Endless Runner 2D Kit

Table of contents

Enemy System

Enemy types

General Rules:

- (_main) blueprint as a parent while the actors in the Viewport are its children.
- Make the sprite collisions disabled + make the player not be able to step on them (the collision component is the main one to interact with other Actors)
- If spawned individually toggle Move on and set speed
- Enemies get destroyed after they reach a certain coordinate
- Each enemy can do x damage

Patrol Enemy

Each blueprint child has its own script for choosing its desired patrol editable targets

- Move toggle (recommend enabling this is you spawn it individually)
    - Speed + Direction
- Death trigger
    - Enemy dies if you are falling and are at greater than ½ character height + Z position of the death trigger
- On collision with the character a certain amount of life is subtracted

Projectile Enemy

Usually faster than the rest of the enemy types.

- Move toggle (recommend enabling this is you spawn it individually)
    - Speed + Direction
- On collision with the character a certain amount of life is subtracted

Simple/Static Enemy

Ground level obstacles the player avoids.

- Move toggle (recommend enabling this is you spawn it individually)
    - Speed + Direction
- Death trigger
    - Enemy dies if you are falling and are at greater than ½ character height + Z position of the death trigger
- On collision with the character a certain amount of life is subtracted

Special Enemy

It's a projectile like enemy that has a random chance of stopping in mid air at a random set of coordinates and charges the players' last position. The player can move away from the last position and the enemy "dies" when it reaches its target.

- Move toggle (recommend enabling this is you spawn it individually)
    - Speed + Direction
- Death trigger
    - Enemy dies if you are falling and are at greater than ½ character height + Z position of the death trigger
- On collision with the character a certain amount of life is subtracted
- Attack trigger
    - An array with X axis coordinates from which a random value is picked
    - Enemy changes color and a spark effect appears

Pickup system

Rewards

General Rules:

- (_main) blueprint as a parent while the actors in the Viewport are its children.
- Make the sprite collisions disabled + make the player not be able to step on them (the collision component is the main one to interact with other Actors)
- If spawned individually toggle Move on and set speed
- Each reward can give x amount of coins
- If the player picked up the coin magnet powerup and the player is in range, the reward will home in on the player and move towards his position

Reward Types

- Low - 1 coin
- Medium - 2 coins
- High -3 coins

Power-Ups

General Rules:

- Make the sprite collisions disabled + make the player not be able to step on them (the collision component is the main one to interact with other Actors)
- If spawned individually toggle Move on and set speed
- Powerups get destroyed after they reach a certain coordinate
- Powerups have a common parent blueprint (_main) that controls only their movement, which is a shared behavior. Each blueprint has its own functionality therefore separate behaviors.
- Each behavior communicates with BP_GameMode which pretty much handles every aspect of the game

Powerup types

- Coin Magnet
    - Pulls reward type actors towards the player for a limited amount of time
    - When it collides with the player it activates the event from the BP_GameMode through an interface blueprint
    - The BP_GameMode communicates with the Player and the individual reward blueprints
- Health

    - When it collides with the player it activates the event from the BP_GameMode through an interface blueprint, adding X amount of health to the Player pool
    - If your health is full you will get extra points
- Invincibility
    - Disables the Player from taking damage from enemies
    - When it collides with the player it activates the event from the BP_GameMode through an interface blueprint
    - If it's active and the Player collides with an enemy you will get extra score points
- Reward multiplier
    - Gives extra an added extra reward amount when the Player picks up a coin
    - When it collides with the player it activates the event from the BP_GameMode through an interface blueprint
- Hover
    - When picked up the player can hover by holding down jump
    - If the powerup is already active you get extra score
    - When its active the environment moves faster to give the impression of extra speed
    - The spawn chances from the Primary or Secondary spawn points are changed to favour more rewards
        - (Note: Make sure that the changed spawn chances are a sum on 100 from BP_GameMode and they reflect the ones in the spawn point manipulated + make sure its reverted back to the old settings)

