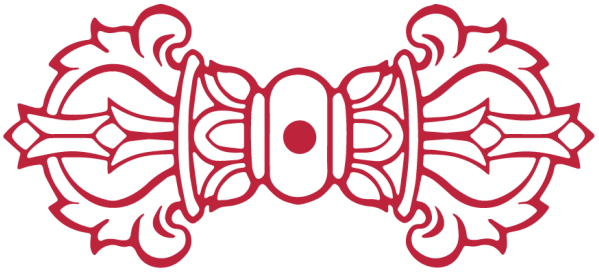

BAJRA

TECHNOLOGIES

A Documentation on

Bug on Wire : Shuriken Circuit

Version 0.1

24 June 2024

BAJRA TECHNOLOGIES

Document Version History

S. No.	Date	Version	Remarks/Summary of Updates	Prepared By
1	24 June 2024	0.1	Created initial release	Suyan Shrestha

Table of Contents

Introduction

What is it about?

Bug on Wire: Shuriken Circuit is an exhilarating game where you step into the shoes of a nimble ninja bug. Choose from three unique ninja characters and test your reflexes as you dodge a barrage of incoming shuriken by swiftly switching between wires. With its fast-paced gameplay and captivating ninja theme, "Bug on Wire: Shuriken Circuit" promises endless excitement and challenges for players of all ages.

Features

a. Player controlled character that can move up and down while switching different lines.
b. Player needs to help the character dodge all obstacles (ninja stars) that come along the line.
c. Multiple playable characters allowing players to choose their favorite.
d. Multiple background music, creating an engaging auditory experience that complements the gameplay.
e. Increasing difficulty for avoiding obstacles as the game progresses.

Requirements Analysis

Following are the basic requirements that was first proposed for the game :

a. Players can earn points through collecting coins or based on the time elapsed, adding a competitive element to the gameplay.
b. Birds are the main obstacle of the game for the bug. If a bird touches the bug, it results in a game over.
c. Engaging sound effects enhance the gaming experience, providing audio feedback for various in-game actions and events.
d. A help screen is available to guide players on how to play the game, providing instructions and tips for a better gaming experience.
e. Includes various assets such as bug and bird images in the form of sprite images, which are integral to the game's design, animation and functionality.
f. A detailed documentation should be made, covering all aspects of the game's development, features, and usage.
g. The project will be managed and submitted via GitLab, ensuring proper version control and collaborative development.

Technical Requirements

a. Use appropriate technologies like HTML, CSS and JS.
b. Implement fluid and seamless animations for character movements and interactions to enhance the visual experience.
c. Develop and render multiple wires that the bug can traverse, forming the primary paths for gameplay.
d. Enable the bug to jump from one wire to another, adding a strategic element to navigating the game environment.
e. Incorporate precise collision detection to ensure accurate interactions between the bug and obstacles.
f. Implement a robust score-keeping mechanism that tracks points based on either the time elapsed or coins collected.
g. Create a feature to display the highest score achieved, encouraging players to beat their previous records.
h. Gradually increase the bug's speed or number of bugs per second as the game progresses, introducing an escalating challenge for players.

Commits

a. Add bug module and try parallax animations

i. In this initial commit, the bug module was added and parallax animations for background were experimented for the project to enhance the visual experience.
ii. Created the main character for the game, referred to as the "bug."
iii. Implemented the basic properties and behaviors of the bug, including movement and interaction with the wire.

b. Add obstacles and loading screen

i. In this commit, the focus shifted to core game functionality. Parallax animations were set aside, and obstacles along with a loading screen were introduced.
ii. Obstacles increase the game's difficulty, and the loading screen improves the user experience during the game initialization.
iii. Implemented logic for random obstacle generation to create dynamic gameplay.
iv. As the game progresses, more and more obstacles would get generated in a shorter time.

c. Add home screen and collision detection

i. In this commit, the loading screen was improvised to a fully built home screen, and collision detection was introduced.
ii. The home screen provides a user-friendly starting point, while collision detection enhances gameplay by introducing interactions between the bug and obstacles.

