**Project Documentation: Survivor**

### **Introduction**

Survivor is a top-down 2D survival game developed for Windows PC. Players face relentless waves of enemies in procedurally generated fields, collecting mana orbs to level up and unlock abilities. Escalating difficulty encourages adaptation and persistence as players combine weapons and skills to survive.

**Key Features**

* Endless procedural map generation.
* Projectile and melee combat systems.
* Leveling system with meaningful upgrades.
* Immersive pixel art and sound design.

Survivor takes inspiration from *Vampire Survivors* and *Magic Survival*, focusing on growth through challenge and adversity.

### **Setup Instructions**

1. **Prerequisites:**
   * Unity 2022.3 or later.
   * Visual Studio Code (or equivalent IDE).
   * Unity's 2D Game Kit.
2. **Installation:**
   * Clone or download the repository.
   * Open the project in Unity.
   * Import required packages via Unity Package Manager.
   * Press **Play** in Unity Editor to launch the game.

### 

### 

### 

### 

### **Gameplay Overview**

In Survivor, players collect mana orbs dropped by enemies to level up and choose new abilities or weapons. Combat involves projectile weapons (e.g., knives) and melee options (e.g., garlic aura), all within procedurally generated fields. Random item drops enhance replayability, and escalating challenges keep gameplay engaging.

### **Code Architecture**

The codebase is modular, with components organized into categories like **Player**, **Enemies**, **Weapons**, **Map**, and **PickUps**.

#### **Player Systems**

* **Movement:** PlayerMovement.cs handles player movement and direction.
* **Stats and Leveling:** PlayerStats.cs tracks attributes (health, speed, experience) and manages leveling logic.
* **Collection:** PlayerCollector.cs manages interaction with collectible items using the ICollectable interface.

#### **Enemy Systems**

* **Behavior:** EnemyMovement.cs drives enemy pursuit of the player.
* **Stats:** EnemyStats.cs tracks health, damage, and death behavior.
* **Configuration:** EnemyScriptableObject.cs defines reusable enemy data like speed and damage.

#### **Weapons**

* **Controller:** WeaponController.cs handles weapon cooldowns and attacks.
* **Projectile Weapons:** ProjectileWeaponBehaviour.cs manages projectile movement, damage, and pierce logic.
* **Melee Weapons:** MeleeWeaponBehaviour.cs governs close-range weapons like garlic aura.
* **Specialized Behavior:** Scripts like GarlicBehaviour.cs and KnifeBehaviour.cs provide unique weapon functionalities.
* **Weapon Configuration:** WeaponScriptableObject.cs defines stats such as damage, speed, and cooldowns.

#### 

#### 

#### **Map and Procedural Generation**

* **Chunk Management:** MapController.cs generates and optimizes map chunks.
* **Terrain Interaction:** Map elements interact dynamically with player movement.

#### **Items and Drops**

* **Collectibles:** ExperienceGems.cs grants experience to players upon collection.
* **Drop Rates:** DropRateManager.cs determines item drops upon enemy destruction based on probabilities.

### **Development Notes**

**Current State**

* Core mechanics are complete, including procedural generation, combat systems, and leveling.
* Placeholder assets are used for visuals and enemies.

**Known Issues**

* Optimization is needed for large enemy counts.
* Limited enemy types; additional behaviors required.
* UI integration for player and enemy health is incomplete.

**Planned Enhancements**

* New weapon types and ability synergies.
* Environmental hazards and obstacles.
* Advanced AI for enemies.
* Visual and audio feedback for key events (e.g., leveling up, item collection).

### 

### 

### 

### 

### 

### 

### **Key Interactions**

* **Player and Enemies:** Enemies pursue the player and deal damage upon collision, while weapons counter these threats.
* **Collectibles and Items:** Collectibles (like experience gems) increase experience and trigger level-up logic.
* **Weapons and Terrain:** Procedural terrain dynamically supports weapon mechanics and enemy spawns.

This documentation provides a comprehensive overview of the game’s structure and functionality, catering to both development and future enhancements. Further iterations will refine gameplay systems and introduce additional content.