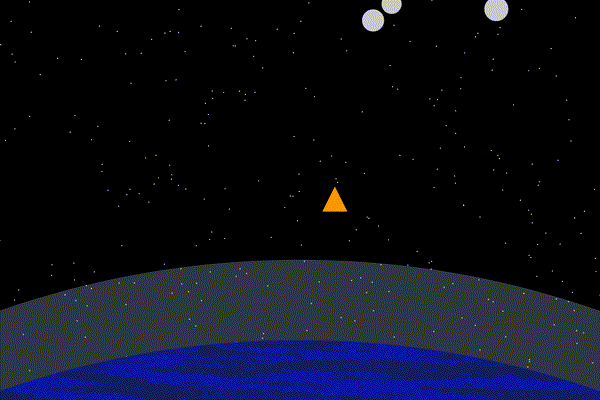
steroid game clone



In this assignment the goal is to complete an asteroids game clone complete with realistic movement, gravity and collision checks as in the image above.

Start by downloading the template code attached.

[asteroidGameCloneTEMPLATE](https://github.com/mastrogiorgis/CM2030StudentCode/raw/main/week%2003%20-%20forces%20and%20physics/asteroidGameCloneTEMPLATE.zip)

**Please note: This is a graded assignment. This is a draft of the assignment made available before course work submission is open, only minor changes may occur.**

**Where do I submit?**

There are eight graded assignments like this on the course. This one and three more make up the midterm assignment and are submitted in week 10. Nevertheless, we strongly recommend that you complete this graded assignment in the week it has been assigned and not wait for the midterm submission. You need to master the material included in these before you continue with the next weeks. Once you complete it, save it somewhere safe, do not share online using the webspace and upload it with the others when prompted in week 10.

**Steps to complete**

Step 1: In bulletSystem.js complete the function edges() by looping over the bullets array and for each bullet check if it has left the screen. If it has, remove the specific bullet from the array. (Hint: Use splice() to remove the bullet from the array and think about how you splice from an array correctly). Remember: If you want to call a function or use any variable that belongs to the class you'll need to prefix it with the word "this". So in this case it would be: this.bullets.myFunction();

Step 2: In spaceship.js update the interaction() function and fill out the missing information in order to simulate the correct behavior in the system. When the left arrow is pressed the spaceship should move left, when the right arrow is pressed it should move right and so on. Please note that the spaceship won't move until you have completed the next step too.

Step 3: In spaceship.js update the move() function similarly to how the move() function works in asteroidSystem. Make sure you limit the velocity vector to maxVelocity so that the spaceship does not accelerate too much. Notice how, unless we fire the rockets in the opposite direction, the spaceship will keep moving. There is no friction in empty space.

Step 4: In sketch.js complete the isInside() function that takes the location of two circles and their diameters and returns true if they overlap, false otherwise. You could check it works, by creating a dummy circle around the mouse and checking if isInside() returns true.

Step 5: In sketch.js complete the checkCollisions() function so that you check collisions between the spaceship and all asteroids. Hint: You'll need to loop over all the asteroids and use the inInside() function you just programmed. If it returns true then you'll call the gameOver() function. If you've done things right, if the spaceship is hit by an asteroid the game will end. Check before proceeding.

Step 6: In sketch.js add more functionality to the checkCollisions() function to check if an asteroid has crashed onto earth. If you do things right then the game should end when that happens. Check before proceeding.

Step 7: In sketch.js add more functionality to the checkCollisions() function to check if the spaceship has collided with the earth and if it has it ends the game. Check before proceeding.

Step 8: In sketch.js add more functionality to the checkCollisions() function to check if the spaceship is inside the atmosphere. If it is, the spaceship's setNearEarth() function is called.

Step 9: In spaceship.js complete the setNearEarth() function. When the spaceship enters the earth's atmosphere it's affected by the earth's gravity. Create a "downwards-pointing" vector of strength 0.05 which pulls the spaceship towards the earth. The atmosphere also introduces friction and the spaceship can't move forever like in empty space. It will decelerate unless it fires its engines. Create a force called friction that's 30 times smaller than the velocity of the spaceship, points the opposite direction to velocity and is then applied in the the opposite direction.

Step 10: In sketch.js add more functionality to the checkCollisions() function to check if any of the bullets of the spaceship have hit any asteroids. If they have, then call the destroy() function of the asteroid object, passing it the index of the asteroid to destroy.

Step 11: Implement two of the ideas for further development below. Points given to ambitious learners.

**Ideas for further development:**

* Make the spaceship pretty by adding jets thrusters which activate from the opposite side of movement just like on a real spaceship.
* Keep score of how many asteroids you have hit.
* Make the program get progressively harder by spawning asteroids more frequently as time goes by.
* Make it your own by customising the look of the game (make sure you maintain same functionality or you might lose points during grading).

**Marking rubric**

* Step 1 - [1 point]: edges() function is correct.
* Step 2 - [1 point]: interaction() function is correct.
* Step 3 - [2 point]: Spaceship moves correctly. No friction in space and velocity limited by maxVelocity.
* Step 4 - [1 point]: isInside() function works as it should.
* Step 5 - [1 point]: If asteroid hits spaceship: game ends.
* Step 6 - [1 points]: If asteroid crashes on earth: game ends.
* Step 7 - [1 points]: If spaceship crashes on earth: game ends.
* Step 8 - [1 points]: If spaceship inside atmosphere: call the spaceship's setNearEarth() function.
* Step 9 - [2 point]: If spaceship inside atmosphere, earth pulls it towards it, air friction affects movement.
* Step 10 - [1 point]: If any bullet hits asteroid: destroy asteroid.
* Step 11 - [4 points]: Has learner implemented any of the ideas for further development?
* S​tep 12 - [4 points]: Code presentation: formatting, comments, variable naming.

Mark as completed

Like

Dislike

Report an issue