Name: Jiecao Wang Student #: 997974811 CDF: g3wangjj
Name: Elio Tanke Student #: 999629282 CDF: c5tankee

Game Title: Tap Tap Bug

Basics: Tap the bugs before they eat all the food. You have 60 seconds.

Core Features:

Bug	Points	Speed Level 1	Speed Level 2	Probability of Appearing
Orange	1	60	80	0.3
Red	3	75	100	0.3
Black	5	150	200	0.4

You may pause during the game
Timer: 60 seconds countdown
5 Food Items: apples, oranges, plum, pear, and bananas are randomly created at the bottom half of the view port

---

Rules:

The game ends when you run out of time or the bugs eat all the food.

Challenges:

1. Try to get more than 100 points!
2. Prevent the bugs from eating any of your food.

Design:

--Bugs--

Bugs are drawn using canvas, with circles, ovals (via circles and scaling), and lines. Bugs are created every 1 to 3 seconds and are moved every 60th of a second. Movement is described below. They are created with multiple properties: type, speed, food target, and score. The type is the color and chosen randomly with certain probabilities. The speed depends on the type. The food target is the closest food determined by the distance function. If the food is eaten by another bug, the food target changes. The same happens once the bug eats one food. The score is the score gained by the player when the bug is killed.

--Food--

The food is drawn using circles and lines. This was namely done through trial and error to make things look good. They are positioned

randomly in the second half of the canvas, using the math modules random function and multiplying the function by the range it can exist in and adding the minimum value. Food disappears when a bug is within a specific radius of the food.

--Movement--

Every 60th of a second, we redraw the food and bugs. However, we also update the bugs positions so that they move to the closest food to them. This is done using the distance formula. We go through each bug in an array, and update their position to move toward the nearest food.

The formula is:

movement of X = (bug speed)*(change in X)/(distance*(frame rate))
movement of Y = (bug speed)*(change in Y)/(distance*(frame rate))

These numbers are incremented to the X and Y coordinates of the bug every 60th of a second.

--Timer--

We initially record the time that the game starts via the system time. Then, when we calculate the time elapsed (60 frames per second) by checking the system time again and subtracting it from the total 60 original seconds given. Because this is not an integer, we round this number down for display purposes and change the html element by getting its ID.

--Pause--

An event handler was used to detect when the button was pressed. It is designed not to function when the game is over.

When you click pause, bug creation is stopped (bugs are created on a random interval between 1 and 3 seconds), bugs stop being updated (for movement), and pause button is changed to play via get element by ID method.

When the game is paused and you click the play button, the game resumes by doing the following: It updates the "start time" to the

current system time so the timer can adjust accordingly, it starts the timer again, it begins creating bugs again, and it begins moving the bugs again. Then the play button is changed back to pause via the same method as before.

--Tap--

An event handler is used to detect taps. When a tap is detected, the X and Y coordinates of the tap are recorded. Then, using the distance formula, we check if there are any bugs within the defined radius. To kill bugs, we also consider their shape, and as they are rectangular, we created 9 cases: 4 corners, 4 sides, and the bug's body, where the distance depends on where the taps coordinates are.

When a bug dies, it fades out, and is removed from the bug list so it is not drawn again.

Testing:

Testing High Score:

1. To test if the high score has the correct value we did the following:
- Check if the initial value is 0 on the homepage.
- Obtain a high score in level one and check if it is recorded on the homepage.
- Obtain a lower score and check if the high score has not changed.
- Obtain a higher score and see if the new score has been recorded.

2. We do the same for level two.

3. Once this is done, we check to see if the high scores correspond to the right levels on the homepage

Testing Pause Button

1. To check if the pause button functioned correctly, we did this:
- Click pause button to see if all bugs have stopped.
- Wait 4 seconds to see if any bugs have been created.
- Check to see that the timer has stopped.

2. To check if we can resume from pause, we did this:
- Click on resume button and check if all the bugs start moving again.
- Check that the timer is moving again from the paused time.
- Check that new bugs are being created by waiting at least 4 seconds.

3. Checking the integrity of the pause button.
- See that when paused, bugs cannot be killed.

- Cannot pause once the game is over.
- Once resumed, timer considers the milliseconds between intervals that its resuming from.

Testing Timer

1. Timer should decrement when game starts
- Check that as soon as game starts, there are 59 seconds left.

2. Timer should decrement every second.
- Check with your clock to see that every second, the timer decrements 1 second.

3. Timer should stop when it reaches 0 seconds.
- Check that the timer stops decrementing.

Testing Bug and Food Behaviour

1. Check that bugs eat the correct food.
- Observe that the bugs move toward the closest food.
- Observe that if another bug eats the food it was trying to eat, it moves toward the next closest.
- Observe that once a bug eats one food, it moves to the next closest.

2. Check that bugs move at the correct speed.
- Observe that orange is slowest, red is faster, and black is fastest.
- Observe that in level 2, bugs become faster than the speeds in level
- Observe that bugs are moving smoothly to the eye and one cannot tell that it is moving at intervals.

3. Check that bugs are created every 1 to 3 seconds at correct probabilities.
- Time each interval between bugs to see that the intervals are between 1 and 3 seconds.
- Keep a set of data for how many of each type of bug appeared and calculate probabilities and match them with theoretical probabilities.

4. Observe that bugs do not walk out of the canvas dimensions.

5. Observe that there are 5 foods in the second half of the canvas.

Testing Game Over

1. Check that game ends when food is gone.
- Let all the bugs eat all the food.
- Observe that the game over popup appears.

2. Check that game ends when timer goes to zero.
- Play the game and survive until timer reaches 0.

- Observe that the game over popup appears.

3. Check Game Over integrity.
- Check that when game ends, the timer stops.
- Check that the bugs stop moving.
- Check that no more bugs are created.
- Check that you cannot kill anymore bugs.
- Check that you cannot press the pause button.

Testing Game Over Popup

1. Check that all buttons and data are there.
- Check that the restart button and exit button are there.
- Check that current score is there and correct. (play the game and test out a score)

2. Test the exit button.
- Click the exit button and see if you go back to the start page.

3. Test the restart button.
- Click the restart button and see if the game restarts and the bug speeds have not changed.

Testing Tap and Kill

1. Check that bugs within a 30px radius of a click will die.
- Using canvas, draw a circle every time you click with a radius of 30px.
- If a bug is in the radius, check to see if it dies.

2. Check that multiple bugs die at once.
- Draw the 30px radius per click and see if all bugs in the radius die.

END OF DOCUMENT