**Unity 3D Project**

**A**

**Project Report Submitted**

in the Partial Fulfillment of the Requirements

for the award of

**Bachelor of Technology**

in

**Computer Science and Engineering**

by

**ARCHIT SHARMA(2004480100001)**

**UDAYBHAN KUMAR(2104480109014)**

Under the Guidance of

**Prof. Fatima Khan**

**Department of Computer Science & Engineering**

****

**Neelam college of enginnering and technology,**

**Agra**

**to the**

****

**DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY**

**LUCKNOW (UP), INDIA**

**MAy-2024**

**DECLARATION**

*I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.*

*Signature:*

*Name : ARCHIT SHARMA*

*Roll No. : 2004480100001*

*Signature:*

*Name : UDAYBHAN KUMAR*

*Roll No. : 2104480109014*

**CERTIFICATE**

This is to certify that *Project Report entitled – Unity 3D Project* which is submitted by *Archit Sharma (21004480100001) Udaybhan kumar (*21004480109014)in partial fulfilment of the requirement for the award of degree B.Tech. in Department of CSE, of Dr A.P.J. Abdul Kalam Technical University, U.P, Lucknow, is a record of the candidate own work carried out by him/her under my/our supervision. The matter embodied in this Project report is original and has not been submitted for the award of any other degree.

### Date: Project Guide

**Prof. Fatima Khan**

**Agra**

**Acknowledgment**

*It gives us a great sense of pleasure to present the report of the B.Tech Project undertaken during B.Tech Final Year. We own special debt of gratitude to our guide Mrs. Fatima Khan Department of CSE, Neelam College of Enginnering and Technology, Agra for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day*.

*We also take the opportunity to acknowledge the contribution of Prof. Ankit Kumar, Department of CSE, , Neelam College of Enginnering and Technology, Agra, for his full support and assistance during the development of the project.*

*We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.*

|  |  |  |
| --- | --- | --- |
| *Signatur*  *Name* | *e*  *: ARCHIT SHARMA* | *Signature:*  *Name : UDAYBHAN KUMAR* |
| *Roll No.* | *: 2004480100001* | *Roll No. : 2104480109014* |
|  |  |  |

**TABLE OF CONTENT**

1. **Introduction**
2. **Project Setup**
3. **Scene Setup**
4. **Player Controller**
5. **Environment interaction**
6. **UI Development**
7. **Game Mechanics**
8. **Polishing and Optimization**
9. **Deployment**

**10. Conclusion**

1. **Introduction**

Welcome to "Gun Game: Tactical Warfare", an immersive Unity 3D project designed to provide players with an adrenaline-pumping experience of combat in a dynamic virtual environment. In this game, players are thrust into intense battles where their survival depends on quick reflexes, strategic thinking, and precise marksmanship.

**Objective:**

Your mission is clear: eliminate all enemy forces that stand in your way. Whether it's battling through waves of AI-controlled adversaries or competing against other players in multiplayer mode, your primary goal is to emerge victorious by mastering the art of combat.

**Gameplay Features:**

**1.** **Diverse Arsenal**: Explore a wide range of firearms, from pistols and shotguns to assault rifles and sniper rifles. Each weapon offers unique attributes and handling characteristics, allowing players to customize their loadout according to their preferred playstyle.

1. **Dynamic Environments**: Engage in fast-paced firefights across a variety of meticulously crafted environments, including urban cityscapes, dense forests, and desolate wastelands. Adapt to the ever-changing terrain as you seek cover, flank enemies, and seize tactical advantages.
2. **Enemy AI**: Encounter intelligent enemy AI that reacts dynamically to your actions. Prepare for challenging encounters as foes employ advanced tactics, coordinate attacks, and exploit weaknesses in your defenses. Every battle presents a new test of your combat skills.
3. **Progression System**: Earn experience points and unlock new weapons, equipment, and abilities as you progress through the game. Customize your character to suit your preferred playstyle, whether it's specializing in long-range sniping, close-quarters combat, or stealthy infiltration.
4. **Multiplayer Mode**: Challenge friends and players from around the world in exhilarating multiplayer battles. Test your skills in team-based skirmishes, free-for-all deathmatches, and objective-based missions. Form alliances, coordinate strategies, and prove your dominance on the battlefield.

**Conclusion:**

"Gun Game: Tactical Warfare" offers an immersive and action-packed gaming experience that will keep players on the edge of their seats. With its stunning visuals, realistic gun mechanics, and intense multiplayer battles, this Unity 3D project promises to deliver hours of adrenaline-fueled excitement for both casual gamers and hardcore enthusiasts alike.

