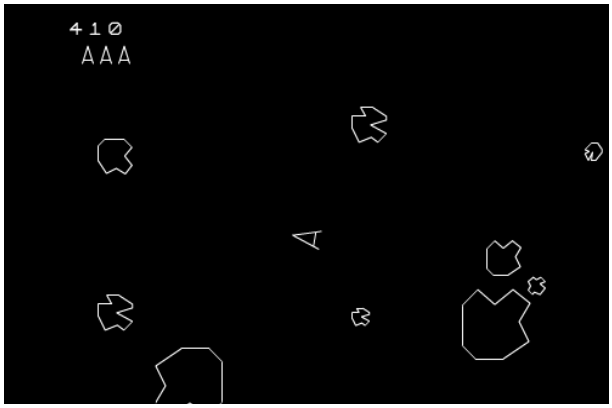
Analysis

Introduction

Games are a popular form of entertainment that take form in many different ways. For my project I have decided to take the arcade game Asteroids which has fallen out of popularity in recent years and I have decided to make the game more complex and add more variety to the way the user will play it. One way I plan to improve the difficulty is to take out the multi-lives functionality. Thus making the game more challenging and allowing less room for error. I would like to still use a lot of functionality from the Asteroids game but expand on the core mechanics by adding micro-gravity which I hope will make the game less predictable and therefore make the game more interesting and challenging and I also want to add a Power-Up system which creates small power-ups that enables the player to become stronger in a particular and unique way. Furthermore I would like to add another game mode that wasn't in the original game. I hope to make this game mode use some of the same functionality but make the way the user plays different. I hope this will also make the game less one dimensional, by adding different ways to play the game. I also intend to improve aspects of the game from the original. One of the things that the original game didn't have was a random rock creation system instead the game would display premade images of rocks. These were just displayed at different sizes and angles as you can see below in the screen shot of the original asteroids game.

Current Game

The base game asteroids has a basic mechanic that allows a player to move a ship that can shoot bullets. When they collide with rocks the score is increased and the rock is destroyed. Most asteroid arcade machines have the ability to save high scores with a 3 letter name. It is estimated that 70,000 arcade units were sold and $150 million dollars was earned in sales of these machines and a further $500 million was made by people who bought these machines and used tokens or coins as a way to operate the machine[1]. Currently there is no official version available for purchase for any console, pc or any handheld device which has been released in the at least the last 5 years. The only available way to play the game is now through unofficial games on free gaming websites. On the right is an example of the original version of the game.



https://cdn.tutsplus.com/gamedev/authors/legacy/Steven%20Lambert/2012/10/28/asteroids.png

End User

I intend my project to be aimed at gamers with little time but who want something challenging but also something that will make them want to continue playing in order to better their current high score. Furthermore I also want my game to be played by people who want to improve things like their reaction time and their hand-eye coordination skills. I am also going to assume has the user has no previous experience with games and technology in general. Therefore it is of high importance that my User Interface is as simple as possible whilst being fully functional. This means that everything should be clearly set out and should be easy to navigate.

1 - https://en.wikipedia.org/wiki/Asteroids_(video_game)

A lot of games that are played nowadays are mobile games they have risen substantially due to the invention of the smart phone in the past decade. Most people lead busy lives and these games allow people to pick up games for just a few minutes at a time. Although I do not plan to create a mobile game I still want my game to fit this idea that a user can play the game if they only have a short amount of time. So whilst I want my game to appeal to gamers I also want my game to appeal to people who live busier lives. Which nowadays is a large majority of society. I intend to achieve this by making sure that an individual game doesn't last for a large amount of time.

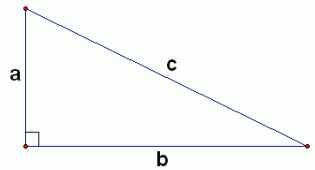
Objectives

Below is a list of the core objectives I want to have in my game.

