_array = empty array
   4. bullets\_array = empty array
   5. ticks = 0
   6. tock = 60
   7. total\_score = 0
   8. total\_lives = 3
   9. already\_pressed = False
   10. stary = main menu background image
   11. game\_over\_screen = the game over screen image
2. Create Classes:
   1. Player
      1. Initialization
         1. Define x
         2. Define y
         3. Define png image used
      2. Create drawing procedure called draw
         1. Draw the player image corresponding to the x and y
   2. Bullet
      1. Initialization
         1. Define x
         2. Define y
         3. Define png image used
         4. Define the speed the bullet will move at
         5. Define what is the right side of the class using image size
      2. Create procedure called event
         1. y = y – speed
         2. Draw the bullet image corresponding to the x and y
   3. Asteroid
      1. Initialization
         1. Define x
         2. Define y
         3. Define the horizontal velocity of the asteroid as vx
         4. Define the horizontal velocity of the asteroid as vy
         5. Pass in a random number
         6. Define png image using the random number passed in
         7. Define what is the right-side of the class using image size
         8. Define the height of the class using image size
         9. Define whether the Asteroid is hidden or not
      2. Create drawing procedure called draw
         1. If the hidden == False:
            1. Draw the asteroid corresponding to the x and y
      3. Create a procedure called move
         1. If the hidden == False:
            1. Set y to the current y + vy
            2. Set x to the current x + vx
      4. Create a procedure called collision
         1. If the hidden == False:
            1. Sets height to the image height + the current y
            2. Set right-side to the image width + the current x
            3. To keep the asteroid from going off screen do
            4. If x <= 0:

Set x to 0

1. Procedures/Functions
   1. asteroids\_spawn(ticks,tock,astEntire\_array)
      1. astG\_array as empty array
      2. ticks += 1
      3. tock -= 0.01
      4. if ticks >= tock:
         1. Spawn in random parent asteroid and the two corresponding children asteroids
         2. Append them all to an array called astG\_array
         3. Append astG\_array to astEntire\_array array
         4. ticks = 0
      5. return ticks, tock, astEntire\_array
   2. key\_presses(already\_pressed,bullets\_array)
      1. if key “a” is pressed change player.x -= 15
      2. if key “d” is pressed change player.x += 15:
      3. if already\_pressed == False and “space” is pressed:
         1. already\_pressed = True
         2. bullet = Bullet((player.x+30),(player.y))
         3. bullets\_array.append(bullet)
      4. elif already\_pressed == True and key[pygame.K\_SPACE] == False:
         1. already\_pressed = False
      5. return already\_pressed,bullets\_array
   3. collision(bullets\_array,astEntire\_array,total\_score)
      1. loop\_end = False
      2. for i in length of bullets\_array:
         1. for j in range length astEntire\_array
            1. If parent.hidden == false and if a parent asteroid is hit by a bullet

parent asteroid

set hidden to true

child asteroid 1

set hidden to false

set x to the same as parent

set horizontal velocity to -0.5

set y to the as the parent

child asteroid 2

set hidden to false

set x to the same as parent but ad the width of child 1

set horizontal velocity to +0.5

set y to the as the parent

total\_score += 10

delete current bullet from bullets\_array

loop\_end = True

end j loop

* + - * 1. If parent.hidden == True and if child asteroid 1 is hit by a bullet

set hidden to true

total\_score += 10

delete current bullet from bullets\_array

loop\_end = True

end j loop

* + - * 1. If parent.hidden == True and if child asteroid 2 is hit by a bullet

set hidden to true

total\_score += 10

delete current bullet from bullets\_array

loop\_end = True

end j loop

* + - * 1. if loop\_end == True:

end i loop

* + 1. return bullets\_array, astEntire\_array, total\_score
  1. lose\_lives(astEntire\_array,total\_lives):
     1. for m in length of astEntire\_array:
        1. for n in length of astEntire\_array[m]:
           1. if astEntire\_array[m][n].y >= screen\_bottom:

total\_lives -= 1

for i in range(3):

