Overview:

This project is a recreation of the classic asteroids game, implemented in WebGL and Javascript. The object of the game is simple: control a small ship and survive for as long as possible in a 2-dimensional world by avoiding and destroying asteroids. You can fire bullets at the asteroids to break them into smaller asteroids or destroy them completely. If the ship collides with an asteroid it is destroyed, and the game is over. Both the ship and the asteroids can wrap around the screen, disappearing off one edge and reappearing on the other. You score points based on how long you stay alive and how many asteroids you destroy.

(Left) The classic Asteroids game. The asteroids and ship are represented with a white outline, the bullets are represented as white dots, all on a black background.

(Left) My recreation. There are many color and style changes as well as added functionality/features.
**Gameplay/Interaction:**

The goal of this game is to achieve as high a score as possible. You accumulate points by surviving in the game area and destroying asteroids. The current score is displayed in the upper left corner. When the game loads you are prompted to click anywhere to begin. The controls are:

- **W:** Move the ship in the direction that it is pointing, building up speed
- **A/D:** Rotate the ship left or right respectively
- **Space:** Shoot a bullet in the direction the ship is pointing
- **Restart Button:** Click to reload the game at any point
- **Pause Button:** Press to pause the game, press again to unpause

**Design/Features:**

- **Geometries:** I implemented 4 geometry classes: ship asteroid, bullet and explosion. The ship is created at the end of main(). The asteroids are created with random sizes when the renderer counts fewer than 8 asteroids on the canvas. When a sufficiently large asteroid is destroyed, two asteroids are created at the same location with half the size of the original. Bullets are created when the space bar is clicked, but only after it has been lifted since the last click (to prevent holding down and spamming bullets). An explosion is rendered when the ship is destroyed.

- **Physics:** Inside the ship class I implemented momentum values, rotation values, a boost value, a position vector, a direction vector, and corresponding matrices for transformation. When the ship is boosting the momentum values are incremented by a small amount according to the direction vector during each render. When it is not boosting the momentum values are decremented towards 0 by a much smaller amount to simulate a very small friction force. A maximum speed is enforced to help with control. The other classes operate using a position value and velocity values, incremented each render. The bullets are given velocity values in the direction the ship is facing and the asteroids are given random velocity values.

- **Collision Detection:** Collision detection is implemented in a loop inside of the Renderer.render() function. For each geometry, the id is checked and then compared with other relevant geometries. The distance between the both position vectors is compared with the sizes of both and a collision is called on overlap. Geometry values and the geometries array are modified based on the type of collision.

- **HUD:** I implemented a Heads Up Display to display the current score, show functionality buttons, and display the start and game over screens. It is a canvas with 2d context laid over the 3d canvas that the game is played on. Click listening is bound to the HUD canvas for button interaction.