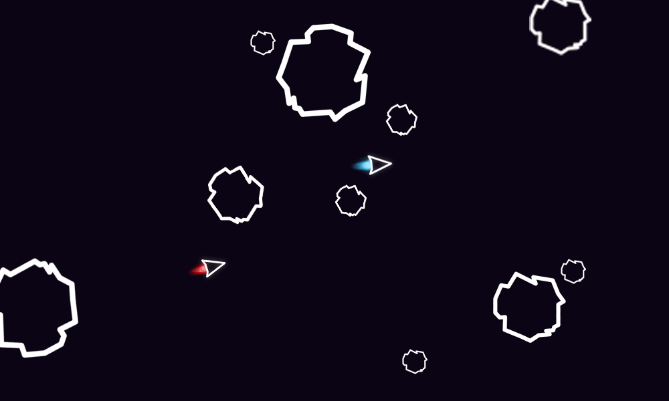
Game Project

AS91906 and AS91907 - Do a big programming project and plan it well.

# Project Selection

I have multiple ideas for what kind of game I could develop for this standard, they each require a different skillset to implement. I need to be careful with my selection so that I don’t pick a project too hard for me to implement or one that isn’t fun for players.

I drew concepts for each of the options to make them more clear and get a better understanding of what would be required to develop them. These were also useful for getting feedback from potential players.



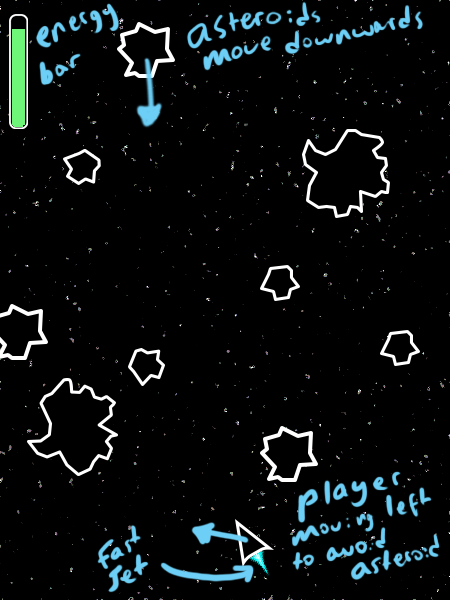
Side-scrolling two player racing game. Circuit racing game.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Side-scrolling two player racing game | Fixed camera, two player circuit racing game. | Single player asteroid shooter game.  (concept on next page) |
| Plus | Would be lots of fun for players | Also lots of fun | I understand most of the code that I will need to use for the game.  Could still be fun for players.  Difficulty can be adapted to allow lots of different skill levels.  Easy to test. |
| Minus | The need for two players would make it difficult to test code as I write it.  I don’t know how to balance moving the players relative to each other while still moving the asteroids past them, this would require a lot of nasty code. | The need for two players would make it difficult to test code as I write it.  Writing code for collisions with curves would be very difficult for me. | Could be a little less interesting |
| Interesting |  | The physics for ship control would be interesting to learn. |  |

I asked some people in the target audience for these games which they would prefer to play and got mixed answers, because I could not draw any clear conclusion from this research, I decided to pick the game which best suited my abilities.

I will develop the third option as it best suits my abilities but still provides plenty of opportunity for learning new skills. I think that I can make this concept into an entertaining game.

# Concept

I want to make an arcade style game where the player must navigate through many randomly generated obstacles as they move down the screen towards the player. The frequency of these obstacles will increase as the player continues upwards. The player can fire a thruster to move faster but cannot stop moving.

The player will be able to shoot up at obstacles and move horizontally but both will deplete the player’s ‘energy’ which regenerates at a set rate. If the player’s energy falls below a set amount their manoeuvrability will be limited, and they will not be able to shoot.

The aim of the game is to get as far through the obstacles as possible.

I will develop this game using Python3.8 and PyGame as I have experience with these, and I know that they have all the tools necessary to build this game.

## Suitability for audience

I want to make this game suitable for a wide range of ages and experience levels, so I need to ensure that the game is appropriate both for children and adults. I will also aim to use symbols in menus to make the game accessible to those that do not speak English well.

To allow players with various levels of experience to enjoy the game I want to implement difficulty options that the player can select from ranging from easy to extremely hard. I want the easiest version to be playable for children with minimal experience with games, and I want the hardest levels to challenge very experienced players.

The gameplay and control need to be intuitive to those with gaming experience and to those without. So I will need to make use of well-established conventions for gameplay and control so that I don’t confuse players that have lots of experience in other games and will expect mine to work in a certain way. I will also need to make sure that these conventions can be understood by new players without outside instructions.

# Components

Because of the program’s complexity it would be very hard to debug issues if I write the entire program before doing significant testing. To make sure that issues are manageable I need to test modules individually as I complete them. This also means that I need to plan the order in which to develop modules.

I will develop the functions for displaying the menu and game, as well as the code for switching between them. These parts will establish the overall structure of the program, so writing and testing them first will ensure that I put all the other parts of the program in the right places.

## Player

The player will be a class that inherits methods from the pygame sprite class. It will primarily handle moving and drawing the player on screen and will need the following methods:

Initialisation method to set up all the variables used by the player including placing it on the screen and reading and sprites from files.

