Technical Design Document

DMED 521, Projects II

Alpha Squad Adventures

Developed By

Denis Morozov	Bob Kreut	Chelsea Chang	Bensson Wu	Yuxin Wang
Project Manager	*Technical Artist*	*Game Designer*	*2D Artist*	*Game Developer*

In collaboration with

Table of Content

Introduction

This document breaks down the processes of setting up the source control, unreal engine, and how to use the existing blueprints.

Get Started

Setting up Source Control

Needs more detailed info

1. Create an account on GitLab
2. Start a new project
3. Connect the Project to Source Control
4. Push the Project to Source Control

Installing Unreal Engine

1. Download Epic Store
2. Download & Install Unreal Engine 4.26.1

Installing Android Studio

1. Download Android Studio 3.5.3 from the following link;
2. Follow this easy to use official guide.

Installing Apple Software

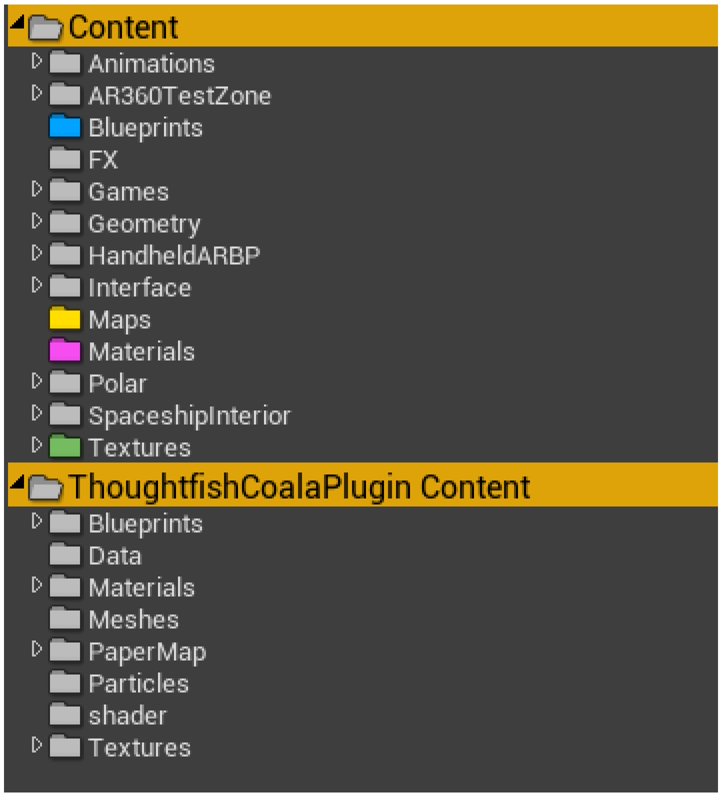
1. Install XCode from the AppStore

Unreal Engine

Architecture

There are 2 primary folders in the project:

- **Content** for all non-GPS blueprints, assets, textures, materials and everything else;
- **ThoughtfishCoalaPlugin Content** contains all GPS-related items.



Content
- Animations
- AR360TestZone
- Blueprints
- FX
- Games
- Geometry
- HandheldARBP
- Interface
- Maps
- Materials
- Polar
- SpaceshipInterior
- Textures

ThoughtfishCoalaPlugin Content
- Blueprints
- Data
- Materials
- Meshes
- PaperMap
- Particles
- shader
- Textures

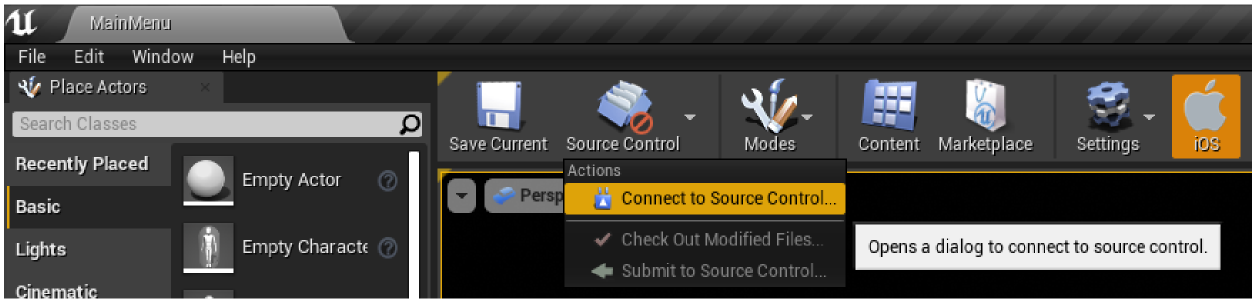
Plugins

The game utilizes the following plugins:

1. Coala GPS Plugin
2. Apple ARKit
3. Google ARCore
4. Google ARCore Services
5. Mobile Location Services

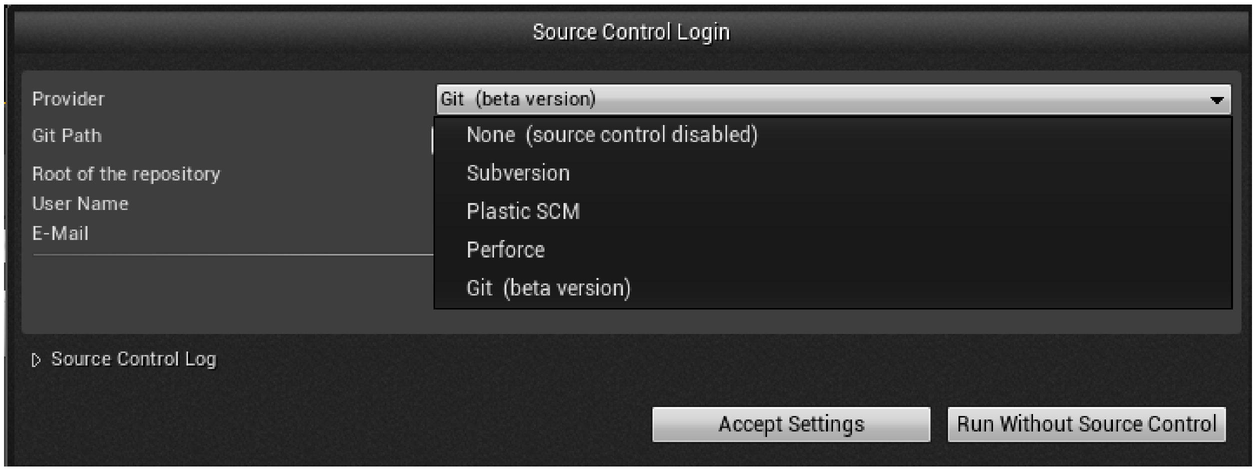
Connecting to Source Control

1. Click Source Control in Unreal Engine and select Connect to Source Control;



Opens a dialog to connect to source control.

