**RPG Arena Documentation**

# **AR Integration with Unity AR Foundation and Google XR Code**

## **Overview:**

The game incorporates Augmented Reality (AR) features using Unity's AR Foundation and Google XR Code. This integration allows the game to project 3D game elements into the real world, creating an immersive and interactive experience for players.

## **AR Foundation in Unity**

### Introduction

AR Foundation is a cross-platform framework provided by Unity that enables the development of AR applications. It abstracts the underlying AR SDKs, such as ARCore (for Android) and ARKit (for iOS), allowing developers to create AR experiences that work seamlessly across multiple platforms.

### Key Features

* **Plane Detection:** AR Foundation provides plane detection capabilities, allowing the game to identify flat surfaces in the real world where game elements can be placed.
* **Anchor Points:** The framework supports anchor points, ensuring that virtual objects remain fixed in the real world, even as the user moves their device.
* **Light Estimation:** AR Foundation can estimate the lighting conditions of the real world, enabling more realistic rendering of virtual objects.
* **Session Management:** Manages AR sessions, handling the lifecycle and state transitions of AR experiences.

### **Implementation**

1. **Setup:** Install the AR Foundation package and ARCore/ARKit XR plugins in Unity.
2. **Configuration:** Configure AR settings in the Unity project, ensuring compatibility with both Android and iOS platforms.
3. **Plane Detection:** Implement plane detection to identify horizontal and vertical surfaces using ARPlaneManager.
4. **Object Placement:** Place game elements (e.g., enemies, obstacles) on detected planes using ARAnchorManager.
5. **Interaction:** Allow players to interact with AR elements, such as shooting enemies or navigating around real-world obstacles.

## **Google XR Code**

### Introduction

Google XR Code extends the capabilities of ARCore, providing additional features and functionalities for AR development.

### Key Features

* + **ARCore Extensions:** Enhanced features and tools that improve ARCore’s performance and capabilities.
  + **Cloud Anchors:** Allows sharing and persistence of AR experiences across devices, enabling multiplayer AR interactions.

### Implementation

1. **Setup:** Integrate Google XR Code with the Unity project.
2. **Cloud Anchors:** Implement cloud anchors to allow multiplayer experiences and persistent AR content.
3. **Extended Tracking:** Use ARCore extensions for improved tracking and stability of virtual objects.

## Gameplay Integration

### Endless Survival Mode

The AR integration introduces an endless survival mode where players must continuously fight against waves of enemies. The game leverages AR plane detection to place enemies and obstacles in the real world, creating an immersive and dynamic gameplay experience.

### Key Mechanics

* **Enemy Spawning:** Enemies spawn on detected planes and engage the player in combat.
* **Real-World Interaction:** Players navigate their physical environment to avoid or engage enemies, adding a layer of physical activity and immersion.
* **Resource Management:** Players must manage limited ammo and health packs placed in the AR environment, encouraging exploration and strategic planning.

### User Experience

* **Immersion:** The AR features enhance immersion by blending virtual game elements with the real world.
* **Interactivity:** Players interact with both the physical and virtual environments, creating a unique and engaging gameplay experience.
* **Replayability:** The endless survival mode and dynamic enemy placement ensure that no two gameplay sessions are the same.

# **Game Design Document (GDD)**

## **Game Title**

RPG Arena WebGL game using Unity Software.

## **Genre**

* Single Player
* Adventure
* 3rd Person

## **Game Elements**

* Exploration
* Combat (Fighting Monsters)

## **Player:**

* Single Player

## **Technical Form:**

* 3D with mesh characters and graphics

## **View:**

* 3rd Person Camera

## **Platform:**

* Windows, Linux, WebGL

## **Language:**

* C#

**Game's concept:**

You are a character with a rifle and limited ammo. Your objective is to clear the level without dying. Monsters will start chasing you when you are 8 tiles close to them. You can play this game using two strategies:

1. Kill all enemies, but there are limited bullets.
2. Play stealthily and avoid as many enemies as possible.

## **Gameplay Outline:**

The player navigates through various levels, each filled with monsters and obstacles. The player must manage their limited resources (ammo and health) and make strategic decisions on whether to engage or avoid enemies. The game ends when the player either clears the level or dies.

**Mechanic****s:**

**1. Character Movement Controls:** W/A/S/D or Arrow Keys: Move forward, left, backward, and right. Mouse: Control camera and aim and shoot.

