Technical Design Document for GravityGame

31186408 Jiangye Song

Contents

Project Overview

Core Mechanic

The GravityGame is a puzzle platform game, and it is inspired by the game "vvvvvv". The core mechanic of this game is that the gravity of the character will flip when character jumped.

Unlike a conventional platform jumping game, player cannot perform a small jump to get to another platform in this game. This creates a unique and challenging gameplay experience that requires the player to think creatively and strategically about how to navigate through the levels.

As players exploring the tutorial levels, they will discover different item and components in the game. The check points, crowns, green cube, blue platforms, red cube or bullet, SpaceStation and yellow blocks. These will make the game more challenging and interesting.

Overall, the core gameplay mechanic in GravityGame is a simple and easy to understand but creates a unique and engaging gameplay experience. The gravity flip mechanic and the other gameplay elements work together to challenge and reward the player.

Target Platform

Controls are essential in GravityGame, it requires both strategic thinking and precise timing. Player must move to a correct direction quickly while also considering the effects of gravity on their movement. In this case, the best platform for this game should be Personal Computers and Consoles.

On a PC or Console, player should have access to a keyboard & mouse or a controller. Player can control flip, move with keyboard or joystick & button, and move their camera with a mouse or another joystick.

Larger screen size is required for player to view the map clearly, far, and quickly. It will be hard if player does not have enough response time when they encountered different element in the game.

While the game could potentially be developed for mobile devices, the experience of the game will decrease. Player will need a finger from one hand to control their movement and a finger from another hand for controlling the camera and jump the same time. The problem is the screen size of a mobile phone is usually smaller than a PC or a Console (TV), means that player needs to move the camera more frequently than other platforms. This would result in low response time and would make the game much more difficult to play and potentially frustrating for the player.

Game Mechanic

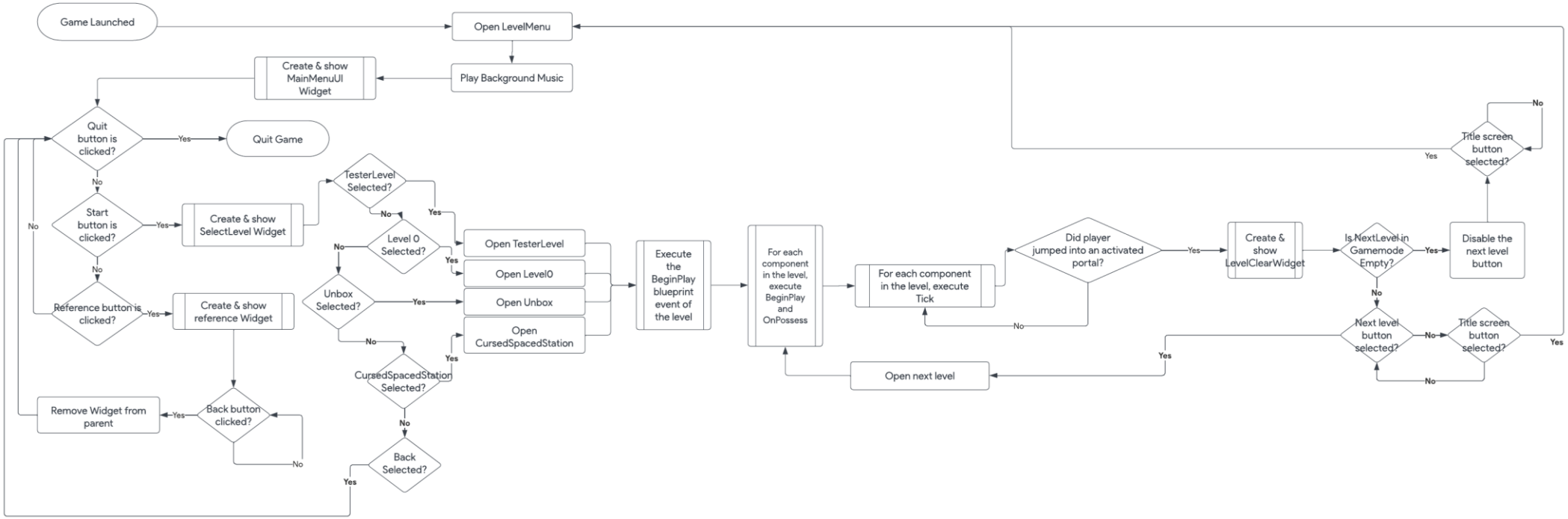
This section will detailly discuss all the mechanics in GravityGame. It will use flowchart, UML diagram and sequential to explain the gameplay, created classes, movement, and core mechanics.

Gameplay Flowchart

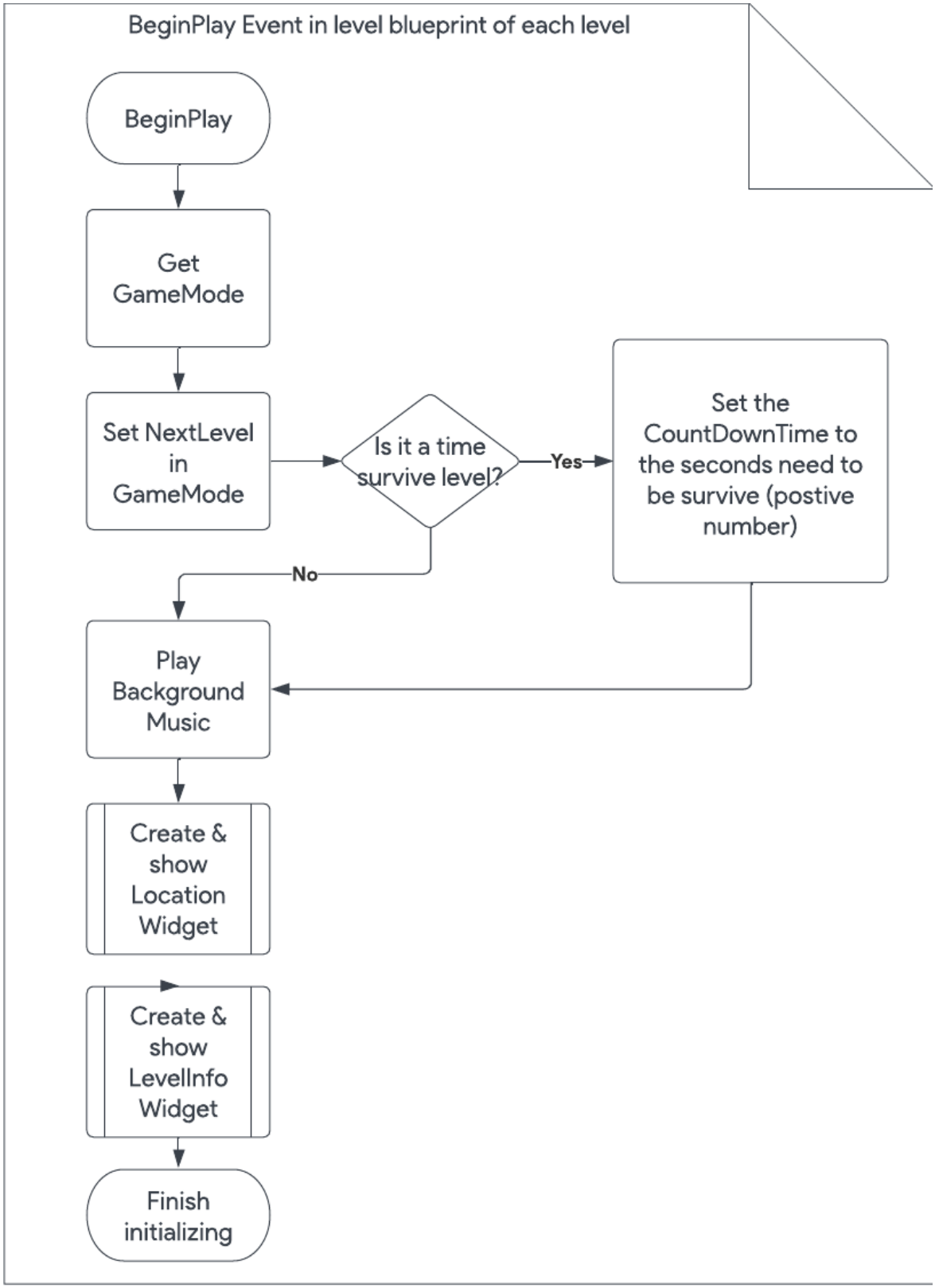
This flowchart is created via Lucidchart.

(Turn over for previews)

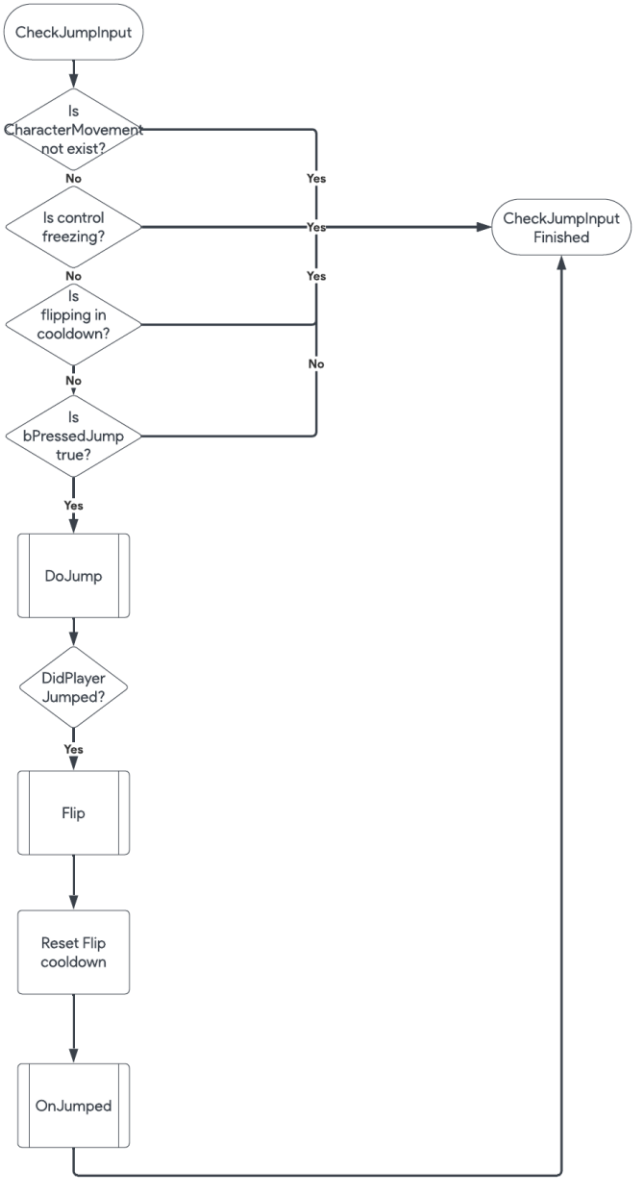
Overall Flowchart

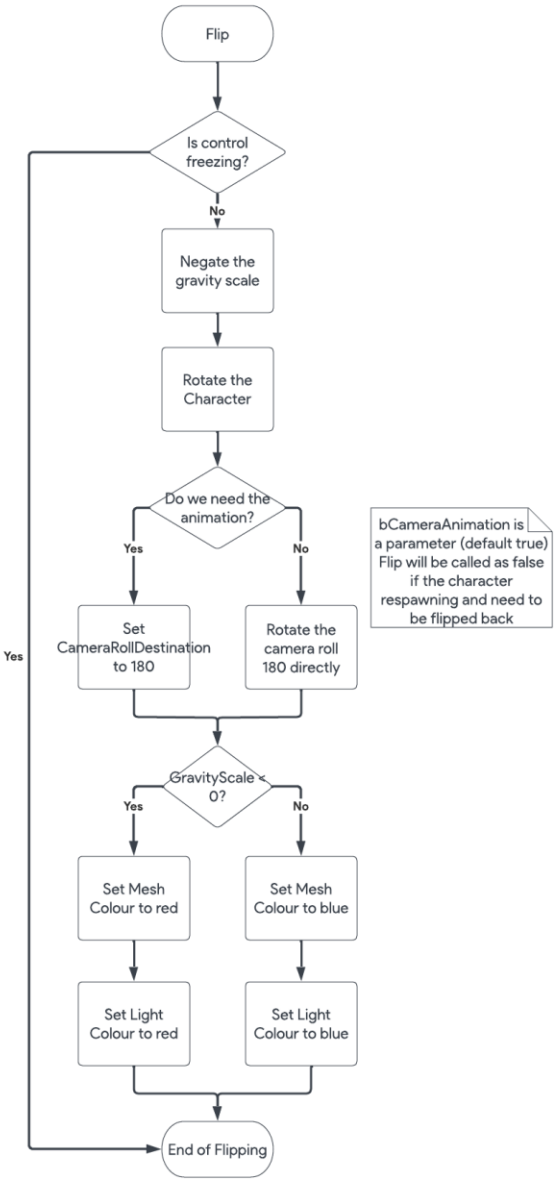


Level BeginPlay Event Flowchart



Jump and Flip Flowchart





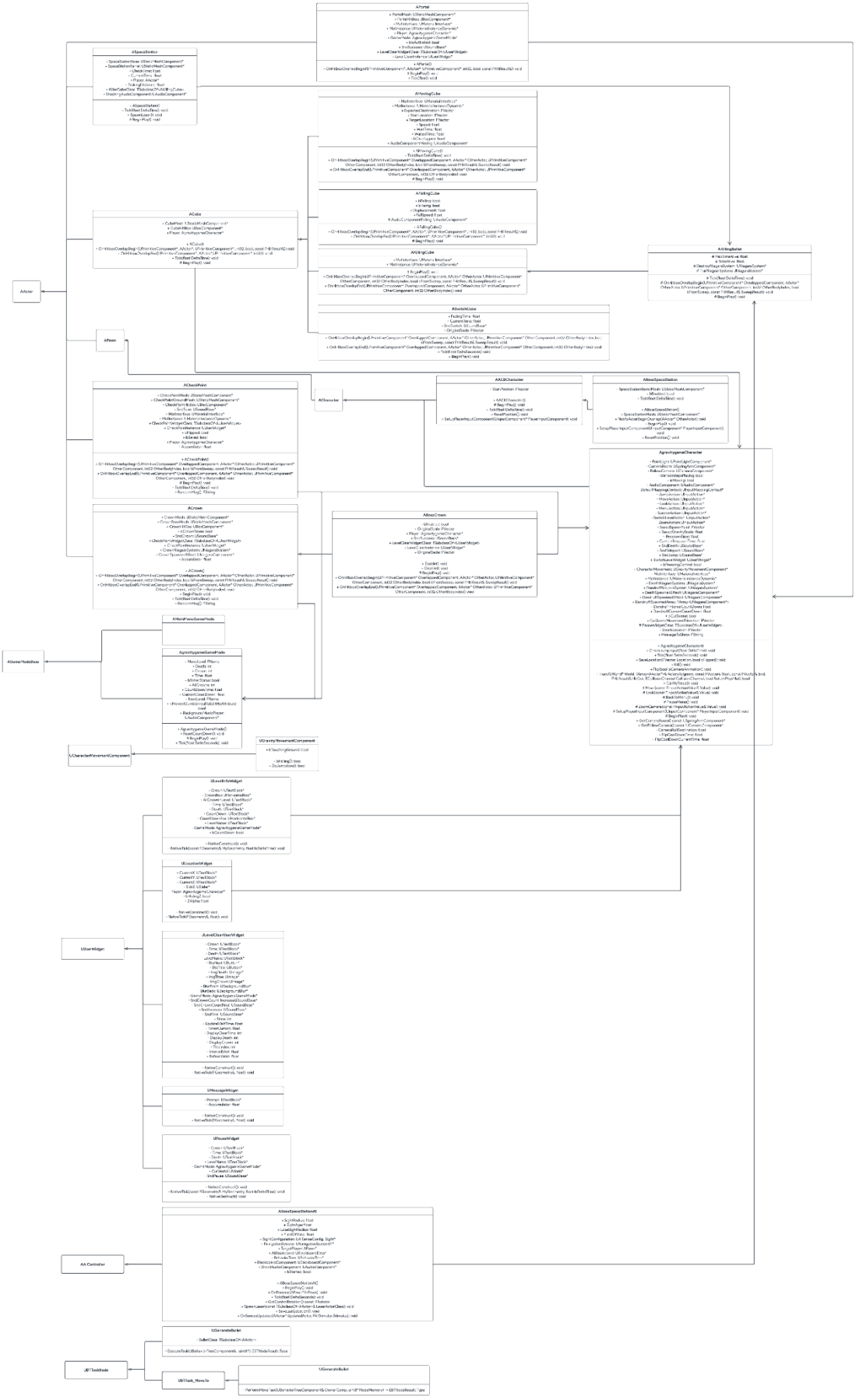
UML Diagram

As the unreal function name is too long, this UML uses direction from left to right rather than from top to down.