astEntire\_array[m][i].hidden = True

end n loop

* + 1. return total\_lives
  1. bullet\_delete():
     1. For each bullet in bullets\_array
        1. if bullet.y <= 0:
           1. delete bullet
           2. end loo
  2. def game\_text(total\_lives,total\_score):
     1. font = adequate font
     2. Draw total\_score on top of background with font
     3. Draw total\_lives on top of background bellow score with font
  3. events(astEntire\_array,bullets\_array):
     1. Draw player.image
     2. For every bullet in bullets\_array:
        1. bullet.update\_position()
        2. bullet.draw()
        3. bullet.collision()
     3. For every array in astEntire\_array:
        1. for each item in the array: #should be 3
           1. asteroid.update\_position()
           2. asteroid.draw()
           3. asteroid.collision()
     4. For every array in astEntire\_array:
     5. if all items in array.hidden == true
        1. delete that array from astEntire\_array:
        2. end loop
  4. read\_data\_from\_file()
     1. open file as read
     2. read data line by line
        1. for each comma split value the value by name and score
        2. array = [name,each\_score]
        3. append array to the scoreboard\_array
     3. close file
     4. return scoreboard\_array
  5. name\_and\_score(scoreboard\_array, total\_score):
     1. ask user to input name
     2. array = [name,total\_score]
     3. append array to the scoreboard\_array
     4. return scoreboard\_array
  6. bubblesort\_data(scoreboard\_array):
     1. Repeat the sort code for all pos in array
        1. Repeat the sort code for all pos in array
           1. If current array item is smaller than the next item in the array swap their position using temporary variables
     2. return scoreboard\_array
  7. write\_data\_from\_file(scoreboard\_array):
     1. open file as write
     2. for each array in scoreboard\_array:
        1. write the first item in the current array to the file
        2. write "," to the file
        3. write the second item in the current array to the file
        4. take a new line in the file
     3. close file
     4. return scoreboard\_array

1. While game\_playing:
   1. If close button clicked set game\_playing to false
   2. Draw background
   3. ticks, tock, astEntire\_array = asteroids\_spawn(ticks, tock, astEntire\_array)
   4. already\_pressed, bullets\_array = key\_presses(already\_pressed, bullets\_array)
   5. bullets\_array, astEntire\_array, total\_score = collision(bullets\_array, astEntire\_array, total\_score)
   6. total\_lives = lose\_lives(astEntire\_array, total\_lives)
   7. bullet\_delete()
   8. game\_text(total\_lives, total\_score)
   9. events(astEntire\_array, bullets\_array)
   10. if total\_lives <= 0:
       1. scoreboard\_array = read\_data\_from\_file()
       2. scoreboard\_array = name\_and\_score(scoreboard\_array, total\_score)
       3. scoreboard\_array = bubblesort\_data(scoreboard\_array)
       4. write\_data\_from\_file(scoreboard\_array)
       5. while total\_lives <= 0 and game\_playing:
          1. Display the top 5 players with there names and scores sorted in descending order

**Implementation**

Python Code

# Astroids Game

# pygame initialisation

import pygame

import random

pygame.font.init()

pygame.init()

#Defines Screen Size

screen\_width = 800

screen\_bottom = 600

#Sets Screen Size

screen = pygame.display.set\_mode((screen\_width,screen\_bottom))

clock = pygame.time.Clock()

#Sets Name of game as Asteroids

pygame.display.set\_caption("Asteroid Defense Game")

#Global------------------------------------------------------------------------

game\_playing = True

#===2-d Array====

#astEntire\_array is a 2-d arrayof asterods

#arrays get appended in the asteroids spawn function

astEntire\_array = []

bullets\_array = []

current\_time = 0

time\_until = 180

total\_score = 0

total\_lives = 3

already\_pressed = False

stary = pygame.image.load("background\_stars.png").convert\_alpha()

stary = pygame.transform.scale(stary,(screen\_width,screen\_bottom))

game\_over\_screen = pygame.image.load("game\_over\_screen.png").convert\_alpha()

game\_over\_screen = pygame.transform.scale(game\_over\_screen,(screen\_width,screen\_bottom))

#------------------------------------------------------------------------------

#Creats a record called Ship

class Ship:

#Constructor, defines the properties used within the method in the record

def \_\_init\_\_(self):

#Loads the png image file to be used for the ship

self.image = pygame.image.load("Asteroid\_blaster3.png").convert\_alpha()

#Defines the ships x

#Starts in the middle

self.x = (screen\_width/2)

#Defines the ships y

self.y = screen\_bottom - self.image.get\_height()

#Creats a method called draw

def draw(self):

#Draws ships corrosponding to the position, which was defined in the constuctor

screen.blit(self.image,(self.x,self.y))