**2.** **Project Setup : Gun Game Development**

Ensure you have Unity 3D installed on your system. The recommended version is Unity 2020.3 LTS (Long Term Support) or newer for stability and compatibility with the latest features and packages.

**IDE (Integrated Development Environment):**

You can use any code editor or IDE of your choice, such as Visual Studio or JetBrains Rider, for scripting in C# within Unity.

**Necessary Plugins/Packages:**

1. **Unity Hub**: Use Unity Hub to manage multiple Unity versions and projects efficiently. Download and install it from the official Unity website.
2. **Standard Assets**: Unity's Standard Assets package provides essential resources such as character controllers, vehicles, and environment assets. Import this package into your project to access pre-made assets that can accelerate development.
3. **Cinemachine**: Cinemachine is a powerful camera system for Unity that enables dynamic camera movements, transitions, and composition. Install it via the Unity Package Manager to enhance your game's visuals and cinematic quality.
4. **ProBuilder:** ProBuilder is a versatile 3D modeling tool integrated into Unity for creating and editing 3D geometry directly within the editor. It's useful for prototyping levels, props, and environmental elements. Install it through the Unity Package Manager to streamline level design workflows.

**Setting up the Environment:**

1. **Create a New Unity Project**: Open Unity Hub, create a new project, and select the 3D template. Choose a project name and location on your system.
2. **Import Necessary Packages**: Once inside the Unity editor, navigate to the Package Manager window. Install the Standard Assets, Cinemachine, and ProBuilder packages if they're not already included in your project.
3. **Set Up Scene and Assets**: Design your game scene by importing assets such as terrain, props, and character models. Utilize ProBuilder to create custom level geometry or modify existing meshes.
4. **Scripting**: Begin scripting enemy behaviors, player controls, weapon mechanics, and game logic using C# scripts. Attach scripts to appropriate GameObjects to define their functionality within the game.
5. **Testing and Iteration**: Test your game frequently to identify bugs, refine gameplay mechanics, and optimize performance. Use Unity's built-in debugging tools to diagnose issues and make necessary adjustments.

**Conclusion:**

With Unity 3D, an appropriate IDE, and essential Plugins/packages installed, you're ready to embark on the exciting journey of Gun Game development. Follow these setup instructions to create a captivating gaming experience where players must eliminate enemies to emerge victorious.

**3. Scene Setup for Gun Game Development**

**1. Terrain Creation:**

* Create a new terrain by selecting GameObject -> 3D Object -> Terrain.
* Use the Terrain Tools to sculpt the terrain to fit your level design, including hills, valleys, and obstacles.
* Apply textures and foliage to the terrain to enhance visual appeal and provide cover for players and enemies.

**2. Lighting Setup:**

* Add directional light to simulate sunlight and create shadows in the scene.
* Adjust light settings such as intensity, color, and shadows to achieve the desired ambiance.
* Utilize additional light sources (point lights, spotlights) to illuminate specific areas or enhance visual effects.

**3. Camera Setup:**

* Create a new camera GameObject to serve as the player's perspective.
* Adjust the camera's position, rotation, and field of view to provide optimal visibility and gameplay experience.
* Implement camera controls (e.g., mouse or joystick input) to enable players to move the camera view freely or focus on specific targets.

**4. Environment Assets:**

* Populate the scene with environmental assets such as buildings, foliage, rocks, and other props to create a realistic and immersive setting.
* Place interactive objects like doors, switches, or destructible elements to add depth to the gameplay and encourage exploration.

**5. Player and Enemy Spawning:**

* Position spawn points for players and enemies strategically throughout the level.
* Ensure spawn points are placed in safe locations for players and provide sufficient cover to avoid spawn camping.
* Implement spawning logic to dynamically generate enemies based on player proximity or predefined triggers.

**6. Weapon and Ammo Placement:**

* Scatter weapon pickups and ammo crates across the level to supply players with firepower and ammunition.
* Distribute weapons strategically to encourage exploration and risk-reward decisions.
* Use scripting to respawn weapons and ammo over time or upon player interaction.

**7. Navigation and Pathfinding:**

* Set up navigation meshes or waypoints to enable AI-controlled enemies to navigate the environment intelligently.
* Implement pathfinding algorithms (e.g., A\* algorithm) to calculate optimal routes for enemies to pursue players or patrol designated areas.

**8. Audio Integration:**

* Add background music, ambient sounds, and sound effects to enhance immersion and atmosphere.
* Trigger audio cues based on gameplay events such as gunfire, explosions, and enemy encounters to provide feedback to players.

