Assignment 2: JavaScript + Canvas

Develop a single-player game using HTML5 and JavaScript (with the jQuery library, if desired). **This assignment is paired (two people per team).** Use the Piazza teammate finder if you need to find a partner.

Game Specifications:

Objective: The object of the game is to stop black holes from eating the solar system – planets, nebulae, alien spacecraft, space junk – by quickly clicking on all of the black holes as they appear before the time is up.

Description: Before you is a solar system filled with all sorts of objects: planets, moons, nebulae, asteroids, spacecraft, space junk, and more. However, black holes appear every few seconds and start to pull objects towards them! If an object hits the centre of a black hole, the black hole eats it, meaning a loss of 50 points from your starting total of 200 points. Additionally, once a black hole has had its fill, it will disappear back into the ether.

The game is time-based, and each level lasts 60 seconds. The game ends when the time is up or when the black holes have eaten all of the objects in the solar system. If at least one object remains, you can move on to the next level.

There are three types of black holes in the game: blue (slow pull), purple (medium pull), and black (fast pull). Each time you destroy (mouse click) a black hole, you receive points (but all objects it's eaten are gone forever): 5 for blue, 10 for purple, and 20 for black. Once a black hole has eaten a certain number of objects – 3 for blue, 2 for purple, and 1 for black - it will disappear, meaning you lose the opportunity to earn points.

Details: The detailed specifications are as follows:

- The game has **two pages**:
    - Start Page
    - Game Page

- For the **Start Page**:
    - Show the user's **high score**. Should be 0 by default and updated as the user progresses through the game.
        - Use HTML5 Local Storage to store and retrieve a list of high scores for the user.
    - Provide a "Start" button that allows the user to start the game at Level 1; see Figure 1 for a **wireframe**, which depicts the unstyled elements and their relative sizes and positions within the page. It's up to you how to style these elements.

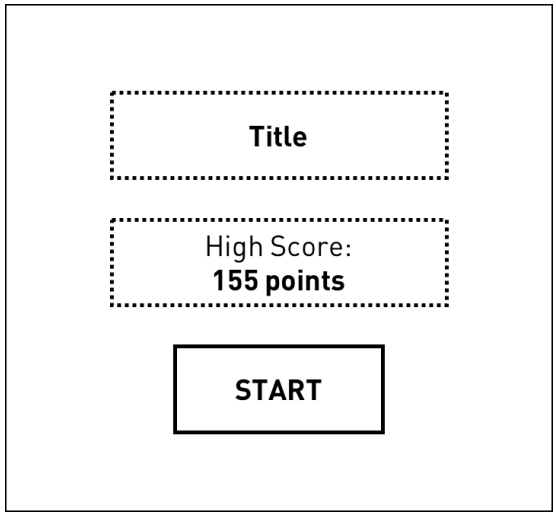

Figure 1. Wireframe of Start Page.

- For the **Game Page**:
    - Contains the Game Canvas.
    - Once the user **finishes Level 1**, the game should display a transitional screen based on the wireframe in Figure 2.
        - Clicking the "Next" button should start Level 2.
    - Once the user **finishes Level 2** (the last level), the game should show a transitional screen similar to the one presented in Figure 2 with the user's final score and a "Finish" button.

- Clicking the "Finish" button should take the user back to the Start page, where their top scores are displayed in descending order.

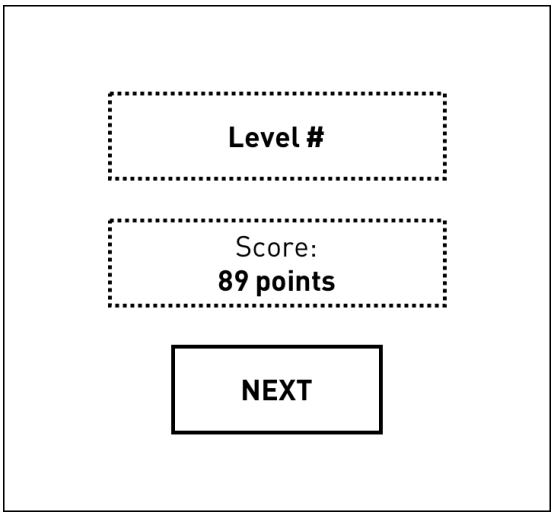

Figure 2. Wireframe for transitional screen.

- For the **Game Canvas** (built using the HTML5 canvas element):
  - See Figure 3 for an annotated wireframe. As with the Start Page, the look & feel of the game is up to you.