Holders

Place the desired child blueprints in here and form patterns of power-ups, rewards or enemies blueprints
Consider this as a parent and its content blueprints as children; if the parent moves so will the children
General Rules:

- The holder moves towards the player so that he can avoid or not, the contents of the pattern.
- If it reaches a certain point the holder gets destroyed
- If it reaches a certain point the spawn event gets triggered
- Any kind of actor blueprint can be dragged in the holders
- If a blueprint component has its move toggled on, it will move faster than the rest of the blueprints from the pattern
- The principle behind holders are similar, therefore you can make different combinations of rewards ,enemies or powerups

Holder Types

Combos enemies

- A combination of enemies

Patrol enemies

- Make separate patrol enemies blueprints with their own patrol patterns before you drag them in. Take care where you spawn the holder plus the patrolling pattern of the actor blueprint

Simple enemies

- A simple combination of ground level enemies

Reward holder

- Combinations of reward blueprints in a holder

Spawn point system

Spawn points are triggered when a certain Holder reaches a certain destination. If holder A reaches point X, it will trigger an event dispatcher in BP_GameMode, which in turn is “heard” by the spawn points.
General Rules:

- You can set a percentage for each type of actors/holders you want to spawn (explained below!)
- Make sure that the chances to spawn x,y,z… actors/holders is a total of **100% (!)**. If it's not 100% then the spawn points will not spawn and you will get a log message.
- An initial reward holder will be spawned at first after the game begins so that the entire spawn system will be kickstarted
- An initial value is picked from a max of 100 (%) and based on this number you will get your actors/holders spawned
- Each spawn points has its set of Actors/Holders arrays that are called from specific functions. Make sure that the arrays are filled and there is **no (none)** element being called (!)
- The spawn points choose a type of array (%) and a random element in those arrays
- If you want combinations of actor types, you must create a separate holder for each spawned actor, because the arrays in the functions need to be set to a certain class (ex: spawn from reward holder class = only rewards in that specific class; spawn a

combination of reward + enemy actors = you must create a separate holder and set the array to that holder class)

(!) How (%) chance to spawn works:
To avoid overlapping chances (%) you must compare generated numbers between each other. For example if X has a 40 % chance to spawn and Y has a 60 % chance to spawn, if you not compare the generated numbers between each other, you will get both X and Y spawned, because a 60 value picks whatever is below it, including the 40. So for each actor added to the pool you must compare the totals of its previous added actors and the total of all the actors, so you can situate it between a min and a max that does not overlap with the rest of the spawn values.

1. if generated number <= x then do THING 1
2. if generated number > x && <=x+y then do THING 2
3. if generated number > x+y && <=x+y+z then do THING 3
4. if generated number > x+y+z && <=x+y+z+w then do THING 4

Spawn point types

The primary type is for spawning “generic” holders while the secondary spawner is used to spawn special actors (power-ups, special enemy etc. (actors must have Move variable toggled on (!))

Primary spawner

- Spawns a reward by default so it will trigger the entire system
- Has holders composed of actos spawned
- It’s triggered when the holder reaches a certain point
- Has a chance to spawn holders separately (ex: x=20%, y = 70%, y=10% => total 100%)
- Activated through the BP_GameMode

Secondary spawner

- When a holder reaches its spawning destination coordinates, it will always trigger the Primary Spawner and will have a chance (%) to trigger the secondary spawner which is a special actor spawner.
- Has a chance to spawn actors separately (ex: x=20%, y = 70%, y=10% => total 100%)
- Activated through the BP_GameMode

Game Controller

Handled in BP_GameMode, is the main communicator between blueprints in the game. All the blueprints has a certain link with this main blueprint that stores different types of data.

I holds different systems together through the use of interfaces. Each actor (reward, powerup, enemy) when it collides with the player sets off its respective interface event that communicates with this blueprint, changing values or triggering other events in turn.