This game used the template of Third Person as a starting point. Classes provided by unreal and does not contain any modification will not show its attributes and functions in this UML, shows as a simple class shape.

This diagram is created via Lucidchart.

(Turn over for preview)



Movement Mechanics

Movement is essential in GravityGame. The movement mechanics does not change much in compared to original Unreal Engine Third Person character movement mechanics. Character can move left, right, forward, backward as usual. The character will be blocked if there's a cube is a higher (at least 46 higher) than the Z axis of the block cube that character stands on. When character is no longer steps on cube, they will keep falling until they fall to a cube again.

The initial gravity scale was modified to 0.5, this is to allow player to have more response time when they are falling.

The max walk speed is 800.0 cm/s, to allow player to have better control for their character. The friction of the character movement has been set to a huge value to avoid the Inertia. The mass of the character is 0 to avoid gravity acceleration. These can resolve unnecessary and unintentional difficulty of the game.

When character is upside down, the character can stand and walk on the ceiling as they usually do on the ground. Moving speed, friction and mass is the same, but the gravity scale will be -0.5 to allow character fall upwards.

The jump velocity is not useful in this game, as the character will fall to another direction of their original gravity. However, it must have a value that greater than 1 to allow the character shift to falling state rather than keep the walking state.

Controls

The control does not change in compared to original Unreal Engine Third Person character default control. Player can use their keyboard & mouse or a controller to move left, right, forward, backward, jump and look.

The detailed binding shows as follows.

Mapping	Action	Description	Control Binding	Modifiers
Walking Mapping	Move Forward	Used to move the character forward relative to the camera direction	W Up	ForwardDirection
	Move Left	Used to move the character left relative to the camera direction	A Left	RightDirection, Negate
	Move Backward	Used to move the character backward relative to the camera direction	S Down	ForwardDirection, Negate
	Move Right	Used to move the character right relative to the camera direction	D Right	RightDirection
Jumping Mapping	Jump StopJumping	Used to make the character jump and flip	Space Z Left Shift	JumpKeyHoldTime, JumpCurrentCount
Looking Mapping	Look	Used to rotate the camera	Mouse Movement	LookAxisVector
Fullscreen	Toggle Fullscreen	Used to switch between Fullscreen and windowed	F11	N/A
Suicide	Kill	Used to suicide to prevent soft lock	Q R	RespawnTime
Menu	BackToMenu	Used to go back to main menu	Backspace F2	N/A
Pause Game	PauseMenu	Used to pause the game and show pause menu	Tab P Escape	N/A
Zoom Camera	ZoomCamera	Used to zoom the camera in or out	Mouse Wheel Vertical	FollowCamera.RelativeLocation

The input of movements will be converted to a 2D vector (MovementVector). Unreal will first get the Camera rotation to find out which direction is forward; then it calculates the right and forward direction by GetUnitAxis, and add the movement to the character using the value of the MovementVector and right and forward directions. To make sure the movement is corresponded to the camera of character

all the time, when the character is flipped, the RightDirections will be negate, as the camera has rotated.

The input of looking will be converted to a 2D vector (LookAxisVector) as well. Unreal will directly add the value of the vector to the controller. To make sure the looking is corresponded to the camera of character all the time, when the character is flipped, the both the yaw and pitch to be added to controller will be negate, as the camera has rotated.

Core Gameplay Mechanic

Mechanic Overview

The core gameplay mechanic in GravityGame is the ability for the player character to flip gravity when jumping. When the player character jumps, the game will rotate the character and flip the gravity, causing the player character to fall in the opposite direction.

To progress through each level, the player character must collect at least one crown, which will activate the portal at the end of the level which can leads to the next level. This creates an additional objective for the player to complete and adds a sense of progression to the game.

The player character's location is saved at green save points, which can be used to "respawn" the player character if they fall to their death or are killed by red cube or bullet. If the player character's Z location is lower than 0 or higher than 2000, they will be "killed" and respawned at their last save point.

The gravity flip mechanic interacts with the other gameplay mechanics and elements in the game. Green movable block will carry the character standing on it to its destination. Blue platforms will fall when the player character steps on them, creating new opportunities for traversal and platforming. Red cubes and bullets will kill the player character instantly, creating hazards that the player must avoid. Space stations will summon red bullets that the player character must avoid or overcome, adding an additional layer of challenge to the game. Finally, yellow blocks will flip the player character's gravity when touched, creating new challenge for movement and traversal.

Mechanic Functionality

Jumping & Flipping

When player press the jump button, it will set bPressedJump to true. This will be detected in CheckJumpInput, the character will then try to jump by calling DoJump. If the character perform jump successfully, it will try to flip itself. Character GravityScale will be negated, and the character will be rotate to upside down. It will set to CameraRollDestination to 180, this variable will be used in Tick, which will make the camera rotate 10 degree a frame until it has rotated to a value of CameraRollDestination degree. When character have negative gravity, the velocity must be a positive number because the in GravityMovementComponent, when gravity scale is negative, it judges IsFalling by checking the Z velocity. Finally, the character will update its colour of lighting to let the player know the current gravity (red for negative gravity and blue for normal gravity).

Killing and respawning

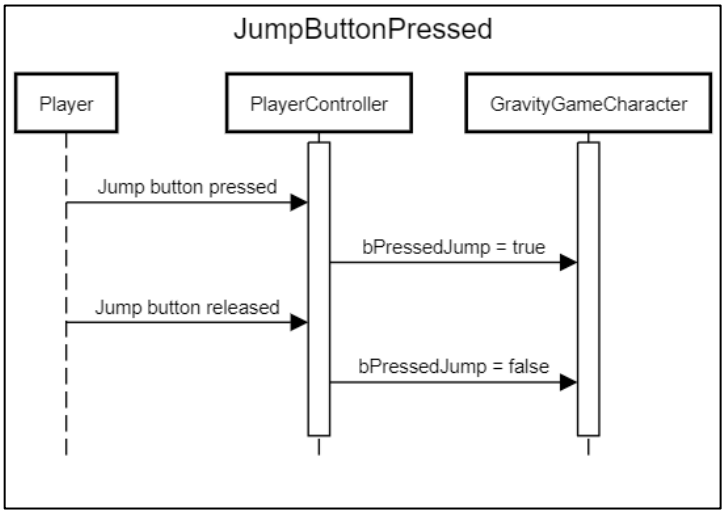
While character Z axis is not in 0-2000, Tick in the character will notice that and "kill" the character. Character will not be destroyed when died in this game because we need to respawn them in a short time, destroy the character will make it complex.

Attribute CurrentRespawnTime in this class not only represent the current timer before respawning the player, but it also represents whether player should be considered as dead or alive. Initially, or after player have been respawned, CurrentRespawnTime will equals to -1.0f to state that this timer is not activated, and the character is alive. When trying to "kill" the character, it will first confirm that the CurrentRespawnTime is equal (actually less than or equal in c++) to -1.0 which means the player is alive. It will set the CurrentRespawnTime to 0.0f to activate this timer. Meanwhile, the Tick function will try to lock the character during the respawn time by setting character to the location they died. Once the timer CurrentRespawnTime is greater than RespawnTime, it will be set to -1 again to state the character is alive. The character will be teleport to its SavedSpawnPoint, which may previously be modified by a CheckPoint, it will also check whether the gravity is the same with the time player saved the location, character will be flipped if necessary.

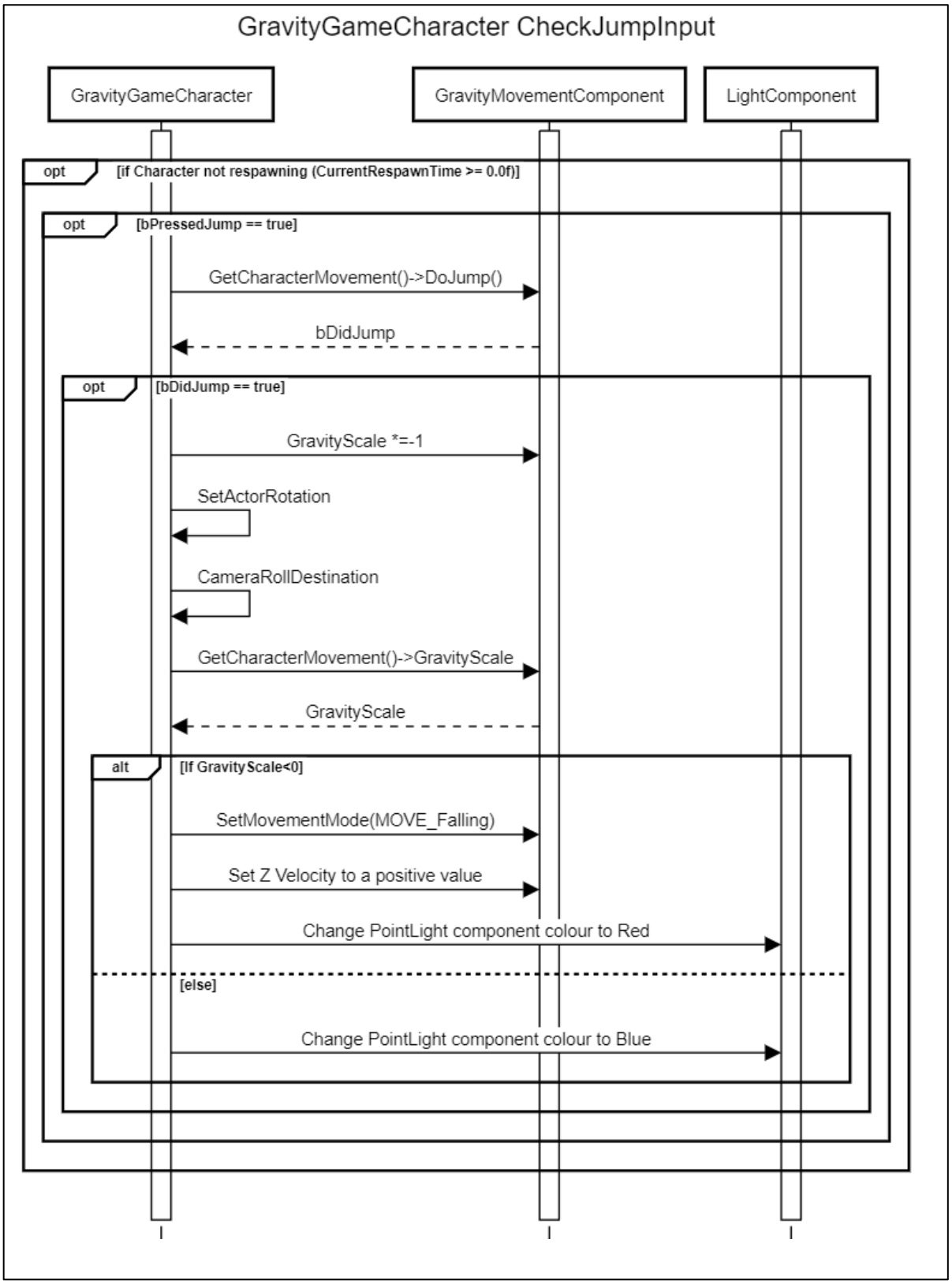
Sequence Diagrams

All sequence diagram is created via sequencediagram.org.

