**Project Based Learning Report**

on

**Designing a 2D maze game using UNITY**

Submitted in the partial fulfillment of the requirements

For the Project based learning in (**AUGMENTED REALITY & VIRTUAL REALITY**)

in

Electronics & Communication Engineering

By

**PRN Name of the Student**

**2014111056 Sankalp Kanikar**

**2014111076 Mansi lal**

**2014111050 Rohit Kumar Gupta**

Under the guidance of Course In-charge

Dr. M.V PATIL

Department of Electronics & Communication Engineering

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering,

Pune – 4110043

**Academic Year: 2023-24**

**Bharati Vidyapeeth**

**(Deemed to be University)**

**College of Engineering,**

**Pune – 411043**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**CERTIFICATE**

Certified that the Project Based Learning report entitled, “ **Design a 2D maze game where players navigate through a maze to reach the exit”**is work done by

**PRN Name of the Student**

**2014111056 SANKALP KANIKAR**

**2014111076 MANSI LAL**

**2014111050 ROHIT KUMAR GUPTA**

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **AUGMENTED REALITY & VIRTUAL REALITY** of Bachelor of Technology Semester VII, in ELECTRONICS AND COMMUNICATION ENGINEERING

**Date:**

**Dr. M.V PATIL Dr. Arundhati A.Shinde**

**Course In-charge Professor & Head , ECE**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SERIAL NO.** | **CONTENT** | **PAGE NO.** |
| 1. | PROBLEM STATEMENT | 4 |
| 2. | INTRODUCTION | 5 |
| 3. | STEPS INVOLVED & ASSETS USED | 6-9 |
| 4. | SOFTWARE USED | 10-11 |
| 5. | RESULTS AND PROJECT OUTCOMES | 12-13 |
| 6. | CONCLUSION | 14 |

**PROBLEM STATEMENT:**

**Design a 2D maze game where players navigate through a maze to reach the exit**

**Introduction**

This report outlines the design of a 2D maze game where players navigate through a maze to reach the exit using the Unity game development software. The game is designed to provide an entertaining and challenging experience for players, encouraging them to use their problem-solving and navigation skills to find there way out of the maze.

**Game Objective:**

The primary objective of the game is for players to navigate through a series of mazes and reach the exit point as quickly as possible. The game may feature multiple levels, each with increasingly complex mazes to solve. Players aim to achieve the fastest completion times and can earn rewards or points for their performance.

**Gameplay:**

* **Controls:** Players will control a character, typically from a top-down perspective, using keyboard arrow keys or touch controls on mobile devices. The character moves within the maze grid and changes direction at intersections.
* **Maze Generation:** Unity's tilemap system can be used to create and design maze layouts. Procedural generation can be implemented to ensure maze variety, using algorithms like recursive backtracking or Prim's algorithm.
* **Obstacles:** The mazes will contain a variety of obstacles and challenges, including locked doors, keys, moving obstacles, and traps that impede the player's progress.
* **Scoring:** Points or rewards can be awarded based on factors such as completion time, the number of attempts, or the number of collected items within the maze.
* **Progression:** As players advance through the game, the mazes become progressively more complex, with narrower paths, dead-ends, and more obstacles.

**Visual and Audio:**

* **Graphics:** Unity's 2D rendering capabilities can be leveraged to create maze walls, characters, and obstacles with distinct visual styles. The game's art assets should be designed to enhance navigation and visual appeal.
* **Audio:** Background music and sound effects, such as footsteps, door openings, and victory fanfares, will contribute to the player's immersion and engagement.

### **Maze Generation**:To ensure maze variety, Unity's tilemap system can be employed to construct mazes. Procedural maze generation algorithms can be implemented within Unity to create unique maze layouts.

### **User Interface:**The game should include user-friendly menus for starting a new game, selecting difficulty levels, and viewing high scores. Unity's UI tools make it relatively easy to design and implement the user interface.

### **Cross-Platform Compatibility**:Unity offers the advantage of cross-platform compatibility. The game can be developed for various platforms, including Windows, macOS, iOS, Android, and web browsers, with minimal adjustments required.

STEPS INVOLVED:

Creating a maze in Unity is a popular project for game developers and enthusiasts. Mazes can be used in various game genres, such as puzzle games, adventure games, and even horror games.

Steps involved in creating this project are as follows:

**1. Setting up your Unity project:**

* Launch Unity and create a new 3D project.
* Ensure you have a camera in the scene, as this will be the player's perspective.