Score System - GameMode blueprint

- Each x seconds the score gets incremented by a certain amount
- The score starts after the player presses space (Start widget)
- Special cases exist when the score bigger amounts than the one incremented (see below)

Reward System - GameMode blueprint

- When a reward actor collides with the player you will get an x amount through an interface event
- If you have a reward multiplier powerup active you will get an extra reward amount added to the reward given by the coins you pick up
- The reward amount is stored in a variable which updates the save game actor

Powerup system - GameMode blueprint

Reward multiplier

- When the player collides with this powerup, the interface event is triggered and the multiplier value amount is communicated in this event
- The interface also has an editable timer until it resets
- When this is active, the default reward amount is multiplied by this value

Coin magnet

- When the player collides with this powerup, the interface event is triggered and the CoinMagnetColi compunent in the Player blueprint gets acivated. When a reward actor enters this collider, it will move towards the current player position (homing)
- The interface also has an editable timer until it resets

Invincibility system

- When the player collides with this powerup, the interface event is triggered and the damage interface is disabled. This way the player does not take any damage from the enemies it hits
- The interface also has an editable timer until it resets
- When active, the player gets extra score when it collides with enemies

Health system - death, hurt, ui -

- When the player collides with this powerup, the interface event is triggered
- You can set a Maximum amount of health points to the pool
- The current health is changeable and affected by the colliding enemies and is displayed in the UI
- When you run out of health (current) the player dies which turn off all
- When you get hit by an enemy the current health drops by a certain point and set off the Hurt event (if you have 1 health left, the Death event is triggered and Hurt is disabled to avoid conflicts)
- You can set the maximum health and the current health in the blueprint
- If you pick up a health powerup and your health is full you will get extra score

Damage system - GameMode blueprint

- When the player collides with an enemy, the interface event is triggered and the amount of damage set in the enemy blueprint is communicated and subtracted from the current health of the player
- The player cannot be damaged while Invulnerability powerup is active

Spawn System - GameMode blueprint

When a holder reaches its spawn event trigger point, it will communicate with the BP_Gamemode blueprint which will trigger two even dispatchers. This in turn causes the events to trigger in the spawning points because they are actively “listening” for these events

Save game/data system - GameMode blueprint

The data saved refers to the amount of coins picked up and the score the player has.
The data is saved in BP_SaveGameData, which contains a variable that holds the coins and an array of **10** elements that would hold the score. The idea is that these get saved in variables in the BP_GameMode blueprint which in turn are saved in the save game slot. (!) Make sure that the save game score array has the same number as the BP_GameMode score array (!)

- Coins amount get actively updated while the score is saved once per death
- The data is saved in a slot called ”SaveSlot”, which is saved locally in a file on your drive
- When the game starts the save slot is loaded if found, and if it’s not found an empty save game slot is created.
- The save game slot can be cleared and deleted (Score Widget)
- When a coin is picked up and the reward interface is triggered, the reward amount is sent to the save game slot instantly adding to the existing save game coin variable

- When the player dies the score is saved in the save game slot. First the score is stored locally in a variable which in turn changes the values of the variable in the save game slot.
- The score is used in a local score list (Score Widget)
- The new score is added to an array (adding elements to an array expands the array!). To counter the expanding array and move the data from n to n+1 each time, the last index of the array is always removed so it will make way for new data on the top. This way what was stored in the first index moves to the second and so on and if the array surpasses its length (10) the last index (11) gets removed.

Difficulty System - GameMode blueprint

- The system starts with a delay and increments every 60 sec (editable)
- Each iteration of the multiplier changes the values (%) of each of the spawned elements (make sure it's 100%!)
- When it reaches its max the reward blueprints give you more coins and the player takes more damage (editable)
- The number of iterations is dependant on an array, when it reaches its last index the difficulty reaches its peak

User interface

Start

Character idle dialogue

The character "talks" to you from a chosen array of texts. The texts are picked randomly and are displayed at certain intervals between each other

Game start

- The game starts when the user presses SPACE and exists when the user presses ESC
- By pressing space you start the main systems in the BP_GameMode blueprint and disable the character dialogue
- The first spawn will be triggered from here
- The Ingame interface is displayed from here
- The cursor is enabled from here so you can click pause in the Ingame menu

InGame(HUD)