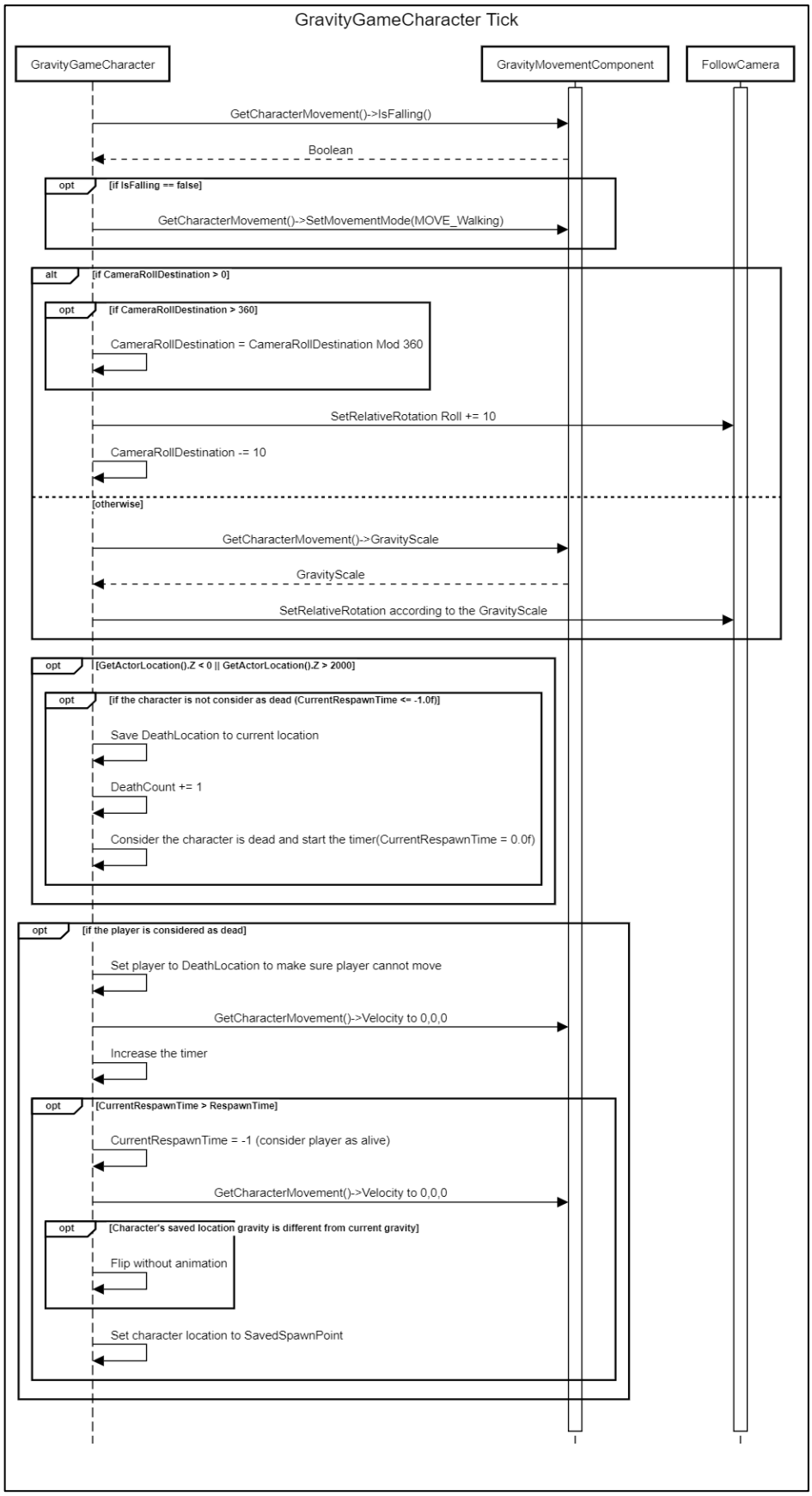
Jump Button Pressed



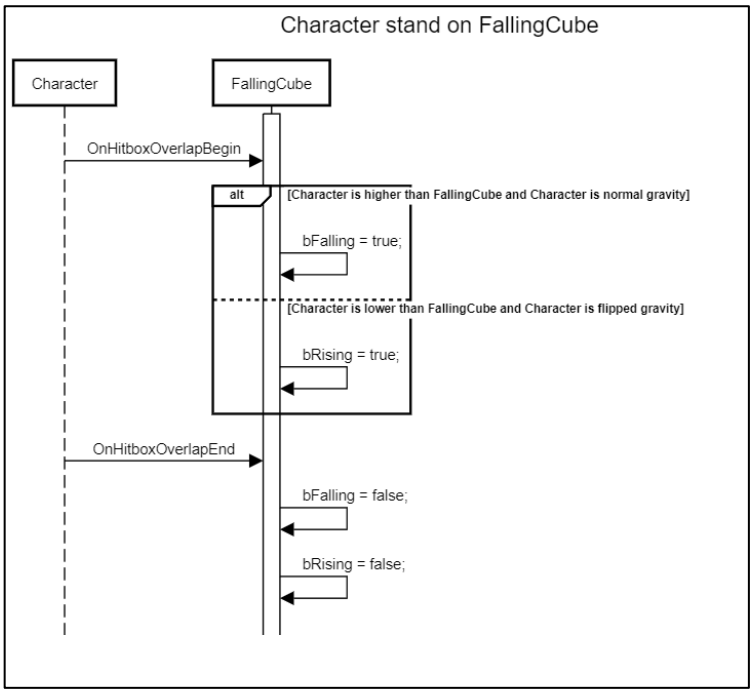
GravityGameCharacter CheckJumpInput



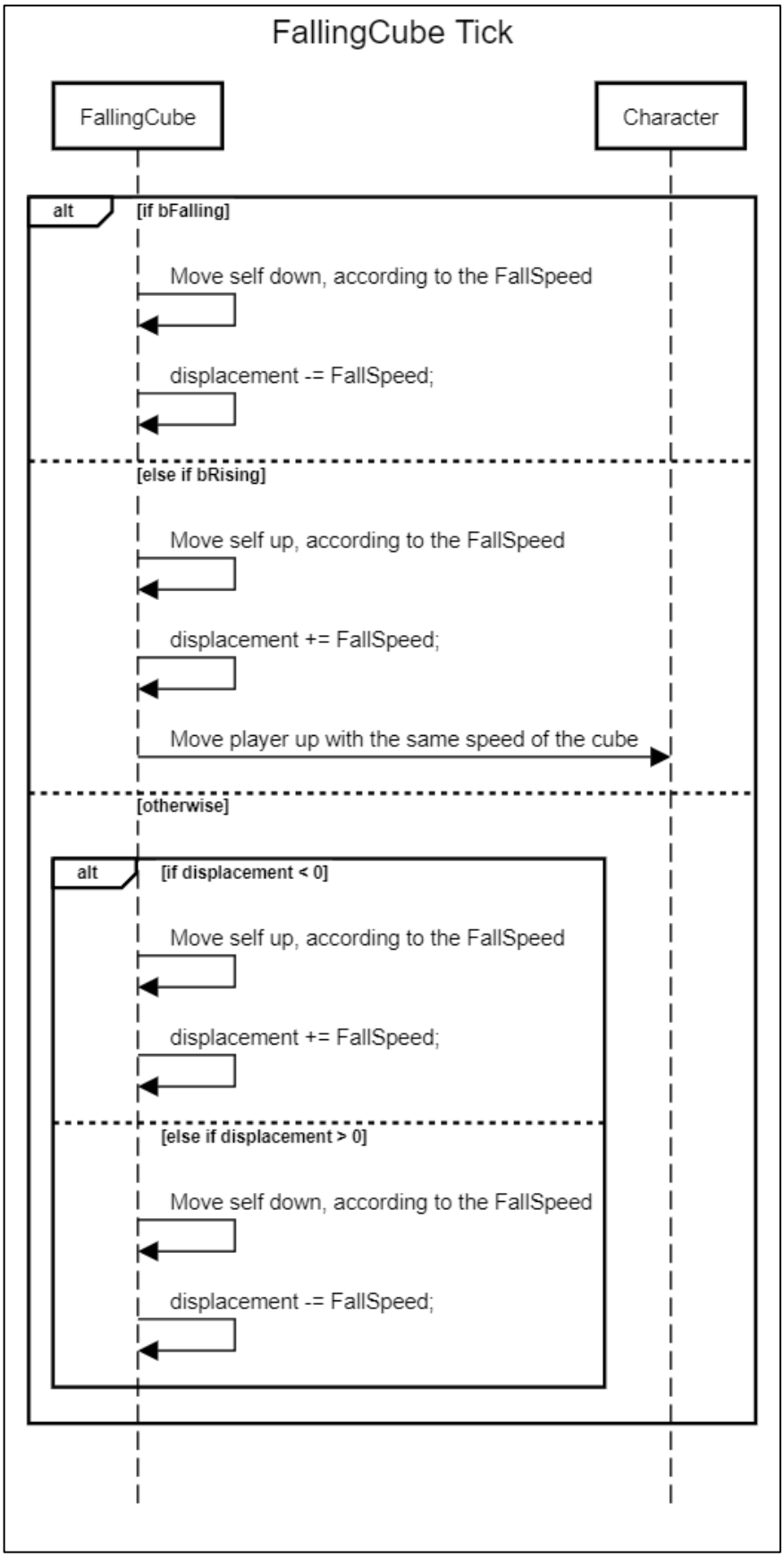
GravityGameCharacter Tick



Character stand on FallingCube



FallingCube Tick



UI Design

Menu UI

As a game, Main Menu is a critical component of GravityGame. The purpose of the Main Menu is to provide players with a clear and intuitive interface that allows player to start the game smoothly.

LevelMenu is the default level when the game starts, it will play the background music, create the MainMenuUI widget, add the widget to viewport and show the mouse.

MainMenuUI Widget

Overview

The MainMenuUI widget contains a background, a title as text block and six buttons, which are "START", "LEVELS", "HOW TO PLAY", "OPTIONS", "REFERENCE" and "QUIT".

Functionality

The "START" button is the most important element of the UI which allows the player to start the game. By using blueprint, it opens Level0 (Open Level by Object Reference) when it is clicked by the user (On Clicked).

The "LEVEL" button is used to open the SelectLevel widget so that players can select a level to play. By using the blueprint, it creates and add the widget to viewport when it was clicked (On Clicked).

The "OPTION" button is not yet implemented and is disabled for now. It is aimed to show options (e.g., volume, key controls, etc) for the users.

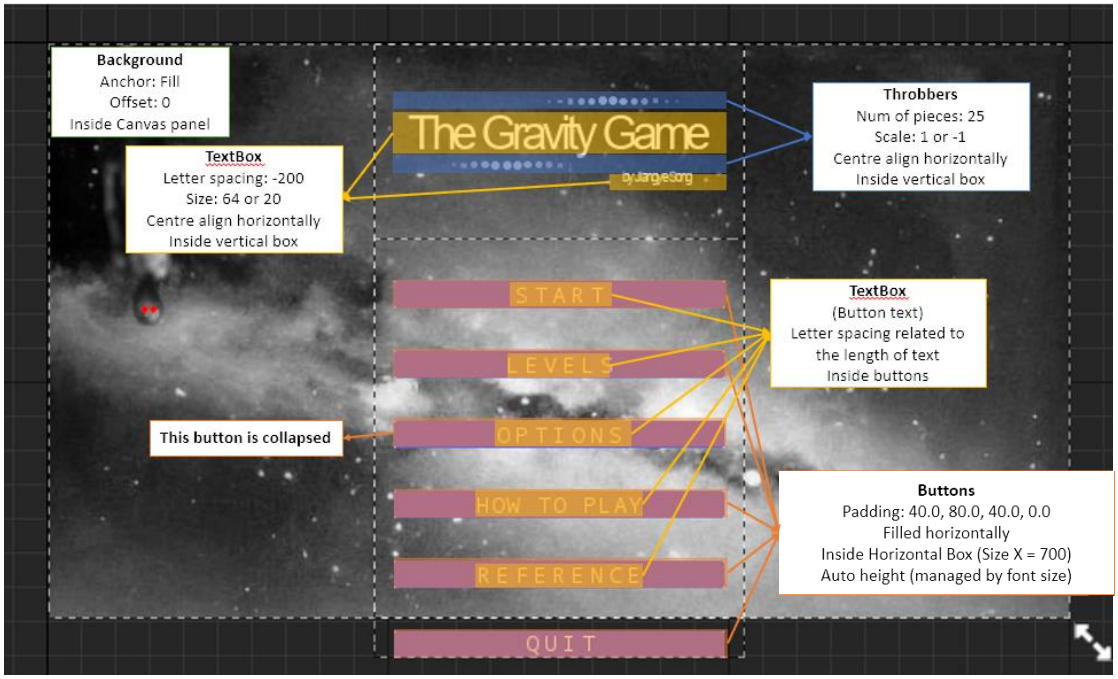
The "HOW TO PLAY" button is to create H2PUI widget to show the instruction of the control. By using the blueprint, it creates and add the widget to viewport when it was clicked (On Clicked).

The "REFERENCE" button is to show the reference of this project. It will create a ReferenceUI Widget (see the next part). By using the blueprint, it creates and add the widget to viewport when it was clicked (On Clicked).

The "QUIT" button will quit the game. By using the blueprint, it will quit the game (Quit Game) when it was clicked (On Clicked).

In-Game Screenshot

Annotated Screenshot



ReferenceUI Widget

Overview

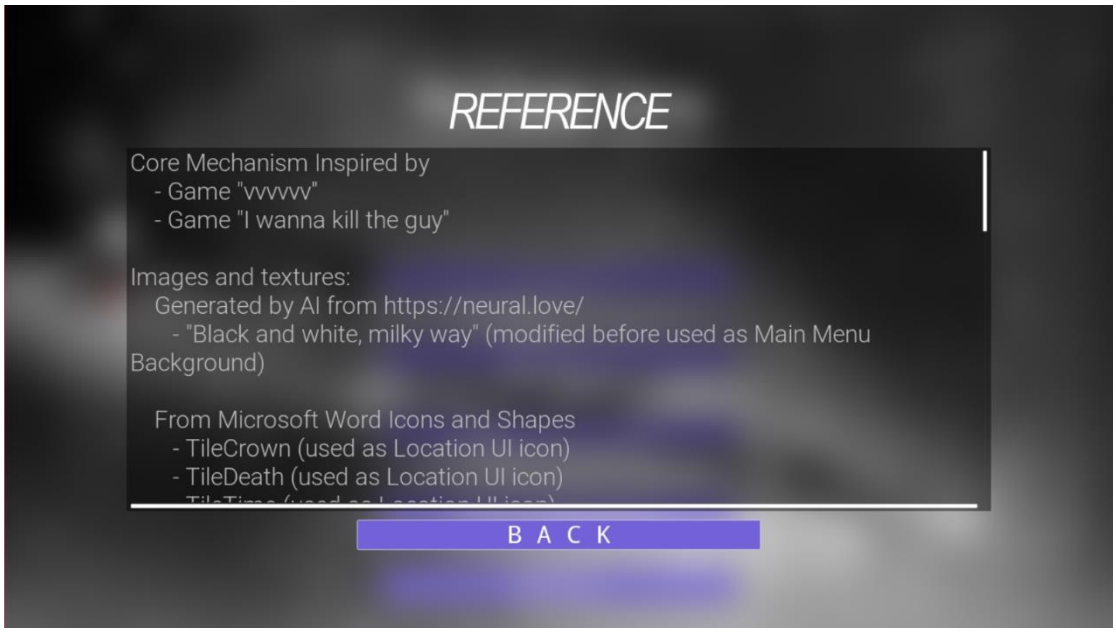
ReferenceUI aim to show the reference for this project.

It contains a title as text block, a multiline editable textbox in read only mode to show content, a back button, a blur effect to make text more clearly and another disabled read-only multiline editable textbox at the back to prevent mis-click.