2. Select the software of your choice and click Accept Settings;

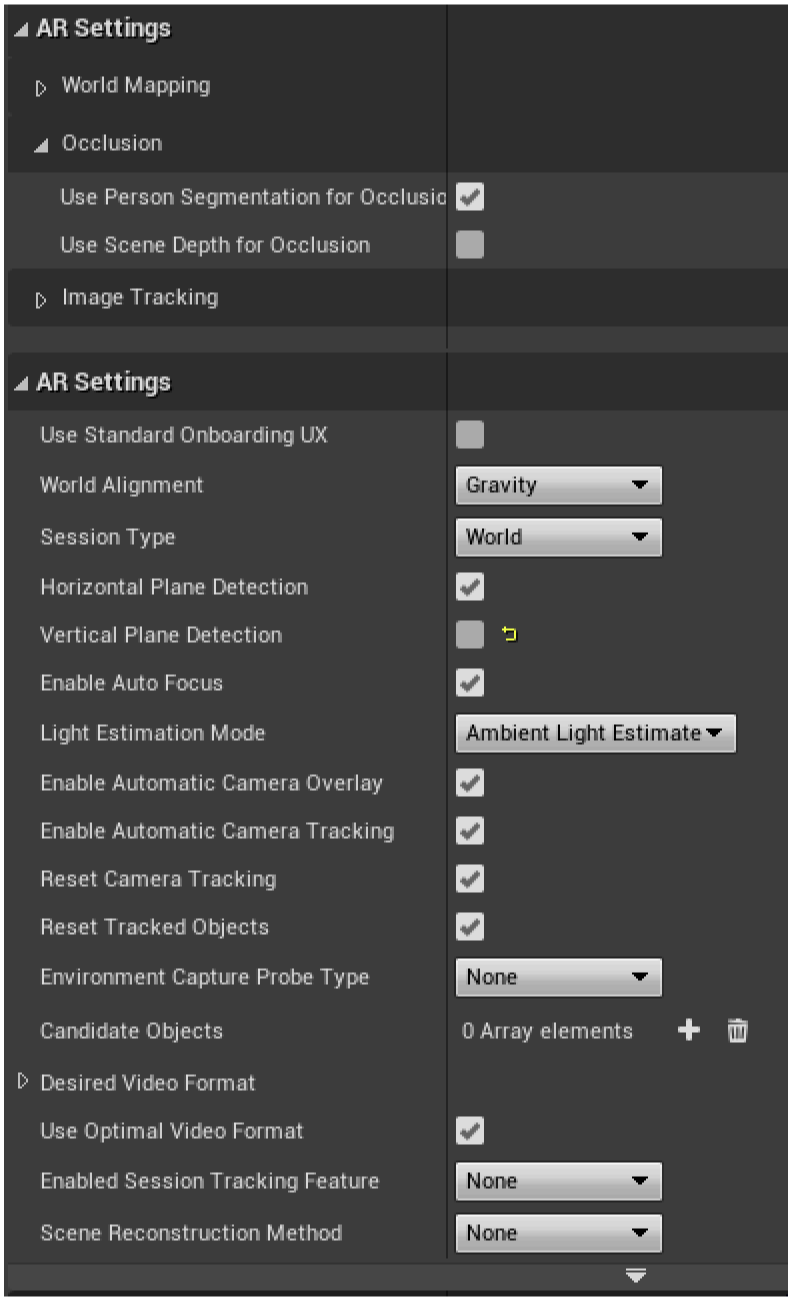


3. Source control is successfully set up and the changes can be pushed and pulled.

Blueprinting System

Setting Up AR Project

- The project is based on the Handheld AR Template. Because of this, many of the plugins and settings are already enabled for AR.
- The AR360 mode and AR Camera-On mode are set up to use different SessionConfig files so they are capable of having different settings applied to them for gameplay. However, currently they are both using the same settings and the only change is Vertical Plane Detection has been disabled so aliens won't spawn on walls.
  - "Use Scene Depth for Occlusion" was enabled so aliens would be occluded by real world objects, but this caused crashing to occur on Android devices so it was disabled again
  - The AR Gameplay Component Classes are using the Engine's AR Classes for planes, points, face, etc.

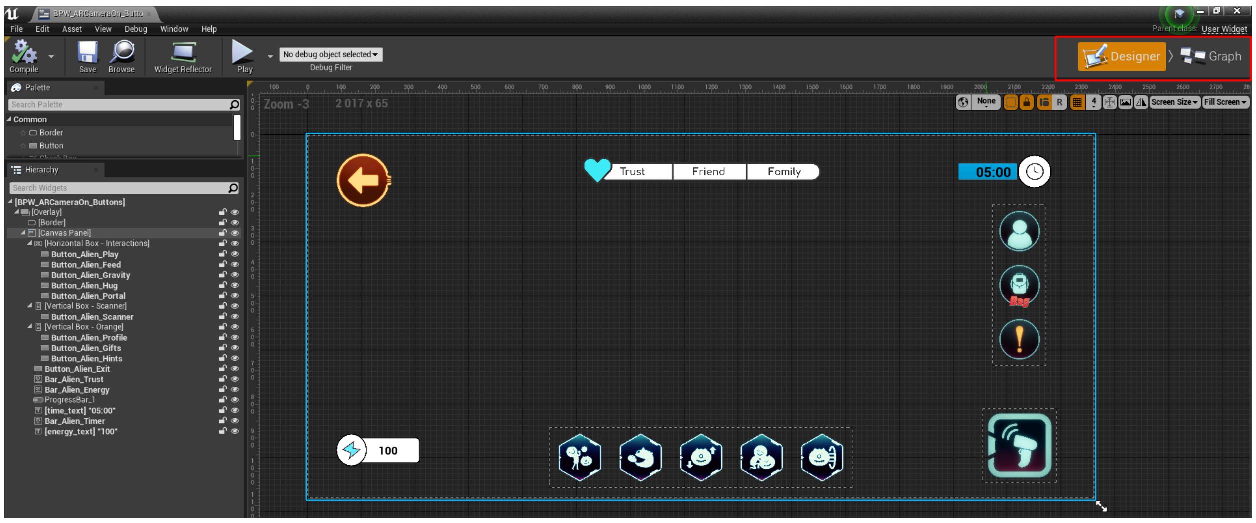


AR Settings	
World Mapping	
Occlusion	
Use Person Segmentation for Occlusio	✓
Use Scene Depth for Occlusion	☐
Image Tracking	
AR Settings	
Use Standard Onboarding UX	☐
World Alignment	Gravity
Session Type	World
Horizontal Plane Detection	✓
Vertical Plane Detection	☐
Enable Auto Focus	✓
Light Estimation Mode	Ambient Light Estimate
Enable Automatic Camera Overlay	✓
Enable Automatic Camera Tracking	✓
Reset Camera Tracking	✓
Reset Tracked Objects	✓
Environment Capture Probe Type	None
Candidate Objects	0 Array elements
Desired Video Format	
Use Optimal Video Format	✓
Enabled Session Tracking Feature	None
Scene Reconstruction Method	None

