(Autonomous Institution Affiliated to VTU, Belagavi)

Department Of Computer Science and Engineering

Software Development Club

Internship Report

Submitted by

Sumith B H	USN: 1MJ21CS217
Shivam Sharma	USN: 1MJ21CS199
Ganya J	USN: 1MJ21CS217
Chethan G	USN: 1MJ21CS048

Under the guidance of

Ms. Thejaswini M

Department of Computer Science and Engineering

MVJ College of Engineering

In Partial fulfillment for the award of degree

of

Bachelor of Engineering

In

Computer Science and Engineering

2021-22

COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

MVJ COLLEGE OF ENGINEERING, BENGALURU- 560067

(Autonomous Institution Affiliated to VTU, Belagavi)

Department Of Computer Science and Engineering

CERTIFICATE

Certified that the internship titled Software Development Club was carried out by Sumith B H (1MJ21CS217), Shivam Sharma (1MJ21CS199), Ganya J (1MJ21CS0) and Chethan G (1MJ21CS048) in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections / suggestions indicated during the internal assessment have been incorporated in the internship report deposited in the department library. The internship report has been approved as it satisfies the academic requirements in respect of internship work prescribed by the institution for the said degree.

Signature of Guide	Signature of Head of the Department	Signature of Principal

MVJ COLLEGE OF ENGINEERING, BENGALURU- 560067

(Autonomous Institution Affiliated to VTU, Belagavi)

Department Of Computer Science and Engineering

DECLARATION

We, **Sumith B H (1MJ21CS217), Shivam Sharma (1MJ21CS199), Ganya J (1MJ21CS0) and Chethan G (1MJ21CS048)** students of Second Semester B.E., Department of **Computer Science and Engineering,** MVJ College of Engineering, Bengaluru - 560067, hereby declare that the Internship Titled Software Development Club has been carried out by us and submitted in partial fulfillment for the award of the degree of Bachelor of Engineering in **Computer Science and Engineering** during the year 2021-2022.

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other University.

Place: Bengaluru

Date:

Name	**Signature**
1. Sumith B H (1MJ21CS217)	
2. Shivam Sharma (1MJ21CS199)	
3. Ganya J (1MJ21CS064)	
4. Chethan G (1MJ21CS048)	

ACKNOWLEDGEMENT

We thank the college, MVJ College of Engineering for conducting the internship.

We express sincere gratitude to our beloved Principal, Dr. Mahabaleshwarappa and Vice Principal, for their appreciation towards the internship.

Our sincere thanks to M.B Sudhan, Associate Professor and Head of **Computer Science and Engineering**, MVJCE for his support and encouragement.

We would also like to extend our gratitude to Lourdu Anthony Raj, Professor for his guidance in carrying out the internship.

We are indebted to our guide, (Ms. Thejaswini M, Assistant Professor) Department of Computer Science and Engineering for her whole hearted support, suggestion and invaluable advice throughout the Internship and also for the help in the preparation of this report.

We thank all the technical and non-technical staff of **Computer Science and Engineering** department, MVJCE for their help.

ABSTRACT

"Spooky Run" is a web based 2d arcade style game that allows players to play as a lost dog finding its way back home through a spooky forest. The game is built using vanilla JavaScript and utilizes HTML5 and the `canvas` element to render graphics and handle user input. Players can attack various monsters using different techniques and score points. The game also includes magic element which allows the player to perform rolling fire attack. Optimization techniques were used to ensure smooth performance on a variety of devices, including desktop computers and mobile devices. Overall, "Spooky Run" offers a rich and stable gameplay experience for fans of arcade style games.

List of Tables

List of Figures

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Gaming in the Field of Software Engineering

In the fast-growing field of software engineering and development and even more rapidly growing sector of game development the future is hard to predict. We are working with this game as our Software Development Club Internship project and as part of our degree we choose this type of work for doing better with development cycle, development period, graphics, scripting, adopting, new technology, animation.

In general software project is a project focused on the creation of software. Consequently, success can be measured by taking a long at the resulting software.

In a game project, the product is a game. But here comes the point: A game is much more than just its software. It must provide content to become enjoyable. Just like a web server: without content the server is useless, and the quality cannot be measured. This has an important effect on the game project. The software part of the project is not the only one, and it must be considered in connection to all other parts: The environment of the game, the story, characters, game plays, the artwork, and so on.

1.2 About the Project

There are many different ways to create a game for the web. Here are a few options:

Use a game engine: Game engines like Unity and Unreal Engine can be used to create games that can be played in a web browser. These engines provide a lot of functionality and make it easier to create complex games.