An update position method that looks at the state of the input keys and energy bar to update the player’s position.

Drawing the player will be handled by pygame’s sprite functions and will be done each cycle in the main loop.

A method to check if the player is colliding with any obstacles. This will make use of the sprite functions to quickly check collisions against all the obstacles which will be other sprites stored in a separate container.

A destroy method called when the player collides with an obstacle. This will handle animating the players destruction and removing them from the screen.

## Obstacles

These will also be a class with inheritance from the pygame sprite class. Because there will be many obstacle objects in the game at once I need to make the obstacles as simple as possible to keep the game running smoothly.

## Menu loop

I will need a separate loop for the menu that listens to the players keyboard inputs and selects menu items accordingly. The menu loop should have two states, a main state that has three options: play, controls, and exit. Its second state should be a short list of the games key bindings.

## Game loop

This will handle playing the game. It will take user input and take actions based on it. It will call all the other functions needed for the game including updating game objects and drawing them on screen.

# Version control

As I am developing this project, I need to keep regular backups of the files and I need a way to revert to previous versions when I make mistakes.

For this I have decided to use GitHub. GitHub backs my project up to its servers and allows me to apply changes to the backups as versions. These versions can be easily reverted so If I try something that doesn’t work I can revert to a previous version and try again.

<https://github.com/Incompetentexercise/13PAD-game-project>

# Testing plan

The game’s main mechanics need to be tested to ensure that they are enjoyable for players.

## Player control

One of the key mechanics is the ship’s control, the player must always keep moving but can briefly accelerate. I have previously made a game with a very similar control mechanic but my testing of this was mostly limited to myself playing it as this was a small hobby project and was not made public. To make sure this control scheme works I will do more testing with a variety of people. I may also test a version of the game with different movement speeds to see if people find it more enjoyable. I will make sure that I get testers from multiple different demographics and experience levels

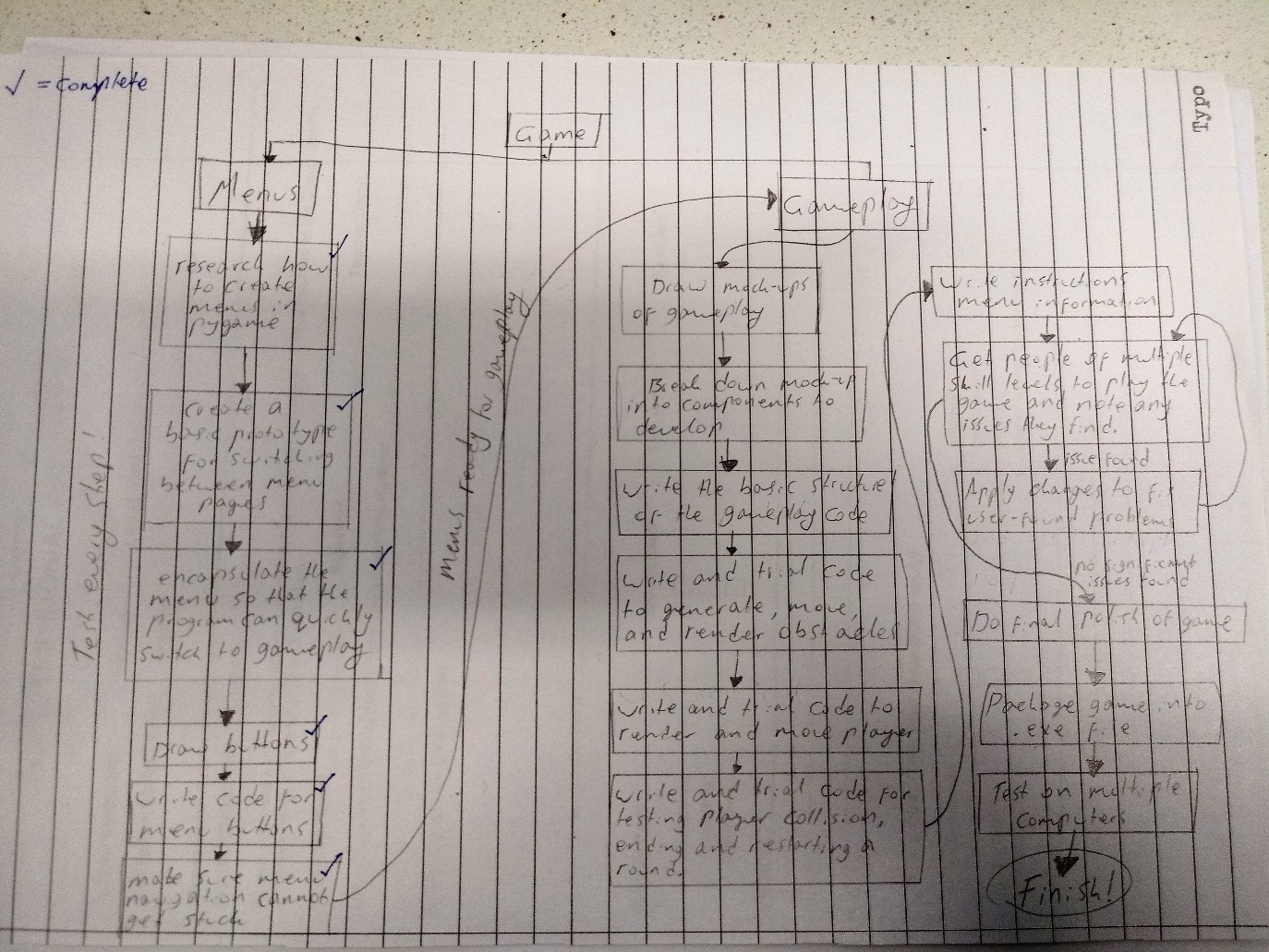
## Difficulty levels

I will need to get players from many ages and experience levels to try the game at different difficulty levels to ensure that I have enough variance in the difficulty to make the game fun for the majority of people.