- Starting the AR Session is found in the BP_Pawn of the level
  - Because of an Android Specific bug with Unreal 4.26.1, it is not possible to start and stop the AR sessions. Code should be added on Event End Play to Stop AR Session once the bug is fixed to save battery life.
  - This bug is also the reason why AR is still enabled in the AR-Explore Mode

Interfaces & Emotion Grid

All interface elements are designed and blueprinted using the Widget Blueprints. All namings of these widgets start with **BPW**_[Name]. The Widget Blueprints consist of 2 modes: Designer & Graph - that can be selected from top right corner

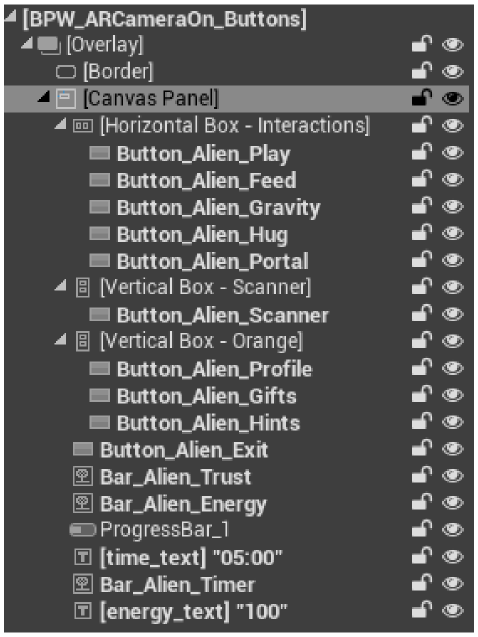


```
[BPW_ARCameraOn_Buttons]
  [Overlay]
    [Border]
    [Canvas Panel]
      [Horizontal Box - Interactions]
        Button_Alien_Play
        Button_Alien_Feed
        Button_Alien_Gravity
        Button_Alien_Hug
        Button_Alien_Portal
      [Vertical Box - Scanner]
        Button_Alien_Scanner
      [Vertical Box - Orange]
        Button_Alien_Profile
        Button_Alien_Gifts
        Button_Alien_Hints
      Button_Alien_Exit
      Bar_Alien_Trust
      Bar_Alien_Energy
      ProgressBar_1
      [time_text] "05:00"
      Bar_Alien_Timer
      [energy_text] "100"
```

Designing Widgets

A typical widget in the game uses the following items:

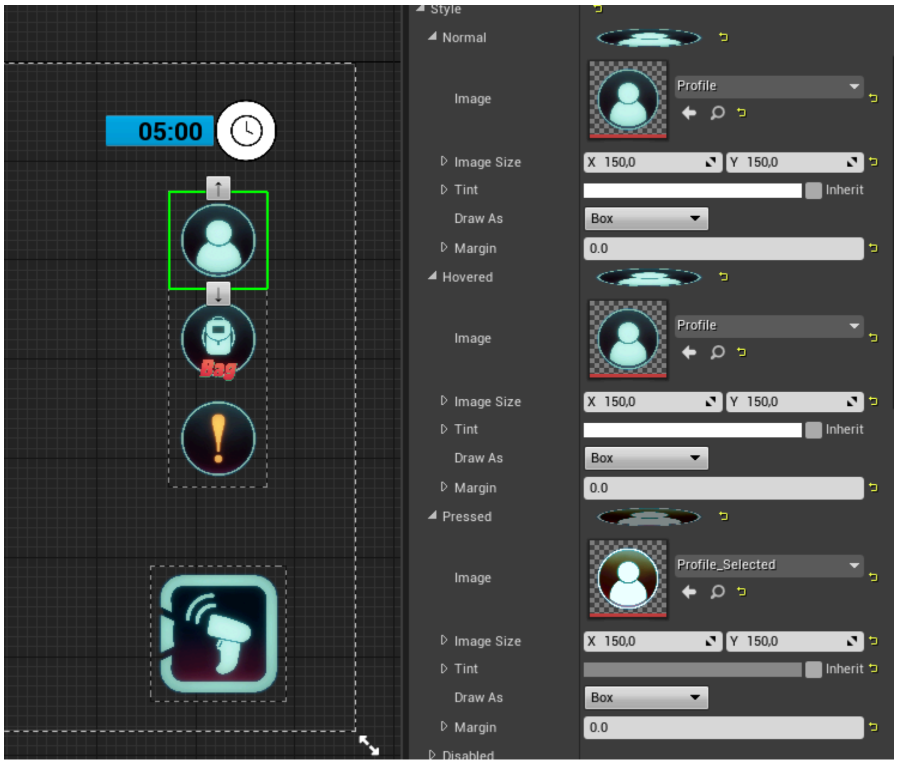
- Overlay - blocks player input behind its area
- Border - background
- Buttons - clickable areas
- Images
- Vertical & Horizontal Boxes - easy way to group the buttons
- Progress Bar
- Text



```
[BPW_ARCameraOn_Buttons]
  [Overlay]
    [Border]
    [Canvas Panel]
      [Horizontal Box - Interactions]
        Button_Alien_Play
        Button_Alien_Feed
        Button_Alien_Gravity
        Button_Alien_Hug
        Button_Alien_Portal
      [Vertical Box - Scanner]
        Button_Alien_Scanner
      [Vertical Box - Orange]
        Button_Alien_Profile
        Button_Alien_Gifts
        Button_Alien_Hints
      Button_Alien_Exit
      Bar_Alien_Trust
      Bar_Alien_Energy
      ProgressBar_1
      [time_text] "05:00"
      Bar_Alien_Timer
      [energy_text] "100"
```