1) Display a menu that waits for a user input that is listed on the screen without throwing an error when an incorrect button is pressed.
2) When a game mode is called through a menu the corresponding function should be called to set the necessary variables and to display and update the game at the set frames per second.
3) Display a user ship from the correct image from the "Textures" Folder.
4) Allow the user ship to move and rotate by using the four arrow keys. Up and down should add speed to the ship and left and right should rotate the ship. This movement should be smooth and fluent.
5) Display rocks that have a random shape within certain bounds. This should be implemented by using 12 different random variables that will dictate the length from the centre each point it so that every rock is randomly generated each time.
6) Then assign a random speed and trajectory to these rocks. Rocks should also travel in towards the centre and shouldn't spawn and immediately be removed from the screen
7) Allow the user ship to shoot bullets (Depending on if the game mode needs the ability to shoot bullets) at the same angle the ship is facing and assign a set speed that is quicker than the user ship so that the bullet doesn't go back through the ship.
8) When a rock, small rock or bullet leave the screen they should be removed from the corresponding list so as not to cause lag from to many entities.
9) When the player ship leaves the screen its coordinates should be moved to the opposite side of the screen at the same angle. This should be done at a slight overlap so that the player ship can never be fully off of the screen as to avoid confusion or create invulnerability if a ship is hid there.
10) Create collisions between the user ship, rocks, small rocks and bullets.
11) When a bullet hits a rock, small rocks should be created and the score should increment by 50. Both bullet and rock should be removed from the bullet list and the rock list in order to not be displayed.
12) When a bullet hits a small rock the score should increment by 5 and the small rock should be removed from the list so as not to be displayed and the bullet should likewise be removed from the bullet list.
13) If a user collides with a rock or small rock then the specific game mode function will call the end game function which will end the game.
14) If a user has attained a higher score than any of the scores stored in the corresponding high score list then the user will be asked to enter a name in which to save against their score. Then should be sent to the end game screen which summarises their statistics from that game and will be prompted to exit.
15) High scores should be stored in a list then sorted and displayed when a high score menu is called with the name next to the corresponding score.

16) Every game mode that has the ability to destroy rocks should add the amount in each game destroyed to a total “rocks destroyed” variable in order for it to continue incrementing.
17) All high-score lists and the amount of rocks destroyed should be stored in separate .json file. This means that high scores will be kept when the program is not running.
18) If no high score is set then the user should be returned to the end game screen where they should be prompted to exit.
19) Throughout the default game mode the game mode function should call for a pick-up for a power-up to be randomly placed within 20 pixels from the edge of the screen as to not overlap with the edge.
20) When a player collides with a pick-up they should gain the corresponding power-up for the remaining 10 seconds from the spawn. This way the Pick-ups will be spawned at the same time as the rocks are spawned which makes picking up power-ups more challenging if rocks spawn at the same time the user will have to evade the rocks in order to get the power-up.
21) When a small rock comes close to a bigger rock they should be affected by a micro-gravity function which should adjust the trajectory of the smaller rock so that it heads towards the bigger rock.
22) When a small rock overlaps with a bigger rock the small rock should have the same speed and angle as the bigger rock to show that the small rock is now attached to the bigger rock.
23) All game modes should record the time elapsed by the player by using a variable that get incremented every frame and then work out the total time the game has been live.
24) The default game mode should keep track of the score throughout the game and then feed it to the end game function in order for it to be displayed at the end of the game and if necessary use it as a high-score.
25) The survival game-mode should use the amount of seconds alive as the high-score and this should be given as a parameter of the end game function.
26) Every 10 seconds each game mode should increase through difficulty. The default game mode should increase the amount of rocks being spawned as well as the speed that the rocks spawn at. The survival game mode should increase the amount of rocks that are spawned but should also increase the speed more severely to make the game progressively harder.
27) Along with time and rocks destroyed the default game mode should display the score and the level which the game is at.
28) All of this should be carried out with no noticeable lag or drops in frames