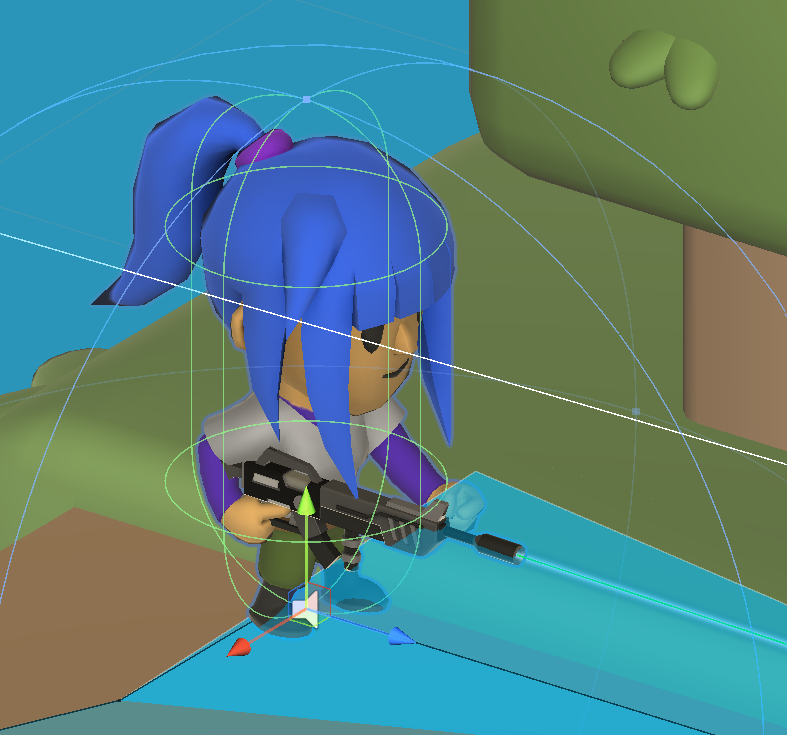
**2. Aiming and Shooting Aim:** The player uses the mouse to aim their rifle. Shoot: Left mouse button to shoot. Each shot reduces the bullet count.

**3. Health System Health Bar:** Displayed on the UI to show the player’s health. Damage: Health decreases when attacked by monsters. The game ends when health reaches zero.

**4. Stealth Mechanics Detection Radius:** Monsters

**Characters:**

1. **Player character:** Player character that user control to progress through the levels.

****

1. **Enemy characters:** These are the characters that will attack the player to kill when you enter into the particular radius of an enemy area.

****

**Level Design:**

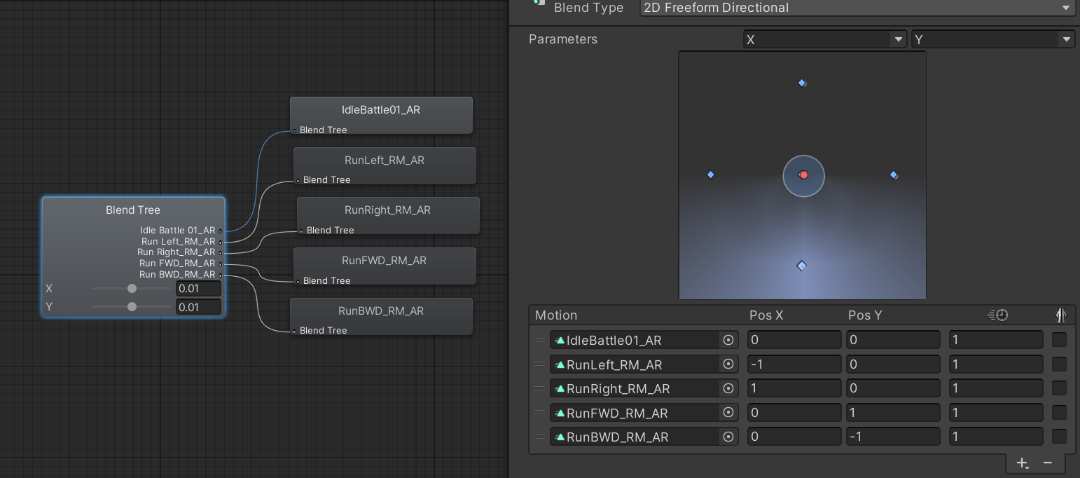
We have different types of tiles and props to build the User Interface (UI). I have used the transform positioning system to arrange these blocks to create the level.