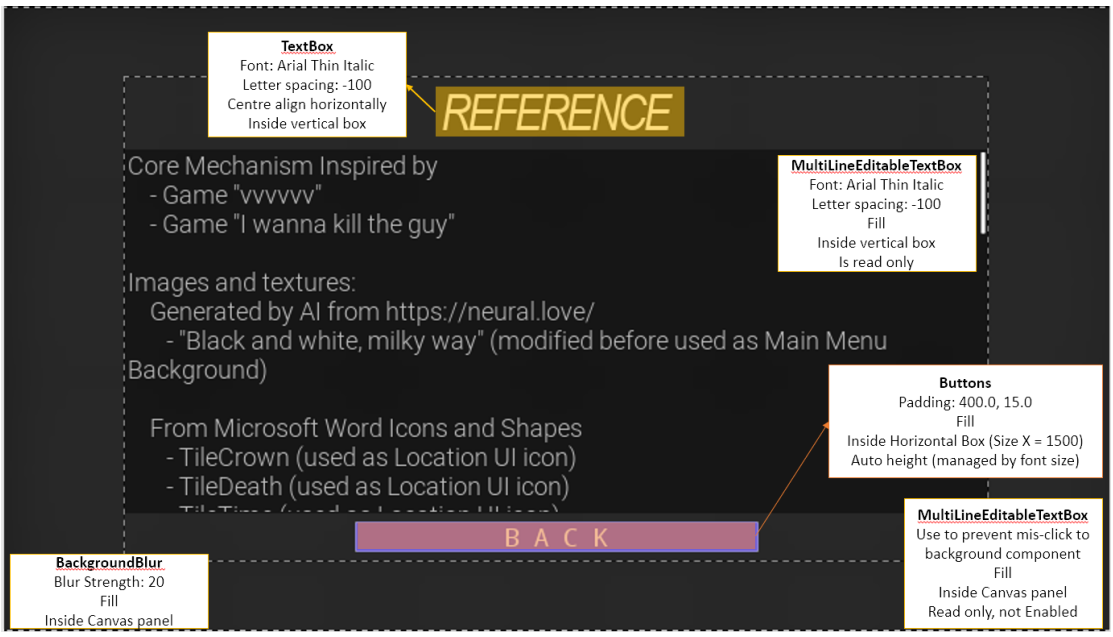
Functionality

By using blueprint, when back button is clicked, it will remove self (the widget) from its parent (Remove from Parent), which destroy the widget and the MainMenuUI on the back will reveal again.

In-Game Screenshot



Annotated Screenshot



H2PUI

Overview

H2PUI aim to show the instructions for the game.

It contains lots of text blocks, a back button, a blur effect to the background and another disabled read-only multiline editable textbox at the back to prevent mis-click. The textboxes include the title text block and tip text block, which are on the top of the page, other text blocks were put into 3 vertical box to display like a table.

Functionality

By using blueprint, when back button is clicked, it will remove self (the widget) from its parent (Remove from Parent), which destroy the widget and the MainMenuUI on the back will reveal again.

In-Game Screenshot

Annotated Screenshot

Select Level Widget

Overview

Select level UI aims to provide levels for user to select.

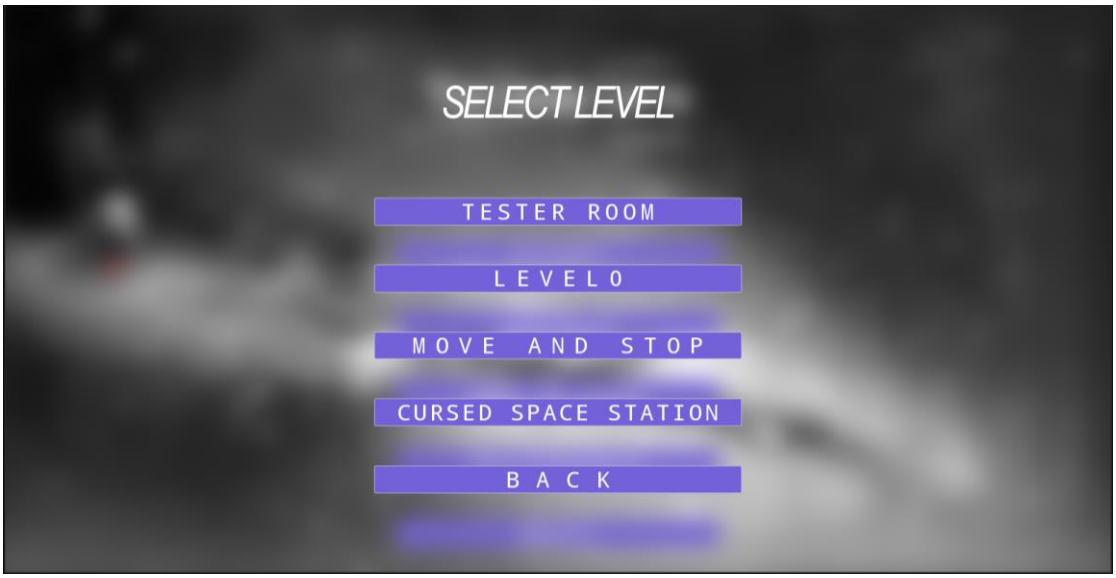
It consists of a title as text block, 4 buttons for different levels, a back button, a blur effect to make text more clearly and another disabled read-only multiline editable textbox at the back to prevent mis-click.

Functionality

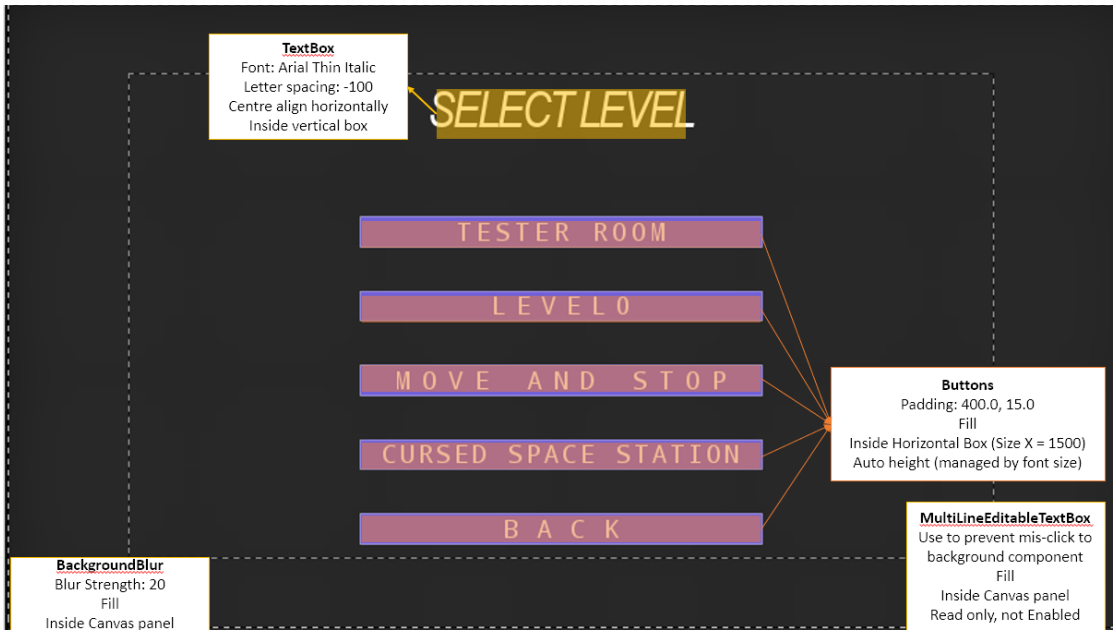
By using blueprint, when a level button is clicked, it will load corresponding level by Open Level by Object reference.

By using blueprint, when back button is clicked, it will remove self (the widget) from its parent (Remove from Parent), which destroy the widget and the MainMenuUI on the back will reveal again.

In-Game Screenshot



Annotated Screenshot



LevelClearWidget

Description

LevelClearWidget shows when character reached an activated portal. It provides the level and player information and give players access to the next level or title screen.

It contains a title as text block. It will display the level name, number of crowns collected, clear time, and death count.

Functionality

While player reach the portal, the portal saves the clear time to the GameMode. The widget can read the clear time (GameMode->ClearTime).

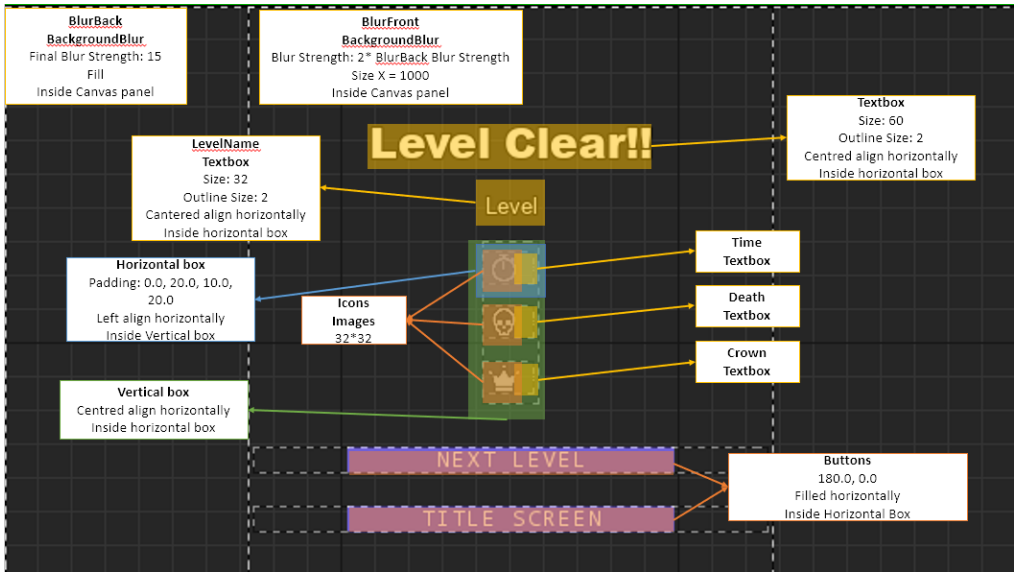
This UI divide the display process into 8 stages, controlled by an integer stage and a switch statement. When a text (level name) is fully displayed or a number (time/death/crown) reaches the real value or a gap (wait time between each components) have waited enough time, it enters the next stage by reset the timer (TimerCurrent), timer interval (UpdateWaitTime) to the next stage's interval, and add stage to 1. In the end, it displays all the information needed and display the buttons to MainMenu (TitleScreenButton) and Next Level.

By using blueprint, when TitleScreenButton is clicked, it will load LevelMenu by Open Level by Name.

This widget will read the NextLevel FName attribute from GameMode, which is usually set during the Map initialization process. By using blueprint, when NextLevelButton is clicked, it will load the level in NextLevel FName by Open Level by Name.

In-Game Screenshot

Annotated Screenshot



PauseWidget

Overview

PauseWidget pause the game and show a pause menu to the user. It allows users to regain the access to mouse or exit the level. It contains a text block at the top, another text block for level name, four buttons ("RESUME","START OVER","LEVELS" and "TITLE SCREEN") and the current level info at the bottom.

Functionality

When the player presses the pause button (P/Esc/Tab key), the player character will create and add the Pause widget to the viewport, it will pass the current world to this widget as an attribute; meanwhile, the player character will show the cursor and set the world to pause.

The "RESUME" button is aimed for resume the game. By using blueprint, it will remove itself from the parent; at this time, the override function NativeDestruct() in C++ will be called. It will un-pause the world in its attribute if that world still exists (will not exist if the game is exiting).

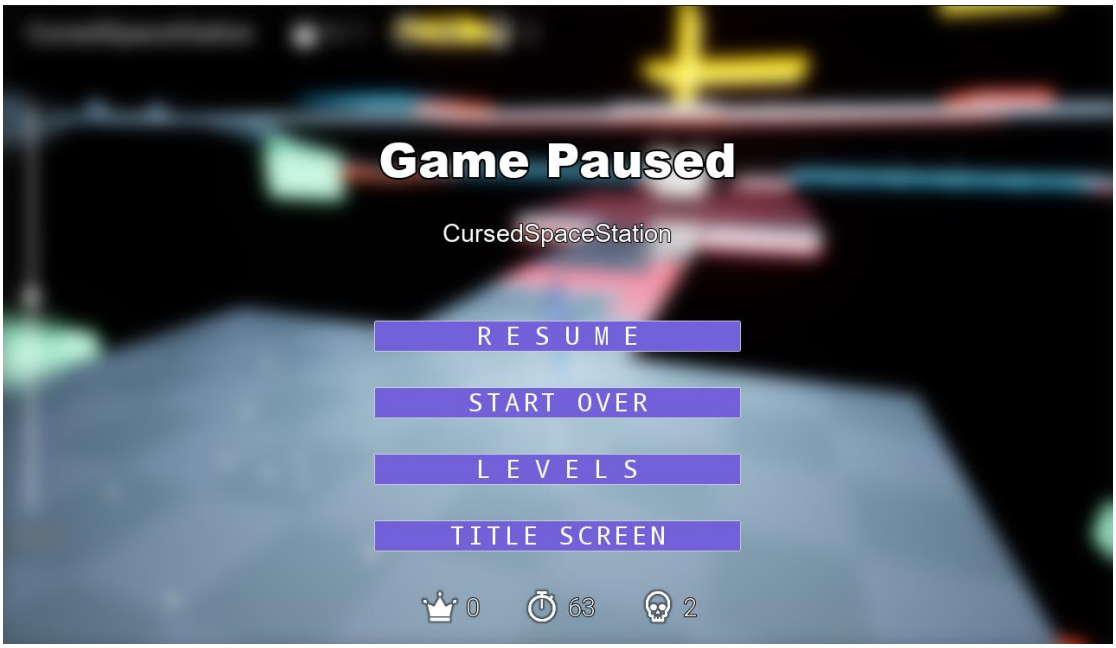
The "START OVER" is to restart the level from beginning. By using blueprint, it reads the current world name, and open the current world again by name.

The "LEVEL" button is used to open the SelectLevel widget so that players can select a level to play. By using the blueprint, it creates and add the widget to viewport when it was clicked (On Clicked).