Mathematical Formulae



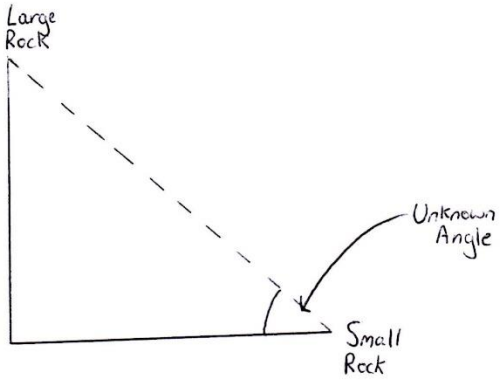
$a^2 + b^2 = c^2$

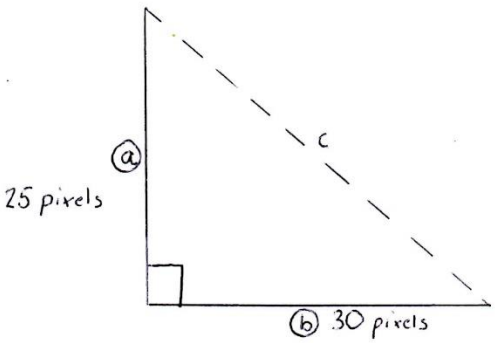
I intend to use both Pythagoras’s theorem and the sine rule in order to create micro-gravity. I plan to do this by finding the gradient between one rock and another rock. Then use this to work out the correct trajectory and therefore to create the impression of micro-gravity between smaller and larger rocks.

$$\frac{a}{SinA} = \frac{b}{SinB} = \frac{c}{SinC}$$

The method in which I intend to use to find the gradient between each rock is by firstly using two rocks coordinates one being a small rock and one being a large rock. If they come within a certain distance then the function will be ran. Then I intend to find the length of the hypotenuse that connects the two rocks. After finding this I can use both a length I already know and the length of the hypotenuse to work out the angle of the small rock to the large rock. In order for this to look like

micro-gravity I intend to keep updating the angle at which the small rock travels as well as increase the speed. Below is an example of an angle calculation.





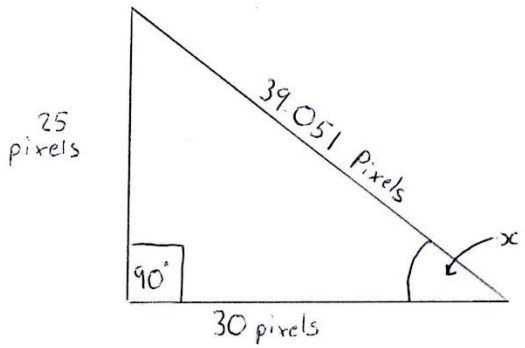
Pythagoras ~ $a^2 + b^2 = c^2$

~ $25^2 + 30^2 = c^2$

~ $625 + 900 = 1,525$

~ $\sqrt{1525} = 39.051$ (3dp)

$c = 39.051$

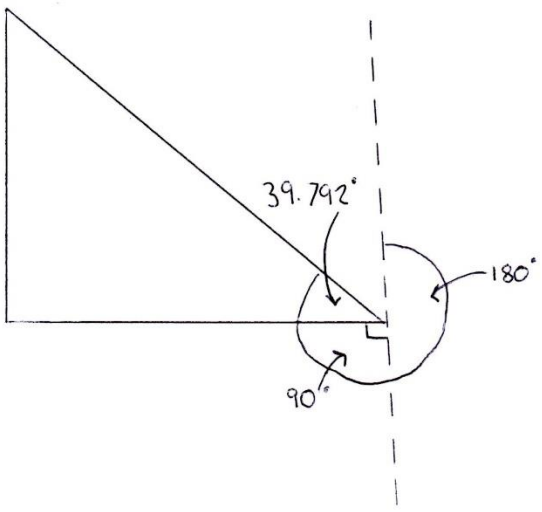


Sine Rule ~ $\frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)}$

~ $\frac{\sin(90)}{39.051} \times 25 = 0.64$

~ $\sin^{-1}(0.64) = 39.792°$

Angle $x = 39.792°$



Angle from Small Rock to Large Rock:

$180 + 90 + 39.792 = \boxed{309.792°}$