#Creates a record called Bullet

class Bullet:

#Constructor, defines the properties used within the method in the record

#Also passes in the players/ships x and y

def \_\_init\_\_(self,x,y):

#Loads the png image file to be used for the bullet

self.image = pygame.image.load("blue\_bullet.png").convert\_alpha()

self.image = pygame.transform.scale(self.image,(20,40))

#Defines the bullets x as the x passed in

self.x = x

#Defines the bullets y as the y passed in

self.y = y

#Defines the the speed of the bullet to be used later on

self.speed = 10

#Defines the right side to be used later on

self.rightside = 0

#Creates a procedure called update

def update\_position(self):

#Takes the bullets speed from the bullets y value

#which causes the bullet to move upwards

self.y -= self.speed

#Creats a method called draw

def draw(self):

#Draws ships corrosponding to the positions(x,y) value

screen.blit(self.image,(self.x,self.y))

#Creates a procedure called update

def collision(self):

#Rightside is the bullet images bottom + the bullets x position

self.rightside = self.image.get\_height() + self.x

#Creats a record called Asteroid

class Asteroid:

#Constructor, defines the properties used within the method in the record

#Also passes in the x,i,j,hidden

def \_\_init\_\_(self,x,ast\_num,random\_num,hidden):

self.random\_num = random\_num

self.vx = 0

#Defines vy as as a random real number between 0.5 and 2.0

self.vy = 1

#Defines hidden as the hidden passed in

self.hidden = hidden

#Loads the png image file to be used for the Asteroid and uses the i and j

#property to determine what png will be choosen

self.Ast = pygame.image.load("Asteroid\_"+str(self.random\_num) + str(ast\_num)+".png").convert\_alpha()

#Defines bottom as 0 which will be changed later on in the program

self.bottom = 0

#Defines rightside as 0 which will be changed later on in the program

self.rightside = self.Ast.get\_width()

#Defines the x as the x passed in

self.x = x - self.rightside

self.y = 0 - self.Ast.get\_height()

#Creats a procedure called draw

def draw(self):

#When hidden == False DO

if self.hidden == False:

#Draws asterids corrosponding to the x and y properties

screen.blit(self.Ast,(self.x,self.y))

## pygame.draw.line(screen, (100, 200, 255), (self.x, self.bottom), (self.rightside, self.bottom), 1)

#Creates a procedure called move

def move(self):

#When hidden == False DO

if self.hidden == False:

#Changes the y value by its y value plus its vy value

#which causes the Asteroid image to be drawn slightly further down

self.y = self.y +self.vy

self.x = self.x + self.vx

#Creates a procedure called collison

def collision(self):

#Sets the bottom property to the asteroid png images bottom + the asteroids y

self.bottom = self.Ast.get\_height() + self.y - 20

#Sets the rightside property to the asteroid png images width

self.rightside = self.Ast.get\_width() + self.x

if self.x <= 0:

self.x = 0

player = Ship()

########################################################################################################################################

########################################################################################################################################

#===2-d Array====

def asteroids\_spawn(current\_time,time\_until,astEntire\_array):

#--asteroid\_spawn---

#sets an empty array to be filled with asterid classes

#and will ultimatly be appended to astEntire\_array

astG\_array = []

current\_time += 1

#time\_until == 60 at start

#Every time the program cycles time\_until will decrease

#meaning over time asteroids will spawn in more rapidly

time\_until -= 0.02

if current\_time >= time\_until:

#This will create a new set of asteroids

#Their is 4 random parent asteroids that it could be

#This determine which one it will be

randoms = str(random.randint(1,4))

#Parent asteroid

astG\_array.append(Asteroid(random.randint(1,screen\_width),1,randoms,False))

#Child asteroid 1 is decided by the parent asteroid

astG\_array.append(Asteroid(random.randint(1,screen\_width),2,randoms,True))

#Child asteroid 2 is decided by the parent asteroid

astG\_array.append(Asteroid(random.randint(1,screen\_width),3,randoms,True))

astEntire\_array.append(astG\_array)

#Resets current\_time to 0

current\_time = 0

return current\_time,time\_until,astEntire\_array

########################################################################################################################################

########################################################################################################################################

def key\_presses(already\_pressed,bullets\_array):

#---presskeys---

#Creats a varible that calls the pygame method "key"

key = pygame.key.get\_pressed()