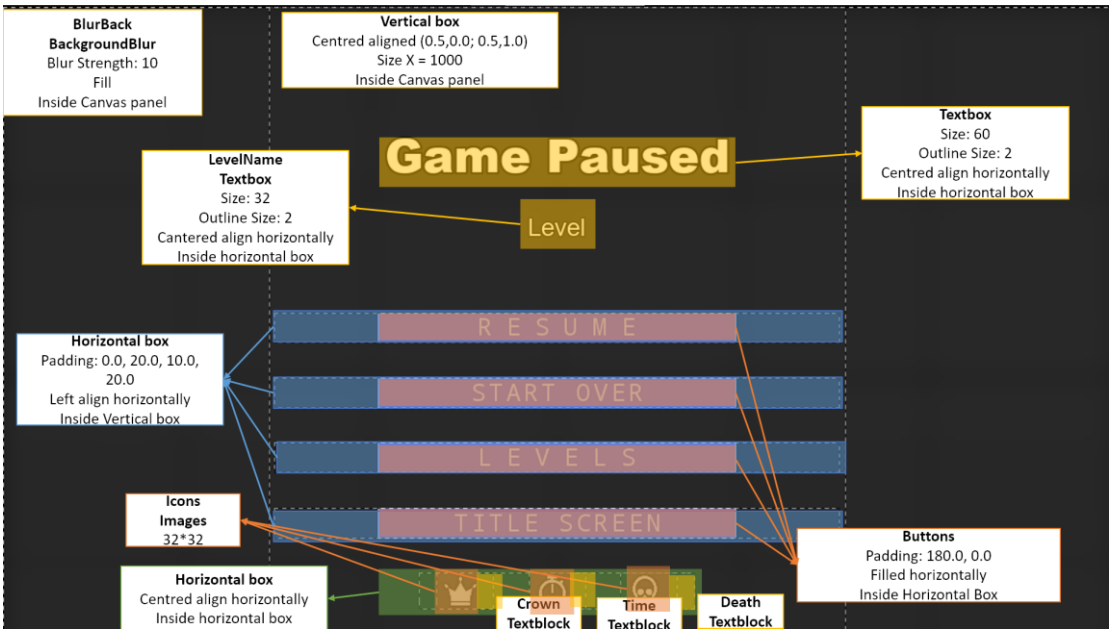
The "TITLE SCREEN" button is to go back to the title screen. By using blueprint, it loads the LevelMenu world by name (Open level by name).

The implementation of the level info at the bottom of this widget is similar to LevelInfoWidget, which provides the current status and progress of the level.

In-Game Screenshot



Annotate Screenshot



In-Game UI

Main In-game UI is separate into 2 as we might not need part of UI in some special levels (e.g., no level info widget in LevelClear or a bonus level (may not appear in final game)).

LevelInfoWidget

Overview

LevelInfoWidget shows on the top of a level. It provides the level progression and information.

It contains images as icons and text blocks. It can display the level name, number of crowns collected, number of all crowns in the map, count down if applicable, time elapsed, and death count.

Functionality

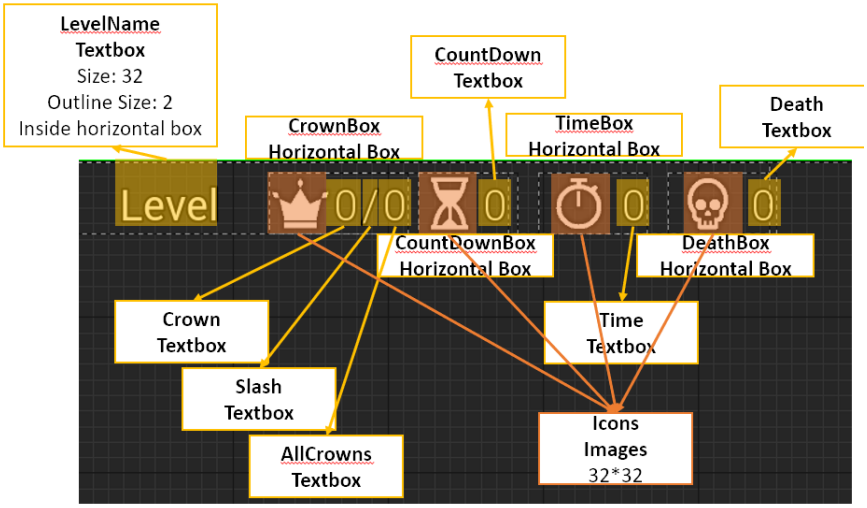
When it is created (in NativeConstruct), the widget will get the level name from the world (GetWorld()->GetName()). It will get the character as well and save the character's pointer into its attribute.

The Gravity Game Mode will keep tracking the death time, crown taken (crown taken will be modified by a Crown when overlaps) and count down. For every tick(NativeTick) afterwards, it will keep reading the death count (GameMode->GetDeath()) and crown taken (GameMode->GetCrown()) from the character and update the value to the view. It also counts the game time inside the tick and show it as seconds as an integer.

By default, the countdown box is collapsed. While the CountDownTime in GameMode is greater than 0 (CountDownTime = -1 in most levels), that means this level is a survival level. The widget will collapse the Crown section temporarily and show the countdown box (CountDownBox->SetVisibility(ESlateVisibility::Visible)).

In-Game Screenshot

Annotated Screenshot



LocationWidget

Description

LocationWidget aims to tell the player about their character's height. It also warns the player if the character is in a danger Z axis as character will be killed when it is outside the range 0-2000.

It consists of text boxes to show X, Y and Z axis (but X and Y views are now invisible). It also contains a slider to show their Z axis more geometrically and directly.

Functionality

As the same for LevelInfoWidget, LocationWidget will get and save character's pointer from NativeConstruct. It will keep track of player's location from NativeTick and update them to views.

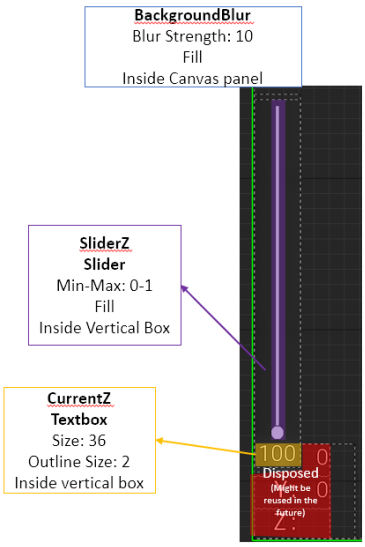
When player is in a danger zone (Z <= 400 with normal gravity OR Z >= 1600 with negative gravity), a Boolean "bZInDanger" will be triggered, which will decrease the alpha of Z axis text block a little bit every tick. Once the alpha hit 0, another Boolean "bHidingZ" is negated and alpha will try to rise a little bit every tick again. Once the alpha hit 1 the Boolean "bHidingZ" will be negated again and alpha will go down again. In this way, the Z-axis text block will be flashed which will warn the player to flip as soon as possible.

Another Boolean will be true when player is very dangerous (Z <= 200 with normal gravity OR Z >= 1800 with negative gravity), which will add/deduct more alpha of Z-axis text block which will make the flash more frequent. It also "disables" the slider to let the slider shows it disabled colour scheme (red).

The slider cannot be dragged by the user because it is set to the percentage of the location of the user every tick. Disable/enable is only for the colour scheme. A blur effect has added to make it show more clearly.

In-Game Screenshot

Annotated Screenshot



MessageWidget

Description

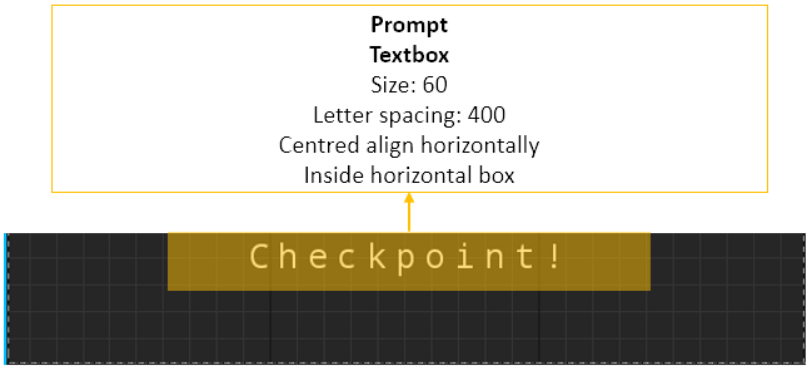
MessageWidget is a simple widget to show a message on screen and fading out in 3 seconds.

It consists of a text box of the message.

Functionality

After it spawned, it will reach the player character and get the Message.

Annotated Screenshot



Dynamic Materials

Dynamic Material 1- BadMaterial

Overview of Effect

Bad Material is used for KillingCube / KillerBullet. It will play a pulse animation when it kills the player character to indicate itself as a murder.

Effect Description

When a player character overlapped with the killing cube, it set the MaxEmissive parameter to 1, while player character leave (respawn somewhere else). MaxEmissive is reset to 0.
The MaxEmissive used to limit the amplitude of a sine wave, while the sinewave is generated using the game time. The result sine wave will increase the brightness of the colour by add the result float directly to RGB value.

Inspiration / Reference Images

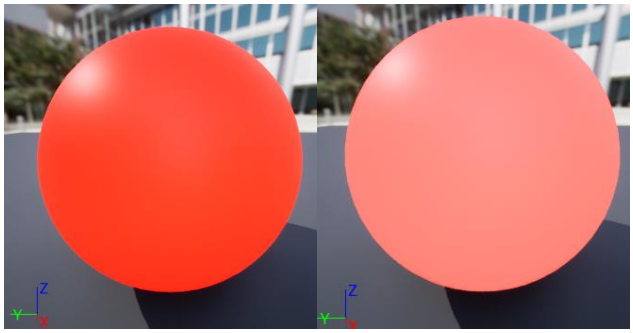
The pulse animation inspired from "plants vs zombie":

(Retrieved from: https://www.youtube.com/watch?v=VZhaETT4zNs)

Where this game uses this animation to guide the player the component to click on.

I have used this animation to show the player which component have caused their death, so that player can learn from it and avoid it for the next attempt.

In-Engine Screenshots

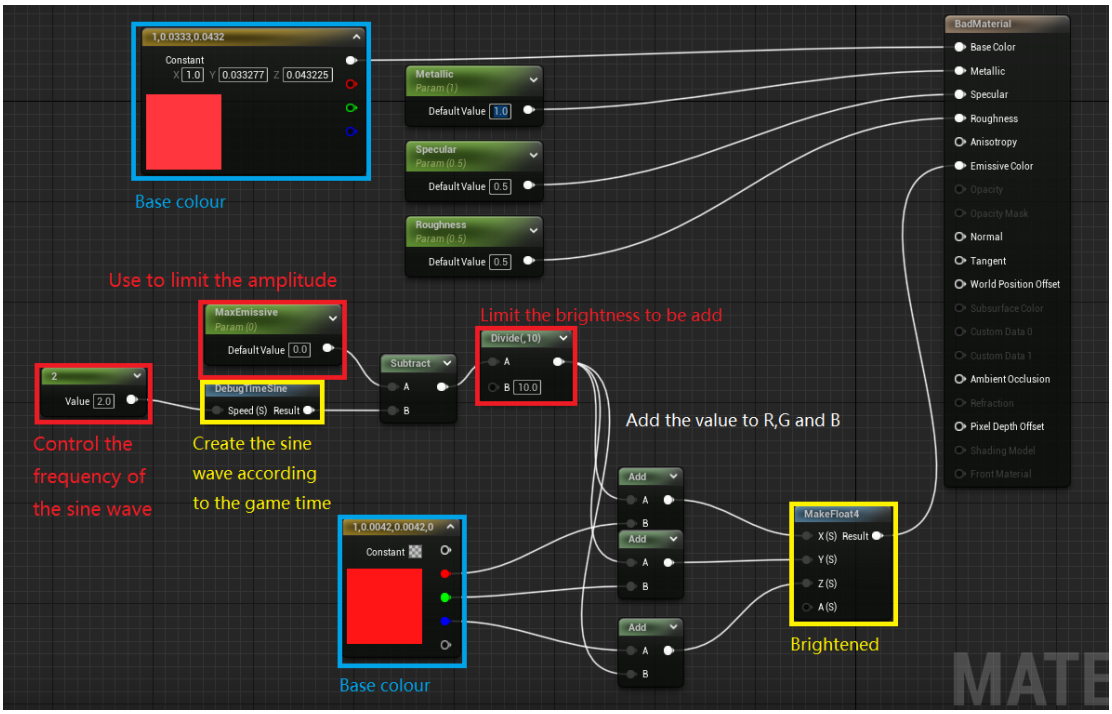


The texture is playing pulse animation and will pulse more violently when the MaxEmissive is set to 1.

Properties and Values

Property	Description of Purpose	Default Value
MaxEmissive	Used for limiting the pulse level.	Float 1(0)
Metallic	Control the Metallic of the material (Not used)	Float 1(1)
Specular	Control the Specular of the material (Not used)	Float 1(0.5)
Roughness	Control the Roughness of the material (Not used)	Float 1(0.5)

Node Graph



Dynamic Material 2- CharacterDisappearMaterial

Overview of Effect

CharacterDisappearMaterial is used so that player character skin colour can be changed when flipping. This material also handles the character death animation. This material indicate the current gravity scale of the player, and the death animation cue the player the character is dead.

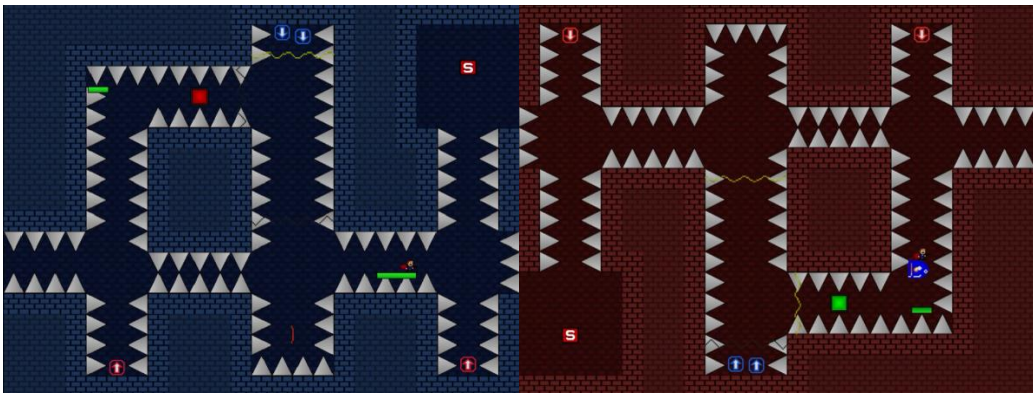
Effect Description

While player character flipped, the colour will be switch to light blue if character gravity scale is positive and will be switch to light red if is negative. This is done by set the parameter Colour to a specific value.

While player character died, the Time parameter will be activated and bond to the CurrentRespawnTime variable, which represent the current respawn progress. This will help to control the animation progress. In each tick, the green layer of the perlin texture with a deduction of the animation progress will be set to the opacity mask of the texture, which performs a dissolved animation in the end.

Inspiration / Reference Images:

Changing colour base on gravity is inspired by "I wanna kill the guy":



(Retrieved from: https://www.youtube.com/watch?v=qgL3x-6pc0k)

The game changes the colour of tiles of the blocks / background blocks to indicate the gravity status of the player.

In this game, the colour of the skin / light is changed which have similar effect.

