csc309: Programming on the Web - A2 readme file

Submitted by:

Jana Micaela Gablan

Jennifer Leung

cdf IDs:

c3gablan

c4leungj

Game Design:

When the page is loaded, the form start page allows users to select the game level and to start the game. On start, the hidden canvas and game menu is set to visible and the start page hidden. This is handled by switchView(), which also initiates keeping game scores, the timer, toggles the game-over variable indicator, and triggers init().

The game environment is set by the functions mostly under the draw bug section on the script.js file. drawgame() draws the food on the canvas, of whose locations are randomized. The location of the remaining foods are stored in a global list, foodlist[].
blackbug(), redbug() and orangebug() initiates the bugs. These functions set the canvas location where the bugs would be summoned, sets their motion speed depending on their distance to the closest food and these properties are stored on a global list poslist[].
drawbb(), drawrb() and drawob() makes the bugs move by rendering the animation heading towards a food, eating it, and then moving on to the next closest food.

The timer, play/ pause button and highscore displays sit on gamemenu. timer() handles the game timer, the play/ pause button is indicated by toggle "freeze" and the high score is set with setscore().

When either all the food is eaten or the time has ran out, gameover() hides the canvas and displays the form end, which allows users to go back to main menu or replay the game. The main menu brings the user back to the start page, while the replay game allows the user to instantly play again in the same level.

The init() function calls bugspawn() and captureclick(). bugspawn() decides which of the three bugs are to be summoned on the canvas. captureclick() on the other hand, handles mouse clicks on the canvas, taking the mouse click coordinates and calls buginrange(). buginrange() checks with all living bugs if any of them are within a 30px range of the click; they are indicated to be dead, and the function adds up the score, accordingly by each bug's color, for the current game. As captureclick() just then continuously waits for a mouse click on the canvas until the game ends.

How game functionalities are tested:

Functionalities are mainly tested by trial and error. One us are coding using Adobe Dreamweaver and the other, Firefox's and Chrome's web developer tools to increment completed work and track values in the code. Some functionalities, such as game menus, which are simpler were implemented first; and how the user interacts with the canvas, were coded later. Writing more codes were built on top little by little; and so the game was built through an iterative and incremental development design method.

Note on bug's motion speed:

The bugs do not move at the required speed. However, the black bugs run faster than the red, which run faster than the orange ones; and they move nice and smoothly on the canvas. Moreover, the bugs' speeds are accordingly adjusted for level 2.

How game objects are designed:

The bugs in particular, are not actually objects. poslist[] stores an array of properties for each bug, of which properties are set and calculated by the blackbug(), redbug() and orangebug() functions. These properties include an indicator whether the bug is alive or not, the coordinates of the nearest food around them, their motion speed in accordance to the game's level and etc. In addition to the list of properties is their animation on the canvas. drawbb(), drawrb(), drawob() handles the redrawing of the bugs' images on the canvas to render animation for black bugs, red bugs and orange bugs, respectively.

Aides used:

http://stackoverflow.com/questions/19041434/html5-canvas-move-directly-to-point
http://jsfiddle.net/loktar/CajbL/
Some ideas from the links above were used to create the animation for the bugs.

W3schools tutorials
Lecture Notes