**2. Creating the Maze Structure:**

* You can create your maze structure using Unity's built-in 3D modeling tools, or you can import pre-made models if you prefer.
* For a simple maze, you can use cubes to create walls. Adjust the scale, position, and rotation as needed to form the maze layout.
* Ensure there's a clear starting point and a destination point within the maze.

**3. Adding a Player Character:**

* Create or import a player character, which can be a simple first-person or third-person controller.
* Place the player character at the starting point within the maze.

**4. Player Controls:**

* Implement player controls to allow movement within the maze.
* You can use Unity's Input System to handle player movement (e.g., WASD or arrow keys for navigation).

**5. Maze Interactivity:**

* If you want to make your maze more engaging, consider adding interactive elements, such as switches, traps, or collectibles.
* Use Unity's scripting system (C#) to handle interactions and gameplay mechanics.

**6. Implementing Collision:**

* Ensure that your maze walls have colliders attached to them. This is necessary to prevent the player from moving through walls.

**7. Lighting and Environment:**

* Adjust lighting and environmental settings to create the desired atmosphere in your maze. Unity's Lighting and Post-Processing features can help you achieve the desired mood.

**8. Camera Control:**

* Implement camera control to follow the player character smoothly through the maze.
* Unity's Cinemachine package can help you set up camera behavior and transitions.

**9. Testing and Debugging:**

* Test your maze to make sure the player can navigate through it and complete the objective.
* Use Unity's debugging tools to identify and fix any issues.

**10. Building the Game:**

* Once you're satisfied with your maze, build the game for your target platform (e.g., PC, mobile, VR).

ASSETS USED:

When creating a maze game in Unity, various assets are required to build the maze itself, add visual elements, and enhance the overall player experience.

Some of the assets used in this project are:

1. **3D Models**: Depending on the style of your maze, assets for walls, floors, ceilings, doors, switches, collectibles, and any other objects within the maze are needed. One can create these assets using modeling software like Blender or find them on the Unity Asset Store.
2. **Textures and Materials**: High-quality textures and materials can make the maze visually appealing. One need brick textures for walls, floor textures, and materials for various elements in your maze.
3. **Sound Effects**: Audio assets are crucial for creating a captivating atmosphere. You might need ambient sounds, footsteps, door opening sounds, or other audio cues for interactions.
4. **UI Elements**: If your maze game includes menus, health bars, timers, or other user interface elements, you'll need UI assets. Unity's UI system or assets from the Asset Store can help with this.
5. **NavMesh Assets**: To enable AI characters to navigate the maze, you'll need to create a NavMesh. Unity's built-in NavMesh components can be used for this purpose.
6. **Skyboxes**: A well-chosen skybox can enhance the visual quality of your maze game, especially if it's set in an outdoor or large indoor environment.
7. **Advanced Camera Control**: Assets like Unity's Cinemachine can provide more advanced camera control and transitions to create a cinematic experience.
8. **3D Path finding Systems**: If your maze includes complex AI characters, you might need 3D path finding systems like A\* Path finding Project or RAIN AI for Unity.
9. **Scripting and Code Assets**: Depending on your game's mechanics and interactions, you may require code assets or plugins for specific gameplay features.

**NAV MESH ASSETS**: NavMesh assets are a fundamental part of creating interactive 3D environments where characters or AI entities need to navigate intelligently. They play a vital role in video game development and various applications that require simulated movement within 3D spaces.

A NavMesh is a 3D representation of the game world, typically generated from the game's geometry, which is then used by artificial intelligence (AI) agents to navigate through the environment. It consists of polygons that define where characters can move, and these polygons are connected to create a network of possible paths