#When "a" key is pressed DO

if key[pygame.K\_a]:

#Deducts 10 from the ships x

#meaning when its drawn again it will be drawn more to the left

player.x -= 15

#When "d" key is pressed DO

if key[pygame.K\_d]:

#Adds 10 to the ships x

#meaning when its drawn again it will be drawn more to the right

player.x += 15

#When "SPACE" key is pressed AND already\_pressed == True DO

if already\_pressed == False and key[pygame.K\_SPACE]:

#Sets the boolean already\_pressed to false

already\_pressed = True

#Calls the bullets\_spawn procedure that is created out with the ship record

#Creats a procedure called bullets\_spawn

#To get bullet x, I added the ships x position + the shps width/2

bullet = Bullet((player.x + (player.image.get\_width()/2)),(player.y))

bullets\_array.append(bullet)

elif already\_pressed == True and key[pygame.K\_SPACE] == False:

already\_pressed = False

return already\_pressed,bullets\_array

########################################################################################################################################

########################################################################################################################################

def collision(bullets\_array,astEntire\_array,total\_score):

loop\_end = False

#---collisions---

for i in range(len(bullets\_array)):

for j in range(len(astEntire\_array)):

#1.Checks if a parent asteroid is hidden if not

#2.checks if the bullets y is less than or equal to the bottom of the asteroid

#(the bottom being its y + its image height)

#3.checks if the bullets x is less than or equal to the right side of the asteroid

#(the rightside being its x + its image width)

#4.checks if the bullets rightside is more than or equal to the x of the asteroid

#(the rightside being its x + its image width)

if astEntire\_array[j][0].hidden == False and \

bullets\_array[i].y <= astEntire\_array[j][0].bottom and \

bullets\_array[i].x <= astEntire\_array[j][0].rightside and \

bullets\_array[i].rightside >= astEntire\_array[j][0].x:

#Sets the parent aseroid to hidden

astEntire\_array[j][0].hidden = True

#Sets the first child aseroid hidden

astEntire\_array[j][1].hidden = False

#Sets first child aseroid the same x as the parent

astEntire\_array[j][1].x = astEntire\_array[j][0].x

#Sets the new horizontal velocity

astEntire\_array[j][1].vx = -0.5

#Sets first child aseroid the same y as the parent

astEntire\_array[j][1].y = astEntire\_array[j][0].y

#Sets the second child aseroid to hidden

astEntire\_array[j][2].hidden = False

#Offsets the second child asteroid so it looks like they've been split in half

astEntire\_array[j][2].x = astEntire\_array[j][0].x + astEntire\_array[j][1].Ast.get\_width()

#Sets the new horizontal velocity

astEntire\_array[j][2].vx = 0.5

#Sets second child aseroid the same y as the parent

astEntire\_array[j][2].y = astEntire\_array[j][0].y

#Increases score by 10 every successful collision

total\_score += 10

#Deletes the bullet that collided with the asteroid

bullets\_array.pop(i)

#Sets the loop end to true so that it can later end the entire i loop to avoid a looping error

loop\_end = True

#Ends the j loop to avoid index out of range error

break

#1.Checks if the first set of child asteroids are hidden if not

#2.checks if the bullets y is less than or equal to the bottom of the asteroid

#(the bottom being its y + its image height)

#3.checks if the bullets x is less than or equal to the right side of the asteroid

#(the rightside being its x + its image width)

#4.checks if the bullets rightside is more than or equal to the x of the asteroid

#(the rightside being its x + its image width)

elif astEntire\_array[j][1].hidden == False and \

bullets\_array[i].y <= astEntire\_array[j][1].bottom and \

bullets\_array[i].x <= astEntire\_array[j][1].rightside and \

bullets\_array[i].rightside >= astEntire\_array[j][1].x:

#Sets hidden back to true if bullet collides with it

astEntire\_array[j][1].hidden = True

#Increases score by 10 every successful collision

total\_score += 10

#Deletes the bullet that collided with the asteroid

bullets\_array.pop(i)

#Sets the loop end to true so that it can later end the entire i loop to avoid a looping error

loop\_end = True

#Ends the j loop to avoid index out of range error

break

#1.Checks if the second set of child asteroids are hidden if not

#2.checks if the bullets y is less than or equal to the bottom of the asteroid

#(the bottom being its y + its image height)

#3.checks if the bullets x is less than or equal to the right side of the asteroid