**Conclusion:**

By following these steps, you can create captivating game scenes for your Gun Game project in Unity. Remember to playtest the scenes regularly to ensure balanced gameplay, smooth performance, and an enjoyable player experience.

1. **Player Controller Setup:**

**1. Creating the Player Controller Script:**

* Open your preferred code editor (e.g., Visual Studio or JetBrains Rider).
* Create a new C# script named "PlayerController" or a similar descriptive name.

**2. Implementing Player Movement and Shooting:**

* Within the PlayerController script, define variables for player movement speed, jump force, and shooting mechanics.
* Use Unity's Input system to capture player input for movement (e.g., WASD keys or joystick) and shooting (e.g., left mouse button or gamepad trigger).
* Implement functions to handle player movement, including character rotation, forward/backward movement, strafing, and jumping.
* Implement a shooting function to instantiate bullets or projectiles from the player's weapon, applying force or damage to enemies upon collision.

**3. Attaching the Script to the Player GameObject:**

* Save the PlayerController script and return to the Unity editor.
* Select the player GameObject in the scene hierarchy.
* Drag and drop the PlayerController script from the project folder onto the player GameObject in the Inspector window.

**4. Testing and Iteration:**

* Playtest the game to ensure that the player character responds correctly to input and interacts with the environment as intended.
* Fine-tune movement parameters, such as speed and jump force, to achieve smooth and responsive controls.
* Debug any issues with player movement or shooting functionality by using Unity's debugging tools and console logs.

This script provides basic functionality for player movement, rotation, jumping, and shooting. Adjust and expand upon it according to your specific game design requirements.

1. **Environment Interaction Setup:**

**1. Implementing Pickup Items:**

* Create a script to handle the behavior of pickup items (e.g., weapons, ammo crates).
* Use collision detection to detect when the player interacts with a pickup item.
* Upon interaction, modify player inventory or attributes accordingly (e.g., add ammo to the current weapon).
* Destroy or deactivate the pickup item to prevent further interaction.

**2. Opening Doors:**

* Create a script to manage the opening and closing of doors in the game world.
* Use triggers or raycasting to detect when the player approaches a door.
* Upon interaction (e.g., pressing a designated key), trigger animations or physics-based movement to open or close the door.
* Implement optional locking mechanisms or key requirements for specific doors to add depth to gameplay progression.

**3. Code Snippets:**

**Conclusion:**

By following these instructions and utilizing the provided code snippets, you can add engaging environment interactions to your Gun Game project in Unity. Experiment with different interaction mechanics and refine them based on your game's design and player feedback.

1. **UI Development for Gun Game:**

**1. Creating UI Elements:**

* **HUD (Heads-Up Display):** Design elements like health bars, ammo counters, and mini-maps to provide essential information to players during gameplay.
* **Menus:** Create menus such as the main menu, pause menu, settings menu, and game over menu. Include options to start or resume the game, adjust settings, and quit the game.
* **Dialogs**: Implement dialogs for in-game messages, notifications, and prompts. These can include alerts for low health, ammo pickups, or mission objectives.

**2. UI Scripting:**

* **HUD Manager Script**: This script manages updating the HUD elements based on the player's status (health, ammo, etc.).
* **Menu Manager Script**: Handles menu navigation, button clicks, and overall menu functionality.
* **Dialog Manager Script**: Controls the display and behavior of in-game dialogs, including showing and hiding them based on game events.

**3. Attaching UI Scripts to Game Objects:**

* Create empty Game Objects in your scene to act as managers for each type of UI element (HUD Manager, Menu Manager, Dialog Manager).
* Attach the corresponding script to each manager Game Object to handle UI functionality.
* Link UI elements (such as Text or Button components) to the appropriate variables in the UI scripts via the Unity Editor.

1. **Code Examples:**

**HUD Manager Script:**

**Menu Manager Script:**

**Dialog Manager Script:**

**Conclusion:**

With these UI development guidelines and code examples, you can create intuitive and immersive user interfaces for your Gun Game in Unity. Tailor the UI elements and functionality to match the theme and gameplay mechanics of your game, providing players with an engaging and enjoyable experience.

1. **Game Mechanics for Gun Game Development:**

**1. Scoring System:**

* Objective: Keep track of the player's performance and provide feedback on their progress.
* Implementation Steps:

1. Create a Score Manager script to manage scoring logic.
2. Award points for actions such as defeating enemies, completing objectives, or collecting items.
3. Update the UI to display the player's current score.