## Intuitive gameplay

To make sure that the game is intuitive to a wide range of players I will ask people to play the game without giving them instructions. If they can play the game with minimal initial confusion, then the game has satisfied the requirement of being intuitive. If the players need instruction to play or enjoy the game, then I will record where they had difficulty and change the problematic elements until players find the game easy to learn.

# Overall Project plan

Due to the size of the project I need a plan for what tasks I should get done and in what order. So I drew this to show the flow of the project. This gives me a much better idea of how much I need to get done and is handy for quickly being able to see what I should do right now.

Apologies for the shoddy picture

# Menu planning

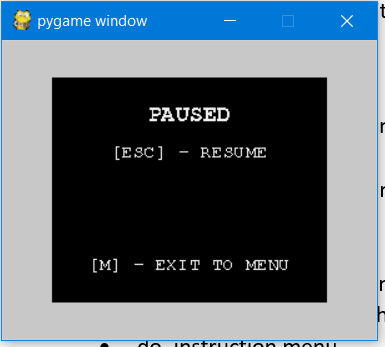
Pygame does not have a built-in way of handling menus so I need to make my own. I was unsure about how to manage all the different menu screens, so I did some research into pygame menu handling. I found this video that gave me the idea to have each menu screen as its own function that would handle rendering its unique elements and handling relevant user input from the keyboard and mouse. It also showed me that I could detect button presses by checking if the mouse pointer is over an element’s rect and then checking if the mouse button is pressed.

I decided to make the menu a class with methods for drawing each menu screen as well as supporting methods for rendering and for selecting which screen to display.

## First prototype

I will first make a very minimal menu as a prototype to work out how to switch between screens. This will also reveal the flaws in my initial plan so that I can make a better final outcome.

### Methods

* \_\_init\_\_  
  defines all the variables and gui elements needed for the menus
* do  
  looks at the menu state and then calls the method to draw that screen.  
  called externally
* do\_main\_menu  
  draws all the text for the main window and then loops through the event list and responds to relevant ones. Switches the state of the menus or game accordingly.
* do\_instruction menu  
  same as above but with different text
* do\_pause\_menu  
  “”

### Testing

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Pass** |
| Try switching between all menus | Menus are switched quickly. Every sub-menu can be exited |  |
| Go into game, pause, resume, exit to menu | Game switches states quickly and without artifacts. Switching between all menus works |  |
| Press escape on main menu | Exits program |  |

## Buttons

The first prototype used keyboard shortcuts to move between screens. This is easy to implement but not intuitive for users, so I needed to make a better interface. I want to do this by replacing keyboard shortcuts with buttons.

These buttons need to provide visual feedback to the user. I want them to have three states: idle, hover, and pressed. The hover state will make the button appear slightly different when the mouse hovers over it, this will confirm to the user that this is a button and makes the menu more interesting. The pressed state shows the button depressed for as long as the mouse button is held down for. I want the buttons to execute their linked command when the mouse button is released.

I learnt from the first prototype that pygame elements such as text take a lot of code to create so I need a good way to organize this. I will make the buttons a class that has a method for updating the button’s state, and a method for rendering the button.

I will use images for the buttons to give me the most options for their appearance. Each button will need an image for each of its states. I may need to trial multiple button designs to see which one users like the most.

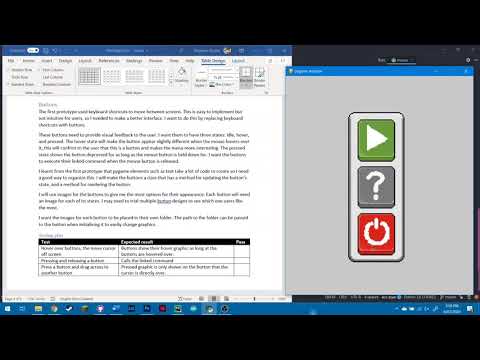
I want the images for each button to be placed in their own folder. The path to the folder can be passed to the button when initialising it to easily change graphics.

This will require a lot of changes to be made to my code and I could easily break the game so I will make a separate git branch to make these changes on. This way I can easily switch back and forth between the button and keyboard navigation versions. I will merge this back into the main branch once the changes are stable.

