Tap Tap Bug

Christiaan Rutten, c3rutten

Vera Sipicki, c3sipick

Basics:

The basic goal of this game is to stop the ants from eating your food by squashing them before it's too late! This is accomplished by tapping on them before they reach the food.

Rules:

-Game ends if:

-60 seconds pass

-all 5 pieces of food are eaten

-Food:

-start with 5 pieces

-disappear as soon as ants reach them

-Bugs:

-black: fast (150px/sec), worth 5 points, 0.3 probability of appearing

-red: medium (75px/sec), worth 3 points, 0.3 probability of appearing

-orange: slow (60px/sec), worth 1 point, 0.4 probability of appearing

-all bugs spawn at the top of the screen and head straight to the nearest food

-new bugs enter every 1 to 3 seconds

-level 2 increases move speed of ants

-taps within a radius of 30px kills an ant

Functions

Game start: checks which level was selected, chooses bug speed based on that. Hides main menu and shows game canvas and info bar. Loads all images, puts food on the canvas and starts the game.

Game end: checks for high score, and changes it if one was achieved. Brings up the end game menu which allows for restarting the game or returning to the main menu (which hides the end screen, info bar and game canvas, and shows the main menu).

Pause: triggers an if statement in the animate function that prevents clearing the canvas, updating bug positions or drawing. Also triggers and if statement that prevents touches to the canvas from killing bugs while paused. Stops the timer, and changes the pause button to resume button. On resume restarts timer and allows the game to progress.

Animate: works on a timeout variable, which calls animate 60 times a second. This updates bug positions, clears the canvas and redraws them.

Timer: has a variable that is assigned an interval of 1000 (each second) that calls a function that increases a variable by one and updates the timer display element.

Taps: canvas has an event listener on mousedown that gets the taps coordinates and compares them to every bug, deleting every bug it hits and adding the appropriate amount of points to the score.

Classes

Bug: stores its coordinates, speed, image and dimensions. Also has functions to draw the bug, as well as how to target and move towards food.

Food: stores x y coordinates of the food, whether or not it has been eaten, and a function to draw the food.

Algorithms:

First of all, each bug would use the Euclidean distance formula to find the closest food. At that point, I would calculate the x coordinate distance between the closest food and bug, and the y coordinate distance. I would also store the Euclidean distance in the class for checking how much more the bug needs to travel. The angle at which the bug needs to travel was calculated using the tangent of the x and y coordinate differences. Afterwards the bug moved towards the food until the difference was less than 1.5 (a small radius around the food).

Challenges:

-Had difficulties with figuring out how best to delete bugs when tapped, settled on splicing them out of the bugs array so that they would not be redrawn.

-Targeting new food once the first target was eaten proved difficult; occasionally the next food would provide null coordinates.