Use a JavaScript library: There are many JavaScript libraries that can be used to create games for the web. Some popular options include Phaser, Pixi.js, and Three.js. These libraries provide APIs for creating game objects, handling user input, and rendering graphics.

Use pure JavaScript: You can also create games using just JavaScript, HTML, and CSS. This can be a good option for simple games, but can become more complex as the game becomes more feature-rich.

Regardless of which approach you choose; it is important to consider the performance of your game and make sure it runs smoothly on a variety of devices. You may also want to consider hosting your game on a platform like Steam or itch.io to make it easy for players to find and play your game.

JavaScript is a popular language for creating games that can be played in a web browser. Here are a few steps you can follow to create a game using JavaScript:

- Set up your development environment: You will need a text editor to write your code and a web browser to test your game. You may also want to use a tool like Webpack or Parcel to manage your assets and help with the build process.
- Design your game: Decide on the theme, mechanics, and features of your game. You may want to sketch out a plan or create a prototype to help visualize your game.
- Create the game world: Use HTML, CSS, and JavaScript to create the game world, including the background, characters, and objects. You can use the HTML canvas element to draw graphics and handle user input.
- Implement the game mechanics: Use JavaScript to implement the logic of your game, including player movement, collision detection, and scoring.
- Test and debug your game: Playtest your game and fix any bugs or issues that you encounter.
- Publish your game: Host your game on a web server or platform like Steam or itch.io so that other people can play it.

There are many resources available online that can help you learn more about creating games with JavaScript. You can also find libraries and frameworks that can help you get started with game development more quickly.

A 2D JavaScript game is a type of video game that is created using the JavaScript programming language and is designed to be played within a web browser. These games can range from simple, browser-based puzzles and arcade games to more complex, feature-rich experiences that require more advanced programming skills to develop.

‘Spooky Run’ is a single player arcade style game emphasizing speed, technique and accuracy. The main character of ‘Spooky Run’ is a dog that is cursed and lost in a spooky forest and must escape the forest to reach its home. The forest is filled with ghosts and monsters so it hard for it to escape the forest. The curse on the dog provides it with 5 lives and a rolling fire attack to kill the monsters. The main mission of the game is to score a certain number of points within the time limit to exit the forest safely.

1.3 Objective of the Internship

The objective of a software development internship is to provide a hands-on learning experience for students or professionals who are interested in software development. Internships can provide opportunities to work on real-world projects, gain practical skills and knowledge, and build a professional network.

Some specific objectives of a software development internship might include:

1. Developing skills in programming languages and technologies: Interns can learn about different programming languages and technologies and gain practical experience working with them.
2. Learning about the software development process: Interns can learn about the software development life cycle, including planning, coding, testing, and deployment.
3. Working on real-world projects: Interns can work on real-world projects under the guidance of experienced professionals, gaining practical experience and developing a portfolio of work.
4. Building a professional network: Interns can network with industry professionals and build relationships that can be beneficial for their future careers.
5. Learning about the culture and work environment at a company: Interns can learn about the culture and work environment at a company and get a sense of what it would be like to work there full-time.

Overall, the objective of a software development internship is to provide a valuable learning and professional development opportunity that can help interns prepare for a career in software development.

1.4 Methodology

There are many different approaches that can be taken when creating a web-based game. Here is a general methodology that you might follow when creating a game for the web:

Planning and design: The first step in creating a web-based game is to plan and design the game. This might involve deciding on the theme, mechanics, and features of the game, as well as creating a prototype or sketching out a plan.

Building the game world: Next, you will need to build the game world using HTML, CSS, and JavaScript. This might involve creating the background, characters, and objects, as well as handling user input and rendering graphics using the HTML canvas element.

Implementing the game mechanics: Use JavaScript to implement the logic of the game, including player movement, collision detection, and scoring. This might involve writing code to handle user input, update the game state, and render the game to the screen.

Testing and debugging: Playtest the game and fix any bugs or issues that you encounter. This might involve running the game in a web browser and testing different scenarios to make sure everything is working as intended.

Publishing the game: Host the game on a web server or platform like Steam or itch.io so that others can play it. This might involve building and deploying the game using a tool like Webpack or Parcel, and setting up any necessary hosting or distribution platforms.

CHAPTER-2

SOFTWARE AND OUTCOMES

2.1 General Description

The Software and Outcomes describe the functional outcome of this project. It also provides a virtual image for the combination of both structured and unstructured information of our project “Spooky Run”.

An infinite runner game is a type of video game in which the player character runs automatically and the player's task is to navigate through an endlessly scrolling environment, avoiding obstacles and collecting power-ups or other items along the way. The outcome of an infinite runner game typically depends on the player's skill and ability to react to the on-screen action.

