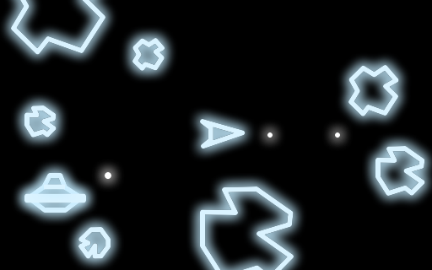
Applied Math 2 - Asteroids Game Project

# Project Structure

We’ve now spent nearly a term and a half learning about math as it applies to the domain of game development. It’s now time to work on a small project that will incorporate the skills and technologies you’ve been studying.

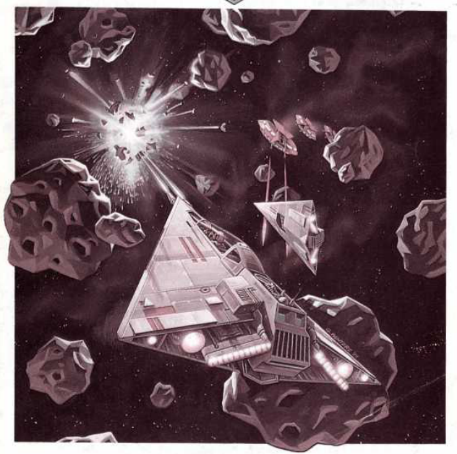
This is a “Choose Your Own Adventure” project. Each feature you complete levels up your mark, but you don’t need to complete them all. Level up your mark until you’re satisfied with your project or until you run out of time.

# Project Objective

It’s time to use your Javascript, P5.js, and applied math skills to build a simple video game.

You will be implementing the classic game Asteroids, originally released by Atari in 1979. (Kyle was two. You did not exist.)

[You should start by playing a few rounds of the game here](https://games.aarp.org/games/atari-asteroids).

While playing, start to think about the different applied math topics you’re going to need to leverage to implement this game.

# Asteroids Gameplay

* The player controls a triangular ship.
* The player can rotate the ship and fire its engine to move.
* The ship eventually comes to a stop when the engine is off.
* The player can teleport the ship through hyperspace to a random location on the screen.
* The player can fire bullets from the front of the ship.
* The objective of the game is to destroy asteroids and saucers.
* The player loses a life when:
  + It crashes into an asteroid.
  + It crashes into a saucer.
  + It is shot by a saucer.

# Basic Asteroids Features (1 Mark Each ➡️ 14/16 Marks Total)

1. ~~A player can control the rotation of their ship using two keyboard keys.~~
2. ~~A player can fire the ship’s engine by using a keyboard key. (The ship’s existing velocity will be affected by the force supplied by the engine.)~~
3. ~~The ship will slow down and eventually stop once the player stops firing the engine.~~
4. ~~The player can teleport to a random location on the screen. (They will insta-die if they teleport into an other game object.)~~
5. ~~The game starts with a set number of large asteroids, each moving in a random direction. (All asteroids move at the same speed.)~~
6. ~~The player loses a life when:~~
   1. ~~They collide with an asteroid of any size.~~
   2. ~~They collide with an enemy saucer.~~
   3. ~~They are shot by an enemy saucer.~~
7. ~~When the player dies, their ship is returned to the centre of the screen.~~
8. ~~The player’s score increases when they shoot down:~~
   1. ~~Large Asteroid: 20 Points~~
   2. ~~Medium Asteroid: 50 Points~~
   3. ~~Small Asteroid: 100 Points~~
   4. ~~Large Saucer: 200 Points (If implemented.)~~
   5. ~~Small Saucer: 1000 Points (If implemented.)~~
9. ~~When a large asteroid is hit, it breaks into two medium asteroids. When a medium asteroid is hit, it breaks into two small asteroids. Small asteroids disappear when shot.~~
10. ~~The player gains an extra life for every 10,000 points they score.~~
11. ~~The game ends when the player loses all their lives. (They start with three lives.)~~
12. ~~The player’s score and number of lives remaining is displayed at the top of the canvas.~~
13. ~~The movement of all game objects (ship, asteroids, saucers, and bullets) will wrap-around the screen edges while maintaining their momentum.~~
14. There are different sound effects for the engine, shooting, the presence of saucers, the destruction of objects, hyperspace jumps.
15. You’ve got music playing on a loop in the background of your game.
16. ~~The smaller asteroids move slightly quicker than medium ones. Medium asteroids are quicker than larger ones.~~