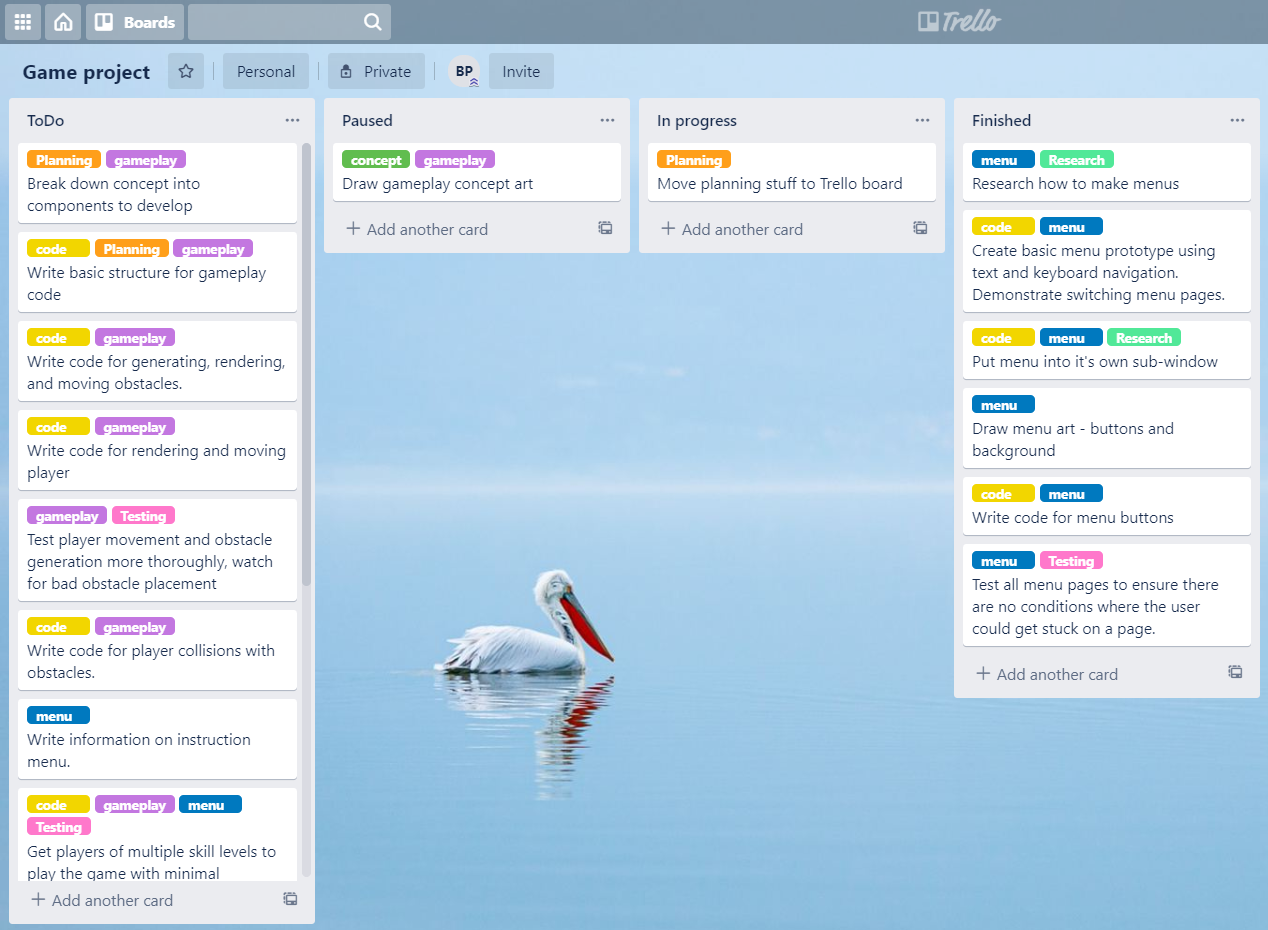
### Testing plan

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Pass** |
| Hover over buttons, the move cursor off screen | Buttons show their hover graphic as long at the buttons are hovered over. |  |
| Pressing and releasing a button | Calls the linked command |  |
| Press a button and drag across to another button. | Pressed graphic is only shown on the button that the cursor is directly over. |  |

### Menu testing video

[](https://www.youtube.com/embed/-tuKJza5L_A?feature=oembed)

# Trello Board

The paper plan I had drawn for the project was useful for initial planning but was inconvenient for working on the project. I decide

d to shift the planning to a Trello Board that would allow me to manage tasks more easily.

I added labels to the ‘cards’ so I could easily see what part of the project each card was in and what kind of work it would require.

In previous projects I have found that the In Progress list gets cluttered with partially complete, paused tasks. To avoid this issue I added a Paused list to put any tasks that I am halfway through and will come back to later.

# Obstacle planning

Each obstacle will be randomly assigned a big, medium, or small image to use for rendering and collisions. When initiated, each obstacle should apply a random rotation to the image that it is given to make the game more visually interesting. I will try assigning each obstacle a slightly different velocity but this could make the game too confusing so I will need to test this, I may choose to disable it on lower difficulty levels.

## Collisions

The sprite object in pygame has many useful methods so my obstacle class will be created as a child of the sprite class. This is especially useful for collisions as it includes the [sprite.collide\_mask(sprite1, sprite2)](https://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_mask) method which lets me check for collisions by checking whether two bitmaps are overlapping. For each obstacle I will need to run [pygame.mask.from\_surface(surface)](https://www.pygame.org/docs/ref/mask.html#pygame.mask.from_surface) to make a mask that includes all the non-transparent pixels in the obstacle image. I will also need to get a mask for the player sprite. Then I can check collisions between the player and each asteroid. If this is too slow I may need to restrict the check to only include obstacles that are far enough down the screen to hit the player.