In most infinite runner games, the player's ultimate goal is to achieve the highest score possible by running as far as possible without dying. When the player character hits an obstacle or falls off the screen, the game is over and the player's score is recorded.

Some infinite runner games may also have additional objectives or challenges that the player can complete for bonus points or rewards. For example, a game might task the player with collecting a certain number of coins or reaching a certain distance within a certain time limit.

Overall, the outcome of an infinite runner game will depend on the player's ability to react quickly and make strategic decisions in the face of constantly changing gameplay challenges.

2.2 Product Perspective of the Game

Software product development is a paradigm shift from routine application maintenance and support in the software industry. Development of a game product from scratch is a significant challenge for any organization. It requires considerable investments in terms of effort and cost and confirms client involvement, knowledge about client market (example: Google Play).

From a product perspective, a web-based game is a digital product that is designed to be played within a web browser. This type of game typically relies on a combination of programming languages, libraries, and frameworks to create the gameplay experience, and may also require additional software or plugins to run within a browser.

One of the key benefits of web-based games is that they are generally easy to access and play, requiring only a computer or mobile device with a web browser and an internet connection. This makes them highly accessible to a wide range of players, regardless of their geographic location or the type of device they are using.

Web-based games are also often designed to be played in short bursts, making them well-suited for casual or on-the-go play. Many web-based games are free to play, with developers relying on advertising or in-game microtransactions to generate revenue.

Overall, web-based games are a popular and convenient way for players to enjoy a wide range of game experiences, from simple browser-based puzzles and arcade games to more complex, feature-rich experiences.

2.3.1 Normal Outcomes

Normal outcomes of our project are:

1. User friendly efficient and lucrative system.
2. Minimum maintenance cost.
3. Availability of expected requirements within the PC/Mobile configuration.
4. Easy to operate.
5. Professional build

2.3.2 Expected Outcomes

Expected outcomes of our project are:

1. Develop system within limited cost.
2. Maximum high definition.
3. Minimum hardware requirements.
4. Design whole system with efficient manner.

2.4 User Experience of Our Game

“Spooky Run” is an arcade style game. Since it is a browser game it can be played on PC, android phone, IOS and other platforms having a web browser and an active internet connection.

After running the game, the UX view of the game will appear on the screen. The term UX means User Experience which is used to explain all aspects of a person’s experience with a system.

The UX view presents the name of the game below which a set of instructions to play the game are provided. Then the gamer can press the ‘Enter’ key to start playing the game. The player can also toggle full screen mode using the ‘Fullscreen mode’ button at the top of the screen and can exit the full screen mode by using ‘esc’ key. If the player were to win or lose the game, he can restart the game by pressing the ‘Shift’ key on the keyboard.

2.5 Software Requirements

2.5.1 User Interfaces

The user interface (UI) of a web-based game refers to the various elements and features that allow players to interact with and navigate the game. A well-designed UI can help make the game more enjoyable and intuitive to play, while a poorly designed UI can be frustrating and confusing for players.

Ultimately, the goal of a web-based game's UI is to provide a clear and intuitive interface that allows players to easily navigate and interact with the game, without getting in the way of the gameplay experience.

Every game must contain a starting screen so it can be friendly enough and gamers can be welcomed to the game. The starting screen of our game consists of instructions and title in a simple arcade form.

2.5.2 Hardware Interfaces

Hardware interfaces for web-based games allow players to use physical devices, such as game controllers or other input devices, to interact with and control the gameplay of a game that is being played within a web browser.

There are a few different ways in which hardware interfaces can be used to support web-based games. One option is to use a device that is specifically designed to work with web games, such as a gamepad that connects to a computer via USB or Bluetooth. These types of devices often come with drivers or software that allows them to be recognized and used by web browsers.

Another option is to use more general-purpose input devices, such as a keyboard or mouse, to play web-based games. Many web games are designed to be played with these types of devices, and may offer support for multiple control schemes or custom mapping of controls to different keys or buttons.

Regardless of the specific hardware interface being used, the goal is to provide players with a more immersive and intuitive way to interact with web-based games, rather than relying on traditional input methods such as a mouse and keyboard.

‘Spooky Run’ is a multi-platform arcade style 2d gaming web application and is functional on both mobile and PC. Gaming date is temporarily stored in the cache and later cleared once the page is left or refreshed this makes it a lightweight game and properly functional without any lag.

The game uses touches and swipes on mobile and tablets. Keyboard inputs and mouse inputs are used in PC to interact with the game.