- The current score, coins and health are displayed here
- The pause menu is accessible from here by pressing the pause button

Pause

- Resume takes you back to the game which has been paused
- Restart reloads the level entirely (keeps the save game)
- Score takes you to the scores menu
- Quit stops the game
- The current score is displayed here at the top of the menu

Score

This is where the coins and scores are displayed. These are taken from the save game slot

- There are score 10 entries (save game array size) each with its own function getting a specific index of the save game score array
- When a score is saved (added) it will be displayed as the first index pushing the rest down creating a cascading effect for your scores (n+1)
- You total numbers of coins picked up ever is also displayed here. Its retreived from the coin variable in the save game slot
- When the save game slot is empty everything is set to 0 from the BP_GameMode blueprint
- You can reset your scores from here which clears the save game array, setting everything to 0. This also restarts the game
- You can delete your save game slot, restarting your game. A new save game slot is created when none is found, in BP_GameMode

Text Feedback

Actions like Difficulty increase and powerup pickups trigger an event in the BP_Game mode that displays a certain text in the Hud_TextDisplay Widget.

- The text is displayed for each action separately and it can be changed in BP_GameMode
- The text in the widget as a simple move + fade out animation for effect

Environment - Looping, parallaxing and texture panning

“Ground”

The “ground” blueprint that handles this is called **BP_GroundMove_Sprites_Main**. The blueprint is based on a set of mobile elements, that move from an initial point to a targeted

coordinate on X axis, at a certain speed. When they reach their individual targeted coordinate they get instantly moved to their initial position.

The system would allow you to do this because what you do is just add elements to an array that controls the parallaxing.

- In the EnvironmentROOT component group you will find the tilemaps responsible for the “ground” part of the environment. These components are added to an array and the resulting elements’ starting coordinates are then added to another array
- Each element of the array is offset every tick by an individual speed amount, a general multiplier and a direction (-X axis)
- Each element of the array has its own destination stored in an array, and if one element arrives to that destination it will get sent back to its starting coordinate (Loop effect)
- (!) Make sure that the destination + speed + strat coordination arrays match the indexes from the tilemaps array (Example: tilemap index 1 > speed index 1….)
- Each tilemap can be moved at a separate speed, meaning that it will create a parallaxing effect (illusion of depth and movement in 2d)

Background

The background is made from a plane with a moving texture on it. The texture is added in MAT_BackgroundLoop material, to which a panner offset effect is added.

Player Character

The player character is created when the game starts from the PlayerStart spawnpoint. The player can jump N times when space is pressed (double jump). When you press the spacebar the second time, the player gets launched upwards by a certain amount.
The character has a big collision box (CoinMagnetColi) which is part of the Coin Magnet powerup. If this is enabled, coins will be attracted by the Player.

Player death slow motion

When the player’s health reaches 0 the game will slightly slow down for extra effect

Character Selection

A separate widget has been created for this called UI_CharacterSelect. You must first chose which image (texture) to display in the UI.
When you click on a chosen character a function is triggered which saves an array containing the character animations into the Save Game blueprint. Each index has a designated animation which is consistent in all blueprints (0=run; 1=Jump; 2=hurt; 3=death).

The save game array has a chosen default which will be displayed the first time you start the game.
Once you chosen a character the character select widget will not be displayed, making the character change available from the Pause menu.
You can delete the save game file in the Score system which will make the Character Selection screen reappear at the start of the game.

Level Progression

This is indicated by a progress bar from UI_LevelProgress that is activated from UI_Start widget.
The progress bar fills according to an incrementation of the Event Tick Delta time divided by the length of the level you wish to achieve (SecondsToNextLevel). This returns a percentage of the bar. When the bar is full the environment GFX changes (you can do something else if you want).
The sprites are changed in BP_GroundMove_Sprites_Main for each level iteration.
When you want to change the level make sure you increment NextLevel from UI_LevelProgress add another pin to the switch (note: in BP_GameMode the switch must also be changed to reflect this)
In BP_GameMode you can add new functions that change the environment to your needs.