## Overview of class

### Init

* Get methods from parent sprite class
* Assign random position along a line above the top of the screen
* Choose velocity
* Load image with transparency
* Apply rotation to image
* Get mask for image

### Update

* Move position based on velocity, apply multiplier if player is moving extra fast

### Blit

* Draw onto the surface given

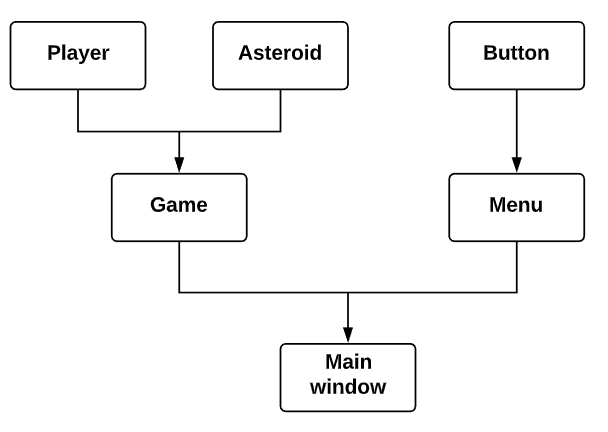
## [Asteroid generation, rendering, and movement.](https://www.youtube.com/embed/MklSzSupf6w?feature=oembed)Testing

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Pass** |
| Run obstacle test program | Obstacles do not appear or disappear while on screen |  |
| All obstacles have a rotation applied |  |
| All obstacles have a slight scale applied |  |

In the testing video a brief stutter can be seen whenever an asteroid is deleted. This is a minor issue and is much less noticeable when I increase the game’s framerate. I think this issue has something to do with the deletion messing with the for loop going through the asteroid list so it sometimes misses an asteroid. I will resolve this issue in the final polish of the game or sooner if it annoys players.

# Integration of components

So far, I have been developing the game in separate parts but I need to bring these together for the game to function as a whole. I will need to make a main file that handles setting up the window, and managing the game and menu loops. I am not currently sure how to do this so I will make a prototype version with minimal planning so that I can try out my ideas and inform myself on how to make a better main program.

To keep the program organized I decided to add another component for the gameplay. The class structure of the program is on the right. The new component Game manages all the inputs and outputs while the players is playing a round. Both the Menu and Game classes have update and blit methods so the main program can choose which needs to be shown or updated at any given time.

Because of how long some of the components are I have split them into multiple files with each file having some code to demonstrate its components. This has allowed me to develop each component individually but is now causing some problems with bringing all the components together. Some parts of the components reference global variables from inside objects, this works fine when everything is on one file but causes some issues when the component is imported to another file. The main issue was that the update method of an asteroid would attempt to delete itself from the global list of asteroids if it had gone off the bottom of the screen. This worked fine when I was testing the asteroids in their own file but when I tried to use the asteroid class from inside the Game class it was not able to access the list of asteroids. I could have changed where the asteroid class tried to get the list of asteroids from but I decided it would be more time efficient to have the update method of the asteroid return False if it needed to be deleted and then have the Game class handle deleting the asteroid.

## Player control testing

At this point I had an opportunity to get a friend to test the game, so I asked her to give it a go with minimal instruction. She has some experience playing games but does not spend a large amount of time on them, so she is indicative of a typical user.

When the game started she immediately tried to control the ship with the arrow keys instead of WASD so I need to add these as a control option.

While playing she did not notice the flickering issue with the asteroids.

She did note that the asteroids tended to clump together into groups which were almost impossible to get through. This is a limitation of my purely random asteroid generation which needs to be looked in to.

She mentioned that having the difficulty increase gradually while playing would make it more interesting and agreed that this could be done in combination with the player selecting a difficulty to start at.

I added these corrections to my Trello to-do list.

# Obstacle-Player collisions

I need to enable collisions between the player and obstacles. Firstly I will write a function that checks for collisions and then prints when it detects a collision. This will allow me to test the collision logic without any extra complication. Then I will write the code to terminate the player’s ship and bring up a death menu with an option to restart.

I want this function to work for bullet-asteroid collisions too, so I need the function to return which objects are colliding.

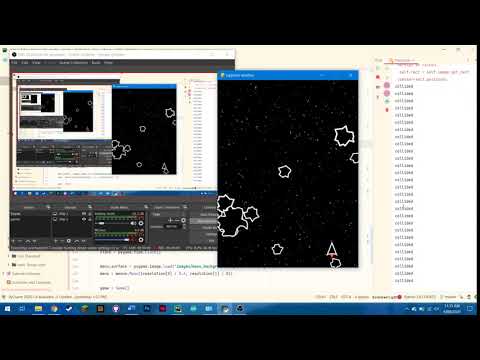
The function will take two arguments, an object, and a list of objects. It will then return which of the objects from the list is colliding with the primary object.

For example, the primary could be a bullet and the list of objects could be all the asteroids. The function would return what asteroids, if any, the bullet is colliding with. The program could then use this information to terminate the bullet and the asteroid that it is colliding with.