2.5.3 Software Interface

The software interface for a web-based game refers to the various software components and features that allow players to interact with and play the game within a web browser. These can include both the underlying code and architecture of the game itself, as well as any additional software or plugins that may be required to run the game within a browser.

Some common elements that are included in our web-based game's software interface are:

- Programming libraries that are used to build and run the game, such as HTML, CSS, and JavaScript.
- Server-side software that powers any online features or multiplayer components of the game.
- Client-side software that runs within the player's web browser, handling tasks such as rendering graphics, handling user input, and communicating with the game's servers.

Overall, the goal of our web-based game's software interface is to provide a stable and reliable platform for the game to run on, and to ensure that players can easily access and play the game within their web browser.

Working tools and platform

- Visual Studio Code
- Mozilla Firefox
- Google Chrome
- Brave

CHAPTER-3

PROJECT MODEL ANALYSIS

3.1 Scenario Based Model

A scenario-based model for a web-based game refers to a design approach in which the game is structured around specific scenarios or situations that the player must navigate or overcome. This can involve completing specific objectives, solving puzzles, or defeating enemies in order to progress through the game.

In a scenario-based web-based game, the game may be divided into levels or stages, each with its own set of challenges and objectives. The player may need to use different skills or abilities in order to complete each scenario, and may also have access to power-ups or other resources to help them along the way.

This model depicts how the user interacts with the system and the specific sequence of activities that occur as the software is used.

Overall, the goal of a scenario-based model in a web-based game is to provide a structured and engaging gameplay experience that challenges the player's skills and abilities while also allowing for a high degree of replay ability.

3.1.1 Use Case Scenario

The following table summarizes the use cases of the system. We have created the use cases based on UX view of the game. The table connects the UX with background programming which are the two important views of the game software.

Level-0	Level-1	Level-2
Game (Spooky Run)	Play	Start Game New Game Exit Game
	Options	Toggle Full Screen Exit Full Screen
	Quit	

Table 3.1: Use Case Scenario

3.1.2 Use Case Description

<u>Play</u>

i. **Use Case: Start Game**

Primary Actors: Anyone playing the game

Goal in context: To start a new game

Precondition:

1. System supports game configuration
2. The file has been triggered to run and the game screen has appeared

Triggers: The player needs to press Enter key on the keyboard

Scenario:

1. Run the game
2. Read the instructions
3. Press Enter key
4. Game is loaded on system

Exception: Game crashed

Priority: Essential, must be implemented

ii. **Use Case: New Game**

Primary Actors: Anyone playing the game

Goal in context: To start a new game

Precondition:

1. Game was played before
2. Game was either won or lost

Triggers: Need to start a new game

Scenario:

1. Game is won or lost
2. Game shows the win or lose screen
3. Press Shift key on the keyboard
4. New game is loaded

Exception: Game crashed

Priority: Essential, must be implemented

iii. **Use Case: Exit Game**

Primary Actors: Anyone playing the game

Goal in context: To exit from the game

Precondition: A game level is played

Triggers: Player needs to exit from the game

Scenario:

1. Gamer is done playing the game
2. Exit the web page

Priority: Essential, must be implemented

<u>Options</u>

i. **Use case: Toggle Full Screen**

Primary Actors: Anyone playing the game

Goal in context: To change the screen configuration of the game

Precondition: Player’s device allows full screen mode

Triggers: Player wants immersive gaming experience

Scenario: Click the Toggle Full screen button

Exception: System doesn’t support full screen

Priority: Expected

ii. **Use case: Exit Full Screen**

Primary Actors: Anyone playing the game

Goal in context: To change the screen configuration of the game

Precondition: Player’s device allows full screen mode

Triggers: Player wants to exit full screen mode

Scenario: Press esc key on the keyboard

Exception: None

Priority: Expected

<u>Quit</u>

Use Case: Quit Game

Primary Actors: Anyone playing the game

Goal in context: To exit from the game

Precondition: A game is played

Triggers: Player needs to exit from the game

Scenario:

3. Gamer is done playing the game
4. Exit the web page

Priority: Essential, must be implemented

3.2 Data Model

If software requirements include the need to create, extend or interface with database or if complex data structure must be constructed and manipulated, the software team may choose to create a data model as part of overall requirements modelling.

A data model for a web-based game refers to the way in which the game's data is structured and organized. This can include data related to the game's levels, characters, objects, and other assets, as well as data related to the player's progress through the game and any achievements or rewards that have been earned.

In a web-based game, the data model is often implemented using a database or other data storage system. This allows the game to store and retrieve data efficiently, and to support features such as saving the player's progress or tracking their achievements.