#(the rightside being its x + its image width)

#4.checks if the bullets rightside is more than or equal to the x of the asteroid

#(the rightside being its x + its image width)

elif astEntire\_array[j][2].hidden == False and \

bullets\_array[i].y <= astEntire\_array[j][2].bottom and \

bullets\_array[i].x <= astEntire\_array[j][2].rightside and \

bullets\_array[i].rightside >= astEntire\_array[j][2].x:

#Sets hidden back to true if bullet collides with it

astEntire\_array[j][2].hidden = True

#Increases score by 10 every successful collision

total\_score += 10

#Deletes the bullet that collided with the asteroid

bullets\_array.pop(i)

#Sets the loop end to true so that it can later end the entire i loop to avoid a looping error

loop\_end = True

#Ends the j loop to avoid index out of range error

break

#Will end the i loop if loop end == true

if loop\_end == True:

break

return bullets\_array,astEntire\_array,total\_score

########################################################################################################################################

########################################################################################################################################

def lose\_lives(astEntire\_array,total\_lives):

#This procedure will take one away from lives when an asteroid goes off screen

for m in range(len(astEntire\_array)):

for n in range(len(astEntire\_array[m])):

if astEntire\_array[m][n].y >= screen\_bottom:

#Remove 1 from total lives

total\_lives -= 1

for i in range(3):

astEntire\_array[m][i].hidden = True

break

return total\_lives

########################################################################################################################################

########################################################################################################################################

def bullet\_delete():

#Delets bullet when it reaches top

for i in range(len(bullets\_array)):

if bullets\_array[i].y <= 0:

#Delets that item in the array

bullets\_array.pop(i)

break

########################################################################################################################################

########################################################################################################################################

def game\_text(total\_lives,total\_score):

#TEXT

myfont = pygame.font.SysFont('Ultra Serif SF', 36)

score\_text = myfont.render(('Score:'+ str(total\_score)), False, (255, 255, 255))

lives\_text = myfont.render(('Lives:'+ str(total\_lives)), False, (255, 255, 255))

#Writes the text to the screen

screen.blit(score\_text,(0,0))

screen.blit(lives\_text,(0,30))

########################################################################################################################################

########################################################################################################################################

def events(astEntire\_array,bullets\_array):

#Draws player on screen

player.draw()

#Runs bullets basic functions

for i in range(len(bullets\_array)):

bullets\_array[i].update\_position()

bullets\_array[i].draw()

bullets\_array[i].collision()

#---asteroids\_events---

#Runs asteroids basic functions

for j in range(len(astEntire\_array)):

#required to loop through each item in the array

#1 parent 2 children for each array

for i in range(3):

astEntire\_array[j][i].move()

astEntire\_array[j][i].collision()

astEntire\_array[j][i].draw()

#Dels asteroid if all set true

for i in range(len(astEntire\_array)):

if astEntire\_array[i][0].hidden == True and \

astEntire\_array[i][1].hidden == True and \

astEntire\_array[i][2].hidden == True:

astEntire\_array.pop(i)

break

########################################################################################################################################

########################################################################################################################################

def read\_data\_from\_file():

#<-----HIGHSCORE TEXT FILE CREATOR----->

scoreboard\_array = []

#LeaderBoard - Saving data to file

#1.--READ DATA--

file = open("leaderboard.txt", "r")

# Read each line and seperate out values using ","

for line in file:

# Removes return value

line = line.strip()

# Splits Values by commas

name,each\_score = line.split(",")

each\_score = int(each\_score)

#Save each line into a 2d array

array = [name,each\_score]

scoreboard\_array.append(array)

file.close()

return scoreboard\_array

########################################################################################################################################

########################################################################################################################################

def name\_and\_score(scoreboard\_array,total\_score):

#2--Input and save name--

name = input(str("Insert name: "))

while len(name) > 4 or len(name) <4:

name = input(str("Insert name that is 4 characters long: "))

array = [name,total\_score]

scoreboard\_array.append(array)

return scoreboard\_array

########################################################################################################################################

########################################################################################################################################

#===BUBBLE-SORT====

def bubblesort\_data(scoreboard\_array):

#3--SORT THE DATA--

if len(scoreboard\_array) > 2:

# Repeat the sort code for all pos in array

for outer in range(len(scoreboard\_array)-1, 0, -1):

# Loops through array to be sorted

for i in range(outer):

