Assignment 2: JavaScript Part One

Course: Web-Based Systems
Number: SENG 513
Semester: Fall 2023
Due Dates: Oct 16
Instructor: Steve Sutcliffe <steve dot sutcliffe at ucalgary dot ca>
Version: 1.1.0

Objective:

To design and develop a unique and interactive browser-based game using HTML, CSS, and JavaScript. This assignment emphasizes the coding and the thoughtful design, planning, and execution of your game idea. You must demonstrate a deep understanding of JavaScript functionalities, interactivity, and dynamic rendering. Before diving into coding, comprehensive game design is required. This will allow you to envision the game's mechanics, user interface, and player experience. The objective is to synthesize design thinking with coding skills, producing an engaging game demonstrating your programming proficiency in JavaScript.

Overview:

This assignment will make up 10% of your grade. You must design a browser-based game (two-player minimum) that **manipulates the DOM** using JavaScript. Appearance and structure can be created using HTML and CSS. You are expected only to use vanilla HTML, CSS (SASS or LESS is ok), and JavaScript. For this first part of the assignment, you will create your game's interface, layout, styling and function prototypes/headers/stubs/drivers/etc. You should plan on creating a game that does not require a backend system; you will only be making a game that "lives" in the browser. You can assume all players are using the same computer. You should create the UI for your game using HTML and CSS and design the function prototypes/headers/drivers/stubs for JavaScript.

Scope and Complexity:

While the freedom to choose any game, simple or extensive, is encouraged, it's essential to manage the scope of your project to align with the time and resources available. ***When picking a more comprehensive game***, like Settlers of Catan, Monopoly, or Risk, you are not expected to implement every detail and rule.

Guidelines for Complexity:

Core Mechanism

Focus on the primary gameplay mechanics that define the essence of the game. For example, in Settlers of Catan, this might involve the mechanics of resource collection, building, and trading, but you might simplify or exclude elements like development cards or the robber.

Scale

Limit the number of elements or components. Using the Settlers of Catan example again, you could limit the game to a fixed number of rounds or reduce the board size.

Interactions

Ensure that there are at least 2-3 significant interactive mechanisms in your implementation. This could be player-to-player interactions, player-to-environment, or any other meaningful gameplay interaction.

Depth over Breadth

It's more important to have a few well-implemented, polished features than many shallow ones. Focus on depth in the mechanics you choose to include.

Clear Milestone

Set clear milestones for your game. What will be the "win" conditions? Is it a certain number of rounds, points, or achieving a specific goal? Define these clearly so that gameplay has a purpose and endpoint.

Remember, the aim of this assignment is to demonstrate your understanding of game design, your ability to code specific functionalities, and your creativity. It's not to produce a complete, commercial-grade game. Keep the scope manageable and focus on quality.

Submission Requirements

File naming and content

Each file in your submission should contain the course, date, assignment, name and UCID in a comment at the top of the file.

E.g. in HTML	E.g., in CSS
<!-- Course: SENG 513 --> <!-- Date: SEPT 10, 2023 --> <!-- Assignment 1 --> <!-- Name: John Doe --> <!-- UCID: 123456789 -->	/* Course: SENG 513 */ /* Date: SEPT 10, 2023 */ /* Assignment 1 */ /* Name: Jane Doe */ /* UCID: 123456789 */

Game Overview

Title: Include the title of your game.

Target Platform: Specify if the game is designed for desktop, mobile, or both.

Game Genre: Classify the game (e.g., puzzle, adventure, strategy, etc.).

Games Objective: Clearly state the primary goal the player needs to achieve, to win or to progress.

Rules of the Game: Briefly explain the rules, penalties, bonuses, or special conditions.

Game Mechanics: Describe the primary actions the players will perform.

UI Design

Using HTML and CSS, design your interface and the essential elements of your game. For example, if you are building a chess game, you may want an area to designate whose turn it is, the pieces lost, etc. Include the details outlined below.

Layout and Structure: Develop the skeletal structure using HTML, laying out significant areas like the game board, score section, control buttons, etc. Ensure alignment and spacing are considered.