**2. Enemy AI:**

* Objective: Create dynamic and challenging opponents for the player to engage with.
* Implementation Steps:

1. Design enemy behaviors such as patrolling, chasing, and attacking using finite state machines or behavior trees.
2. Implement line-of-sight detection to enable enemies to detect and track the player.
3. Include decision-making algorithms for enemies to determine their actions based on factors like proximity, health, and available cover.

**3. Weapon Mechanics:**

* Objective: Provide satisfying and realistic gunplay mechanics for the player.
* Implementation Steps:

1. Create a Weapon script to manage weapon properties such as damage, fire rate, and ammo capacity.
2. Implement shooting mechanics including aiming, firing projectiles, and applying damage to targets.
3. Include reloading functionality with appropriate animations and cool down times.

**4. Puzzle Mechanics:**

* Objective: Introduce puzzles or challenges to add variety and depth to gameplay.
* Implementation Steps:

1. Design puzzles that require problem-solving skills or interaction with the environment (e.g., finding keys to unlock doors, activating switches to progress).
2. Create puzzle elements such as movable objects, pressure plates, and logic gates.
3. Implement feedback mechanisms to provide hints or feedback to the player during puzzle solving.

**Code Snippets:**

**Conclusion:**

By implementing these game mechanics, you can create a dynamic and engaging Gun Game experience for players. Experiment with different variations and combinations of mechanics to tailor the gameplay to your vision and provide an enjoyable challenge for players.

1. **Optimizing Performance:**
2. **Level of Detail (LOD):** Implement LOD groups for complex models to reduce the number of polygons rendered at a distance, improving performance.
3. **Batching**: Combine static Game Objects into fewer draw calls by utilizing batching techniques like Static Batching and Dynamic Batching.
4. **Texture Compression**: Use texture compression formats (e.g., DXT, ETC) to reduce texture memory usage without sacrificing visual quality.
5. **Culling**: Implement occlusion culling and frustum culling to avoid rendering objects that are not visible to the camera, reducing GPU workload.
6. **Shader Optimization:** Simplify shaders and use shader variants sparingly to minimize GPU overhead. Utilize GPU Profiler to identify performance bottlenecks in shaders.
7. **Lighting Optimization:** Opt for baked lighting whenever possible to reduce runtime calculations. Use light probes and occlusion culling to enhance lighting quality while maintaining performance.
8. **Asset Optimization:** Optimize 3D models and textures by reducing polygon counts and using texture atlases to minimize memory usage.

**Improving Visual Quality:**

1. **High-Quality Textures**: Use high-resolution textures for important assets to enhance visual fidelity. Utilize normal maps, specular maps, and ambient occlusion maps to add depth and realism.
2. **Post-Processing Effects:** Apply post-processing effects like bloom, ambient occlusion, and color grading to enhance the overall visual aesthetic of the game.
3. **Dynamic Lighting**: Utilize dynamic lighting effects such as real-time shadows, reflections, and particle effects to create immersive environments and engaging gunplay visuals.
4. **Particle Systems**: Implement particle systems for muzzle flashes, bullet impacts, and explosions to add visual feedback and excitement during combat encounters.
5. **Environmental Details**: Add environmental details such as foliage, debris, and interactive elements to enrich the game world and make it feel more immersive and alive.
6. **UI/UX Design**: Design intuitive and visually appealing user interfaces with clear navigation and feedback elements to enhance the overall player experience.

**Polishing Gameplay:**

1. **Balanced Difficulty:** Fine-tune enemy AI behaviors and weapon mechanics to provide a challenging but fair gameplay experience. Adjust difficulty levels based on player feedback.
2. **Feedback Systems**: Implement audiovisual feedback for player actions, such as hit indicators, sound effects, and screen shakes, to provide immediate feedback and reinforce player actions.
3. **Smooth Controls**: Ensure responsive and intuitive player controls by refining movement mechanics, aiming sensitivity, and input responsiveness.
4. **Bug Fixing:** Conduct thorough playtesting and debugging to identify and fix any gameplay bugs, glitches, or inconsistencies that may detract from the overall experience.
5. **Player Progression**: Introduce progression systems such as unlocking new weapons, abilities, or levels as players advance through the game to maintain engagement and motivation.
6. **Deployment for Gun Game:**

**1. Building for Different Platforms:**

* PC (Windows, mac OS, Linux):
* Go to File -> Build Settings in Unity.
* Select the target platform (e.g., PC, Mac, Linux) and click "Switch Platform."
* Configure build settings (e.g., resolution, graphics quality) and click "Build" to generate the executable file.
* Mobile (iOS, Android):
* Install the necessary platform-specific SDKs and tools (X code for iOS, Android Studio for Android).
* Connect your device to the computer and enable USB debugging.
* In Unity's Build Settings, select the target platform (iOS or Android) and click "Switch Platform."
* Configure player settings (e.g., bundle identifier, minimum API level) and click "Build and Run" to deploy the game directly to the device for testing.

