Assignment 2: Mardi Gras Parade
EECS 493 Winter 2021
Released: 02/26/21
Due: 03/19/20 at 11:59pm

Submission instructions: Please submit your work to Canvas as a zip file, named "hw2_<your_uniq_name>.zip." Please keep the existing folders but feel free to add more folders or files (e.g., js or css). All files, except index.html and README (for bonus features, text only README.md is also fine), should be in folders*. Not following the upload instruction will result in a penalty*.

- Rename the hw2_starter_code folder, which **contains the code you write**, including the index.html and other folders, into hw2_<your_uniq_name>.
- Zip the folder and upload the zip file. Renaming (e.g., “-1”) done by Canvas is fine.

Objective: The objective of this assignment is for you to gain practical experience in building interactive user interfaces with JavaScript/jQuery. No external library is allowed unless instructed to do so.
See this video for a project overview: https://youtu.be/RJ_3LLsWWxk

Instructions: We would like you to create a digital “Mardi Gras” game as outlined in the attached spec below. The application has 6 main components (denominator of 100 points, with 15 extra bonus points possible):

1. Parade floats moving along the parade route - *15 points*
2. Parade floats throwing beads and candy - *30 points*
3. A person (controlled by the user) whose goal is to collect beads and candy - *15 points*
4. A splash screen on page load - *10 points*
5. A scoreboard - *10 points*
6. A settings panel - *20 points*
7. Optional bonus features - *up to 15 points*

Starter code: you will use the starter code we provide to finish this assignment:
https://drive.google.com/file/d/1wu1TnR60RVwt5mmHd-ImM4krXrJ_S8RL/view?usp=sharing
The starter code we provide includes the following:

- The base containers for the game window and scoreboard
- The parade route, and the parade floats (sitting off screen on the left - see the HTML file)
- A person who can move up/down/left/right (via the keyboard arrow keys)
- Other relevant variables and helper functions

Notes:

- You aren’t required to use all parts of the starter code, but it’s there to help you.
- Please refer to Piazza note @274 for modifications and clarifications:
  https://piazza.com/class/kjumyiqcbaf2po?cid=274

- Make sure that your application (webpage) behaves properly on the latest version of Google Chrome. Your graders will use Chrome (hint).

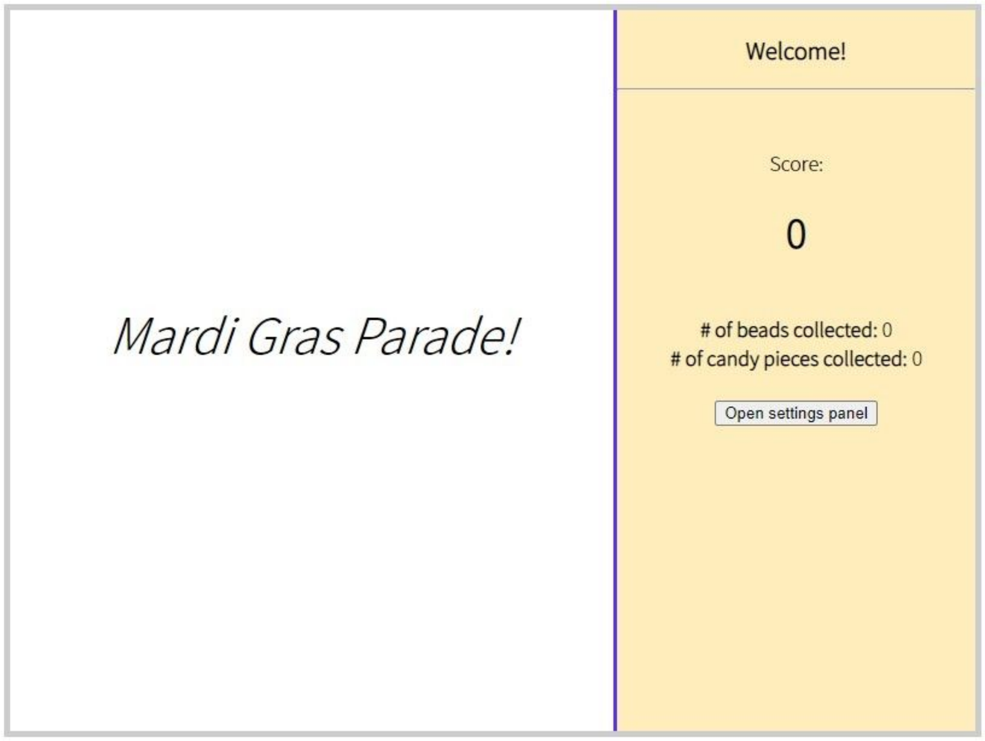
Contents of This Spec:

Blaster Game Demo:
Before starting with this assignment, we *strongly recommend* that you check out this demo on creating a game with JavaScript. The starter code is very similar to the code in this assignment, and this demo will walk you through some starter code as well as some common functionality. The demo is about 45 minutes and will make your progress in this assignment much easier!

Blaster Game Video: (Consider downloading for better quality! Captions have been added also.)
https://drive.google.com/file/d/1ttQV8_Wzm37cXrbEYuBXPAixG_UzQNFZ/view?usp=sharing

Blaster Game Starter Code:
https://drive.google.com/file/d/1r5YC7vx_9l5GBzTYRpGE-5ewFbGN12Lk/view?usp=sharing

Screenshots and Game Overview:
The loading/ splash screen:


Mardi Gras Parade!

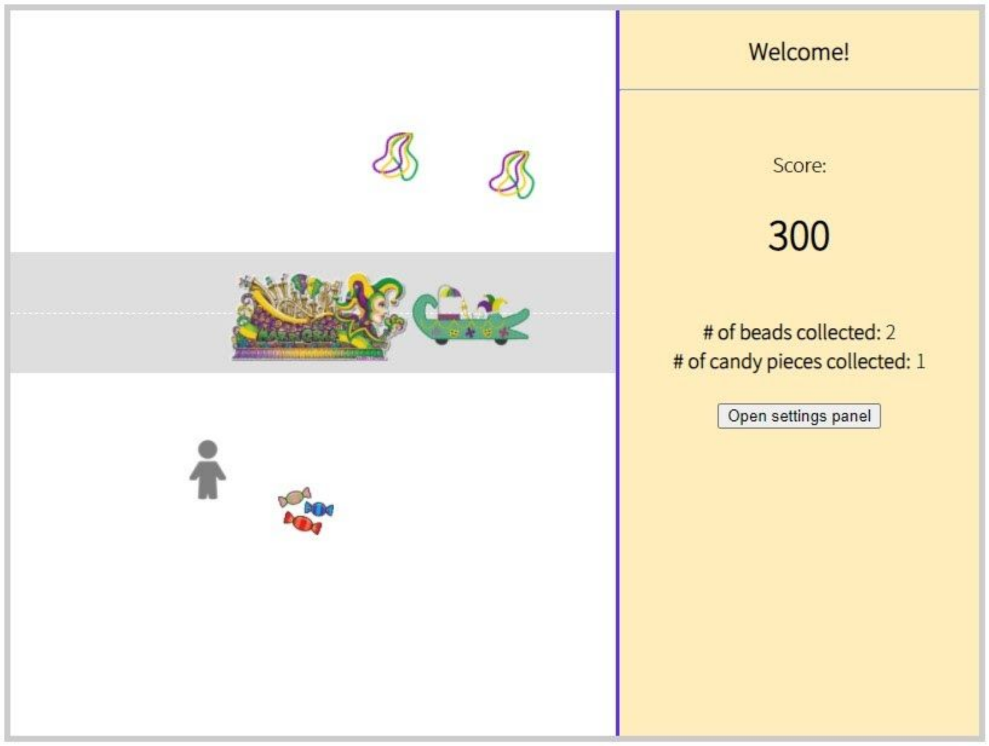
Welcome!