Visual Elements: Incorporate essential visual game elements, including player avatars, game tokens, symbols/icons for starting a new game, etc. NOTE: If you are using game assets you found online, BE SURE those assets do NOT violate copyright laws. Look for CC0 assets. If you need to provide attribution to use those assets freely, include that in your UI Design.

Functionality Design

Design the pieces necessary to make your game functional using descriptions or function prototypes/headers/stubs/drivers. This section should have no implementation except for the code to run your stubs/drivers if you are using them. The goal is to quickly flesh out your thoughts on what is needed to make your game functional. You ***may*** write pseudocode or use flow diagrams to explain the game mechanics.

The key is that another student should be able to look at your descriptions/explanations for code and begin programming the functionality. Their success will depend on how well you design your functionality.

The functionality of your game should include the following features to be implemented in vanilla JavaScript:

1) **Custom Animations**
   Introduce animations that enhance the gameplay experience. This might include animations for player movements, transitions between game states, or visual feedback based on player actions.
2) **Custom Interaction Mechanism**
   If your game involves moving and interacting elements, implement a custom collision detection mechanism. If your chosen game concept (e.g., Chess, Sudoku) doesn't inherently involve collisions, focus on another complex interaction mechanism. For a chess game, this might be the logic to determine legal moves for each piece or a mechanism to check for check and checkmate scenarios.
3) **Custom Algorithms**
   Implement algorithms that drive the gameplay. This could be path-finding for games that involve navigation, sorting algorithms for scoreboards, or any other game-specific algorithm

D2L/Github Submission

Please follow these guidelines for submitting this assignment. Details and instructions will be presented in your tutorials this week.

1) Create and use an online Git repository for your work. We recommend using GitHub. Please check with your TA if you wish to use something different.
2) Make sure to commit your work frequently. Only commits made before the deadline will be considered for grading.
3) Grant access to the TA by adding them as a collaborator on your repository.
4) Once done, submit a single *.txt file to D2L containing:
   i. Your full name.
   ii. Your username for the chosen Git platform (e.g., GitHub, GitLab, Bitbucket).
   iii. A link to your repository.
   iv. The name of the online repository, especially if it's not evident from the link.

Failure to follow these guidelines might impact your grade. Ensure you provide all the necessary details for easy access to your work. It is important that you submit a *.txt file to D2L so we have a place to enter a grade.

Grading [50 Marks]

Grading will include the categories listed below.

File Naming and Content [1]

Do all your files contain the necessary identification and attributions?

Game Overview: Title [1]

Clearly stated title

Target Platform [1]

Clearly stated platform (e.g., Desktop, Mobile, or both)

Game Genre [1]

Game Genre identified.

Game's Objective [3]

3 – Clear, concise, engaging
2 – Players have a basic understanding of what's expected, but there may be some ambiguity
1 – The objective is unclear or vague

Rules of the Game [3]

3 – Rules are clear, comprehensive, and well structured
2 – Most of the essential rules are present, but minor gaps or some ambiguity
1 – Rules are sparse or overly complicated

Game Mechanics [3]

3 – Mechanics are robust, well-thought-out
2 – Game mechanics are decently developed but may lack polish or originality
1 – Mechanics are rudimentary

UI Design: Layout and Structure [5]

5: Exceptional - Comprehensive structure, consistent design elements, exceptional alignment of elements
4: Good – Major sections are defined and organized, some inconsistencies in design or spacing
3: Adequate – Key sections are defined but layout might be missing elements or cluttered
2: Needs Improvement – Basic structure but design inconsistencies and alignment issues
1: Inadequate – Lacks clear structure
0: No layout provided

UI Design: Visual Elements [5]

5: Exceptional – Aesthetically consistent, all game elements present, copyright adherence (e.g., CC0)
4: Good – Most elements incorporated but minor omissions or don't fully align
3: Adequate – Some key elements present but missing components, inconsistencies, or ambiguity
2: Needs Improvement – Only includes a few visual game elements, many inconsistencies
1: Inadquate – Barely incorporates any essential elements, significant inconsistencies
0: No design of visual elements to speak of or unrelated, or blatant copyright violation

Description of Functionalities [5]