# If current array item is bigger than

if scoreboard\_array[i][1] < scoreboard\_array[i+1][1]:

# Swap using temp var

temp\_swap = scoreboard\_array[i]

scoreboard\_array[i] = scoreboard\_array[i+1]

scoreboard\_array[i+1] = temp\_swap

return scoreboard\_array

########################################################################################################################################

########################################################################################################################################

def write\_data\_from\_file(scoreboard\_array):

#4.--Save highscores to file--

# Open the file to write

file = open("leaderboard.txt", "w")

for i in range(len(scoreboard\_array)):

# Write the data score array to the text file

file.write((scoreboard\_array[i][0]))

file.write((","))

file.write(str(scoreboard\_array[i][1]))

file.write("\n")

# Close the text file

file.close()

return scoreboard\_array

########################################################################################################################################

########################################################################################################################################

#----------------------------------------------------------------------------------------------------------------------------------------

#First while starting the program begins

while game\_playing:

# If the player quits

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_playing = False

#Updates screen !important!

pygame.display.flip()

#The fps -max30

clock.tick(30)

#Draws background first

screen.blit(stary,(0,0))

#Spawns asteroids if ready

current\_time,time\_until,astEntire\_array = asteroids\_spawn(current\_time,time\_until,astEntire\_array)

#Inputs for game

already\_pressed,bullets\_array = key\_presses(already\_pressed,bullets\_array)

#This deals with the collsion between the bullets and the asteroids

bullets\_array,astEntire\_array,total\_score = collision(bullets\_array,astEntire\_array,total\_score)

#When asteroids reach bottom of scree a life is lost

total\_lives = lose\_lives(astEntire\_array,total\_lives)

#Delets a for the array if it reaches top of screen

bullet\_delete()

#Displays the players current lives and score on screen

game\_text(total\_lives,total\_score)

#Runs all the events/methods for the classes

events(astEntire\_array,bullets\_array)

#When the lives reaches the following will run

if total\_lives <= 0:

scoreboard\_array = read\_data\_from\_file()

scoreboard\_array = name\_and\_score(scoreboard\_array,total\_score)

scoreboard\_array = bubblesort\_data(scoreboard\_array)

write\_data\_from\_file(scoreboard\_array)

#Game over Screen

while total\_lives <= 0 and game\_playing:

screen.blit(game\_over\_screen,(0,0))

middle = 250

#---TEXT---

for i in range(4):

if i < len(scoreboard\_array):

myfont = pygame.font.SysFont('Ultra Serif SF', 36)

allscore\_text = myfont.render(str(scoreboard\_array[i][0]) + " " + str(scoreboard\_array[i][1]), False, (0, 0, 0))

middle += 50

#Writes the text to the screen

screen.blit(allscore\_text,((screen\_width/2)-120,middle))

pygame.display.flip()

# If the player quits

for event in pygame.event.get():

if event.type == pygame.QUIT:

game\_playing = False

#---------------------------------------------------------------------------------------------------------------------------------------------

pygame.quit()

Iterative/Component Test Results

|  |  |  |
| --- | --- | --- |
| **Iterative** | **Component** | **Done** |
| 4.Create Classes | Player | Yes |
| Bullet | Yes |
| Asteroid | Yes |
| 5.Create globals | Score | Yes |
| 2d Array of Asteroids | Yes |
| Bullets array | Yes |
| Main menu background image | Yes |
| Game-over background image | Yes |
| 13a.On launch display (according to user interface design) | Main menu background image | Yes |
| Score | Yes |
| Lives | Yes |
| Player Ship | Yes |
| 12.Inputs | User can move ship left and right | Yes |
| Ship fires bullet when space pressed | Yes |
| Ask for name when lives reach 0 (input validation no more or less than 4 character inputted) | Yes |
| 6-7.Asteroid | Asteroids spawn every few seconds | Yes |
| Large asteroids split into 2 smaller asteroids when hit | Yes |
| 8-10.Collision | Bullet disappears on asteroid hit | Yes |
| Small asteroids disappear when hit | Yes |
| Score increases on asteroids hit | Yes |
| 11,13b.When lives reach 0 | Display End Screen | Yes |
| Display top 5 best players ranked in descending order with names and score | Yes |