Score:

0

# of beads collected: 0
# of candy pieces collected: 0

Here is a screenshot of gameplay, with the scoreboard on the right:

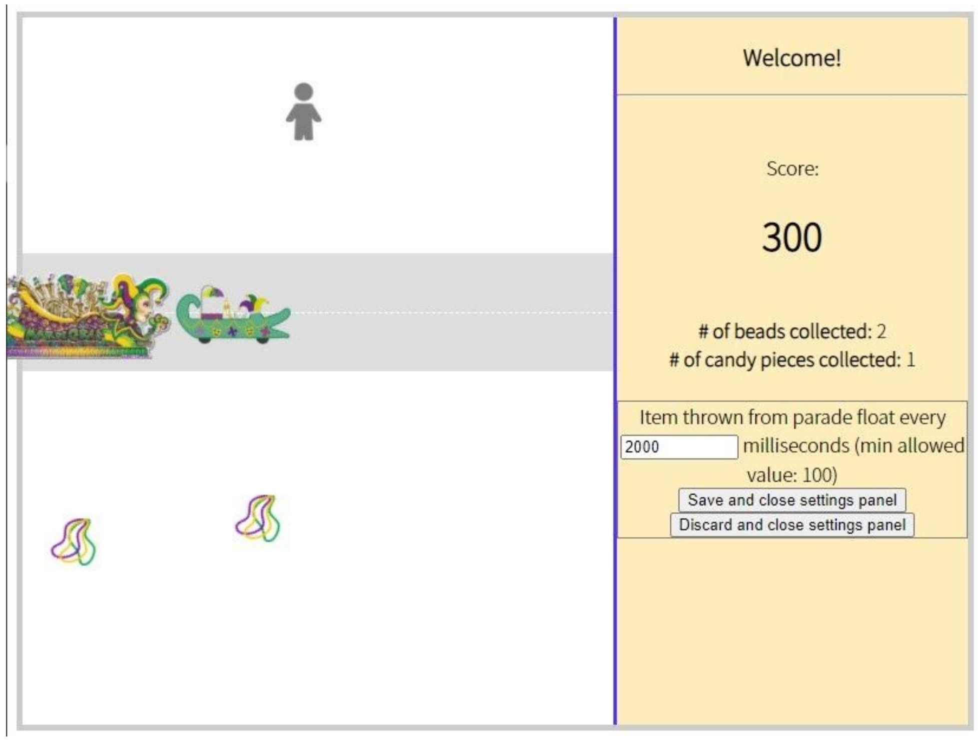

Welcome!

Score:

300

# of beads collected: 2
# of candy pieces collected: 1

The settings panel:



Welcome!

Score:

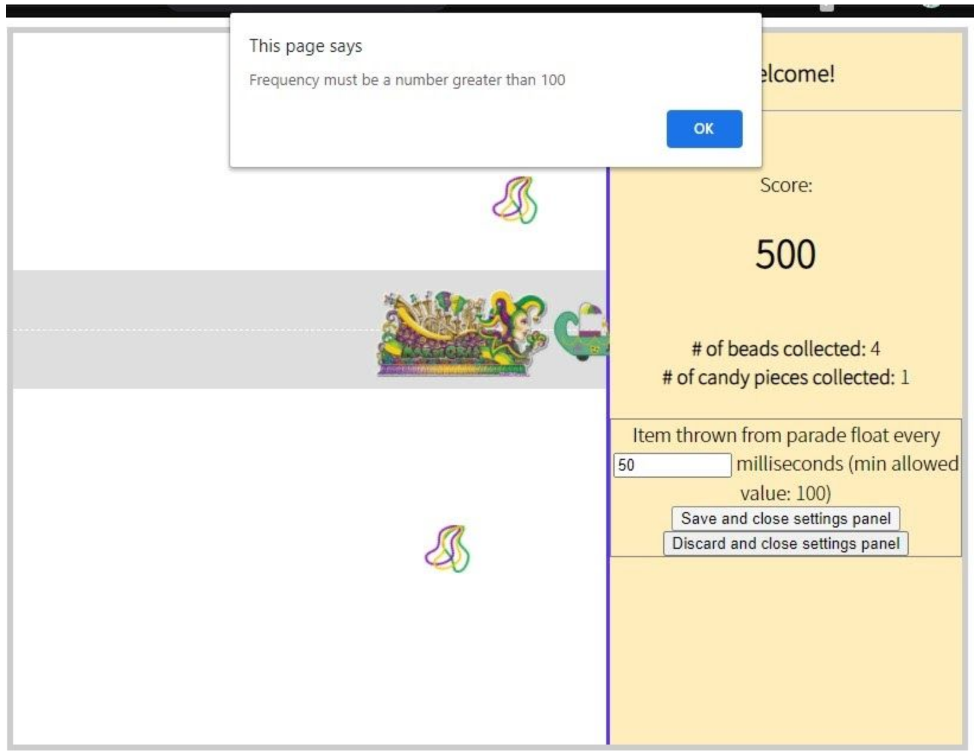
300

# of beads collected: 2
# of candy pieces collected: 1

Item thrown from parade float every 2000 milliseconds (min allowed value: 100)

Save and close settings panel
Discard and close settings panel

An alert about an invalid value:



This page says

Frequency must be a number greater than 100

elcome!

Score:

500

# of beads collected: 4
# of candy pieces collected: 1

Item thrown from parade float every 50 milliseconds (min allowed value: 100)

Save and close settings panel
Discard and close settings panel

Requirements: We outline requirements for each of the game components below. **Everything listed below, unless labeled as "Suggested", is required**. Please be sure to examine the provided video closely to understand the expected behavior

1. **General**
   a. Use relative paths for images. *- 25 points off if not followed.*

2. **Parade floats moving along the parade route** *- 15 points*
   a. *(Already provided in the starter code)* **Parade route:** A long gray rectangle representing the parade street route. It should also have a dashed white line (as most roads have). Suggested values:
      - Street color: #dedede
      - Dashed line: white
      - Street "height": 100px

   b. *(Already provided in the starter code)* **Parade floats:**
      - Alligator float
        - Suggested width: 101px
        - Suggested height: 80px
      - Sparkling queen float
        - Suggested width: 148px
        - Suggested height: 80px

   c. **Parade float movement:**
      - There are 2 floats, and they should move from the left to the right side of the screen.
      - The floats should be approximately vertically centered (equal spacing from the top and bottom) within the parade route
      - The alligator float should be the first one in the parade, followed by the sparkling queen float
      - The floats should be synchronized and move smoothly
        - Suggested speed: 2 pixels per 50 milliseconds
      - Once *both* floats have exited the route on the right side of the screen, they should reappear on the left side of the route.
      - The floats should not obstruct the view of the scoreboard on the right side of the screen (e.g., if they are on the right side of the screen, they should appear behind the scoreboard)
      - If the person walks into the street and in front of the alligator float, the floats should stop moving (until the person moves out of the way). The person can walk behind the sparkling queen float without issue.

3. **Parade floats throwing beads and candy** *- 30 points*
   a. The first float (i.e., the alligator float) should "throw" items from its head.
   b. By default, the float should throw an item every 2 seconds

c. ⅔ of the items thrown should be beads, and the other ⅓ should be candy. You can implement this either as a strict “every 3rd item is candy” or as a more probabilistic “approximately ⅓ of the items will be candy”

d. When an item is thrown, it should move smoothly, and along a single straight path, to its final position. *Hint:* Likely you will want to use timers(s) to incrementally adjust the x and y position of the item.
    - Suggested speed: on the order of 5 to 20 pixels per 50 milliseconds
    - Items shouldn’t jiggle along the path. They should move along a single straight path.
    - The exact throwing speed doesn’t matter. You could throw items all at the same speed, or throw them so that they all reach their destination in the same amount of time. Either implementation is fine.