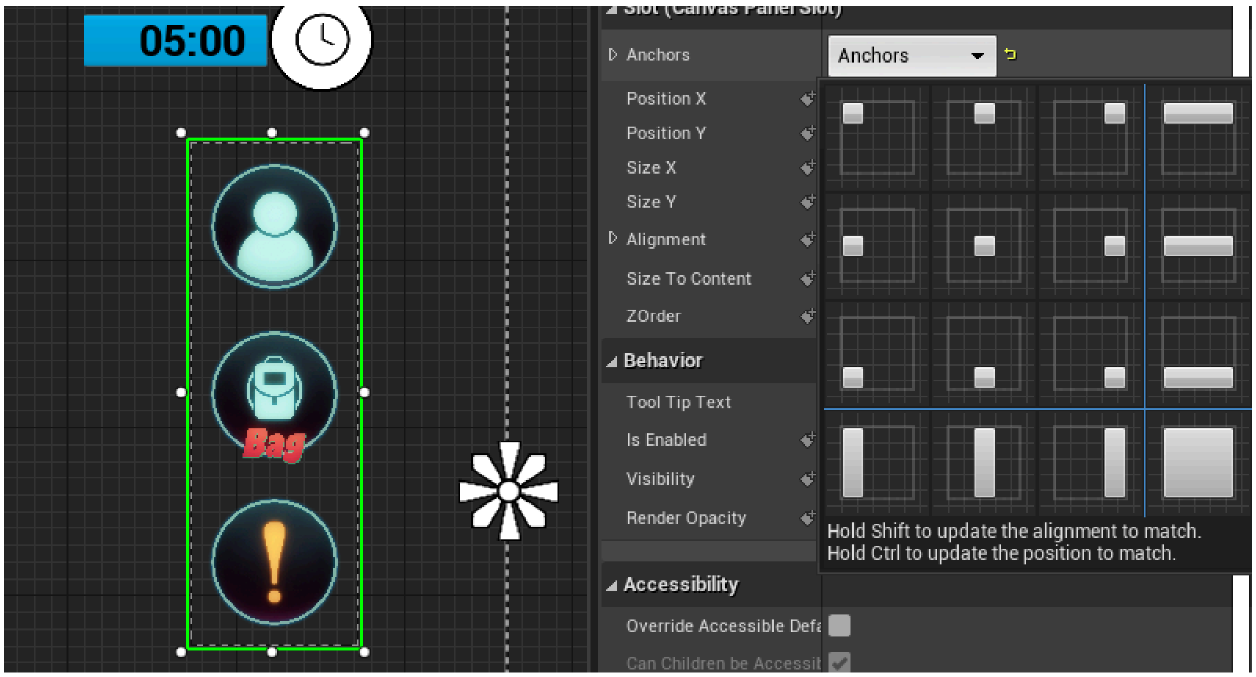
When creating a button, the following items need to be filled:

- Size & Margin (to be set to 0, default is 0.25)
- Normal, Hovered & Pressed Images for buttons
- Tint - change image color to create an illusion of a new icon



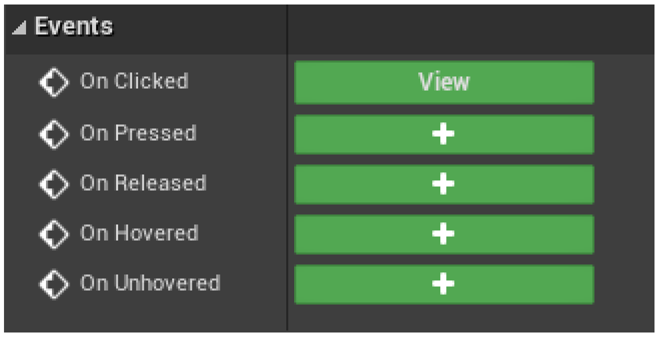
```
Style
  Normal
    Image        Profile
    Image Size   X 150,0   Y 150,0
    Tint         Inherit
    Draw As      Box
    Margin       0.0
  Hovered
    Image        Profile
    Image Size   X 150,0   Y 150,0
    Tint         Inherit
    Draw As      Box
    Margin       0.0
  Pressed
    Image        Profile_Selected
    Image Size   X 150,0   Y 150,0
    Tint         Inherit
    Draw As      Box
    Margin       0.0
  Disabled
```

After creating and grouping the buttons/objects/images, they need to be anchored. When archoned, they will be moving proportionally to the assigned point when the screen is resized on different devices.



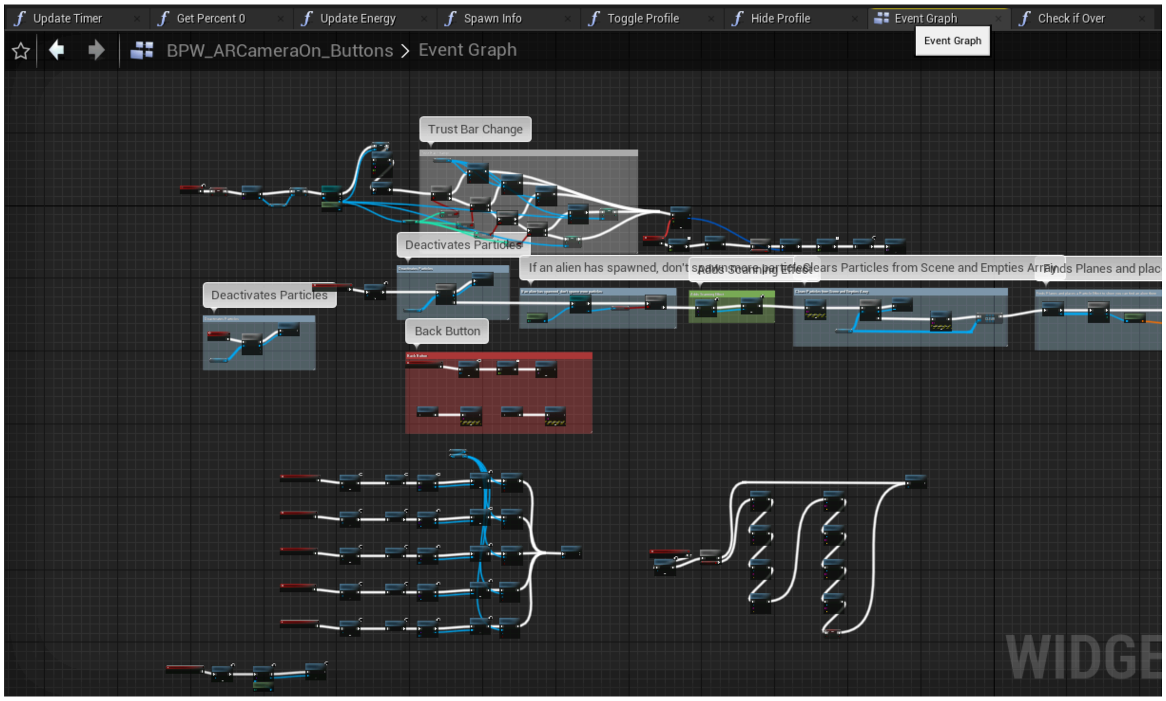
Hold Shift to update the alignment to match.
Hold Ctrl to update the position to match.