**Acceptance and Usability Testing**

|  |  |
| --- | --- |
| Task | Comments |
| Main game displayed on launch |  |
| Move ship left and right with keys “a” and “d” |  |
| Fire bullet with space bar |  |
| Hit an asteroid with a bullet which should increase score |  |
| Get at least 100 score |  |
| Play until lives reaches 0 and then the game-over screen should be displayed |  |
| Input name |  |
| The top 5 scores should be displayed |  |
| Close the game by clicking top right X |  |
| Any further comments? |  |

Please perform the tasks described below and note and comment you have about the usability, efficiency, functionality and robustness of the video-game

**Final Testing**

|  |
| --- |
| Final Testing Table |
| 13a.Main menu displays on launch  Screenshot Evidence of working:  I tested my game to see the main menu matches the requirements specifications as well as the user interface design.  Final Product User-Interface Design |
| 12.Inputs work  Screenshot Evidence of working:  Player should move left(a)  On start When “a” was pressed    Player should move right(d) when pressed  On start When “d” was pressed    When the player presses space and asteroid should spawn on the characters cutting position.  Before space pressed After space pressed      Asks for name when lives reach 0  Just before reaching 0 After reaching 0    Input Validation no more or less than 4 characters for name  Before wrong input After wrong input |
| 6-7.Asteroid functions  Screenshot Evidence of working:  An asteroid should spawn every second or so at the start.  On launch After a few seconds    When a bullet hits a parent asteroid the asteroid should split into 2 smaller child asteroids.  Before collision After collision |
| 8-10.Collision  Screenshot Evidence of working:  Child asteroids should completely disappear when hit. After some testing it does  Before collision After collision    Score increases on asteroid hit. After testing it, it does  Before collision After collision      When the bullet hits the asteroid, the bullet should be deleted. After some testing it does  Before collision After collision |
| 11,13b.Display End Screen, ask for name and Display top 5 best players ranked in descending order with names and score when lives reach 0  Screenshot Evidence of working:  I tested my game to see if the end game screen matches the criteria set by the requirements specifications. It follows the requirements meeting all criteria set in 11b.  Except instead of top 5 players it top 4 as that’s what the end users suggested in the Acceptance and Usability testing.  Final Product User-Interface Design |
| Performance Testing  Screenshot Evidence of working:  The program never goes bellow 29.00fps out of 30.00 fps  Game running normally |

**Evaluation**

Overall, my completed project/solution me all requirements set by the sqa and included many advanced higher techniques which means this project was successful in meeting its requirements. Furthermore, my project successfully fits the project proposals aim which was to create a retro-like style video game inspired by Asteroids and Space Invaders able to display high score and sort them. This project was completed within the deadline and is suitable for the end users of the solution proved by the usability and acceptance survey.

Requirements Specification

I believe that the final project has fulfilled the user requirements, functional requirements in the requirements specification and I’ve fulfilled the user surveys as well.

The final solution meets the user requirements noted because the user requirements were shown to be met in the acceptance and usability end users who tested them. Therefore, the user requirements were met.

The completed solution also meets the functional requirements because the table for the iterative/component testing was filled out one by one when making the solution ensuring every iterative/component in the functional requirements were met.

From testing all aspects of the program, I’ve found that the program was efficient, never dropping below 29fps show in final testing and is rather robust, the final program never crashes when playing the game or from inputting to many or to little characters when inputting name.

The name inputted is validated and stored in a text file safely then read back into the program once rebooted.

Final Testing

During final testing I tested all functional requirements showing how the program works when ran shows that all the functional requirements were met. Furthermore, to test the robustness I inputted several varying values for the name input.

Corrective maintenance was used when I was testing the bullet firing, capabilities. An error occurred when randomly pressing SPACE it fired two bullets instead of one. It was beneficial to this the error before it got to end user testing, I believe in order to allow users to test a fully working program.

Final testing produced no further errors and thus no more corrective maintenance was need.

Acceptance/Usability End user Testing

When deciding who to use for acceptance testing, I chose two people who did a game for their project, one individual who done a website for their project and one individual who never done a project. This was to help me gain feedback on the usability of the game.

During the acceptance testing no errors occurred and the comments made by the user were mostly positive. However, the end users thought an improvement could be made on the Game-over screen. They believed the program would benefit from having a top 4 instead of a top 5 as they felt 5 was too many and didn’t fit very well. So, I had to adjust the program and had to adjust the final testing page because I done the final testing before the end user testing. This was useful feed back as I believed in the long run it looks better.

