Assignment 2: JavaScript

100 Points Possible

25/04/2024

Add comment

22/01/2024 to 30/06/2024

Details

COMP284 Scripting Languages (2023-24) -- Assignment 2: JavaScript

Your task for this assignment consists of two parts:

1. Develop a JavaScript program that provides the functionality stated in the **Requirements section (https://canvas.liverpool.ac.uk/courses/73095/assignments/265339)** below.
2. Make the JavaScript program that you have created accessible and usable via the URL

   ```
   https://student.csc.liv.ac.uk/~<your user name>/game.html
   ```

   taking care of the requirements set out in **Submission and Setup section (https://canvas.liverpool.ac.uk/courses/73095/assignments/265339)** below.

Requirements

The JavaScript program implements a simple game that consists of three stages, *setup*, *play* and *end*. During the *play* stage the game proceeds in *rounds*. The game is played on a grid with 10 x 10 cells. It involves a spaceship that is controlled by the user, a couple of robotic spaceships that are controlled by the computer (that is, your program), asteroids, and (initially inactive) mines. The user and your program are the two *players* of the game. All spaceships can move around on the grid and the robotic spaceships hunt the user's spaceship; the user's spaceship tries to avoid getting caught while trying to activate all the mines on the grid, possibly destroying some or all the robotic spaceships in the process.

The game always starts in the *setup* stage. During that stage the user is shown the grid and can place four different types of objects on the cells of the grid:

- by clicking on a cell and typing the letter "a", an *asteroid* is placed on a cell;
- by clicking on a cell and typing the letter "m", a *mine* is placed on a cell;
- by clicking on a cell and typing the letter "r", a *robotic spaceship* is placed on a cell;
- by clicking on a cell and typing the letter "u", the *user's spaceship* is placed on a cell.

There is no limit on the number of asteroids, mines, and robotic spaceships, but there is obviously only one user's spaceship. No grid cell can initially contain more than one object. If the user types a character that is not among "a", "m", "r" and "u", an error message should be shown.

In addition to the grid, there must be a button that allows the user to end the *setup* stage of the game. If the user tries to end the *setup* stage of the game without placing the user's spaceship, then an error message should be shown and the user remains in the *setup* stage. Otherwise the game continues with the *play* stage.

At the start and during the *play* stage, the user is again shown the grid, initially with all the objects that have been placed o[illegible]e setup stage, plus additional *status information*: The number of the *round* cur[illegible]*umber of inactive mines* on the grid, and the *number of robotic spaceships* on the [illegible]here must be button that allows the user to end the *play* stage at any tim[illegible]

While in the *play* stage, t[illegible] in rounds, each round starts with the user's turn followed by the computer's turn. At the start of a round, the number of the *round* currently played is increased by one (the first round has the number 1), and the *status information* shown to the user is updated.

During his/her turn, the user can attempt to move his/her spaceship horizontally or vertically on the grid by typing one of four letters:

- "a" attempts to move the user's spaceship one grid cell to the left,
- "d" attempts to move the user's spaceship one grid cell to the right,
- "w" attempts to move the user's spaceship one grid cell up,
- "s" attempts to move the user's spaceship one grid cell down.

If the user types any other character, then an error message should be shown, the user's turn does not end, and the user has the possibility to type another character. If the attempted move would result in the user's spaceship ending up outside the grid or on a cell occupied by an asteroid, then the attempt to move fails, the user's spaceship does not move, an error message should be shown, the user's turn does not end, and the user has the possibility to type another character. Otherwise, the attempted move is successful and the user's spaceship changes cells:

- If the cell to which the spaceship has moved was previously empty, then nothing special happens and the user's turn is over.
- If the user's spaceship ends up on a grid cell that contains an inactive mine, then the mine is *activated*, the number of inactive mines shown in the *status information* is reduced by one, the user's spaceship and the mine coexist on the cell. The mine remains *activated* (and in place) for the remainder of the game. If a robotic spaceship is on a cell surrounding an activated mine, it is immediately destroyed and the number of robotic spaceships shown in the *status information* is changed accordingly. The user's turn is then over.
- If the user's spaceship ends up on a grid cell that contains an activated mine, then nothing special happens. The user's spaceship and the mine simply coexist on the cell. The user's turn is over.
- If the user's spaceship ends up on a grid cell that contains a robotic spaceship, then the user's spaceship is destroyed, and the game proceeds to the *end* stage.