In-Engine Screenshots:

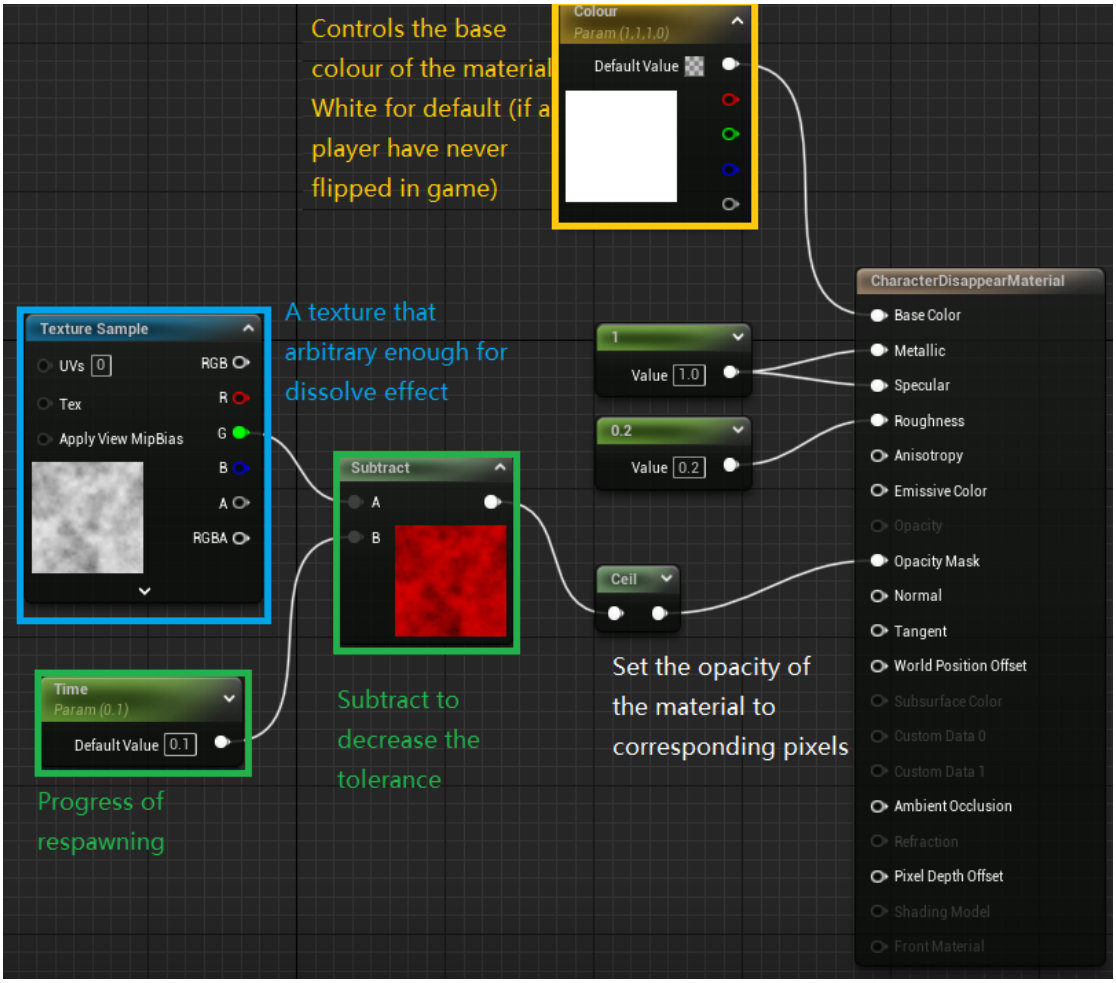
These images shows the status of the material when Time parameter equals to 0.1/0.4/0.5/0.6. The material will completely gone after Time parameter > 1.

These images show the different colour will appear in game when player character in different gravity status.

Properties and Values

Property	Description of Purpose	Default Value
Colour	Used to control colour of material dynamically	Float 4(1,1,1,1)
Time	Used to control the progress of the dissolved animation	Float 1(1)

Node Graph



Dynamic Material 3- MoveMaterial

Overview of Effect

This material is to display the movement status of the movable cube. The movement status of the texture will be synchronized with the movement status of the movable cube.

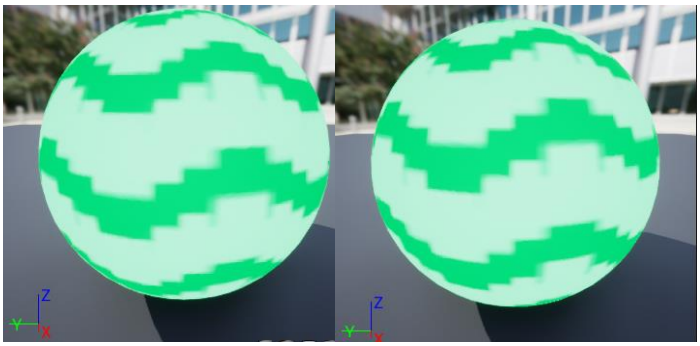
Effect Description

When the cube is moving, it will set the parameter Vibration to its moving speed. The texture will then play the move animation. When the cube stopped moving, it will set the Vibration to 0 which stopped the move animation. (While the move cube is going back, it negates its speed so the animation can play backwards.)

Inspiration / Reference Images:

N/A.

In-Engine Screenshots

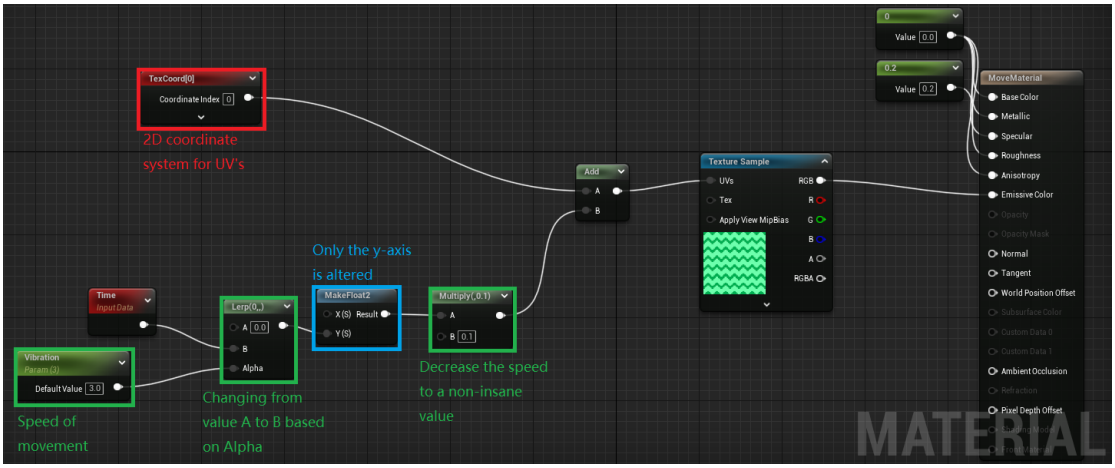


The material is playing moving animation.

Properties and Values

Property	Description of Purpose	Default Value
Vibration	Used to control the speed of the moving animation	Float 1(3)

Node Graph



Physics

Overview of Interaction

The GravityGame contains abundant physics interactions to provide a novel gameplay experience and adding depth to the puzzles. Each interaction introduces unique challenges and mechanics that players must navigate and utilize to their advantage. In this document, it will focus on Falling Cube and provide a detailed description, and then briefly describe all other physical interactions.

Interaction Description for Falling Cube

The desired physics interaction in GravityGame is the Falling Cube mechanic. This interaction increases the depth of the puzzle and add an extra layer of challenge to the gameplay.

Falling Cube react to the player character's presence. When the player character stands on top of a falling cube, it starts to fall, creating new paths or threat of death. Conversely, when the player character is flipped and stands on the back of the cube, it rises, offering alternative routes. The falling cube interaction requires careful timing and positioning to clear the level.

The effect of the Falling Cube mechanic significantly impacts the existing gameplay by introducing puzzle-solving elements and requiring players to think strategically about their movements. Players will need to think how to use / re-use the falling cube or need to act quickly before fall into void. This interaction adds depth and complexity to the platforming experience, enhancing the overall gameplay challenge and providing a sense of accomplishment when successfully utilized.

(The main purpose of displaying the Location Widget is to let player have better control for timing when stepping on the Falling Cube)

How the Interaction Works

The Falling Cube consists of a mesh representing the physical cube and a hitbox slightly higher than the mesh to detect player character interactions.

The Falling Cube utilizes two Boolean variables, namely bFalling and bRising. When the player character enters the hitbox, these variables are set based on the player character's gravity scale and z-location. If the player character has a positive gravity scale and a higher z-value, bFalling is set to true. Conversely, if the player character has a negative gravity scale and a lower z-value, bRising is set to true. When the player character leaves the hitbox, both boolean variables are set to false.

The Falling Cube includes a float variable called Displacement, which tracks the current displacement of the cube from its original location. This value is used to determine the cube's position and movement.

In every tick, the Falling Cube moves either up or down based on the values of bFalling and bRising. If bRising is true, indicating the player character is in a rising

state, the cube moves up at a speed defined by the MoveSpeed variable (which can be adjusted in the Unreal Engine). Conversely, if bFalling is true, indicating the player character is in a falling state, the cube moves down at the same speed. The Displacement variable is updated to reflect the cube's current position relative to its original location.

When both bFalling and bRising are false, indicating that the player character has left the hitbox, the Falling Cube begins moving back to its original position. It gradually moves back using the Displacement value until it reaches 0, aligning it with its starting location.

Inspiration / Reference Images

Initially, a breakable cube that breaks in a small amount of time that like this in vvvvvv was going to be develop:

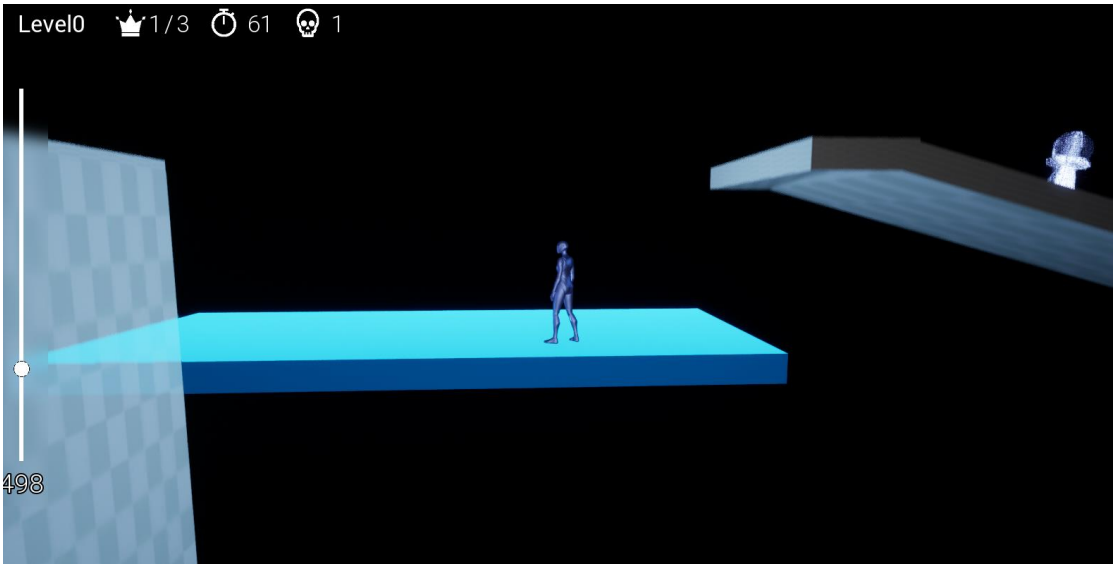
(Retrieved from https://www.youtube.com/watch?v=apwA8UQ_xeE)

However, I found it hard to display the remaining time of the breakable cube, and the modification to the collision or opacity seems difficult at that stage of development. I have then replaced this into current design. It kind of like a Donut Block in Super Mario Maker without a delay.

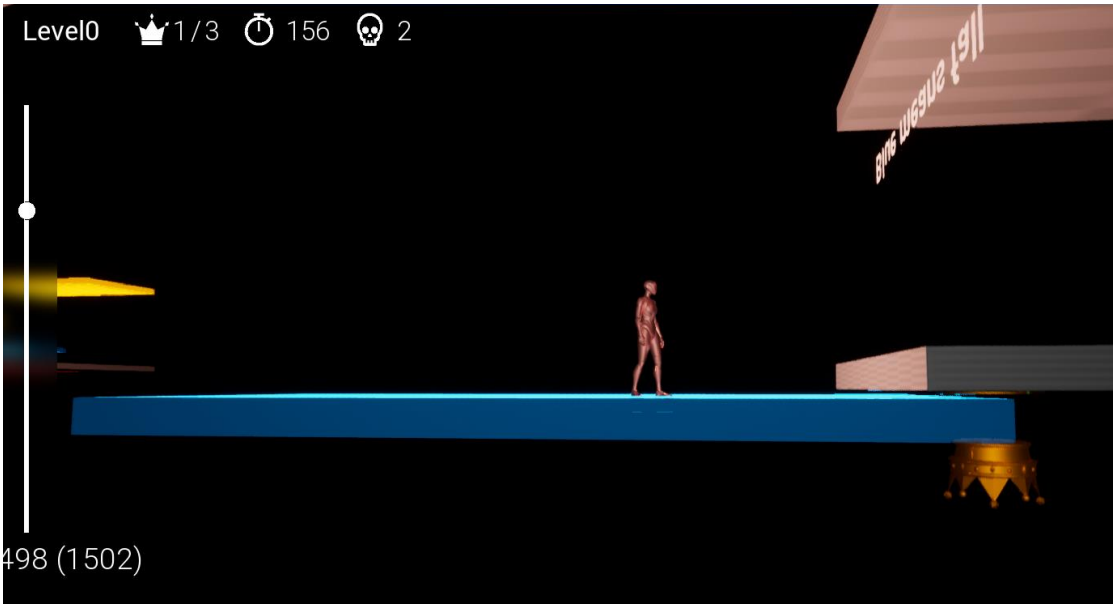
(Retrieved from https://www.youtube.com/watch?v=zdQNtR-LSIc)

In-Engine Screenshots

The player is falling on Falling Cube.



The player is falling inversely (rising absolutely) on Falling Cube.



Properties and Values

Property	Description of Purpose	Default Value
FallSpeed	(Uproperty) Used to control the falling speed of the block	float (1.0)
Displacement	Used to track the displacement to the original location.	float (0.0)
bFalling	Used to indicate if player character is standing on the top of the cube	bool (false)
bRising	Used to indicate if player character is standing on the bottom of the cube	bool (false)
Player	(Uproperty) The player character in the game	AgravitygameCharacter*
CubeMesh	(Uproperty) The static mesh of the cube	UStaticMeshComponent*
CubeHitbox	(Uproperty) The hitbox of the cube	UBoxComponent*

Other Interactions

Checkpoint

Checkpoint save locations for the player character's progress. When the player character touches a checkpoint, the game saves the current location and flips necessary information, such as the spawn point. An animation is played to indicate successful checkpoint activation, and the checkpoint deactivates itself using a Boolean (bSaved) to prevent further triggering.