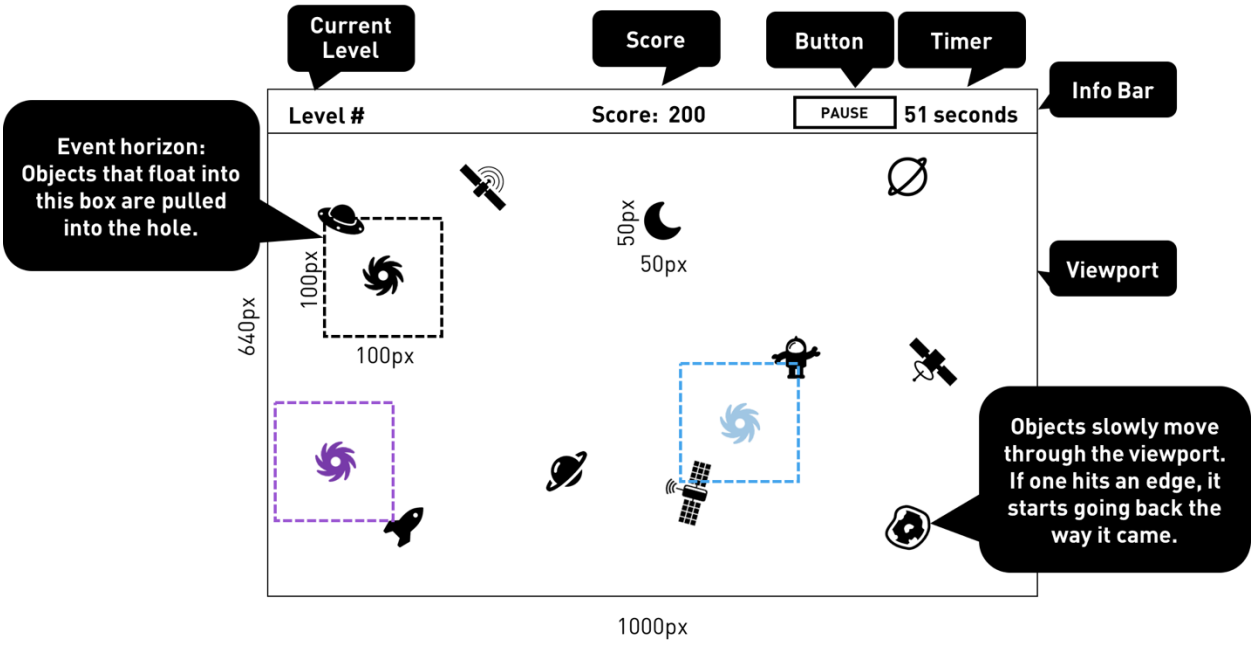

Figure 3. Wireframe for Game Canvas. Note that the dashed boxes around the black holes that indicate the event horizon should be invisible in your version of the game.

- The **viewport** should be 1000x640px.
- The **info bar** should be about 40px in height (but can be taller if need be).
- The **starting score** should be 200 points.
- The **timer** should start at 60 seconds and be counting down.
- There should be a **pause button** beside the timer that should pause the timer and display a "Pause" overlay to indicate that the game is paused.
- The **Level #** should be the current level number.
- Each level of the game should start with a random assortment of **10 objects**, such as planets, spacecraft, asteroids, etc., that are slowly floating in one random direction across the screen at a random starting position. These should be **drawn using canvas**, NOT imported images, and should be sized 50x50px.
    - If an object hits the edge of the screen, it should start going back the way it came OR a new random direction.
    - Objects can float over top of each other.
- A new **black hole** of a random type should appear on the screen every few seconds, depending on the type (see Table 1).
    - The **black holes can be SVG images** sized 50x50px.
    - The dashed line indicating the **event horizon** in Figure 3 should NOT be drawn on screen. It is an invisible element sized 100x100px and has the black hole at its centre.
    - A new black hole and its event horizon **must not be generated over top** of an existing black hole and event horizon. But it can be generated over top of an object.
    - If a **floating object crosses over** the black hole's invisible event horizon, that object should start moving towards the centre of the black hole.
        - If it hits the centre, it should disappear and 50 points should be subtracted from the user's score.
        - However, if the black hole is clicked by the user or it has eaten enough objects, it will disappear and let the object continue on its normal course.
    - If the black hole **eats enough objects** (see Table 1), it should disappear.

- If the black hole is **clicked** by the user (if the black hole falls within 25px of the relative coordinates x,y of the click event), it should disappear and the user should receive points depending on its type (see Table 1).

Table 1. Black hole properties. For relative units, try different numbers to balance the game while maintaining the relative difference between the three types of black holes.

Type	Pull Speed	Disappears After Eating …	Points if Clicked	Appearance in Level 1	Appearance in Level 2
Blue	Slow	3 Objects	5	Frequent	Double of L1
Purple	Medium	2 Objects	10	Infrequent	Double of L1
Black	Fast	1 Objects	20	Rare	Double of L1

Requirements

1) Work in **pairs**. Individuals or teams of three or more are not allowed.
2) Use HTML5, CSS3, and JavaScript. You may also use the jQuery JavaScript library. However, other third-party libraries are *not* allowed.
3) Switching between screens should be done using JavaScript (meaning that screens should be dynamically shown/hidden and there should be no refreshing of the browser window).
4) There should only be one HTML file called **a2.html**.
5) Code should work in Google Chrome 50+.

Rubric:

Criteria	Weight
Page setup as expected, with JavaScript-based switching.	5%
Score (displayed and updated on Start Page as well as the info bar; added to/subtracted from based on specified interactions).	5%

Info bar design, including Level #, score, timer, and pause button.	5%
Working pause button that toggles between pause/unpaused states.	5%
Objects drawn in canvas, moving/floating and rebounding as specified.	20%
Black holes drawn in canvas or with SVG images.	5%
Black holes have an invisible event horizon, appearing and interacting with floating objects as specified based on their type.	20%
Click events operating as specified, e.g., black holes disappearing, allocation of correct points, etc.	20%
Changes in Level 2, particularly appearance frequency of black holes.	5%
Look & feel is applied (styled) to the game and pages and is consistent.	5%
Overall experience of the game.	5%

Deliverable:

All files submitted in a single zip named LASTNAME1_LASTNAME2-a2.zip. At the least, you should include the following files: index.html, style.css, any script (.js) files used, including jQuery if used, any image files (e.g., SVGs), and a readme.pdf file that includes your names, CDF IDs, and preferred contact emails.

Each of you must also submit, individually and confidentially, a Peer and Self Evaluation to MarkUs. The template can be found here: http://www.cdf.toronto.edu/~csc309h/summer/docs/A%23-PeerAndSelfEval-YOURLASTNAME_YOURFIRSTNAME.docx

Due:

Through Markus on June 15th (the Wednesday of Week 6) at 11:59pm.