e. Items should be thrown varying distances (vertical and horizontal) from the parade float, above and below the route. In your implementation, you may consider choosing either a random “final” distance from the float or a random “incremental” distance change (e.g., to perform each 50 milliseconds)
    - Items can be thrown onto the parade route, but they should be thrown above and below it too. Note that items do not have to be thrown onto the parade route.
        - If you decide to throw items onto the parade route, the float does not have to stop for the items and it can drive right over them.
    - Items can be thrown in front of and behind the parade float.
    - There is no range of min/ max distances required, just make sure that items are consistently being thrown onto the game board.

f. Once an item has reached its final position, it should sit there for 5 seconds and then gradually fade away (e.g., over the course of 2 seconds) and be removed from the screen

g. Beads/candy should be thrown within the game window. If the beads/candy go outside the bounds of the game window, the part outside the game window should be hidden (e.g., shouldn’t appear in front of the scoreboard).

h. Suggested dimensions
    - Candy
        - Width: 46px
        - Height: 40px
    - Beads
        - Width: 40px
        - Height: 40px

4. A person (controlled by user) whose goal is to collect beads and candy - *15 points*

a. *(Already implemented in the starter code)* The person moves up/down/left/right, controlled by the keyboard arrow keys

b. When the person comes into contact with beads or candy:
    - The score (in the scoreboard on the right) should increase by 100

        - A yellow circle should appear behind the beads/candy (to provide the visual effect that it has been collected)
        - The beads/candy and yellow circle should gradually disappear (e.g., over the course of 1 second)
        - The counters for the number of beads and candy collected (in the scoreboard on the right) should be updated

    c. The person can cross the street but cannot cross through the parade floats. The person should stop moving (in the same direction as the floats) if a parade float is right in front of the person.
    d. *(Repeated from above)* If the person walks into the street and in front of the alligator float, the floats should stop moving (until the person moves out of the way). The person can walk behind the sparkling queen float without issue.
    e. Your choice: You can choose to allow the person to collect items as they are fading away, or not. We will allow either implementation.
    f. Your choice: You can choose to allow the person to collect items as they are being thrown/ moving across the screen, or not, and only allow them to collect items once they have reached their destination. We will allow either implementation.

5. **A splash screen on page load** - *10 points*
    a. When you first load the HTML page, a splash screen should be shown; the game board (e.g., parade route/floats/person) should not yet be visible.
    b. In large italics, the splash screen should display the text “Mardi Gras Parade!”
    c. After 3 seconds, the splash screen should disappear and the game board should appear. When the game board is displayed, the parade route and person should be visible immediately, and the parade floats should start moving in from the left side of the screen.
    d. During the time the splash screen was shown, there should have been no player or parade float movement behind it; the parade floats and people can only move when they are visible on the screen.
    e. Your choice: You can hide or show the settings option from the splash screen. We will allow either implementation.

6. **A scoreboard** - *10 points*
    a. The scoreboard (within the yellow pane on the right side of the page) should be always be visible
    b. A “Score” label with the current score, which should update based on events on the game board
    c. A “# of beads collected” label with that count, which should update based on events on the game board
    d. A “# of candy pieces collected” label with that count, which should update based on events on the game board

7. **A settings panel** - *20 points*
    a. Beneath the score area should be a “Open settings panel” button; clicking on this button should replace the button with a settings panel
    b. The settings panel should contain the text “Item thrown from parade float every [blank] milliseconds (min allowed value: 100)”, where “[blank]” should be a textfield with default value 2000
    c. The textfield should be editable (the user can click in to type a new value)
    d. There should be a “Save and close settings panel” button
        - When it’s clicked, if the value entered is a number and greater than or equal to 100,
            - the settings panel should be hidden and the “Open settings panel” button should be shown again
            - the game should immediately update to now throw beads/candy at this new frequency
        - When it’s clicked, if the value entered is not a number or is less than 100, a simple alert dialog should be shown (using the “alert” function) displaying the message “Frequency must be a number greater than or equal to 100” and with button “OK”. When the user clicks “OK”, the settings panel should remain open and with the invalid textfield input.
    e. There should be a “Discard and close settings panel” button. When it’s clicked,
        - the settings panel should be hidden and the “Open settings panel” button should be shown again
        - the game should retain the existing throwing frequency
        - The next time you click the “Open settings panel” button, the textfield should NOT contain the invalid input, and instead should contain the existing throwing frequency