# Advanced Asteroids Features (2 Marks Each ➡️ 14/22 Marks Total)

* ~~At set scoring intervals, an enemy saucer will fly across the canvas and shoot at the player.~~
* ~~More often than not, the enemies will be large saucers with very poor aim.~~
* ~~Occasionally small saucers will arrive. They will have decent aim.~~
* ~~The aim of the small saucers improves as the player’s score increases.~~
* When the player’s score is high enough, the small saucers will take the player’s current velocity into consideration when aiming.
* ~~Saucer bullets can also destroy asteroids~~.
* ~~Saucers can be destroyed when they collide with asteroids.~~
* A level system has been implemented.
* Screen shake has been implemented and used when ships and asteroids are destroyed. It’s recommended that you only attempt linear shake, as rotational shake could be much more difficult. See [the video we watched in class](https://www.youtube.com/watch?v=tu-Qe66AvtY) and [the associated slide deck](https://www.gdcvault.com/play/1023146/Math-for-Game-Programmers-Juicing).
* ~~Each bullet fired applies a small amount of knock-back force to the ship.~~
* Players can record time scores on a locally saved leaderboard. (See: [storeItem()](https://p5js.org/reference/#/p5/storeItem))

# Mind Melting Asteroids Features (15 Marks)

* Small saucers navigate around the canvas by firing four corner engines while actively avoiding collisions. They do not rotate. Their motion is physics based, like the ship. The engine firing collision avoidance system needs to work. Small saucers should be able to navigate around the canvas without crashing while firing on at the player’s ship.

# Additional Tasks (2 Marks Each ➡️ 6/14 Marks Total)

* Recorded video submitted to YouTube showing gameplay including win and loss situations.
* 200+ word reflection on the problems and discoveries you made while coding this project.
* ~~You’ve designed your game using at least three different classes.~~
* ~~You’ve designed a reasonable class hierarchy amongst several of your classes.~~
* ~~You’ve coded a way to restart the game after the player wins or loses.~~
* Your game begins with a title screen that includes a play button to start the game.
* You submit a github repo of your project, showing a minimum of 32 commits made throughout the project. (This will mean that you’ve developed the project outside of the p5.js editor. This will be covered in class.)

# Project Marking and Progress Bonus

There will be two in-person marking opportunities for this project:

* Thursday March 2nd, 2023
* Thursday March 9th, 2023

**BONUS OPPORTUNITY**: If you can show that you have completed at least 25 marks worth of work by March 2nd, you will receive an instant bonus of 5 marks. This is a 10% bonus!

# Feature Marks and Project Grade

**Your project grade will be: 100 \* [marks for your completed work] / 50**

The basic project features listed above are worth 1 mark each.

The advanced project features listed above are worth 2 marks each.

The additional tasks listed above are worth 2 marks each.

The mind melting project feature listed above is worth 15 marks. (Although I fear the amount of work required for this feature might exceed the reward!)

# Choose Your Own Adventure

There is a total of 71 marks available (including bonuses) but you only need 50 marks to get 100% on the project. This means you can choose to implement the features that interest you the most.

The type of graphics you use is not defined by the assignment. You can implement the original line-based graphics or use image sprites. 3rd party image sprites are permitted with attribution.

**IMPORTANT:** Be sure to include a list of all the features you believe you have completed. This list should be categorized by basic, advanced, mind melting, and additional tasks. This list can be submitted as a Word/Text file, a Notion page, or within a Learn submission comment.

# Tips, Gotchas, and Best Practices

## Dealing with User Input Involving Many Keys

Dealing with simultaneous keypresses in p5.js can lead to weird results, especially when using the keyPressed() callback function or the isKeyPressed global Boolean variable. Instead, it’s recommended that you use the keyIsDown() predicate function to detect keyboard input.

if (keyIsDown(65)) { // The 'a' key is being pressed.  
 // rotate counter-clockwise  
}  
  
if (keyIsDown(68)) { // The 'd' key is being pressed.  
 // rotate clockwise  
}

if (keyIsDown(87)) { // The 'w' key is being pressed.  
 // Turn on engine.  
} else {  
 // Turn off engine.  
}