Report

Game Designer Instructions

The game designer can change the values in the game by opening the Assets/ScriptableObjects/EnemyDataDefault scriptable object file.

Values:

- Moving saw speed: horizontal speed of the saw
- Moving saw min level: the lowest level at which the saw can be spawned
- Moving saw spawn chance: chance that the saw can be spawned. It is between 0-1. A higher value increases the chance of being spawned.
- Falling rock speed: horizontally falling speed of the rock
- Falling rock min level: the lowest level at which the rock can be spawned
- Falling rock spawn chance: chance that the rock can be spawned. It is between 0-1. A higher value increases the chance of being spawned.

Project

In this project, the manager-behaviour-action structure was used. The objects to be instantiated are under the Assets/Prefab folder. A ragdoll character is used for the player. This character is connected to a gameobject named Player with Hinge Joint. This joint is where the character's hand is. In this way, the character swings freely while holding onto the rock from his hand. If this oscillation is too much, the character can also die by hitting the static rocks on the side. Apart from that, it can be killed by two moving enemies (moving saw, falling rock). There are three scenes in the game.

Class Explanations

PlayerBehaviour: Performs the movement of the player. A lerp is used to make the movement smooth. The move action is coded in the LateUpdate function. Clicking on the next target is also detected in this behavior. The NextTarget function of the GameManager is called when the click is detected.

CameraBehaviour: It allows the camera to follow the player. It is coded into LateUpdate.

GameManager: It is a singleton class where the main logic of the game is coded. The NextTarget function is called when the user clicks on the next target. This function calls the CalculateTargetPos function, which determines the new target point. It calls the corresponding functions on the SpawnManager to create objects around the specified target point.

SpawnerManager: Singleton class responsible for object creation. Takes spawner scripts as a reference. It takes the enemy spawner scripts as a list in the EnemySpawnerBase class type from which they are derived. Thanks to this polymophisim structure, it randomly creates one of the enemies that will be produced at the level reached by the user at a random chance.

SpawnerBase: All classes that derive from this abstract class must take a prefab object as a reference. Deriving classes override the InstantiateObj function to specify the position and rotation at which the object will be created. Object pooling is written in this base class. In this way, the code is not duplicated. All created objects are kept in a list and each time a new object is generated, if the maximum number is exceeded, the oldest object is deleted.

EnemySpawnerBase: Since all enemy objects to be created will have minSpawnLevel and spawnChance fields, this abstruct class is derived from SpawnerBase class. Enemy spawners must derive from this abstract class.

BaseEnemyBehaviour: It is the main abstruct class to be found in Enemy objects. With the OnCollisionEnter function, it detects a collision with the player object and switches to the endgame scene. Since all enemy objects implement this behavior, the base abstruct class is written.

BaseMovingEnemyBehaviour: BaseEnemyBehaviour is the main class of mobile enemies derived from the abstruct class. There is a Speed field and all derived classes must implement the Move function.