d. Complete all functionalities to initial wireframe

i. This commit completes the development of all functionalities outlined in the initial wireframe for the "bug on wire" game.
ii. With this update, the game achieves a fully functional state, incorporating all planned features and interactions.
iii. Until now, a rectangular box for the bug and circles for obstacles had been used.
iv. Also added scoring and highscore mechanisms using local storage.

e. Add sprite images for bugs and obstacles

i. Introduced a new sprite image for the bug character.
ii. Updated the bug module to use the sprite for rendering the bug's appearance.
iii. Added static image for obstacle and applied rotation to it, ensuring that it gets an animated look.

f. Add three new characters

i. This commit introduces three iconic characters from various anime series into the "bug on wire" game, adding unique personalities and visual styles.
ii. Introducing Ichigo Kurosaki from "Bleach".
iii. Introducing Suyan Zetenabe from "Bajra".
iv. Introducing Madara Uchiha from "Naruto".
v. Updated character selection mechanics in Home screen to incorporate them as playable character options.

g. Final Commits

i. Refactored asset paths in the game to use relative paths.
ii. Removed unnecessary code and comments.
iii. Improved codebase readability and maintainability.
iv. Integrated feat/ci_cd branch into the main development branch.
v. Prepared the project for continuous integration and deployment (CI/CD).

Code Snippets

a. Keypress based Update functionality for Bug class

```
update(key) {
  let moveSound = new Audio("music/move.mp3");
  switch (key) {
    case "ArrowUp":
      if (this.y > Math.round(this.canvas.height / 6)) {
        this.y -= Math.round(this.canvas.height / 3);
        moveSound.play();
      }
      break;
    case "ArrowDown":
      if (this.y < Math.round((this.canvas.height / 6) * 5 - 100)) {
        this.y += Math.round(this.canvas.height / 3);
        moveSound.play();
      }
      break;
  }
}
```

i. It first creates a new Audio object, moveSound, with the source file located at "music/move.mp3". This sound is likely played when the object moves.
ii. The switch statement checks the value of the key parameter.
iii. If the key is "ArrowUp", it checks if the current y-coordinate of the object (this.y) is greater than one-sixth of the canvas's height. If it is, it decreases this.y by one-third of the canvas's height, effectively moving the object up, and plays the move sound.
iv. If the key is "ArrowDown", it checks if the current y-coordinate of the object is less than five-sixths of the canvas's height minus 100. If it is, it increases this.y by one-third of the canvas's height, effectively moving the object down, and plays the move sound.

b. How obstacles are generated?

```
export class Obstacle {
  constructor(ctx, canvas, dv, CANVAS_HEIGHT, img) {
    this.ctx = ctx;
    this.canvas = canvas;
    this.y = [CANVAS_HEIGHT / 6, CANVAS_HEIGHT / 2, (CANVAS_HEIGHT / 6) * 5][
      Math.floor(Math.random() * 3)
    ] - 100;
    this.x = this.canvas.width;
    this.speed = Math.floor(2 + Math.random() * dv);
    this.width = 60;
    this.img = img;
  }
```

i. The third line sets the y-coordinate for the obstacle. It randomly selects one of three possible values: CANVAS_HEIGHT / 6, CANVAS_HEIGHT / 2, or (CANVAS_HEIGHT / 6) * 5, and then subtracts 100 from the selected value.
ii. Here, -100 is done to align stars properly and make them visually appealing.
iii. There is also code to set the speed of the obstacle. It will be a random number between 2 and 2 + dv.

c. Collision detection

```
// COLLISION DETECTION
if (
  bug.x <= obstacle.x &&
  bug.x + bug.size + collisionMargin >= obstacle.x &&
  bug.y === obstacle.y
) {
  console.log("Collision detected");
  gameOver = true;
  gameOverSound.play();
}
```

i. First of all, it checks if the bug is on the left side of or at the same position as the obstacle.
ii. Then, it checks if the right edge of the bug is greater than or equal to the x-coordinate of the obstacle.
iii. Then, it also checks if both the bug and obstacle are on same height.