There are many different approaches to designing a data model for a web-based game, and the specific model used will depend on the needs of the game and the type of data it needs to store. Some common elements that might be included in a data model for a web-based game include:

Tables or collections for storing data about the game's levels, characters, objects, and other assets.

Fields or attributes for storing specific data points, such as the name of a character or the value of an in-game object.

Relationships between different data entities, such as the relationship between a character and their inventory items.

Overall, the goal of a data model for a web-based game is to provide a flexible and efficient way to store and retrieve the data that is needed to support the game's gameplay and features.

3.3 Behavioral Model

A behavioural model for a web-based game refers to the way in which the game's characters and objects behave and interact within the game world. This can include how they move, how they respond to player input or other stimuli, and how they interact with other objects or characters in the game.

In a web-based game, the behavioural model is often implemented using programming languages and libraries, such as HTML, CSS, and JavaScript. This allows the game to define the specific behaviours and interactions that should occur within the game world, and to respond to player input or other events in a logical and consistent manner.

There are many different approaches to designing a behavioural model for a web-based game, and the specific model used will depend on the needs of the game and the type of gameplay it is designed to support. Some common elements that might be included in a behavioural model for a web-based game include:

Rules or logic for determining how characters or objects should move or behave.

Conditions or triggers that cause certain behaviours or interactions to occur, such as when a character touches an object or when a player presses a specific button.

Responses or actions that occur as a result of certain behaviours or interactions, such as a character taking damage or a player earning a reward.

Overall, the goal of a behavioural model for a web-based game is to provide a logical and consistent framework for defining the behaviours and interactions that occur within the game world, and to enable the game to respond to player input and other events in a meaningful and engaging way.

The behavioral model indicates how software will respond to external events or stimuli. There are two ways to show these responses. One is state diagram and the other is sequence. Usually, state diagram can be made in two ways, one creating a state diagram for each class and the other is to create a state diagram for the whole system. This game is an object-oriented game and has classes.

CHAPTER-4

DESIGN AND IMPLEMENTATION

4.1 Sprite Animation

A sprite sheet is a single image file that contains a series of smaller graphics, called sprites. Sprite sheets are often used in video games to reduce the number of image files that need to be loaded, which can improve performance.

To create a sprite sheet, you can combine multiple smaller images into a single image file using an image editor or specialized sprite sheet generator tool. Each sprite on the sheet is typically the same size and arranged in a grid pattern.

To use a sprite sheet in a game, you will need to be able to extract individual sprites from the sheet and display them on the screen. This is typically done using a technique called sprite rendering. In sprite rendering, you specify the position and size of the sprite you want to display, and the renderer will extract that sprite from the sheet and draw it on the screen.

Sprite sheets can be used for many types of graphics in games, including characters, backgrounds, and game objects. They can also be used for other types of animations, such as web graphics and mobile app icons.

Sprite animation is a technique used in computer graphics to create the illusion of movement by displaying a series of static images, or "sprites," in rapid succession. This can be used to animate characters or objects within a video game or other interactive media.

To create a sprite animation, typically a series of individual sprites that depict a character or object in different positions or poses are created. These sprites are then played back in a specific sequence at a high frame rate, typically 30 or 60 frames per second, to create the illusion of smooth, continuous motion.

Sprite animation is often used in 2D games and other interactive media, as it allows for relatively simple and efficient rendering of characters and objects. It can also be used in 3D games, although it is generally more resource-intensive to implement.

Overall, sprite animation is a powerful tool for creating dynamic and engaging visuals in video games and other interactive media, and is widely used in a variety of different genres and styles.

Sprite Animations are animation clips that are created for 2D assets. There are various ways to create Sprite Animations. One way is to create them from a Sprite Sheet, a collection of Sprites arranged in a grid. The Sprites are then compiled into an Animation Clip that will play each Sprite in order to create the animation, much like a flipbook.

Sprite Sheets are images containing sequential sprites typically used for animation, much like a flipbook, while sprite atlases are images containing a collection of non-sequential sprites. Sprite atlases are created to pack as many sprites as possible together in a single image in order to optimize 2D games. Sprites within sprite atlas can be animated as well, but they are typically animated using unity.