Crown

Crowns are collectible items that contribute to the player's progress. When the player character collects a crown, the game increments the count of crowns that player collected in current map. The crown visually disappears by removing one of its meshes, and a Boolean (bCrownTaken) is flipped to deactivate it, ensuring it cannot be collected again.

Movable Cube

The interaction was specifically designed to address an issue where movable cubes didn't carry the player character when their gravity scale was negative. This fix allows the player character to properly interact with and ride on movable cubes to travel across the level.

Killer Cube / Killer Bullet

These dangerous objects pose a threat to the player character. Contact with a killer cube or bullet results in the player character's instant death. An animation is played to indicate the current actor is the murder.

Switch Cube

The Switch Cube is a game element that allows the player character to switch their gravity orientation. When activated, the Switch Cube triggers a gravity flip if player character overlaps with it, rotating the player character and causing them to fall in the opposite direction. The Switch Cube deactivates temporarily after activation, preventing repeated gravity switches. Its ability to flip the player character's gravity opens new possibilities for traversal and puzzle-solving, enhancing the gameplay experience.

Portal

The portal serves as the goal for each level. When activated (collecting at least 1 crown) the portal activates, shows the result of the level, and give access to the next level. This interaction triggers the display of the level clear widget, indicating successful completion and progression.

Artificial Intelligence

Overview of AI

In GravityGame, the implementation of Artificial Intelligence (AI) is utilized for the ACE enemies, in particular BossSpaceStation in current development. The incorporation of AI serves to enhance the game's dynamics and introduce increased challenges for players.

The purpose of employing AI for the Boss SpaceStation is to create a formidable opponent that can adapt. A KillerBullet has a limited lifespan and is kind of hard to aim the player character when the character is far away. With help from the navigation of AI enables the Boss to move on the map to player character, which provide a huge threat to the player. By using the blackboard and behaviour tree, the BossSpaceStation can decide whether to move or shoot to the player character in a different situation. This makes the gameplay experience more chanllenging.

AI Description

AI Abilities

BossSpaceStation can feel the location of player character every tick. In order to aim and shoot player character, they will try come closer to the player character’s location. If they get closer enough to see the player character, they will start shooting player, with a random interval between each shoot. If they found themselves cannot move to player character, they will try to move to the last position player character went that is feasible.

Inputs & Senses

AI Senses

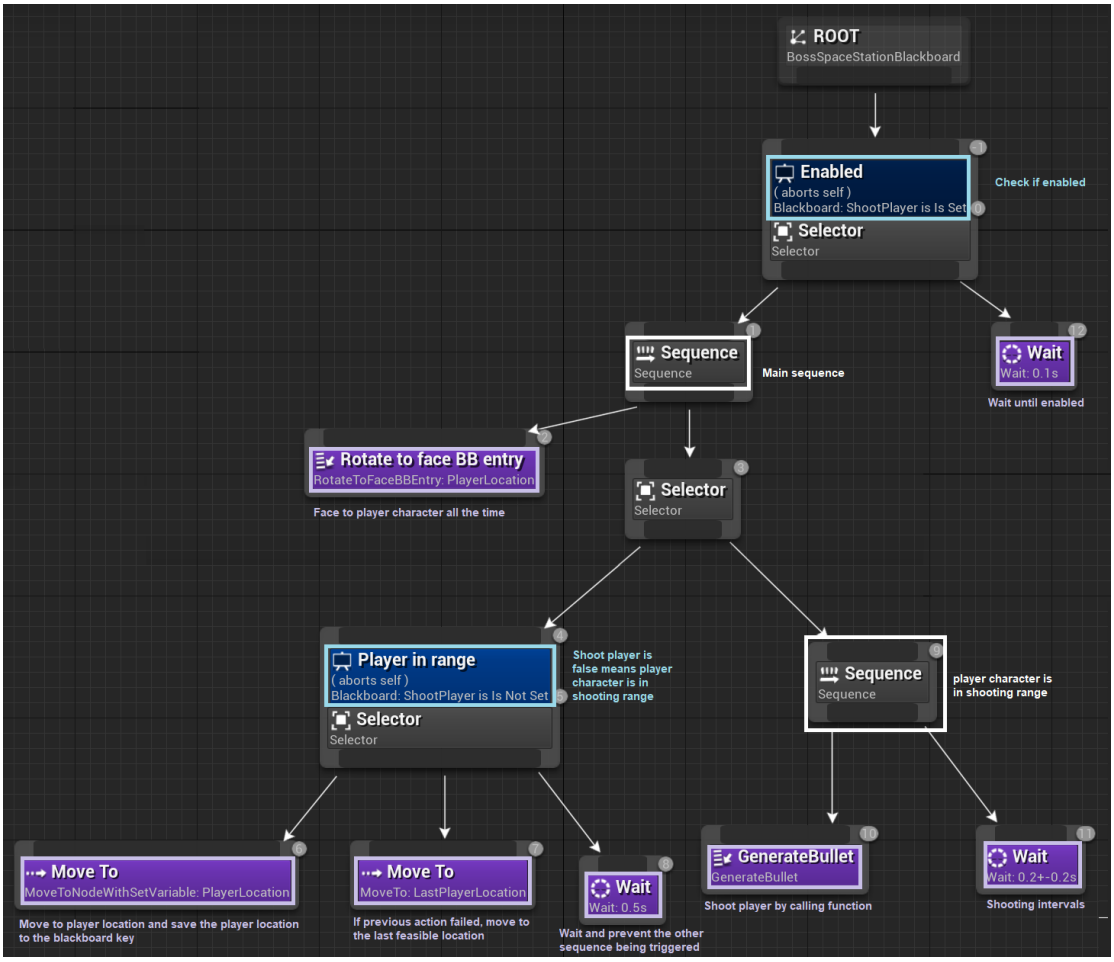
Sense Name	Property	Value
Sight	SightRadius	float (1000)
	SightAge	float (3.5)
	LoseSightRadius	float (1000)
	FieldOfView	float (180)

Blackboard Values

Property	Type	Description	Related Actions
SelfActor	Object	The BossSpaceStation character actor that AI is controlling	/
PlayerLocation	Vector	The location of the player character	MoveToWithSetVariable RotateToFaceBBEntry
ShootBoolean	Bool	A Boolean that state whether the BossSpaceStation should shoot	MoveTo GenerateBullet
Enabled	Bool	Whether BossSpaceStation should start the whole logic	<All others>
LastPlayerLocation	Vector	Stores to the last feasible player character location	MoveTo

BossSpaceStation will update PlayerLocation every tick. If the player is not within a range of radius of 1000 (field of view is 180, which means the BossSpaceStation have a view of 360 degree) or the sight is obstructed, the ShootBoolean is false and the BossSpaceStation will try to move to player character. If the movement is not Failed, the location of the player will also save to blackboard (as LastPlayerLoction). On the other hand, if the movement is failed, the selector will make the BossSpaceStation try to move to LastPlayerLocation; if it failed to move there either, then it will wait until the player character is feasible. Once the ShootBoolean is true, the BossSpaceStation will stop moving and start shoot the player character.

Behaviour Tree Graph



The BossSpaceStation will Rotate to face the player character all the time. It will then will try to move if the player character is not in range. If the player character is in range (which set the ShootPlayer to true) or MoveTo task failed, it tries another branch of the selector, which starts to shoot the player character. After every shoot, the BossSpaceStation will wait for 0.25 seconds with a 0.2 second deviation.

Niagara Particles

Niagara Particle Effect 1- PlayerTrail

Overview of Effect

The PlayerTrail Niagara particle effect generates a bunch of white particles that emanate from the player character's head (also known as dandruff). These particles mimic the player character's gravity and fall in a natural trajectory as the character moves. The effect aims to create a visual trail that allows players to trace their path and serves as a reference point within the game world.

Effect Description

The PlayerTrail Niagara particle effect in GravityGame serves as a visual indicator that continuously generates white particles from the player character's head. These particles fall according to the same gravity as the player character and have a lifespan. When they collide with objects or reach the end of their lifespan, they are destroyed and replaced with stationary particles that remain in place indefinitely.

Inspiration / Reference Images

Inspired by the trail effect seen in Angry Birds, where fired birds leave trails in the air, the PlayerTrail effect provides a similar visual cue to assist players in tracking their movement and direction.

(Retrieved from https://www.youtube.com/watch?v=aiiQ8btusrs)

The purpose of implementing the PlayerTrail effect is to address feedback from test players who reported getting lost and forgetting their starting point within the game. By creating a trail of particles that follows the player character's movement, players are provided with a visual reference of their path. This effect aims to reduce the likelihood of players getting lost and helps them maintain a sense of orientation throughout the gameplay experience.

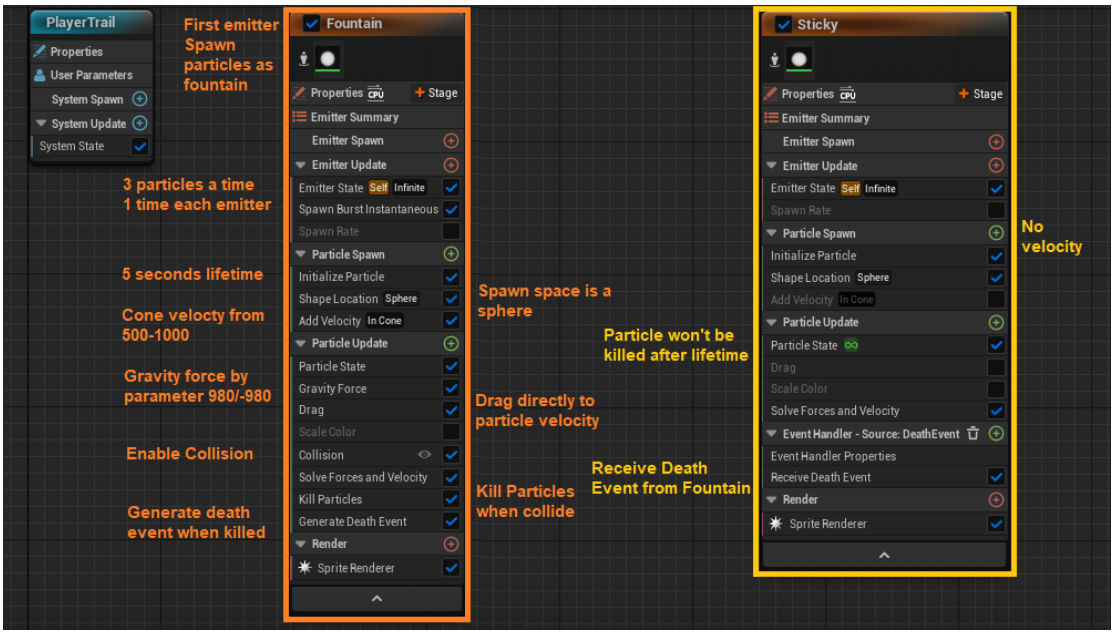
In-Engine Screenshots

The white particles were spawned with a velocity to upward and a gravity force.

Properties and Values

	State	Property	Description of Purpose	Value
Fountain	Emitter Update	Spawn Count	Number of particles spawn in one time	int32 (1)
	Particle Spawn	Add Velocity	Add velocity in cone with random range float to diverge the particles	float (500-1000)
	Particle Update	Lifetime	Lifetime of the particles	Float (5)
		Gravity Force	Controlled by user parameter (Gravity). Used to set the gravity force of the particles	float (-980)
		Kill Particles	Kill the particles when its collision and execute the death event	Particles (HasCollide)
		Generate Death Event	Event Generation Enable status	bool (true)
Sticky	Particle State	Kill Particles When Lifetime Has Elapsed	Whether to keep the particles to live forever	bool (false)
	Event Handler	Source	Source emitter and event	Emitter: "Fountain" Event: "DeathEvent"
		Spawn Number	Number of particles spawn	int32 (1)

Niagara System / Emitters Breakdown



Niagara Particle Effect 2- CrownGotEffect

Overview of Effect

When the player character successfully collects a crown, a burst of particles is unleashed on the screen. The particles consist of vibrant confetti and pyramids that fill the environment, creating a visually stunning effect.

Effect Description

The CrownGotEffect in GravityGame is a Niagara particle system that creates a burst effect consisting of confetti and pyramids. This effect is triggered when the player character collects a crown in the game. The default material for these confetti and pyramids are CrownMaterial; however, it can be change in C++ by changing the parameter.

The purpose of the CrownGotEffect is to provide a visual celebration and reward for the player's achievement, enhancing the gameplay experience and creating a sense of accomplishment.

Inspiration / Reference Images

Burst effects are very common in various games. Similar effects can be found in Super Mario Odyssey, Celeste, etc when collect or interact with items.

(Retrieved from: https://www.youtube.com/watch?v=dd5h5IAB2to)

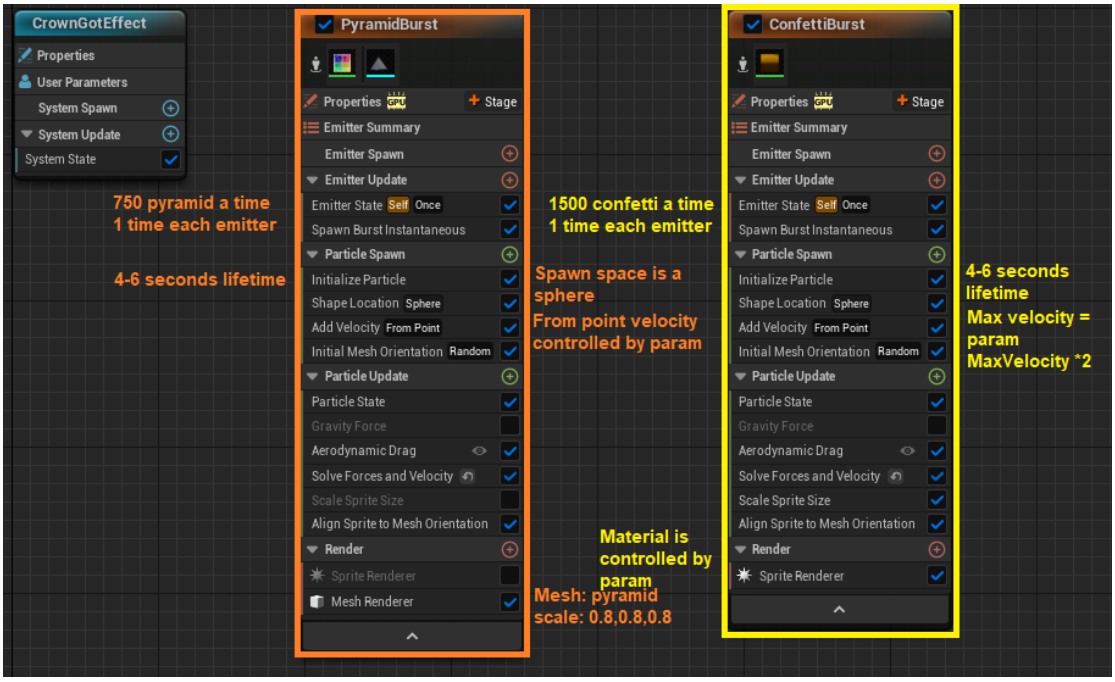
In-Engine Screenshots

A burst contains confetti and pyramids.

