Assignment 3: JavaScript Part Two

Course: Web-Based Systems
Number: SENG 513
Semester: Fall 2023
Due Date: Nov. 6
Instructor: Steve Sutcliffe <steve dot sutcliffe at ucalgary dot ca>
Version: 1.0.0

Objective:

To develop a unique and interactive browser-based game using HTML, CSS, and JavaScript. This assignment emphasizes the coding necessary to implement your design from Assignment 2. You must demonstrate a deep understanding of JavaScript functionalities, interactivity, and dynamic rendering. This is an opportunity to develop the game's mechanics. The objective is to implement the functionality from your original design and reflect on the process, comparing the initial design to the final product.

Overview:

This assignment will make up 15% of your grade. In Assignment 3, you will bring your game design from Assignment 2 to life by implementing the functionality behind your UI. This assignment aims to transform the conceptual work done in Assignment 2 into a working game, utilizing programming skills and tools like ChatGPT or GitHub Copilot to assist in the coding process.

Scope and Complexity:

Managing your scope is crucial if your game design is particularly complex or ambitious. Break down the implementation into manageable parts and focus on the core functionality first. You can add additional features once the basic game is working. Clearly define what will be included in this implementation phase and what might be saved for future iterations. This will help you stay focused and ensure a working game version.

Relatively simple games (e.g., hangman, tic-tac-toe) should aim to complete the entire game, however, you should also focus on the core functionality first and add additional features once the game is working.

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

Depth over Breadth

Having a few well-implemented, polished features is more important 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" or “game over” 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 (reiterate from Assignment 2)

In the README.md file, include the following:

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.

Implementation

Based on your descriptions/prototypes/headers/stubs/drivers, build your implementation for the game. The goal here should be to implement each design element from assignment 2. You ***may*** use AI to help you write the code (such as using ChatGPT or Copilot), but you must indicate which sections were written by AI. You’re your original descriptions/prototypes/headers/etc. for comparison. Additionally, you should be the one to assemble the final structure of the program without the use of AI and be able to describe what the code is doing.

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

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.

Custom Interaction Mechanism

Implement a custom collision detection mechanism if your game involves moving and interacting elements. 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.

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.

Note: Make the marker's life easier by identifying these sections in your code.

Reflection

In a file called reflect.md, Reflect on the implementation process using point form. Focus on what you learned, your challenges, how you managed the game's complexity, and how the final product compares to your initial design. If you used an AI to help write code, reflect on this process and highlight key insights you learned. If you chose not to use AI, explain why you did not need to or want to use it.

Code Explanation

In a file called, explanation.md, explain three complex parts of your code. If ChatGPT or CoPilot wrote these sections, highlight these sections and provide an explanation demonstrating your understanding. The document should be clear, concise, and technical so anyone reading it can understand its functionality and purpose.

Snapshot of your game (Optional)

Provide a short video clip or image of your game that we can show to future students. Write your name and the game title on the video clip or image. By submitting this clip or image, you are giving consent for us to show this to students in future classes. Some of your games look amazing, so it would be awesome if we could use them to inspire future students and even show them off to your classmates in D2L. We will try to post these submissions in D2L this semester so others can look at some of the fantastic games you created.

D2L/Github Submission

Please follow these guidelines for submitting this assignment. Please ask your TA if you need further details.

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) 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. You must submit a *.txt file to D2L so we have a place to enter a grade.

Grading [60 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 [1]

This can be a reiteration from assignment 2.

Implementation of Functionality [10]

10: Excellent - All game elements are implemented and functioning perfectly. The game is stable, bug-free, and runs smoothly.
8: Good - Most game elements are implemented, with minor bugs or issues that do not significantly impact gameplay.
6: Satisfactory - Some game elements are implemented, but some noticeable bugs or issues affect gameplay.
4: Poor - Few game elements are implemented, and some significant bugs or issues severely impact gameplay.
2: Very Poor - Very few or no game elements are implemented, and the game is essentially non-functional.

Reflection [10]

10: Excellent - Provides a comprehensive and detailed description of the process, demonstrates deep insight into challenges and decision-making
8: Good - Offers a thorough description of most parts of the process, shows good insight and self-awareness, though some areas might lack depth or detail.
6: Satisfactory - Gives adequate description of the process, with some insight and self-awareness, but may miss depth in analyzing challenges or connecting to broader learning objectives
4: Needs Improvement - Provides basic and unclear description of the process, lacks depth in insight and self-awareness
2: Poor - Offers limited and/or unclear descriptions, shows minimal insight or self-awareness

Code Explanation [10]

10: Exceptional – Demonstrates a deep and thorough understanding of complex parts of code, presented clearly and accurately
8: Good – Has a well-thought-out structure and demonstrates a strong understanding of the code but may lack depth in some parts
6: Adequate – Some parts lack clarity; demonstrates a basic understanding of code but struggles to articulate explanations for complex sections
4: Needs Improvement – Vague and shows a limited understanding of the code; few complex parts explained

2: Inadequate – Document is unclear, disorganized, demonstrates little to no understanding of the code
0: No meaningful explanations presented.

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

10: Exceptional - The implemented code closely follows the pseudocode, demonstrating a high level of accuracy.
8: Good – The implemented code generally follows the pseudocode, with minor deviations that are well justified.
6: Adequate – The implemented code follows the pseudocode, but several deviations exist.
4: Needs Improvement – The implemented code loosely follows the pseudocode, with many deviations and no justification.
2: Inadequate – The implemented code does not follow the pseudocode, with no apparent connection between them.
0: No implementation of design elements

Integration of Requirements [10]

10: Exceptional – Custom animations, interaction mechanisms and custom algorithms clearly and logically implemented
8: Good – A good number of requirements accounted for and logically implemented
6: Adequate – Several requirements are met, but some implementation issues
4: Needs Improvement – Rudimentary implementation of requirements
2: Inadequate – Only a very basic implementation of requirements
0: No relevant components implemented

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, CSS and/or JavaScript files

Validator [3]

Evaluates if your code passes standard HTML, CSS and JavaScript validators/linters using the rubric:

3: Exceptional - Code passes validators with zero errors, linters show no issues with coding standards.
2: Good – Code generally passes validators and linters with only few minor errors/issues
1: Adequate - When running through the validator, the code has several errors.
0: Blank HTML, CSS and/or JavaScript.

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/JavaScript Code [-10]

The use of non-HTML/CSS/vanilla JavaScript code violates the intended purpose of this assignment. Non-HTML, CSS or vanilla JavaScript code includes but is not limited to the following: JavaScript-based frameworks/technologies, Tailwind CSS, etc. Using any technology other than HTML, CSS and JavaScript will result in removing 15 marks from your overall score. You must receive permission beforehand if you intend to complete the advanced version of this assignment to have this restriction removed.

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 basics?

Advanced Version of the Assignment

An advanced version of this assignment is available for those who wish to explore and incorporate additional technologies beyond the base requirements in the HTML and CSS sections. This option caters to more experienced students or those seeking a more significant 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 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, emphasizing 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.