If at the start of the user's turn, the user's spaceship cannot be moved, then the user's turn should end automatically and the round continues with the computer's turn.

During the computer's turn your program attempts to move each of the robotic spaceships in an order that allows each to move if at all possible. Unlike the user's spaceship, the robotic spaceships are not only able to move horizontally and vertically but also diagonally. Just like the user's spaceship, each robotic spaceship only moves at most one cell in a turn.

- If the user's spaceshi[illegible] immediately surrounding a robotic spaceship, then that robotic spaceship mu[illegible] occupied by the user's spaceship. If the user's spaceship is on a cell surroundi[illegible]e, then both the robotic spaceship and the user's spaceship are destro[illegible]aceship is not on a cell surrounding an activated mine, then only the user's s[illegible]ved. The status information is updated accordingly and the game proceeds to the end stage.
- If the user's spaceship is not on a grid cell immediately surrounding a robotic spaceship, but one or more of those grid cells contains an inactive mine, then the robotic spaceship must move to one of those mines, the mine is removed from the grid, the number of inactive mines shown in the *status information* is reduced by one.
- If none of the surrounding grid cells contains the user's spaceship nor an inactive mine, then a robotic spaceship can move to an arbitrary surrounding cell provided that this move does not take it to a grid cell that is outside the grid or occupied by an asteroid or by another robotic spaceship.
- If a robotic spaceship were to move onto a cell surrounding an activated mine, then the robotic spaceship is destroyed and the number of robotic spaceships shown in the *status information* is reduced by one.
- A robotic spaceship is not allowed to stand still if it can move. However, if a robotic spaceship cannot move at all, then the computer should simply proceed to the next robotic spaceship.

Once an attempt has been made to move each of the robotic spaceships, the computer's turn and the current round ends, and the status information is updated.

The *play* stage ends if one of the following conditions becomes true:

- the user ends the *play* stage (by pressing the button provided for that);
- the user's spaceship is destroyed;
- at the end of a turn there are no robotic spaceships left on the grid;
- at the end of a turn there are no inactive mines left on the grid;
- neither the user's spaceship nor any of the robotic spaceships is able to move.

Once the *play* stage has ended, the game is in the *end* stage. In the *end* stage the program determines the outcome of the game. The outcome is a *win* for the user if there are no robotic spaceships left on the grid but the user's spaceship has survived; the outcome is a *win* for the computer if the user's spaceship has been destroyed and at least one robotic spaceship has survived; otherwise, the outcome is a *draw*. The program should display a message indicating the outcome of the game and then stop. During the *end* stage the program should not react to any user input or actions.

Additional Requirements and Comments:

- The bulk of your JavaScript code should be in a JavaScript library called `game.js`. Before submitting your solution, you should create a copy of `game.js` named `game.pretty.js` in a directory other than your `public_html` directory, say, your home directory. Then make the file `game.js` indecipherable for humans using the command

  ```
  uglifyjs-3 $HOME/game.pretty.js --mangle --compress > $HOME/public_html/game.js
  ```