Properties and Values

	State	Property	Description of Purpose	Value
PyramidBurst	Emitter Update	Spawn Count	Number of particles spawn in one time	int32 (750)
	Particle Spawn	Add Velocity	Controlled by user parameter (MinVelocity-MaxVelocity). Add velocity in cone with random range float to diverge the particles	float (150-2000)
		Lifetime	Lifetime of the particles	float (4-6)
	Render	Mesh Override Material	Controlled by user parameter (Material). The material override to the mesh	Material (CrownMaterial)
		Mesh Render Mesh Scale	The size scale of the mesh (pyramid)	FVector (0.08,0.08,0.08)
ConfettiBurst	Emitter Update	Spawn Count	Number of particles spawn in one time	int32 (1500)
	Particle Spawn	Add Velocity	Controlled by user parameter (MinVelocity-MaxVelocity*2). Add velocity in cone with random range float to diverge the particles	float (150-4000)
		Lifetime	Lifetime of the particles	float (4-6)
	Render	Sprite Renderer Material User Param Binding	Controlled by user parameter (Material). The material of the sprite	Material (CrownMaterial)

Niagara System / Emitters Breakdown

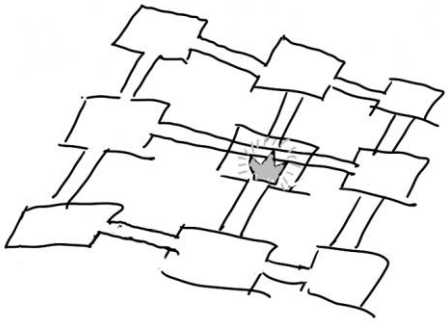
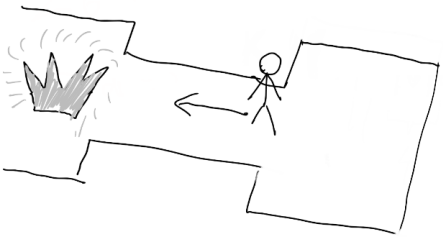
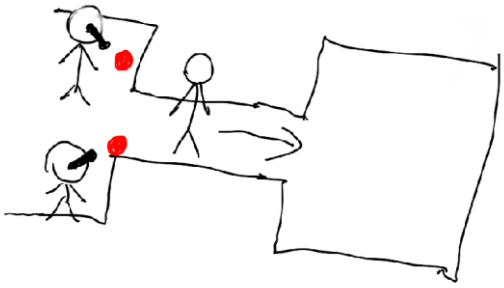


Sequencing / Cinematic

Overview of Sequence

The cinematic sequence in GravityGame occurs when the player enters the boss level known as CursedSpaceStation. The sequence plays out automatically, and the player does not have control over their character during this time. Its purpose is to provide important background information and establish the goal of the current level. It enhances the overall gameplay experience and sets the stage for an exciting survival challenge.

Storyboard

	
Show a quick overview of the map. Freeze player's input, remove all widgets.	Show the crown to the player.
	
Player character approaching the crown. Set bCutScene to true, and CutSceneMovementDirection to 0, 2, 0. Set bCutScene to false a while after.	Set bCutScene to true, and CutSceneMovementDirection to 0, -2, 0. Set bCutScene to false a while after and unfreeze player's input and add widgets back.

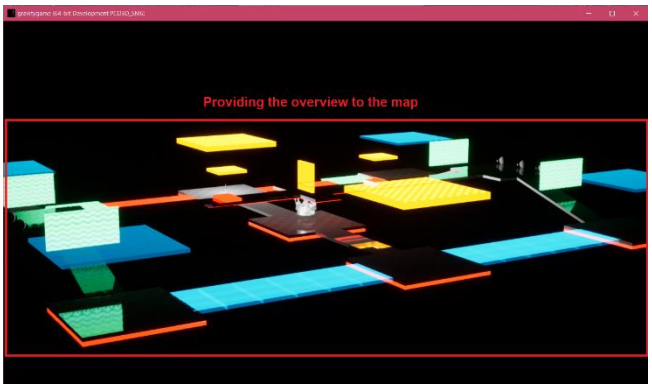
Camera Properties

Duration 0.50 – 2.00

Properties

	Time	X	Y	Z	Roll	Pitch	Yaw
From	0.50	7205	19898	6040	0	-11	-112
To	2.00	-8012	14074	4476	0	-11	-59

Screenshot

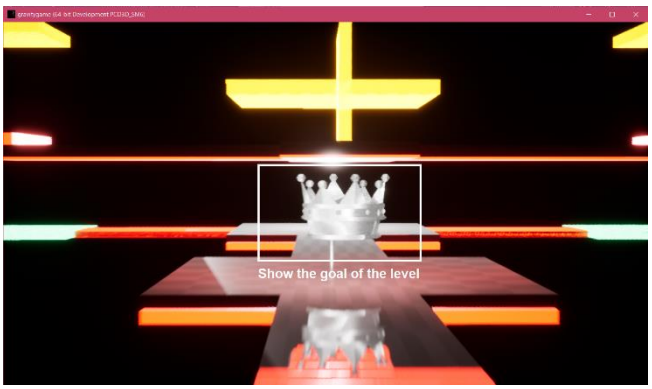


Duration 2.50 – 6.98

Properties

	Time	X	Y	Z	Roll	Pitch	Yaw
From	2.50	-8012	14074	4476	0	-11	-112
Via 1	3.70	-1324	11292	3783	0	-12	-65
Via 2	4.35	10	7066	2144	0	-15	-91
To	6.98	-121	4003	1402	0	-1	-89

Screenshot

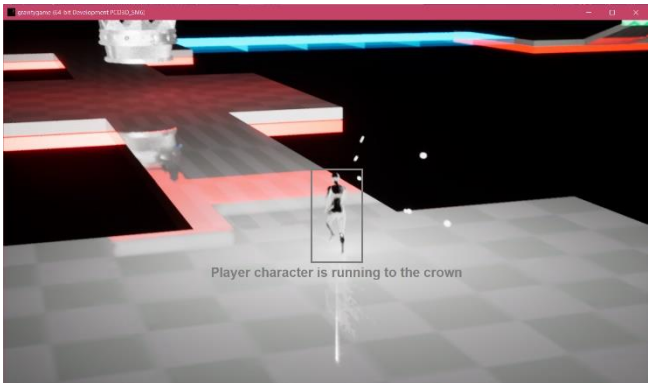


Duration 7.00 – 12.07

Properties

	Time	X	Y	Z	Roll	Pitch	Yaw
From	7.00	713	-4096	1586	0	-17	113
To	12.07	470	-3700	1396	0	-4	97

Screenshot

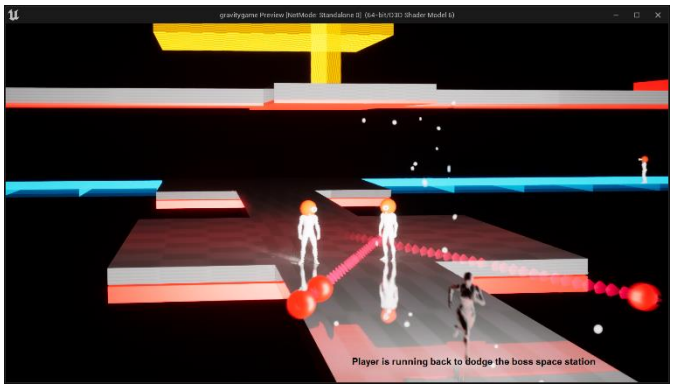


Duration 13.00 – 15.05

Properties

	Time	X	Y	Z	Roll	Pitch	Yaw
Property	12.07	470	-3700	1396	0	-4	97

Screenshot



Scripted Events

Event Name	Time	Actions
Sequence Start Event	0.03	1. Pause the game timer from game mode. 2. Freeze the control of player character. 3. Disable all BossSpaceStations. 4. Remove all widgets.
Sequence Intermediate1 Event	8.00	1. Set the cut scene move direction to approach boss crown for player character. 2. Enable the bCutScene Boolean to make player character start moving.
Sequence Intermediate2 Event	10.00	Disable all BossCrown
Sequence Intermediate3 Event	10.80	Disable the bCutScene Boolean to stop player character from moving.
Sequence Intermediate4 Event	11.00	Enable all BossSpaceStations.
Sequence Intermediate5 Event	13.00	1. Set the cut scene move direction to escape for player character. 2. Enable the bCutScene Boolean to make player character start moving.
Sequence End Event	15.00	1. Resume the game timer from game mode. 2. Unfreeze the control of player character. 3. Disable the bCutScene Boolean to stop player character from moving. 4. Clear the cut scene move direction. 5. Set message to show to "Survive!" 6. Pop that message on screen by add the message widget 7. Add other in-game widgets back.

MetaSound

Sound Effect 1- FootStep

Overview

The MetaSound effect in GravityGame includes a footstep sound system that enhances the auditory experience within the game. By incorporating different footstep sounds based on the type of surface the player character is moving on, the MetaSound effect adds depth and detail to the gameplay experience.

Effect Description

The footstep MetaSound effect is designed to dynamically play corresponding footstep sounds based on the player character's movement on different surfaces. When the player character moves on regular cubes, the footstep sound of footsteps on concrete will be repeatedly played. When the player character steps on FallingCube or MovingCube objects, the footstep sounds will change to metallic footstep sounds, reflect the material of these objects. Meanwhile, controlled by C++ code with ray tracing, The footstep sound will be stopped if player stopped moving or started falling. The footstep sounds add an additional layer of audio immersion, making the gameplay experience more realistic.

Inspiration / Reference:

Footstep sound effect that changes depend on the ground player is on are very common in various games. Similar effects can be found in Minecraft, NieR:Automata, AC:NH, etc when character is walking on different grounds.

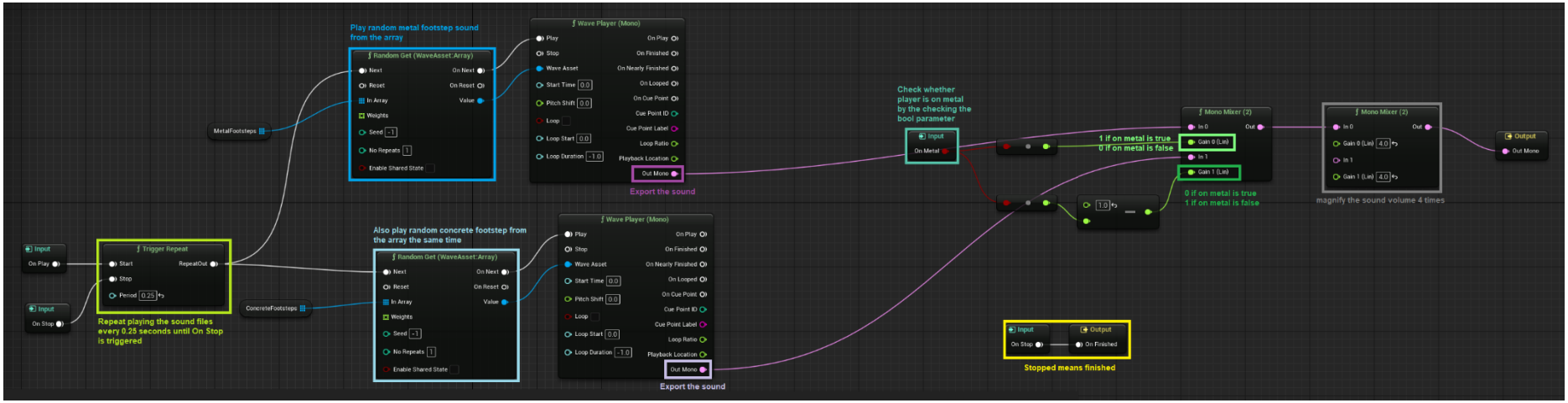
(Retrieved from: myself)

Properties and Values

Type	Property Name	Description of Purpose	Value
Variable	ConcreteFootsteps	Stores all the concreate footsteps wave sound	WaveAsset[7]
	MetalFootsteps	Stores all the metal footsteps wave sounds	WaveAsset[7]
Input	On Metal	Whether player	Bool (false)
	On Stop	Unused. Actions should be done when the sound is intended to be stopped should be followed by this trigger	Trigger

Notice: On Stop is not used because in practice UAudioComponent::Stop() function works better for this metasound when player character crossing a tiny gap.

MetaSound Diagram



Sound Effect 2 – FallingCubeSound

Overview

The FallingCubeSound is a metasound effect that will play when the player character interacts with the FallingCube. It contains three sounds for three stages: FallingStart, Falling, and FallingEnd. The sound effect is triggered when the player steps on a FallingCube, indicating its activation and movement.

Effect Description

The FallingCubeSound metasound consists of three sounds: FallingStart, Falling, and FallingEnd.

FallingStart: This sound serves as the initial sound cue when the player character steps on a FallingCube, signaling its activation. It provides a distinct audio signal to inform the player that the cube is starting its falling.

Falling: The Falling sound will keep repeating after FallingStart sound have played to state that the cube is keep falling. It provides a dynamic element to the game and providing auditory feedback for the cube's movement.

FallingEnd: At any time, if the player character leaves the FallingCube, the FallingEnd stage is triggered. This stage is designed to smoothly conclude the sound effect. It can include a fading and concluding sound element to indicate the cube is no longer falling and starts to restore.

The FallingCubeSound effect provide player cues that aligns with the FallingCube mechanics and enhances the player's experience in GravityGame.

Inspirations / Reference

This kind of sound effect structure is common in various game. Similar effects may be played when an old object is moving slowly. Titles have this kind of effects includes LIMBO, Submachine, Undertale, etc.

(Retrieved from: https://www.youtube.com/watch?v=UtTQty5QHvA)

Properties and Values

Type	Property Name	Description of Purpose	Value
Variable	FallingStart	Store the falling start state wave sound	WaveAsset
	Falling	Store the falling state wave sound	WaveAsset
	FallingEnd	Store the falling end state wave sound	WaveAsset
	On Stop	Unused. Actions should be done when the sound is intended to be stopped should be followed by this trigger	Trigger

MetaSound Diagram