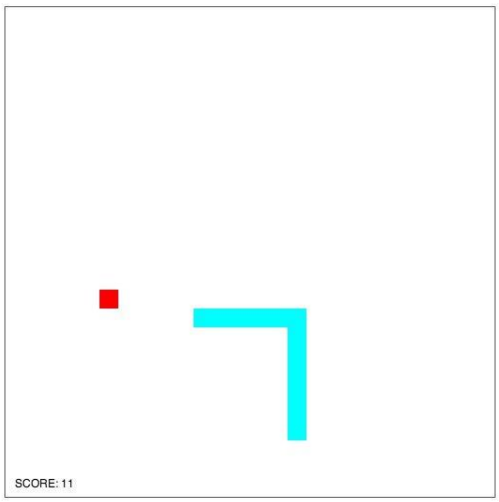
Lesego Nzimande

HTML Games

Academic Year 2020/2021

Video Game Report



Game Requirements

A) The game is a puzzle game.

b) The game is 2d.

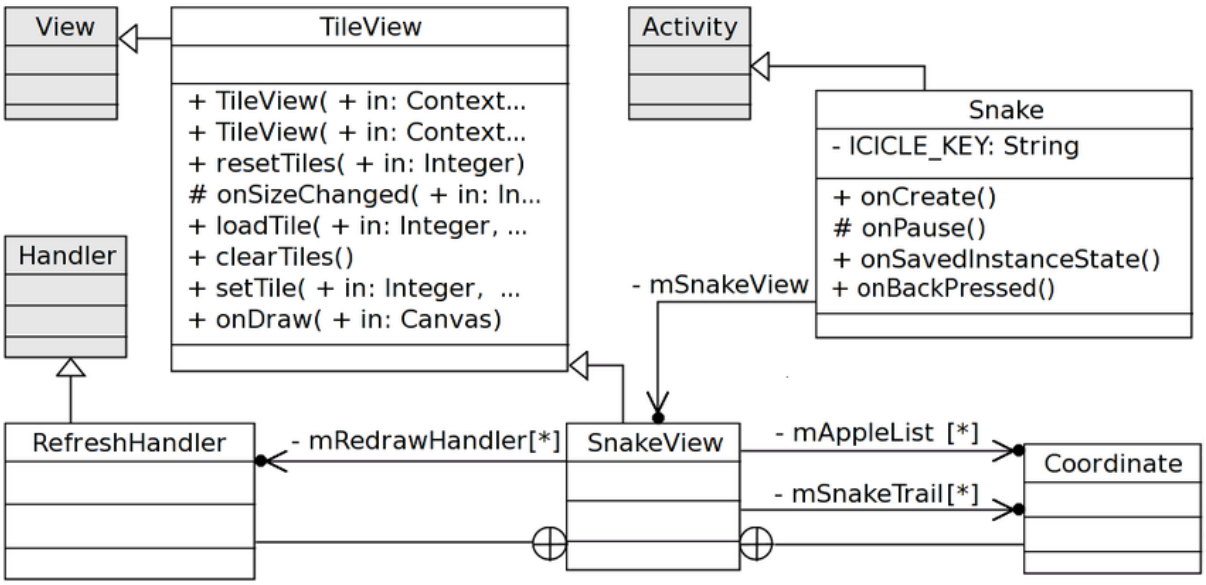
c) The game runs within a user's browser

d) The game is modeled using javascript, HTML5 & css

e) The games use a javascript library

The game is based on the "snake" video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle.

The player controls a square object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. The end of the trail is in a fixed position, so the snake continually gets longer as it moves; the player loses when the snake runs into itself.



Instructions

Step 1: Using the text editor Notepad++ we create a decent canvas for the game.

Step 2: Give the canvas a background color and a border to make our canvas visible, we can give it a border by writing some JavaScript code. To do that, we need to insert <script> tags where all our JavaScript code will go, we can now write some JavaScript code, between the enclosing <script> tags.

Step 3: To display the snake on the canvas, we can write a function to draw a rectangle for each pair of coordinates; after this if you open your browser: you will see something like a dash between the screen.

Step 4: Enable the snake to move.

Step 5: Our next task is to change the snake's direction when one of the arrow keys is pressed. Add a specific code after the drawSnakePart function. There is nothing tricky going on here. We check if the key pressed matches one of the arrow keys. If it does, we change the vertical and horizontal velocity as described earlier.
Notice that we also check if the snake is moving in the opposite direction of the new intended direction. This is to prevent our snake from reversing, for example when you press the right arrow key when the snake is moving to the left.

Step 6: For our food object, we have to generate a random set of coordinates. We can use a helper function randomTen to produce two numbers. One for the x-coordinate and one for the y-coordinate. We also have to make sure that the food is not located where the snake currently is. If it is, we have to generate a new food location.

Step 7: Growing our snake is simple. We can update our advanceSnake function to check if the head of the snake is touching the food. If it is we can skip removing the last part of the snake and create a new food location.

Step 8: To make the game more enjoyable for the player, we can also add a score that increases when the snake eats food. Create a new variable score and set it to 0 after the snake declaration. let score = 0;Next add a new div with an id “score” before the canvas. We can use this to display the score. Finally update advanceSnake to increase and display the score when the snake eats the food

Step 9: There is one final piece left, and that is to end the game. To do that we can create a function didGameEnd that returns true when the game has ended or false otherwise. Now the game is complete.