1. **Generation:** NavMeshes are generated during the game development process. This can be done manually by level designers or automatically using tools or scripts that analyze the level's geometry.
2. **Components:** NavMesh assets consist of several components, including:
   * **NavMesh Surface:** This component is responsible for generating the NavMesh based on the colliders in the scene. It can be configured to handle various types of geometry, including terrain, static and dynamic objects.
   * **NavMesh Agent:** This component is attached to AI characters and defines their navigation properties, such as speed, acceleration, and radius. It allows them to interact with the NavMesh to find paths and move around.
3. **Pathfinding:** Once the NavMesh is generated, AI agents use algorithms like A\* (A-Star) to find the optimal path from their current position to a target location. These algorithms take into account the layout of the NavMesh, obstacles, and other factors to calculate the path.
4. **Dynamic Navigation:** In dynamic environments, the NavMesh may need to be updated in real-time to accommodate changes, such as moving obstacles. Many game engines provide tools and scripts to handle dynamic NavMesh updates.
5. **Usage:** NavMesh assets are widely used in video games, virtual reality simulations, and other interactive 3D applications. They are crucial for enabling characters and NPCs to move realistically through complex environments.
6. **Optimization:** Efficient NavMesh generation and pathfinding algorithms are crucial for maintaining good performance in games. This may involve optimization techniques like hierarchical pathfinding or offloading pathfinding to a separate thread.
7. **Debugging:** Many game engines provide tools to visualize the NavMesh and test pathfinding in the editor. This helps developers fine-tune the navigation for AI characters.

**SKYBOXES**: Skybox is a technique used in computer graphics and 3D rendering to create the illusion of a distant and immersive background in a 3D scene. It's essentially a textured cube or sphere that surrounds the 3D environment, providing a backdrop that simulates a distant sky or environment.

Skyboxes are a valuable tool in 3D graphics and game development, providing a visually appealing and efficient way to create immersive backgrounds for 3D scenes. They contribute to the overall atmosphere and realism of a virtual environment while helping maintain good performance.

**1. Visual Backgrounds:** Skyboxes are used to provide a visually pleasing and immersive background for 3D scenes, especially in computer games and virtual environments. Instead of rendering a detailed 3D world far into the distance, which can be computationally expensive, a skybox is used to represent that distant world.

**2. Box or Sphere:** Skyboxes are typically implemented as a cube (hence the name "skybox") or a sphere. The six faces of the cube or the interior surface of the sphere are textured with images that represent different parts of the distant environment, such as the sky, mountains, cityscape, or any other scenery that fits the scene's theme.

**3. Seamless Textures:** The key to a convincing skybox is creating seamless textures for the skybox's faces. These textures should blend together seamlessly at their edges to avoid visible seams and provide a continuous, realistic backdrop.

**4. Fixed to Camera:** A skybox is usually fixed to the camera, which means that as the camera moves through the 3D scene, the skybox moves with it. This creates the illusion of a distant, stationary background, even though the camera may be exploring the 3D environment.

**5. Performance Benefits:** Skyboxes offer performance advantages over rendering a fully 3D distant environment. They can be pre-rendered and require less computational power to display, making them an efficient way to provide a visually pleasing background without sacrificing performance.

**6. Customization:** Game developers and 3D artists can create custom skyboxes to fit the specific visual style and atmosphere of their game or simulation. This allows for creative freedom in designing the environment's backdrop.

**7. Realism and Immersion:** A well-crafted skybox can significantly enhance the realism and immersion of a 3D scene. It creates the impression of a vast, open world without the need to render all the details of that world.

**8. Variations:** Beyond traditional static skyboxes, some advanced rendering techniques involve dynamic or procedural skyboxes that change in response to factors like time of day or in-game events. These can further enhance the realism and atmosphere of a game.

