JavaScript DOM & UI

Course Teamwork Project Documentation

Tetris

**Team**: Onaga

* Andrei Trajkov (andrei\_pl)
* Denis Kyashif (stinger907)
* Hristo Dimitrov (Ico093)
* Hristo Kovachev (Kovachev) – didn’t participate
* Mihail Bonev (Bonevm01)
* Mihail Vakov (mivak)

**Project URL:** <https://github.com/stinger907/tetris-js>

**Goals**: The goal of this project was to practice animation development with HTML5 Canvas and SVG, including their respective frameworks. We tried to uphold the best practices in JavaScript and satisfy the project requirements.

**Description**: This is a simple implementation of the famous Tetris game where the player has control over the falling shape and tries to place it on the board, in its best possible adjustment. The player gains points when a complete line of blocks is formed, the line is then removed and the game goes on. If a specified, score-related, stage in game is reached - the level is raised, so is the shape’s falling speed. The game ends when there is a board block on the highest board row. The player can afterwards save his/hers score in the high scores ranking.

The game meets the following requirements:

* Use HTML5 Canvas or a Canvas framework like KineticJS paper.js or other
* Use SVG or a SVG framework like RaphaelJS or other
* Create animations, either for the Canvas, SVG or both
* The application must work in the latest versions of the browsers: Google Chrome, Firefox, IE 10/11, Opera and Apple Safari
* Use DOM Manipulations like native DOM API or JQuery

The project source code consists of:

**tetromino.js** - This is the Tetris shape object factory closure. It contains the hardcoded shape templates with every possible direction and Shape constructor function. Only the getRandomShapeAtPosition() function is visible to the outside world.

**background.js** - This file consists of ten colored figures, drawn using RaphaelJS, which are used as an application background.

**tetris.js –** Contains the core application logic, including the gameplay and animation rendering functions.

* drawLogo() - using SVG and Raphael this function draws the logo of the game. The logo consists of a colored figure (part of the game shapes), the name of the game and the logo of HTML5.
* drawBoard() – visualizes the game board with its borders and filled blocks
* drawShape() – visualizes the current falling shape
* drawMetrics() – visualizes the current score, level and the next shape
* hasLeftLimit(), hasRightLimit() - check for obstacles near the falling shape – return true or false
* isOverFallenItem() – checks if after rotation the falling shape is over another item - returns true or false
* findMinMax() – returns the numbers for the leftmost and rightmost position of the falling object. This function is used to keep the shape within the game board when it is rotated near one of its borders
* rotateIfPossible() – if there are no limitations, the falling object rotates on each “up-arrow” click
* addToBoard() – add the current shape to the board when it has no further possible move downwards
* checkForCompleteLines() – if a board row has no empty cell – it is considered a complete line, therefore it is removed from it and all the rows above it are displaced one level downwards
* checkForGameOver() – if the highest board row has at least one non-empty cell – the game ends
* playSound(sound, stopIt) - This function is used for background music and when some arrow button is pressed. The “sound” parameter is the sound you want to use and “stopIt” is a boolean variable that shows what you want to do with this sound. To stop and restart it when you call it every time, or to restart only if the previous call has finished playing it
* run() - function that performs the animation itself. When we have level up, the interval of execution is changing
* Music button click event - Checks the boolean "isMusicOn" for its value and calls switchMusic() to make changes on the music played. Changes remain until the button is pressed again, or until the page is refreshed
* switchMusic() - Make changes according to the value of the boolean "isMusicOn", meanwhile the music can start only after the button for Start new game has been pressed
* Pause button click event – toggles pause/resume game

**index.html** – assembles all the application components. Serves as an improvised application entry point and launcher

**style.css** – contains styles used in several parts of the application user interface