Here is the example of sprite sheet we used in our game:

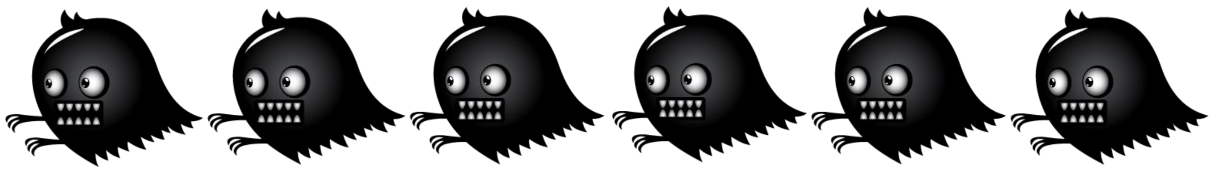

Fig 4.1: Sprite Sheet

4.2 FPS (Frames per Second)

FPS, or "frames per second," refers to the number of times per second that a display device, such as a computer monitor or television, is able to refresh its image. In the context of a 2D game, FPS is an important factor in determining the smoothness and responsiveness of the gameplay experience.

In general, a higher FPS value results in a smoother and more responsive gameplay experience, as the game is able to update the screen more frequently and respond to player input more quickly. On the other hand, a lower FPS value can result in a choppier and less responsive gameplay experience, as the game is not able to update the screen as frequently.

The FPS of a 2D game is typically determined by a combination of factors, including the hardware and software being used to run the game, the complexity of the game's graphics and gameplay, and the frame rate at which the game was designed to run.

To ensure the best possible gameplay experience, it is generally recommended to aim for an FPS value of at least 30 for a 2D game. This should provide a smooth and responsive gameplay experience on most devices and settings.

4.3 Delta Time

Delta time, often referred to as "delta Time" or "dt", is a value that represents the amount of time that has passed since the last frame in a video game or other real-time application. It is typically used to ensure that the game or application runs at a consistent speed, regardless of the performance of the device it is running on.

In a game or other real-time application, the delta time is calculated by the game engine or application each frame. It is the difference between the current time and the time of the previous frame. The delta time is then used to update the game or application logic and render the next frame.

Delta time is an important concept in game development and is used in many other types of real-time applications as well. It can be a useful tool for maintaining consistent performance and ensuring that the application behaves predictably.

To calculate the frames per second (FPS) of a game or other real-time application that is running in a web browser, you can use the delta time value along with the current time to keep track of the elapsed time and the number of frames that have been rendered.

The FPS value can then be displayed on the screen or logged to the console for debugging purposes.

It's important to note that the FPS value will fluctuate over time, depending on the performance of the device and the complexity of the game or application. You may want to smooth the FPS value using an averaging technique to get a more stable reading.

4.4 Parallax Scrolling

Parallax scrolling is a technique used in computer graphics to create the illusion of depth and movement by displaying multiple layers of images at different scrolling speeds. This is often used in 2D video games and other interactive media to create a sense of movement or progression through a virtual world.

In a parallax scrolling effect, multiple layers of images are used to depict different parts of the game world, such as the foreground, midground, and background. Each layer is then scrolled at a different speed, with the foreground typically scrolling faster than the background. This creates the illusion of depth, as it appears as though the objects in the foreground are closer to the player than those in the background.

Parallax scrolling can be used in a variety of different game genres and styles, and is often combined with other visual effects, such as sprite animation or particle effects, to create more immersive and dynamic gameplay experiences.

Overall, parallax scrolling is a powerful tool for creating a sense of movement and depth in video games and other interactive media, and is widely used to enhance the visual appeal and immersion of these types of projects.

Some display systems support multiple background layers that can be scrolled independently in horizontal and vertical directions and composited on one another. On such a display system, a game can produce parallax by simply changing each layer's position by a different amount in the same direction. Layers that move more quickly are perceived to be closer to the virtual camera.

We have used various layers of forest background. These layers move at different speeds. The layer which is closer moves faster than the previous ones in the same direction which creates parallax movement of the background and makes it look lively.

We created each layer as an instance of a class. And stored and rendered them in a specific order from an array. Each layer takes a user set or hardcoded speed value as an argument.

The actions performed by the player such as sitting, running, rolling etc., affects the speed of movement of the background.