**Art Style:**

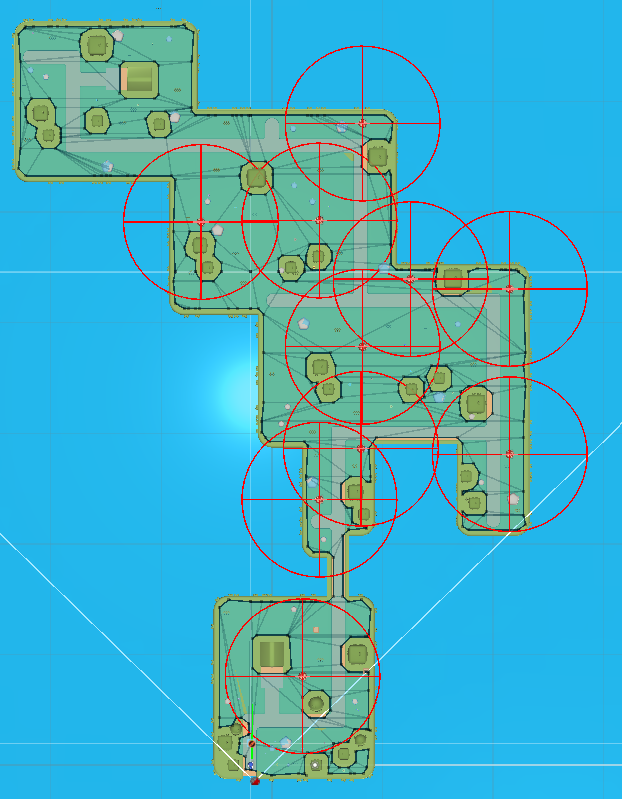
The game will feature a low poly cartoon style with bright and vibrant colors. This artistic choice aims to create a visually appealing and accessible experience that emphasizes clarity and readability while maintaining a playful and engaging atmosphere.

**Key Features**

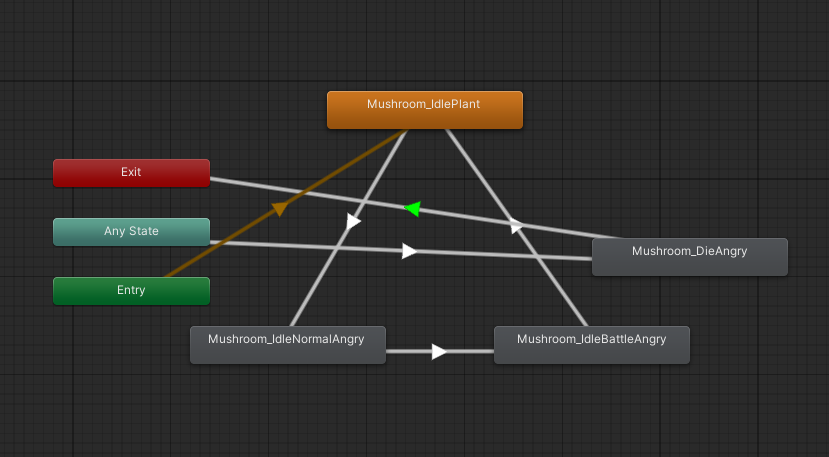
* Animation Controllers –



* Navigation Mesh (NavMesh) for AI pathfinding –



* 3rd Person Camera and Controller –



#### **User Interface:**

The user interface will be minimalistic, providing essential information such as health, ammo count, and objective markers. The UI will be created using Unity's UI pack.

### **Design Guidelines**

* Ensure smooth and responsive character controls.
* Design levels that encourage both combat and stealth gameplay.
* Implement a clear and simple user interface.

### **Game Design Definitions**

#### **Player Definition**

The player character is equipped with a rifle and limited ammunition. The player must navigate through levels, defeat monsters, and avoid death.

#### **Player Properties**

* **Health:** The player’s health decreases when attacked by monsters.
* **Turn Speed:** The speed at which the player can rotate and aim.
* **Bullets:** The number of bullets available for the rifle.