Software Used

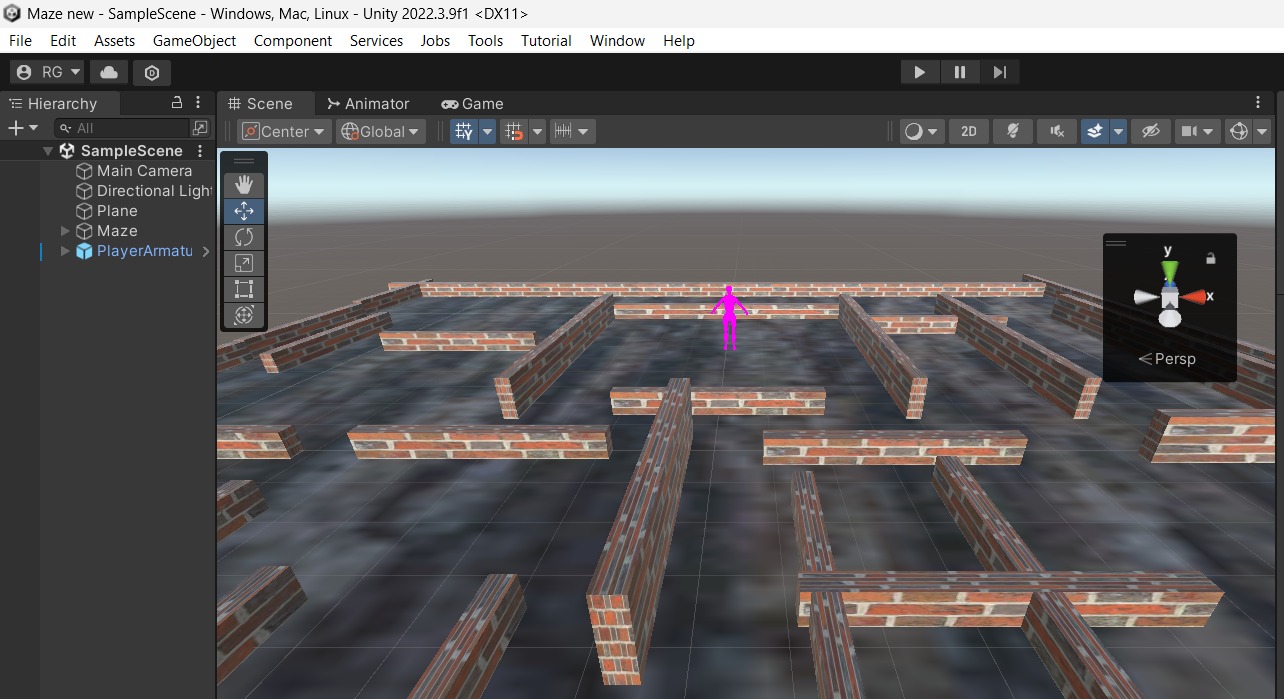
UNITY

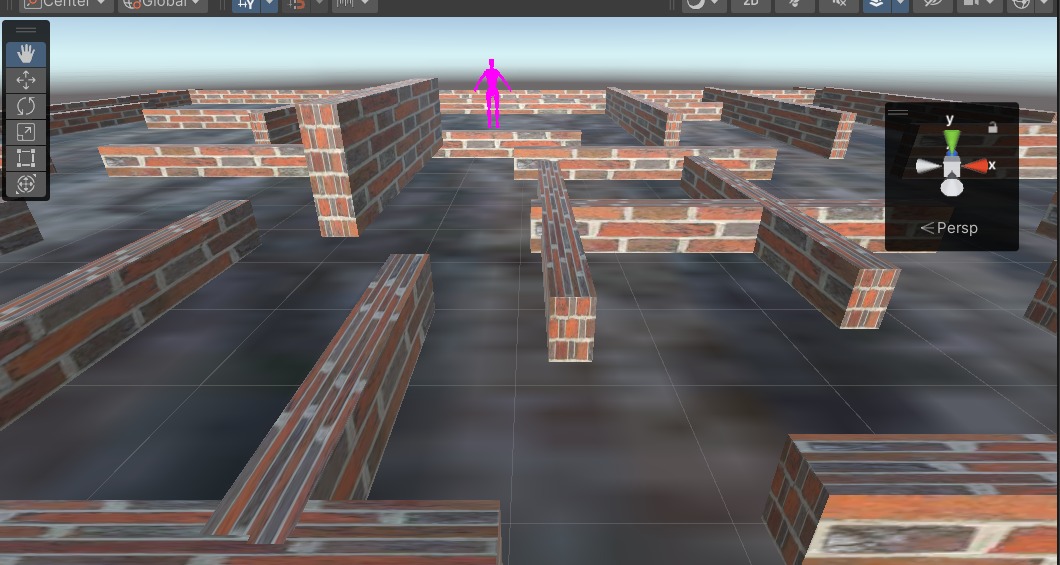
Unity is industry-leading engine provides tools to create and operate amazing games and other real-time interactive experiences and publish them to a wide range of devices it is a versatile and widely-used platform for game development and interactive experiences. Its flexibility, cross-platform support, and large community of developers make it a popular choice for a wide range of projects. Its versatility and flexibility, making it a popular choice for creating both 2D and 3D games, simulations, interactive experiences, and various applications.