5: Exceptional – Comprehensive and covers all aspects of the game's functionalities
4: Good – Covers most of the game's critical functionalities with clarity
3: Adequate – A basic outline of the game's functionalities is present.
2: Needs Improvement – Vague and missing essential functionalities
1: Inadequate – Sparse and lacks clarity on most of the game's functionalities.
0: No meaningful descriptions or no indication of DOM manipulation

Prototypes/Headers/Stubs/Drivers/Pseudocode/FlowDiagrams [10]

10: Exceptional - All necessary and clear organization of all components and excellent planning
8: Good – Most essential components are present, minor areas of oversight or confusion
6: Adequate – Several components are missing or lack clarity
4: Needs Improvement – Components show rudimentary understanding of functionality
2: Inadequate – Basic or minimal components provided, lacking depth and understanding
0: No relevant components

Integration of Requirements [5]

5: Exceptional – Custom animations, collision detection, and algorithms clearly and logically planned
4: Good – A good number of requirements accounted for and clearly planned
3: Adequate – Several requirements are met but some logical issues
2: Needs Improvement – Rudimentary planning of requirements
1: Inadequate – Only a very basic planned implementation of requirements
0: No relevant components planned

Coding Standards [5]

5: Exceptional - Code is consistently well-organized and follows best practices.
4: Good - Code is organized and often follows best practices.
3: Adequate - Code structure is somewhat organized, but deviations from best practices are evident.
2: Needs Improvement - Code lacks clear organization; adherence to best practices is inconsistent.
1: Inadequate - The code is disorganized, making it hard to follow and understand.
0: Blank HTML and/or CSS

Validator [2]

Evaluates if your code passes standard HTML validators using the rubric:

2: Exceptional - Code passes the validator with zero errors.

1: Adequate - The code has several errors when running through the validator.
0: Blank HTML and/or CSS.

Penalties

Potential Copyright Issues for Assets [-15]

Use of copyright assets is strictly prohibited. If you are using assets you did not design yourself, you must include citations of where your assets were collected and that they meet the CC0. Any suspicion of copyright infringement will result in a -15% of your grade on this assignment and a requirement to re-do these assets for the next assignment.

Non-HTML/CSS Code [-10]

The use of non-HTML or CSS code violates the intended purpose of this assignment. Non-HTML or CSS code includes but is not limited to the following: JavaScript or JavaScript-based frameworks/technologies, Tailwind CSS, etc. Using any technology other than HTML and CSS will result in removing 15 marks from your overall score.

Use of !important [-2]

The use of `!important` is generally discouraged in the industry and demonstrates a lack of knowledge in structuring quality CSS. Using `!important` in this assignment will result in the deduction of 2 marks.

Bonuses / Advanced Version Requirements

Bonus Responsive Design [+5]

Designing your game for both desktop and mobile.

Originality [+5]

A unique game of your design.

Complexity [+5]

Given the described design, have you chosen to take on challenges beyond the basic?

Advanced Version of the Assignment

For those who wish to explore and incorporate additional technologies beyond the base requirements in the HTML and CSS sections, an advanced version of this assignment is available. This option is designed to cater to more experienced students or those seeking a greater challenge.

Key Differences in the Advanced Version:

Use of Additional Technologies:

While the standard assignment restricts students to core technologies, the advanced version permits the use of additional frameworks, libraries, or tools in the HTML and CSS sections. Whether it's a CSS pre-processor, a specific framework, or any other tool, you are free to use it to enhance your game's design and responsiveness provided it does not build the interface for you (e.g., you are not allowed to use a WYSIWYG editor).

Stricter Evaluation

Given the allowance for additional tools, the evaluation criteria for the advanced version will be more rigorous. Expectations for design precision, responsiveness, and creativity will be higher.

Shift from Bonus to Requirement

Elements that are listed under the "bonus" section in the standard assignment will become mandatory requirements in the advanced version. This shift ensures that the complexity and depth of the assignment match the additional tools and skills employed.

Grading

While the standard assignment offers some leniency in grading, the advanced version will have a more stringent grading system, with greater emphasis on detail, complexity, and polish.

Note

This advanced version is analogous to a graduate student opting to take an undergraduate course. While the foundational learning objectives remain the same, the depth, complexity, and expectations are adjusted to match the advanced skill set of the participant.

Before choosing this option, ensure you are comfortable with the heightened expectations and are prepared to meet the additional requirements.