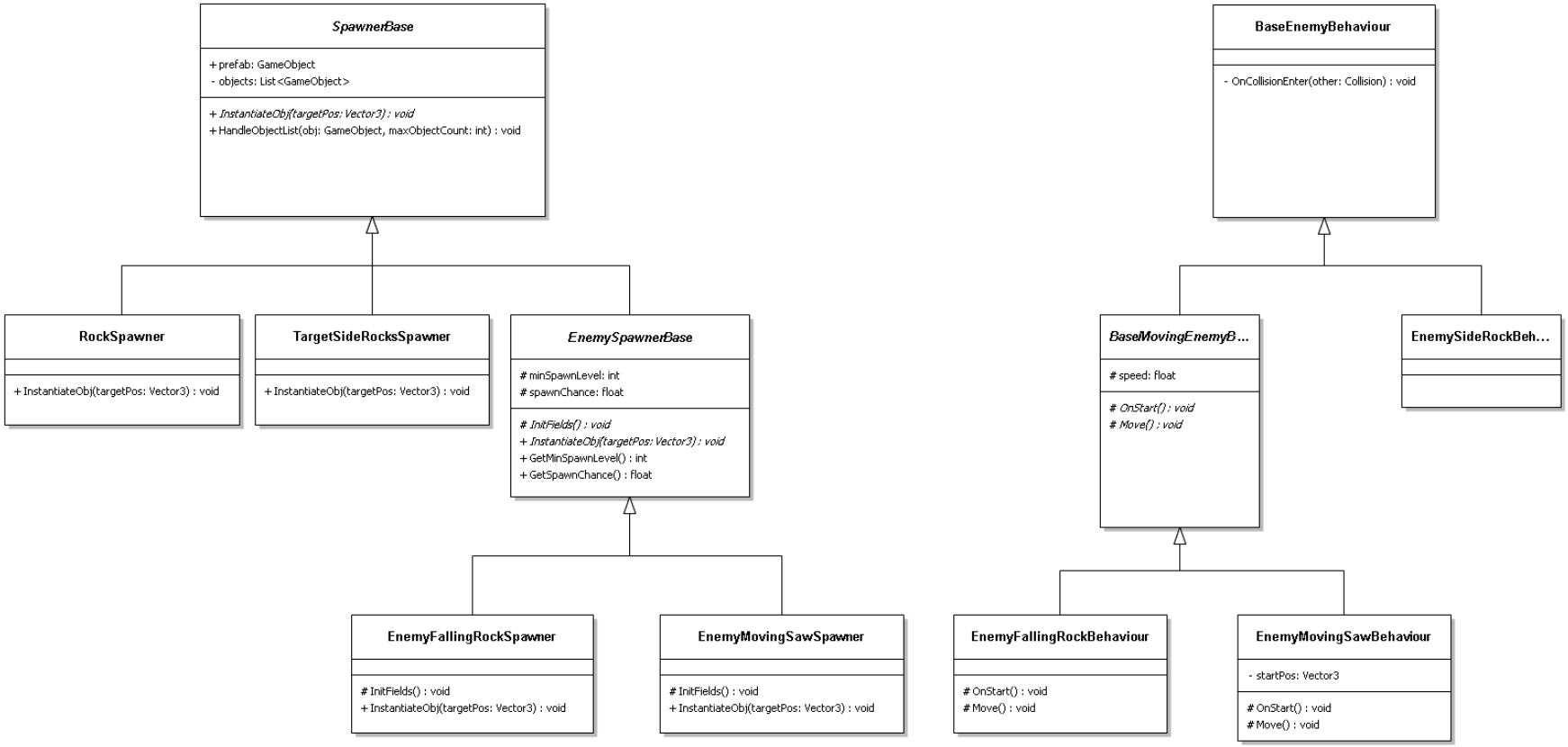
GameDataManager: It takes the scriptable object containing the values used in the game as a reference. It is a singleton class so that all scripts that will use these values can access them.

MenuTransitionManager: It is a singleton class that provides transition between scenes. Occurs in the Menu scene and because it is DontDestroyOnLoad, it is not destroyed when switching between scenes. Scenes to be passed can be added as an additive by creating an overlay scene in the future. In this way, the scene transition is animated.

ActionScripts: These are the scripts in which the functions to be called in the UI event functions are written.

Diagram

Diagram is in the next page and also available as png file.


SpawnerBase
+ prefab: GameObject
- objects: List<GameObject>
+ InstantiateObj(targetPos: Vector3) : void
+ HandleObjectList(obj: GameObject, maxObjectCount: int) : void
RockSpawner
+ InstantiateObj(targetPos: Vector3) : void
TargetSideRocksSpawner
+ InstantiateObj(targetPos: Vector3) : void
EnemySpawnerBase
minSpawnLevel: int
spawnChance: float
InitFields() : void
+ InstantiateObj(targetPos: Vector3) : void
+ GetMinSpawnLevel() : int
+ GetSpawnChance() : float
EnemyFallingRockSpawner
InitFields() : void
+ InstantiateObj(targetPos: Vector3) : void
EnemyMovingSawSpawner
InitFields() : void
+ InstantiateObj(targetPos: Vector3) : void
BaseEnemyBehaviour
- OnCollisionEnter(other: Collision) : void
BaseMovingEnemyB...
speed: float
OnStart() : void
Move() : void
EnemySideRockBeh...
EnemyFallingRockBehaviour
OnStart() : void
Move() : void
EnemyMovingSawBehaviour
- startPos: Vector3
OnStart() : void
Move() : void