8. **Optional bonus features** - *up to 15 points*
    We’ll give you bonus points (up to 15 points, dependent on the difficulty as assessed by the grading staff) if you implement any of these additional features. **If you create bonus features, please include a README file describing what you built.** The README should be a file that can be opened by a text editor (i.e., .txt or .md). It doesn’t have to be super detailed. Describe what you have done, give your thoughts about it’s implementation difficulty, why it is an interesting/ challenging/ fun feature to add, and how it changes the user experience of this game.
    a. Some “game over” state - you’ll need to decide the condition for “game over” as well as some visual effects to indicate the game is over
    b. Play audio, e.g., different audio for when the player collects different items or runs into the parade floats
    c. Additional settings in the settings panel, e.g., how long candy/beads stay visible on the game board

d. Some way for the player to get temporary “special powers” so that they can walk through the floats (e.g., they pick up an invisible cloak or a king cake and then they have special powers)
e. Parade float throwing an item directly at the player, and possibly reducing the score if the player is hit
f. Have an opponent (e.g., a dog, cat, or anything) that will race with the player to see who gathers more
g. Have one or more non-player characters (NPC) who will participate in the parade and interact with the player (e.g., blocking).
h. Other cool and creative ideas you may have that make the game more fun or robust!

Hints:

Hint 1: Input Format for Frequency
If the user inputs a string like “100a”, “parseInt” and “parseFloat” will parse it as “100”. Most students will use parseInt/parseFloat in their implementation, so we will not be too strict about the textbox input. When we test your program we will test with inputs that are clearly numbers (e.g., “5000”) or clearly not numbers (e.g., “abc”).

Hint 2: Changing Item Size
If you are trying to set the width and height for candy and beads elements but it's not working, double check what element you're actually setting the width and height for. If you are setting the width and height on the "div" containing the image, this doesn't actually affect the image size; it's simply a container for the image. You likely will need to set the width and height for the image element itself.

Hint 3: Event Listeners
If you think your event listeners aren't being triggered, one common error occurs when event listeners are created (and attached) before relevant DOM objects exists.
A way around this is to create event listeners that are attached to the body, or another element that exists immediately on the page, and then filter events for a given selector. For example, if I have a UI where I expect items to get added to the page dynamically, and I want to have a "delete" button next to each dynamically added element, I might create the event listener this way:

```
$("body").on("click", ".deleteX", function(event){
 ...
});
```

Here, the "body" is listening for all click events, and is essentially only passing them to the callback if the item clicked actually had the "deleteX" class. See the "selector" arg here: https://api.jquery.com/on/

Hint 4: Item Locations and Alignment
If you are having trouble setting a bead/ candy item location correctly (i.e., you are able to move it, but can't seem to position properly), then check the values you are setting and how they relate to the parade float location. Depending on your implementation, it is possible that the floats and candy/bead items are positioned relative to different ancestors, so it is possible that if you set the same value, e.g., top: 300px, for a parade float and a candy/ bead item that they actually don't get aligned vertically.

There are three potential options for resolving this: (You only need to choose one)

- Manually add an offset yourself, if you know what the vertical offset is between the parade float's ancestor and candy/bead item's ancestor
- Instead of setting x/y location using the jQuery "css" function, you could use the jQuery "offset" function (https://api.jquery.com/offset/). The getter and setter versions of "offset" tell you the left/top offset relative to the "document", so you don't need to worry about relative/absolute positioning.
- You could make sure that you put the parade floats and candy/bead items under the same positioned ancestor, so that the "css" getter and setter functions affect the parade floats and candy/bead items in the same way.