### Change of plan

I realized that this function that I was trying to make already existed in the pygame sprite library. After some more research I found that some other methods could also be replaced by the sprite library, I needed to place the sprites into group objects for these to work which required some small changes to how I call the update and blit methods of the asteroids and player. The blit method was removed because the sprite group object has a draw method that blits all its contained sprites.

I was able to detect collisions using this function:

[](https://www.youtube.com/embed/37ZI2CFD00M?feature=oembed)pygame.sprite.spritecollide(primary, secondaries, False, pygame.sprite.collide\_mask)

This function checks whether any sprites in the secondaries group are colliding with the primary sprite. If there are any collisions it checks whether I passed True or False for the kill argument, if True the function will kill the colliding sprites. The final argument is the function that should be used to check for collisions between each pair that the main function checks.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Steps to fix** |
| Play game and crash into asteroids | “collided” is printed when the ship is in contact with the asteroids | This initially threw an error that one of the masks was a NoneType. I found that on the line of code where I generate the players mask I had forgotten to assign a variable to the result so the self.mask variable remained the None that I assigned it in \_\_init\_\_. |

## Second stage

I now need to make the game respond to collisions. I am leaving out a fancy crash animation for now and will simply switch the game to a menu with an option to restart.

I will switch to the menu in the same way that I switch to the pause menu, changing the main game state and then changing the menu state to ‘death’

The restart button will need to call a function that created a new Game object and switch the game state back to playing the game.

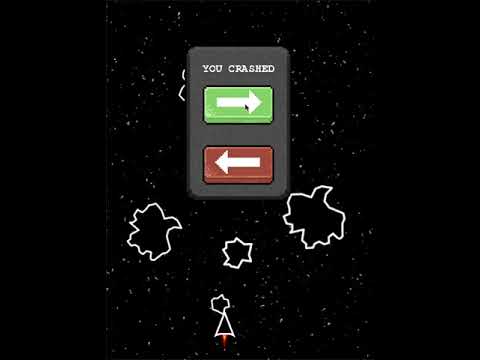
|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Steps to fix** |
| Crash into asteroid | Game stops updating, death menu is displayed | Worked fine first time |
| Press restart button | Game restarts with a fresh set of asteroids | I couldn’t access the programs top level to reassign the game to a new game object from inside the button object.  I learnt how to raise pygame events, give them information, and detect them in other parts of the program.  I made the restart button raise an event with the RESTART command that is detected in the main loop. |