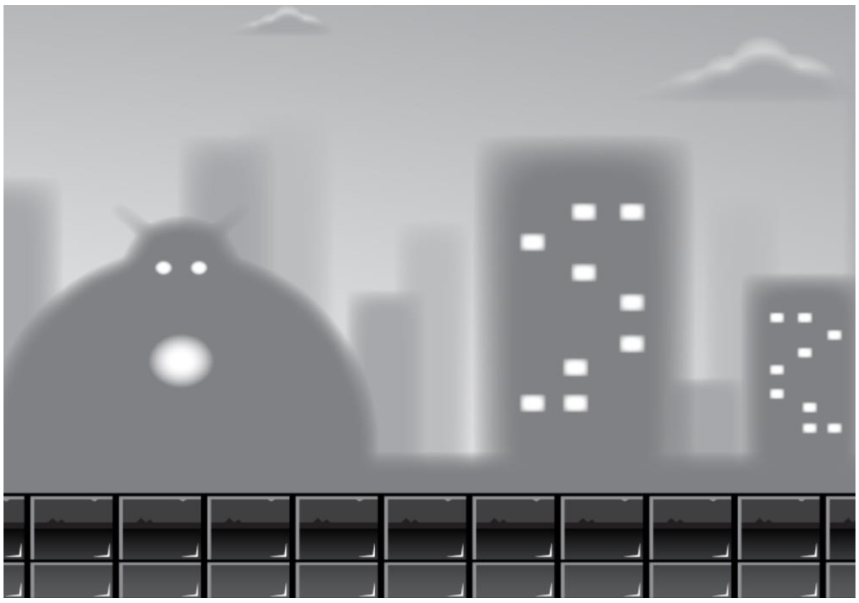

Fig 4.2: Parallax Background

4.5 Canvas Element

The HTML **`<canvas>`** element is used to draw graphics, on the fly, via JavaScript.

The **`<canvas>`** element is only a container for graphics. You must use JavaScript to draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The displayed size of the canvas can be changed using CSS, but if you do this the image is scaled during rendering to fit the styled size, which can make the final graphics rendering end up being distorted.

We have used two canvases to build the game. One of the canvases registers the touches when used in a mobile and the other displays the graphics interface of the game.

The canvas element is supported by most of the browsers hence it's the best choice for a web game.

Initially, the canvas is blank. To draw something, you need to access the rendering context and use it to draw on the canvas. The 2D drawing context features methods for drawing simple 2D shapes such as paths, rectangles, and arcs.

The coordinates in a 2D context begin at the upper left of the **`<canvas>`** element, which is point (0,0) as shown in the following picture:

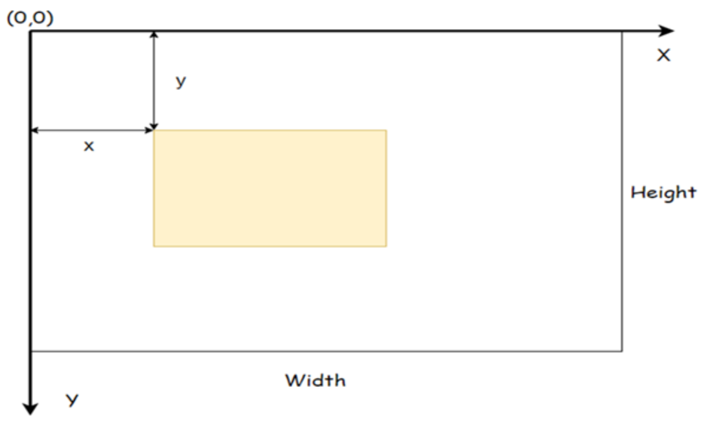

Fig 4.3: HTML Canvas Element

4.6 State Management

State management in a game is used when game invariably switches between different, unrelated modes. At one point, it's drawing the main menu, at another the player is steering his avatar through the game world, eventually it is rolling the credits.

In our game we have used state management techniques to control the player states throughout the game. The states like sitting, rolling, standing idle, running etc need to be accessed immediately when the player clicks the keys and sometimes player might click multiple keys and must be seen that the states are not missed, or the right states are implemented during the game. Using multiple if statements or loops will reduce the efficiency of the program and render the graphics slowly therefore reducing the switching speeds and accounting for bad user experience. This can be overcome using an efficient state management technique or system.

‘Spooky Run’ uses a dedicated class to manage the states and all the sprites related to the specific states are handled by that class. This increases the efficiency of the code and renders the graphics quickly.

4.7 Enemy Types

‘Spooky Run’ has wide variety of enemies which gives the player a great gaming experience. Different enemies have different movement style and different points to be earned when killed.

Enemies are spawned at different times at different positions in the game. The spawn is randomized using the random() method in JavaScript.

The movement patterns of the enemies make the game hard as well as fun to play. All the enemies in the game are animated using sprite sheet animation and are implemented using classes and objects. Once the enemies have left the screen they are removed from memory, thus making memory management efficient.

4.8 UI and Add-Ons

The game has a starting screen presenting information about how to control the character. The background for the entering screen is a snapshot from the game itself to provide a glimpse of the actual game.

Once the player starts the game, various in game details are presented. The score, in game time, player lives, and energy are presented on the top left corner of the game to not obstruct the view of the main play area.