  Make sure that after [illegible] eration your game still works. Also make sure that `game.pretty.js` can [illegible] self and is not in your public_html directory.
- It is possible that dur[illegible] the user does not place any mines on the grid. On entering the *play* stag[illegible] ould recognise that, immediately proceed to the *end* stage, and de[illegible] of the game.
- It is also possible that during the *setup* stage the user does not place any robotic spaceships on the grid. On entering the *play* stage your program should recognise that, immediately proceed to the *end* stage, and declare the outcome of the game.
- It should be possible for the user to see whether a mine is inactive or has been activated. Whether the cells surrounding an activated mine are visually distinct is up to you.
- You should carefully analyse in which situations the user's spaceship and the robotic spaceships might not be able to move in order to correctly end the *play* stage in such situations.
- Ideally your program would move the robotic spaceships in such a way that they increase their chances of destroying the user's spaceship. This could be done by each robotic spaceship trying to decrease the distance to the user's spaceship with each move. But you could also implement a strategy by which the robotic killer spaceships try to `encircle' the user's spaceship in order to increase their chances. They could also `guard' one specific inactive mine, knowing that the user must eventually try to activate it.
- Also, be aware that the requirement that a robotic spaceship must move if it can, means that it might have to move onto a cell surrounding an activated mine and be destroyed. But if alternative moves exist, then they should be preferred.
- JavaScript engines differ from browser to browser. You should make sure that your system works in all commonly used browsers (e.g., Google Chrome, Microsoft Edge, Mozilla Firefox) and on all commonly used platforms (e.g., Linux derivatives and Microsoft Windows).
- Your JavaScript program should only depend on your own code. The use of JavaScript libraries/frameworks, like jQuery, is not be used.
- Your code should follow the [COMP284 Coding Standard (https://canvas.liverpool.ac.uk/courses/73095/files/10520245?wrap=1)](https://canvas.liverpool.ac.uk/courses/73095/files/10520245?wrap=1). This includes pointing out which parts of your code have been developed with the help of on-line sources or textbooks and references for these sources. You must also provide references for any language constructs or functions that you have used that were not covered in the lectures.

A program that deals satisfactorily with these additional requirements and comments, in addition to providing the basic functionality required, will receive higher marks.

Submission and Setup

Make game.html, game.js, and other files (but not game.pretty.js) accessible via the departmental web server so that the game can be played via the URL given at the beginning of the assignment. This playable version of the game must remain available, working, and unchanged from the time of submission to the end of July 2024.

Permissions of the files in your filestore must be such that no other user can view their contents in the filestore. The permis[illegible]c_html directory must be such that any user can obtain a listing of its content.

From the files on the dep[illegible]reate a single zip-file named COMP284-2.zip containing all relevant files (game.ht[illegible]pretty.js, any CSS file, and local images, but no directories) and submit it [illegible]tal submission system at https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284 (https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284) (COMP284-2: JavaScript).

The files submitted must be identical to those set up on the departmental web server. Furthermore, no alterations are allowed to the latter after files have been submitted. If a submitted file and the corresponding file on the departmental web server have different timestamps, then the later timestamp will be used to determine lateness. This applies even if the earlier file is used for marking.

Deadline

The deadline for this assignment is

Thursday, 25 April 2024, 17:00

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties. Please remember that a strict interpretation of `lateness' is applied by the Department, that is, a submission a minute after the deadline is considered to be a day late. Also remember that late resubmissions are not allowed.

Assessment

This assignment will address the following learning outcomes of the module:

- Develop computer-based or client-side web-based applications using an appropriate scripting language.

This assignment will contribute **50%** to the overall mark of COMP284. Failure on this assignment may be compensated by higher marks on other assignments for this module.

Marks will be awarded according to the following scheme, assessing the extend to which the application you have developed by the deadline meets the requirements, has been set up correctly, and submitted correctly:

- Submission, Setup, Error-freeness: 10
- Quality of the interface design: 10
- Correctness and quality of the implementation of the *setup* stage: 12

- Correctness and quality of the implementation of the *play* stage: 46
- Correctness and quality of detecting the end of the game and of the *end* stage: 10
- Code layout, Comments, References, Quality of code: 12

In more detail, the requirements above translate into about 32 criteria that your system must satisfy. Marks are given according to the extent to which the system is observed to behave in the expected way and produces corre[illegible] lesser extent, how well the code is written. Code that has no observable effect will [illegible] o marks.

As stated above, the Uni[illegible] te submissions applies to this assignment, as do the University policy on cou[illegible] n (available at https://www.liverpoo[illegible] uk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf (https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf)) and the University policy on academic integrity (available at http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/ (http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/)). You should follow the COMP284 Lab Rules (https://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/COMP284LabRules.pdf) to ensure that you do not breach the latter policy.

Feedback

You can expect individual feedback for this assignment about three weeks after the deadline. Generic feedback will be provided five days after the deadline. No work can be submitted more than 5 days after the deadline.