**2. Testing and Debugging:**

* Testing:
* Conduct thorough testing on each platform to ensure compatibility and functionality.
* Test gameplay mechanics, UI elements, performance, and stability across different devices and screen resolutions.
* Gather feedback from play testers and iterate on any issues or suggestions.
* Debugging:
* Use Unity's built-in debugging tools, such as the Console window and Profiler, to identify and fix bugs.
* Utilize platform-specific debugging tools (e.g., X code Debugger for iOS, Android Debug Bridge for Android) to diagnose and troubleshoot platform-specific issues.

**3. Publishing to Distribution Platforms:**

* **Steam (PC):**
* Prepare promotional materials (e.g., screenshots, trailers, store description) and create a store page on Steam works.
* Build your game for PC and upload the executable file, along with any additional files (e.g., assets, dependencies), to Steam works.
* Set pricing, release date, and other store settings on Steam works before submitting the game for review.
* Once approved, your game will be available for purchase/download on the Steam store.
* **App Store (iOS):**
* Create an Apple Developer account and set up your app in App Store Connect.
* Generate an iOS build from Unity and submit it to App Store Connect for review.
* Provide required metadata (e.g., app name, description, screenshots) and set pricing and availability options.
* Once approved, your game will be available for download on the App Store.
* **Google Play Store (Android):**
* Create a Google Play Developer account and set up your app in the Google Play Console.
* Generate an Android build from Unity and upload the APK file to the Google Play Console.
* Fill out store listing details (e.g., app title, description, screenshots) and configure pricing and distribution settings.
* After review and approval, your game will be published on the Google Play Store.

**4. Post-launch Support:**

* **Updates and Patches**:
* Monitor player feedback and address any reported issues promptly through updates and patches.
* Release new content, features, or improvements based on player feedback and market trends to keep the game fresh and engaging.
* **Community Engagement**:
* Engage with the player community through forums, social media, and in-game communication channels to foster a supportive and active player base.
* Respond to player inquiries, feedback, and suggestions to demonstrate commitment to the game's ongoing development and improvement.

**10. Conclusion: Gun Game Development**

In conclusion, our Gun Game project is a thrilling and immersive gaming experience that puts players in the heart of intense combat scenarios. Let's summarize its key features and reflect on the development process:

* **Key Features**:
* Engaging Gunplay: Enjoy dynamic gunplay mechanics with a diverse arsenal of weapons, realistic shooting mechanics, and challenging enemies.
* Immersive Environments: Explore meticulously crafted environments ranging from urban landscapes to desolate wastelands, each offering strategic advantages and obstacles.
* Intelligent Enemy AI: Encounter intelligent enemy AI that adapts to your tactics, providing challenging and rewarding combat encounters.
* Customizable Gameplay: Customize your playstyle with a progression system that unlocks new weapons, equipment, and abilities as you advance through the game.
* Multiplayer Mode: Challenge friends and players worldwide in exhilarating multiplayer battles, showcasing your skills and strategic prowess.
* **Development Reflection:**

The development process of our Gun Game has been both challenging and rewarding. We've learned valuable lessons along the way, including the importance of:

* **Iterative Design**: Iterating on gameplay mechanics, level design, and user interface elements based on player feedback and testing results.
* **Optimization**: Implementing performance optimizations and visual enhancements to ensure a smooth and immersive gameplay experience across different platforms.
* **Community Engagement**: Engaging with the player community to gather feedback, address concerns, and foster a supportive and active player base.
* **Continuous Improvement**: Striving for excellence by constantly refining and improving the game through updates, patches, and new content releases.

**Encouragement for Further Exploration and Improvement:**

As we conclude this Gun Game project, we encourage you to continue exploring the world of game development and seek opportunities for improvement and innovation. Whether it's experimenting with new gameplay mechanics, delving into advanced optimization techniques, or embarking on ambitious projects, the journey of game development is limitless and full of possibilities. Keep pushing the boundaries, honing your skills, and creating unforgettable gaming experiences that captivate players around the world.

Thank you for joining us on this exciting adventure in Gun Game development. We look forward to seeing where your creativity and passion for game development take you next!