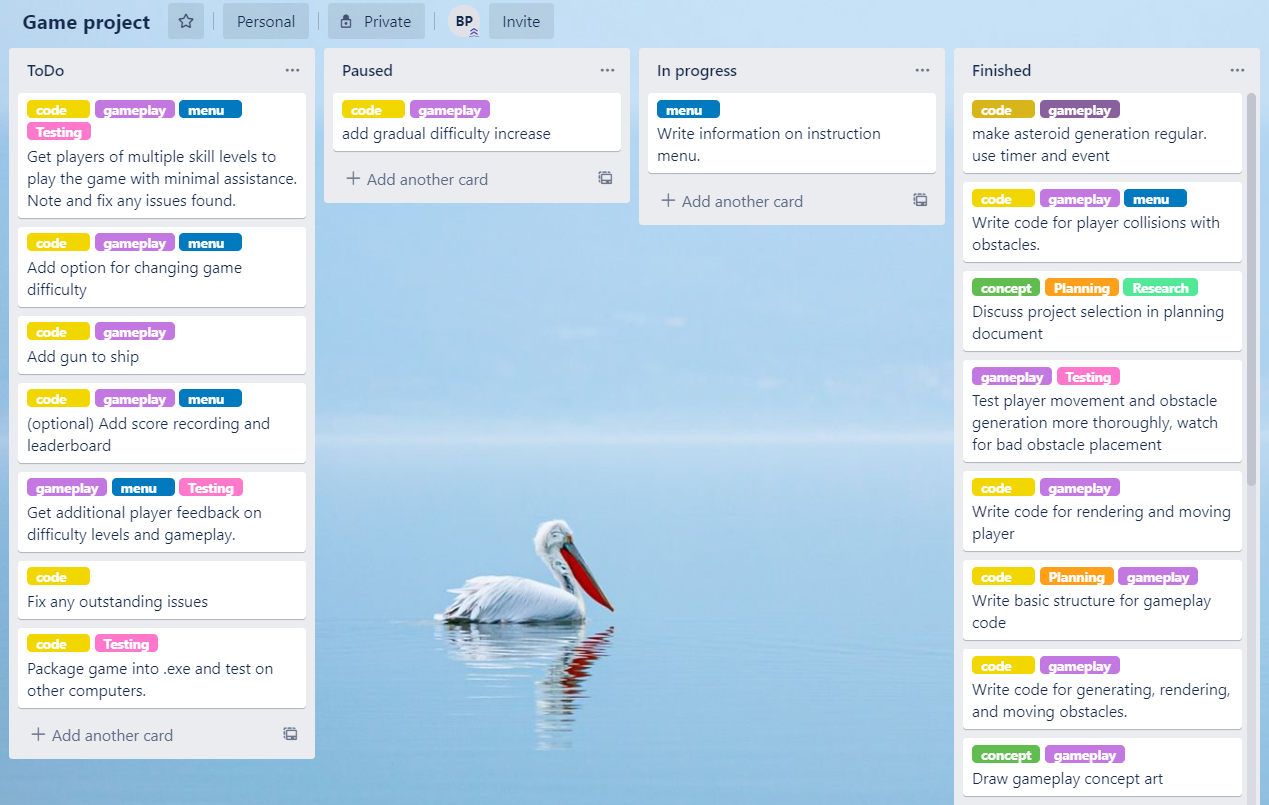
# Even asteroid generation

When I had another person test the game, she noticed that the asteroids tended to come in dense clumps that were almost impossible to get through. This was a consequence of using purely random timing for when an asteroid would generate. While researching events for solving the issues with the buttons I came across [this answer](https://stackoverflow.com/questions/23368999/move-an-object-every-few-seconds-in-pygame) on stackoverflow that mentioned that it was possible to set a timer that would post a pygame event every time a set interval passed. I could use this to post an event which told the game to generate an asteroid a few times per second. I could change the interval to make different difficulty levels.

I created an event called GENERATE\_OBSTACLE and set a timer in the \_\_init\_\_ of the Game object to post this event every 200ms/difficulty. I made difficulty an argument for the \_\_init\_\_ so that I could set the difficulty each time the player started the game. I made the main loop of the game listen for the generate asteroid event and add a new asteroid to the sprite groups whenever it was posted.

### Testing video to show the even asteroid, player death, and restart.

[](https://www.youtube.com/embed/45Xmd8RuDg4?feature=oembed)Please note that the screen recording software causes the game to run at a lower framerate than usual.

I have continued to update the Trello board to keep track of what I need to do.

# Instruction Menu Text

I need to have instructions in the game so that users can check how to play the game if they missed the queues for how it works or they haven’t played enough games to guess the controls yet.

Pygame does not have any way for wrapping text so to write multiple lines I would need a text object for each. This would be tedious to make and maintain. Instead, I will make the instructions in photoshop with a transparent background and then place it on the instruction menu.

## Testing

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Steps to fix** |
| Open instruction menu | Instructions are displayed | The background was not wide enough to fit the instructions because I had forgotten to switch the instruction menu to the wider background that I had made in photoshop. |
| Exit to main menu | Main menu is displayed normally | The wider background for the instructions was showing out the side of the narrower main menu. This was because I was only updating the menu’s surface each tick so the edges of the instruction menu were not being cleared.  I made the entire screen update each tick and made the game blit before the menu so that it would still show in the background. |

# Difficulty levels

To cater to people of different skill levels and allow progressions I need to have difficulty levels that the player can select. I will trial multiple interfaces for this and I need to survey potential players on which they find the most intuitive and easy to use. I will develop these in photoshop and ask people based on the concept images before I develop functional prototypes.

## Concepts and Feedback

Concepts A and B

Concept A requires the player to select a difficulty before pressing the play button. In concept B, each play button corresponds to a different difficulty level.

The first person that I surveyed preferred concept B. They also suggested disabling the play button in concept A until the user selects a difficulty level.

I asked another person how they would assume each menu works, they found the buttons confusing and thought there should be a label to show that they changed difficulty.

## Implementation

### Interface

The buttons that I have made for the other parts of the menus are unsuitable for this so I will create a new class for option buttons. This will use an image with a state for each button depressed. It will have a rect covering the space over each option so that I can check whether the cursor is over any one of the options when the mouse buttons are pressed. The object will contain a state variable that the rest of the program can check to see which difficulty is selected when starting a game.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Steps to fix** |
| Start program | Option buttons are displayed with medium selected | Worked first time |
| Clicking on an option. | Switches the state of the buttons to show pressed one. | The rects were offset from the visible buttons because I had set their positions assuming that the image position would be set from its top-left, it was actually set to the centre so I had to change it to top-left. |

### Setting game difficulty

I decided to make the options call an event that the menu update method would catch and change the game difficulty. When starting the game the Game object would be given the difficulty as a multiplier for the asteroid generation timer. A lower multiplier decreases the number of asteroids generated per second.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected result** | **Steps to fix** |
| Open program, start game at default difficulty | Game runs as usual at the expected difficulty | The event type number overlapped with another event type, I needed to give the event a unique type number. |
| Change the difficulty and play again | Game runs at new difficulty | Worked after above fix. |

Player stuck to bullet, made copy of position variable

Bullet wrong direction, convert direction to radians

# Player testing

At this point I had completed all the major features of the game so I needed to get people to test it out. I tried to get a wide variety of different skill levels represented in the testing so that the game would appeal to both people with little and lots of experience.

## High experience

The first person I got to play the game was young and had lots of experience playing 2D web games with similar controls to mine. He found the interface to be intuitive and had lots of fun playing the game. He asked me several times if he could play it again another time. He had no complaints but suggested a system where the player can gather points and use them to upgrade the ship. I will consider this as an additional feature to add if I get time.

He found the hard level to be almost unplayable and usually stuck to the medium and easy difficulties.