When the player wins the game a win scenario is entered where he’s cheered and an option to restart the game is provided. If the game is lost a game lost scenario is entered with text and option to retry the game.

In addition to player and enemy animation the game also has collision animation, floating messages and particle animation. All the addon animations are made using sprite sheets except the floating messages.

The collision animation is triggered when the player collides with an enemy. A cloud of dust is animated at the spot of collision. The particle animation adds fire during an attack and blood splatters when the player loses a life. The floating messages show the number of points earned when an enemy is killed.

The game also has sound effects added to it. The add-ons and the sound effects make the game more realistic.

4.9 Controls

The game controls are a way to interact with the game. Various controls are provided to interact with the player character.

The controls available in the game are:

JUMP: ‘Arrow Up’, ‘W’

MOVE LEFT: ‘Arrow Left’, ‘D’

MOVE RIGHT: ‘Arrow Right’, ‘A’

SIT (when on ground): ‘Arrow Down’, ‘S’

DIVE (when in air): ‘Arrow Down’, ‘S’

ROLL: ‘Space Bar’

ATTACK: ‘Space Bar’

The ‘W’, ‘A’, ‘S’, ‘D’ keys also control the character allowing the player to use the keys which are comfortable.

4.10 Game Objective

The game can be played on any device since it requires a browser to play. The game is supported by most of the browsers.

The main objective of the game is to kill more than 30 monsters within 30 seconds. If the player loses all his lives, he loses the game. An enemy is killed, and a point is scored only when the player performs a roll attack or dive attack. It consists of instructions page and on pressing the ‘ENTER’ key the game starts.

CHAPTER-5

Conclusion and Future scope

5.1 Conclusion

Creating a game for the web involves a number of steps, including designing the game, building the game world, implementing the game mechanics, testing and debugging the game, and publishing it for others to play.

There are several different approaches you can take when creating a web-based game, including using a game engine like Unity or Unreal Engine, using a JavaScript library like Phaser or Pixi.js, or using pure JavaScript, HTML, and CSS.

Regardless of the approach you take, it is important to consider the performance of your game and make sure it runs smoothly on a variety of devices. You may also want to consider hosting your game on a platform like Steam or itch.io to make it easy for players to find and play your game.

To create a successful web-based game, it is important to spend time planning and designing the game, and to test and debug it thoroughly to ensure a smooth and enjoyable experience for players. With the right tools and approach, it is possible to create engaging and immersive games for the web.

In conclusion, creating a web-based game involves a number of steps, including designing the game, building the game world, implementing the game mechanics, testing and debugging the game, and publishing it for others to play. There are several different approaches that can be taken, including using a game engine, a JavaScript library, or pure JavaScript, HTML, and CSS. It is important to consider the performance of the game and make sure it runs smoothly on a variety of devices. Hosting the game on a platform like Steam or itch.io can also make it easier for players to find and play the game. To create a successful web-based game, it is important to spend time planning and designing the game and to test and debug it thoroughly. Overall, with the right tools and approach, it is possible to create engaging and immersive games for the web.

5.2 Future Scope

There are many areas where JavaScript games could continue to evolve in the future. Some potential areas of focus include:

- Improved performance: As the capabilities of web browsers continue to increase, it will be possible to create more complex and visually impressive games using JavaScript. Developers will need to focus on optimizing their games to ensure smooth performance on a variety of devices.
- Virtual and augmented reality: JavaScript games could potentially be used to create immersive virtual and augmented reality experiences in the web browser. This could involve using web technologies like WebGL and WebXR to create 3D environments and interact with them using VR and AR devices.
- Multiplayer and online features: JavaScript games could incorporate more advanced multiplayer and online features, such as real-time multiplayer gameplay, cross-platform play, and online matchmaking.
- Integration with other platforms: JavaScript games could potentially be integrated with other platforms, such as mobile apps or console games, to create more seamless and interconnected experiences.
- Artificial intelligence and machine learning: JavaScript games could leverage artificial intelligence and machine learning to create more dynamic and unpredictable gameplay experiences. This could involve using AI to generate levels or to create non-player characters that adapt to the player's actions.

These are just a few examples of the many directions that JavaScript games could potentially take in the future. As the technology and capabilities of web browsers continue to evolve, the possibilities for creating innovative and engaging games using JavaScript will only continue to grow.

References

1) Freecodecamp.org
   https://www.freecodecamp.org/
2) MDN Web Docs
   https://developer.mozilla.org/
3) Scrimba
   https://scrimba.com/
4) W3Schools
   https://www.w3schools.com/
5) ChatGPT
   https://openai.com/blog/chatgpt/
6) Bevouliin
   https://bevouliin.com/