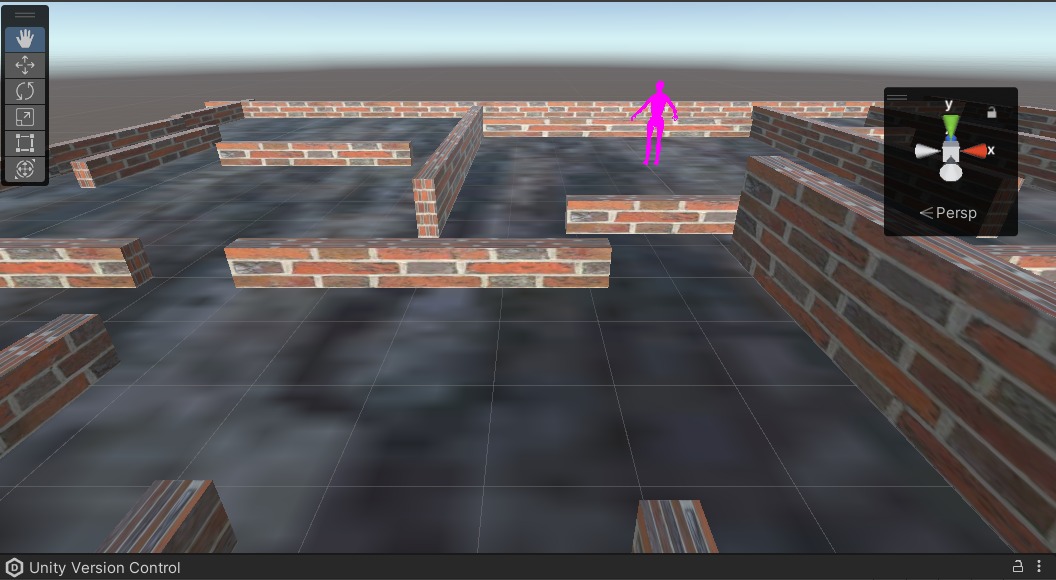
Some key aspects of Unity software are:

1. **Cross-Platform Development**: Unity allows developers to create applications for a wide range of platforms, including mobile devices (iOS and Android), desktop (Windows, macOS, Linux), consoles (PlayStation, Xbox), VR/AR devices (Oculus Rift, HTC Vive, HoloLens), web browsers, and more. This is achieved through a single codebase, saving developers time and effort.
2. **Programming Languages**: Unity primarily uses C# for scripting, making it accessible to developers familiar with this language. However, Unity also supports JavaScript and Boo.
3. **Asset Store**: Unity has a vast Asset Store where developers can buy or download free assets, such as 3D models, textures, plugins, and scripts, to accelerate development.
4. **Visual Editor**: Unity features a user-friendly visual editor that simplifies game and application design. Developers can create and manipulate objects, adjust properties, and arrange elements using a drag-and-drop interface.
5. **Physics Engine**: Unity includes a robust physics engine, which enables realistic simulations of 2D and 3D environments. This is essential for creating games and interactive experiences with lifelike movements and interactions.
6. **Graphics and Rendering**: Unity offers high-quality graphics and rendering capabilities, supporting advanced lighting, shading, and post-processing effects. It is compatible with a wide range of platforms and graphics APIs.
7. **Animation**: Unity provides tools for creating complex animations, including character animation, skeletal animation, and more. This is essential for creating dynamic and engaging content
8. **Networking**: Unity includes networking functionality, making it suitable for multiplayer and online games. Developers can create networked experiences and synchronize data between players and servers.
9. **AR/VR Support**: Unity is a popular choice for creating augmented reality (AR) and virtual reality (VR) applications. It has built-in support for various AR/VR devices, including the Oculus Rift, HTC Vive, and various mobile AR platforms.
10. **Community and Documentation**: Unity has a large and active community of developers. There are numerous tutorials, forums, and documentation available to help users at all skill levels.
11. **Unity Services**: Unity provides a suite of services, such as Unity Ads, Unity Analytics, and Unity Cloud Build, to help developers monetize and manage their games more effectively.

RESULTS :







**CONCLUSION AND PROJECT OUTCOME:**

The project “Designing of a 2D maze game” has helped us to understand the concept of unity software , its advantages and applications and designed an immersive and challenging game, leveraging Unity's powerful tools for graphics and scripting. We have encountered and overcome various challenges, such as maze generation algorithms, pathfinding, and graphical representation.

This project not only helped us in the application based learning but also has built the concept of the unity in which we learnt to create game.

Hence course outcome

CO2 & CO6 – Identify various geometric modeling techniques and apply the required knowledge for analysis Virtual/Augmented Reality Application is achieved .

Appendix:

Github link:

Sources: <https://youtu.be/yGidpBVWPx0?si=sxQlWJqaNGBFLOY4>

<https://faramira.com/implement-mazes-in-unity2d/>