d. Dialog box in Home screen

```
// to show details in loading screen
const showDetails = (dialog, show) => {
  if (show) {
    dialog.classList.remove("hide");
    dialog.showModal();
    foodSound.play();
  } else {
    dialog.classList.add("hide");
    dialog.close();
    foodSound.play();
  }
};
```

i. On the basis of boolean value of show, it decides if hide class (with property display none) is being added or removed.

ii. Then, if show is true showModel inbuilt function of dialog is run, otherwise it is closed.

e. Restart game with spacebar

```
window.addEventListener("keydown", function (event) {
  bug.update(event.key);

  // Check if any dialog is open
  let dialogs = document.querySelectorAll("dialog");
  for (let dialog of dialogs) {
    if (dialog.open) {
      return;
    }
  }

  // Instruction screen
  if (event.code === "Space") {
    document.getElementById("instructionScreen").style.display = "none";
    document.querySelector(".canvas-div").style.display = "block";
    instructionsDialog.classList.add("hide");
    musicSound.currentTime = 0;
    musicSound.play();
    restartGame();
  }
});
```

i. If any dialog box is open in the home screen, then spacebar key press won't work, and the function will be returned beforehand.

ii. On pressing spacebar, HomeScreen will not be displayed using display none, and actual game screen will be displayed.

iii. Then the currentTime property will just reset the music, and while playing music, it will start from the beginning of the music file.

f. Generating obstacles at lesser timespan as the game progresses more

```
obstacleInterval = setInterval(() => {
  gameProgress += 0.01;
  let obstacle = new Obstacle(
    ctx1,
    canvas1,
    2 + gameProgress,
    CANVAS_HEIGHT,
    obstacleImg
  );
  obstacles.push(obstacle);
}, 1000 - gameProgress * 100);
```

i. As the game progresses, new obstacles will emerge.
ii. Then, the value of gameplay will increase by 0.01.
iii. So, setInterval will run in shorter and shorter time.

g. Scoring and High Score mechanisms

```
if (obstacle.x > 0) return obstacle;
score.textContent = (parseInt(score.textContent) + 1)
  .toString()
  .padStart(3, "0");
```

i. obstacle.x > 0 is used to filter out obstacles that have moved past a certain point on the x-axis (left edge of canvas)
ii. Remaining code ensures that the score is always displayed with at least three digits, with leading zeros if necessary.

```
if (localStorage.getItem("highScore") < parseInt(score.textContent)) {
  localStorage.setItem("highScore", score.textContent);
}
highScore.textContent = localStorage.getItem("highScore");
myHighScore.textContent = localStorage.getItem("highScore");
```

iii. Checks if the current score is higher than the stored high score in the browser's local storage.
iv. If the current score is higher, it updates the high score in the local storage.
v. It then retrieves the high score from local storage and displays it in two different elements on the webpage: highScore (for displaying highScore alongside canvas) and myHighScore (for displaying highScore in scoreboard).

Future Enhancements

a. I experimented with implementing parallax backgrounds and dynamic background changes based on game progression. However, these features will need to be deferred to future updates and enhancements.
b. I also attempted to create custom background music, and I achieved some designs tailored to different characters. However, as these designs are not yet perfect, they will be included in future updates and improvements.
c. I will also be implementing a new gameplay mechanic where the ninja character can collect scrolls that appear sporadically throughout the game. Scrolls will appear in smaller numbers to maintain game balance, and will be usable to unlock new characters.

Conclusion

Here's some of the things I learned while working on this project :

a. Gained proficiency in implementing core game mechanics such as character movement, collision detection, and obstacle generation.
b. Learned about sprite animations and how they work.
c. Encountered challenges and learned effective problem-solving strategies within the context of game development.
d. Embraced iterative development cycles to continuously improve features and address feedback.
e. Implemented CI/CD practices to automate testing, building, and deployment processes.