## Low experience

Two of the people who tested the game were around my age and had minimal experience playing games similar to mine although they have played plenty of very different games. Both players enjoyed the game but didn’t realize that they could shoot the asteroids. One of the players was confused about the exit button and pressed it assuming that the button would go forward instead of exiting.

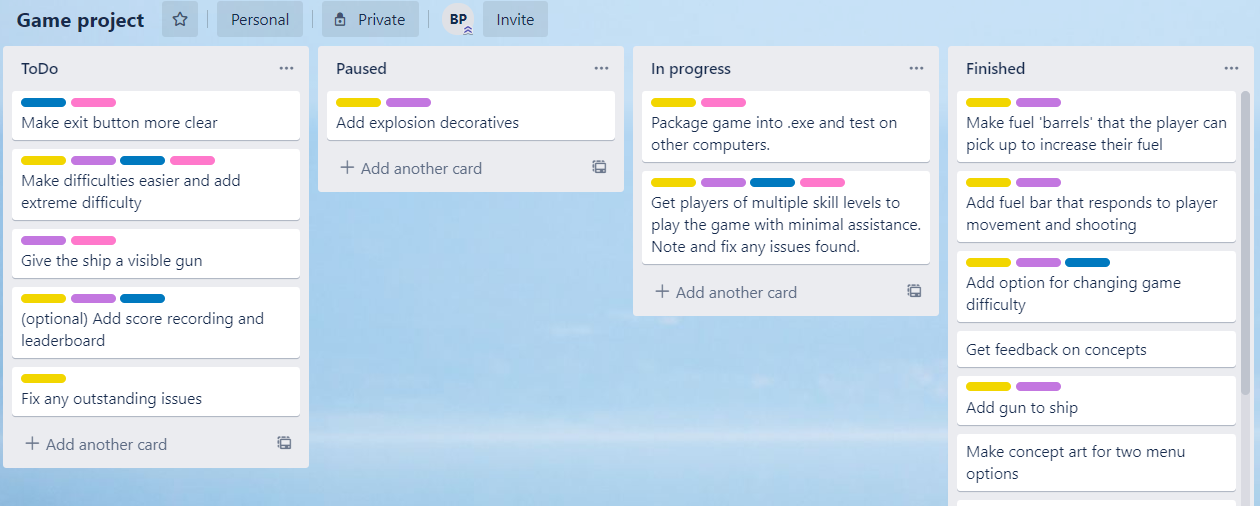
One of the players started on medium difficulty and enjoyed it while finding it challenging. The other player picked easy mode and still found it very challenging.

## Further steps

I will change the difficulties to make them all slightly easier but I will also add an extreme difficulty to challenge very skilled players. I will make the easy mode move asteroids more slowly and slightly decrease the number of asteroids in the medium difficulty.

I will make the exit button less ambiguous and consider removing it as it is redundant.

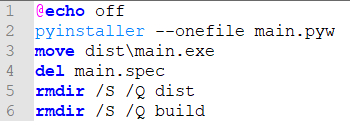
I need to make it more clear that the ship has a gun, I will add a cannon to the ship sprite and consider showing the controls next to the main menu instead of a sub-menu.

I have added these tasks to my Tello board:

# Packaging

Unlike some other common programming languages, python is not compiled before execution. Python scripts run in the python application that interprets the code into machine code as it runs. Because of this python scripts will not run without the python application installed first. This is a problem for distribution of the program because most computers do not have python installed and users cannot be expected to install it themselves. Before I give copies of the game to other people, I need to package it into an executable file that contains all the code needed to interpret python scripts.

This packaging can be done with a program called pyinstaller. I installed this using python pip and then tried packaging my game into a single file. When I tried to run the executable that this produced it crashed with an unspecified error. I found that the packaged file did not include the images and sound files that the game loads so I had to include those folders when distributing it. There was also another error where the exit button was causing the game to crash. Because the button is redundant I decided to remove it.

The packaging process placed the executable in a subfolder and included lots of files and folders that I did not need. Removing these and finding the executable was tedious so I wrote a windows batch script to automate the process. This allowed me to package the program and clean up the folders with one click.

To test that the executable would run on computers without python installed I renamed my python application so that my computer could not find it and tried playing the packaged version of my game. It worked fine.

# Process and planning tool evaluation

Throughout the process planning tools have been vital for organising the project and keeping me on track with the tasks that need to be completed.

I mostly made use of Trello for my project management. I was able to add and rearrange tasks whenever I wanted which was vital for keeping track of my progress and making sure I didn’t forget anything. There were multiple points where I had to take a break from working on the game and needed to look through the Trello board to figure out where I was up to when I came back. The flexibility of the tool was very useful to me, I often needed to change the priority of tasks, pause tasks, or add and remove tasks as I learnt from player testing.

GitHub was very useful for the version management of the project. When I added a new feature or made some other change to the game I would always “commit” those changed to a new version which got backed up to GitHub’s servers. Making versions for all the important changes was vital because when working on a complex program like this it is easy to make a mistake that breaks the game but is extremely difficult to track down. It is also useful for retrieving accidentally deleted files. At any point I can revert my working files to an older version. GitHub is also good for backing up the game files so that if the files on my computer are damaged or lost, I have another source to get them back from. I like that the files are only backed up when I commit changes so my mistakes are not immediately synced between the local and cloud versions.