I believe my choice for the acceptance testing was greatly beneficial to me because, I gained feedback from a variety of users. The two people who were doing a game for their project were able to give me feedback and ultimately tested inputs more severely that the two users who where doing a project of programming. The person who done a website was able to give me more feedback on the structure of the game, and the person who never done a project was able to give me a first-hand view of what it was like to figure out the controls and inputs.

Personal Performance

I used a diary through this project to record my progress as I completed each task. This was very helpful as it allowed to keep track of where I was in the project and what I was to do next as I could just look back. This helped me complete my project within the deadline, with all requirements met.

At first, I hated the documentation, I saw no need for it, I just wanted to program, however my teacher told me how important it was it is in the real world for programmers and overtime documentation began to grow on me. I now fully understand why documentation is so important and I will be sure to document everything I do in future projects.

When I started this project, I wasn’t sure what my game was going to look like or fully function, but after writing the functional requirements and creating a table for iterative/component testing this helped me visualize the stages I was to pseudocode and program my code in. Furthermore, the functional requirements also helped me design a user interface design which allowed me to create the sprites and see how they would/should interact with one another.

Over the course of this project, I have learned a lot. I feel a lot more confident with documentation now. For the implementation, many errors, false fixes and several sessions of crying later I finished the implementation. From my previous higher knowledge of how classes work (which was somewhat lacking) I now feel fully confident on how to uses classes in programming, from how to edit values in them, to how to call them. I also strengthened my knowledge thoroughly on how 2-d arrays work, such as how to get values from them etc. I also done a bit of overkill because my asteroids array was a 2-d array of classes which was necessary for my program however I am fully aware that using to things from one bullet point doesn’t necessary benefitted me in marks. Finally using bubble sort to sort the data allowed me to better understand how to write the bubble sort function as well as how to use it in a program.

Overall, I performed well during the project as my final solution meets all the requirements in the Requirements Specification. I have also overall greatly improved my programming skill and after this project I may start my own during the summer.

Further Developments

Most of my further developments would be to enhance the gameplay, like adding powerup or increasing the speed of the asteroids over time. Things like some, some of which highlighted in the user comments. My initial thoughts were to have this power-up that when pressed the bottom a spacecraft would fly in front of the player dropping a bomb whipping out all the asteroids on screen. However, due to time constraints this was not feasible. I also would have liked to work on the png images using in the game, I dislike the way the “ship” looks and perhaps changing the font to be more adequate and have the game scale with resolution. I believe that with proper planning and less avoidable setbacks (certain stupid errors) some of these features if not all could be implemented. However, because this is the first time I done a project this big before I am somewhat forgiving of myself.

Lessons Learned

If I were to take another project like this in the future, I would know to plan it out more and make better use of my time. It would also be faster because I would have to learn/research less I would no longer have to research how pygame works nor would I have to install the module of pygame into python (pain). Furthermore, I would know the essentials for how records/classes work as well as 2-d arrays and bubble sorting them. All was a lot of research that I wouldn’t have to do again or as thoroughly. This will give me more time to spend on things like power-up etc to increase the gameplay of these games within the same amount of time.

Implications

The intended use of this project was to only be used by me in a classroom/school or home environment, I will let my friends play the game and store their name onto it if they wanted to.

There could be some legal implications of this project how ever it only stores their score and whatever name they wanted to input within 4 character, but I believe this does break todays data protection act as no real malicious intent could be used by this data nor am, I profiteering from it. Furthermore, I made all the images my self handmade and only took slight inspiration from some areas thus the copyright, designs and patents act should not be breached. This project should not be breaking any legal implications as it will only be used and tested in school and at home on my pc.

Apart from the computer there is no economic implication of this product everything used to make or play this project was free.

The only real environmental implication from this project will be from the computer developments, the manufacturer and running of a contributes to the carbon foot print also uses a lot of energy. My game isn’t energy intensive and thus has very little environmental impact.

This project is a single player game thus there is nothing socially inherent about it apart from peers to peers comparing high scores there is nothing social.

**Bibliography**

[www.courses.scholar.hw.ac.uk/](http://www.courses.scholar.hw.ac.uk/)

Used to learn varies procedure such as bubble sort as well as classes

[www.w3schools.com](http://www.w3schools.com)

Used to learn about 2-d arrays

[www.pygame.com](http://www.pygame.com)

Used to learn how to use the pygame module for python