## 

## **Game Flowchart**

mathematica

Copy code

Start -> Level Select -> Level -> Game Over

### **Player Definition**

* **Character Model:** 3D mesh with animations
* **Weapon:** Rifle with limited ammunition

### **Player Properties**

* **Health:** Maximum of 100
* **Turn Speed:** Adjustable for smooth aiming
* **Bullets:** Limited, replenished at certain points

### **User Interface**

* Health Bar
* Ammo Count
* Time
* Score
* Enemy health

## **Art and Graphics**

* **Environment:** Detailed 3D environments with interactive elements
* **Characters:** High-quality 3D models with smooth animations
* **Effects:** Particle effects for smoke, sandbox
* **UI**: unity sample UI pack

## **Sound and Music**

* Background music for each level
* Sound effects for gunfire, monster growls, and environmental sounds

## **Iteration 1: Core Game Mechanics**

### Objectives

* Develop the fundamental mechanics of the game.
* Implement character movement and shooting without detailed character models.

### Key Features

* **Character Movement:** Basic controls for moving forward, backward, left, and right.
* **Shooting Mechanic:** Implement shooting mechanics with a simple placeholder for the rifle.
* **Camera:** Third-person camera following the character.
* **Basic Environment:** Simple low poly environment for testing movement and shooting.

### Outcomes

* Establish core movement and shooting mechanics.
* Basic environment to navigate and test core gameplay functionality.

## **Iteration 2: Added Animation and Graphics**

### Objectives

* Introduce animations and improve the visual appeal of the game.
* Replace placeholders with low poly character models and environmental assets.

### Key Features

* **Character Animations:** Walking, running, shooting, and idle animations.
* **Visual Improvements:** Low poly character models and bright, cartoon-style textures for the environment.
* **Enhanced Environment:** More detailed terrain, structures, and vegetation.

### Outcomes

* Characters and environments look more polished and visually appealing.
* Smooth and responsive character animations that enhance gameplay experience.

## **Iteration 3: Added Enemy AI and Navigation Mesh**

### Objectives

* Introduce enemy AI to create challenging gameplay.
* Implement a navigation mesh for realistic enemy movement.

### Key Features

* **Enemy AI:** Basic behavior patterns including patrolling and chasing the player when detected.
* **Navigation Mesh (NavMesh):** Pathfinding system to ensure enemies can navigate the environment effectively.
* **Enemy Animations:** Walking, attacking, and idle animations for enemies.

### Outcomes

* Enemies actively patrol and chase the player, adding challenge and excitement to the gameplay.
* Enemies navigate the environment smoothly using the NavMesh.

## **Iteration 4: Gun Rotation Lock**

### Objectives

* Restrict the gun's rotation to prevent unrealistic 360-degree rotation.
* Ensure the gun's aiming mechanism is functional and visually coherent.

### Key Features

* **Gun Rotation Lock:** Restrict the gun’s rotation to 345 to 10 degrees to prevent it from rotating a full 360 degrees.
* **Aiming Mechanics:** Fine-tune aiming to ensure it remains accurate and responsive within the restricted range.

### Outcomes

* Realistic and coherent gun aiming mechanics.
* Prevents unrealistic and potentially disorienting 360-degree gun rotation.

## **Summary**

Through these iterations, the game evolves from a basic prototype into a more refined and visually appealing experience with complex mechanics and AI behavior. Each iteration focuses on adding significant features and polishing existing ones to create a cohesive and engaging game.

### Future Iterations

* **Iteration 5:** Implement health and resource management systems.
* **Iteration 6:** Introduce stealth mechanics and additional environmental interactions.
* **Iteration 7:** Optimize performance and polish UI elements.
* **Iteration 8:** Conduct playtesting and finalize gameplay balancing and bug fixes.

By following this iterative development process, the game will progressively improve in terms of mechanics, visuals, and overall player experience.

**Feedbacks:**

* **Player 1:** The game's graphics are impressive. The attention to detail in the environment design creates an immersive experience. The character models and animations are smooth, though there could be more variation in character designs to enhance visual diversity.
* **Player 2:** The game is very smooth to play and explore, it just needs little bit polishing more. If we talk about game levels, they are designed very thoughtfully. Camera movements can be improved.

**Game Play uploaded link** - https://play.unity.com/en/games/89a05f4a-c160-4728-b073-f47643349768/rpg-arena