Hangman

Sabbella Srinivasa Reddy
sabbels@g.clemson.edu
Clemson University
Clemson, SC

Madhav Ashok
mashok@g.clemson.edu
Clemson University
Clemson, SC

Vinod Ramavath
vramava@g.clemson.edu
Clemson University
Clemson, SC

Abstract

Hangman is a classic word puzzle challenge game, an old classic game where one player thinks of a word while the other tries to guess it by proposing letters. Each wrong guess causes a hangman figure to be gradually drawn, where when the complete hangman is drawn, the player loses the game, so the player has to guess the word before the clock runs out, i.e., before the figure is completed, to win the game.

Objective

The simplicity of Hangman is what makes the game so unique. It simply needs two participants and a love of wordplay. When one player has an idea for a word, the other player tries to figure it out by offering letters. A hangman figure is gradually drawn with each wrong guess, raising the difficulty level. The word-picker wants to test their opponent with a tricky, hidden word, while the guesser seeks to figure out the word before the hangman is completely drawn.

We have created the classic hangman game as a computer game that can be played online anywhere on a website using JavaScript and HTML. In this game, participants are challenged to put their word-guessing abilities to the test by attempting to piece together a secret word. To offer an engaging game experience, the code develops a user-friendly interface with buttons, input fields, and notifications. Players can start a brand-new game and get tips to help them with their guesses. The game's objective is for participants to accurately guess the word while being conscientious about their finite number of wrong guesses. This serves as the basis for a pleasant and addicting word-guessing game because of its appealing features and functionality.

Rules

- **Game Start:** The player's click on the "Play" button initiates the game. A word is chosen randomly from a predetermined list to begin the game.
- **Word Guessing:** The player's primary goal is to uncover the word buried letter by letter. They hit Enter after entering their educated estimates into the input slot.
- **Limited Chances:** The player only has so many chances or lives, which are less with each wrong estimate. They lose the game if they run out of opportunities.
- **Correct Guesses:** When a player successfully guesses a letter that is a component of a concealed word, that letter is revealed in the word's proper location.
- **Clues:** Players may ask for clues and receive information about the concealed word by clicking on a voice bubble symbol. The number of times hints have been utilized is recorded by the game.

Technology Stack Used

HTML: The game's web page is structured using HTML, which defines the game board, input fields, and button placement.
CSS: CSS is used to design the look of the game's elements, such as the fonts, colors, and layout, making the game more aesthetically pleasing.
JavaScript: Word selection, player input verification, game status monitoring, and victory or loss management are all handled by JavaScript, which is also in charge of the logic behind the game.
Google Fonts: For the fonts used in the game.
Animista: For CSS animations.

Setup

- **Environment Preparation:** Ensuring we had a working development environment with a code editor (e.g., Visual Studio Code), Node.js, and a web browser installed on your computer.
- **Coding:** Making sure that our code works.
- **Web Hosting:** Decide where to host our Hangman game.
- **Project Structure:** Organize our project's directory structure. Creating folders for HTML, CSS, JavaScript, images, and audio files.
- **HTML File:** Creating an HTML file for the game's user interface.
- **CSS Styling:** Creating a CSS file to style the game's elements, applying fonts, colors, and layout.
- **JavaScript Logic:** Creating a JavaScript file to implement the game's logic.

Deployment

- **Web Hosting:** Choosing a web hosting service and uploading our project files to the hosting server using FTP or other deployment methods.
- **Testing:** Testing our game thoroughly to ensure that it functions as expected.
- **Security Considerations:** Using HTTPS for secure connections and ensure your server has the necessary security configurations.
- **Optimization:** Optimizing our code, assets, and images for faster loading times.
- **Launch:** Once satisfied with our game's functionality and appearance, we launched it for public access.

Third-Party Credits:

- FontAwesome
- Lottie Player JS
- Speech Bubble Maker

Reflection

Challenges Faced:

- Agreeing on the game's features and design was one of the early difficulties we faced. We had various ideas and preferences for the user interface, gameplay elements, and theme. It took some brainstorming sessions and compromises to arrive at a design we all liked.
- The most challenging part was finding and correcting bugs and error situations. Therefore, it was difficult to test every scenario. As a result, several problems managed to sneak through the gaps and detract from the game's overall quality, which we could rectify later.
- Another difficulty was ensuring an enjoyable and seamless user experience. The game needed to be challenging and satisfying while being playable by various players, so we had to find the right balance. It was essential to get input and make changes.
- Choosing the best programming language and framework was challenging since we had yet to gain any prior gaming development expertise, including the technological components that required some study. Additionally, it took much work to integrate the front-end and back-end to retain the game state seamlessly.

What worked:

- Based on our individual talents and interests, we split the tasks. This made the development process more efficient, enhancing the efficiency of our workflow.
- Our accomplishment was primarily due to open and honest communication. We regularly connected to discuss our achievements, obstacles, and subsequent initiatives. It enabled us to fix problems and align with the project's objectives quickly.
- Early development of a simple game prototype was a wise choice. We were able to improve the game's fundamental mechanics and design since it offered us a concrete starting point. We were able to build on this basis as we went along.
- During the development process, we gathered input from friends. Their opinions offered insightful information and assisted us in determining where we might improve. It was an excellent technique to ensure the game's intended audience would find it engaging.

What Didn’t Work:

- Initially, we had grand ambitions for the game, including online multiplayer capabilities and a substantial word database. However, we concluded that, given our time constraints, these features could have been more practical. It was crucial to have a realistic perspective on our capabilities.
- Initially, we concentrated on making a game that operated flawlessly on contemporary browsers and devices. However, we should have considered the significance of cross-browser compatibility, which caused some users to have unforeseen problems. We had to put in additional time to make the game more inclusive.

Lesson Learned:

- Especially when working with a small team, it's important to begin with a limited, controllable scope. Your game can grow and include additional features as you acquire experience.
- Give user input and experience a priority. You may adapt your game to your audience's tastes by testing regularly and soliciting feedback from prospective players.
- Within the team, effective communication is crucial. Uncertainties may be avoided, and the project can be kept on schedule with the help of regular meetings, progress reports, and problem tracking.
- It's crucial to gain a basic understanding of the technological requirements and tools required for developing games. Although you don't have to be an expert, a working knowledge of the selected technologies is helpful.