To create an event for when the player interacts with the button, scroll to the bottom of the details page and select the event you want to set up. After clicking on the +, you will be redirected to Graph Section



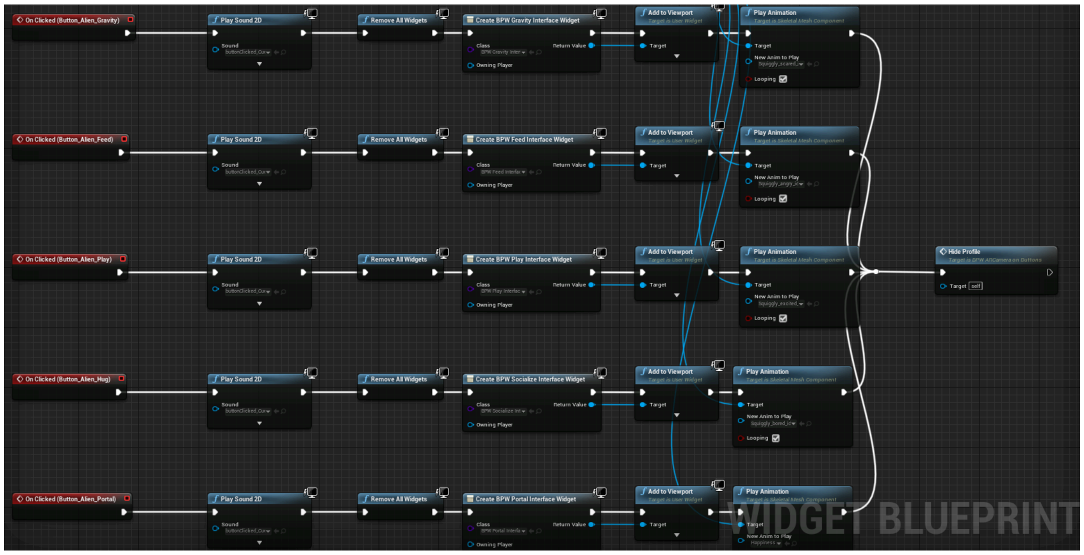
Coding Widgets

Continuing with BPW_ARCameraOn_Buttons as an example, let’s navigate to the Graph section. It consists of the functions, which are custom events called by the blueprint and Event Graph, the main work space to create interactive interfaces.



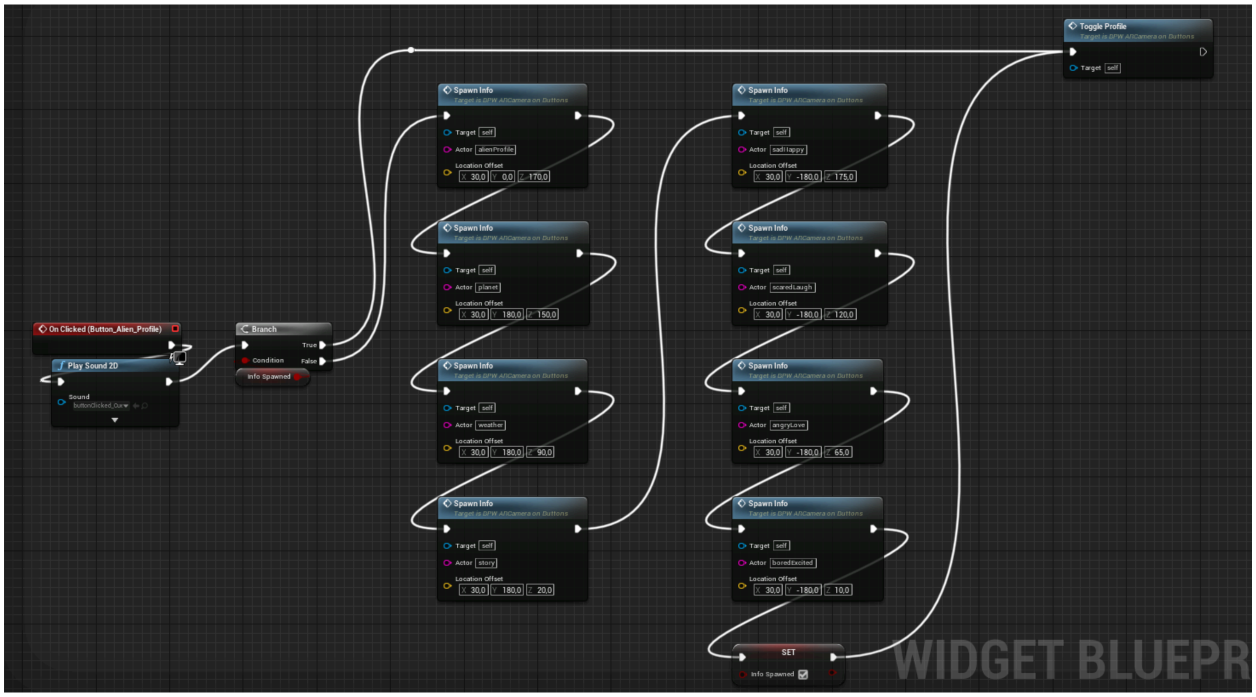
Using OnClicked as an example, the click of a button will cause a sequence of the following events to play out. In the example below:

1. Play a sound when the button is clicked
2. Remove all Widgets from viewport
3. Create new Widget for the selected activity
4. Add Widget to the viewport
5. Play animation on the Alien
6. Hide Alien profile (in case the player is viewing the Alien's profile when clicking activity)


On Clicked (Button_Alien_Gravity)
On Clicked (Button_Alien_Feed)
On Clicked (Button_Alien_Play)
On Clicked (Button_Alien_Hug)
On Clicked (Button_Alien_Portal)
Play Sound 2D
Remove All Widgets
Create BPW Gravity Interface Widget
Create BPW Feed Interface Widget
Create BPW Play Interface Widget
Create BPW Socialize Interface Widget
Create BPW Portal Interface Widget
Add to Viewport
Play Animation
Hide Profile
WIDGET BLUEPRINT

A more complicated example is how we spawn Alien profile in AR Camera-On Mode:

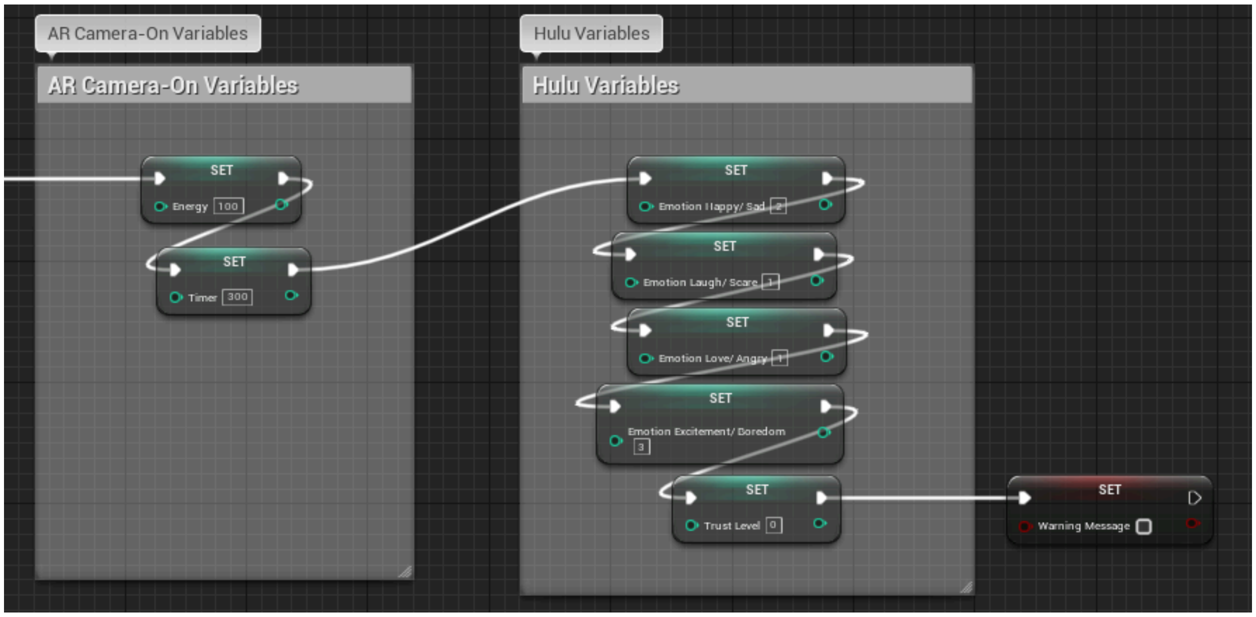
1. Play a sound when the button is clicked
2. Create an IF statement to check if the Profile is turned on
    a. If the Boolean Info Spawned is false spawn all actors separately
    b. If True, hide it with Toggle
3. For the false branch, each actor is spawned at a different location, based on the Alien's position in the environment.
4. After spawning all actors, set the Boolean Info Spawned to True, so that when the button is clicked again, the profile will be hidden



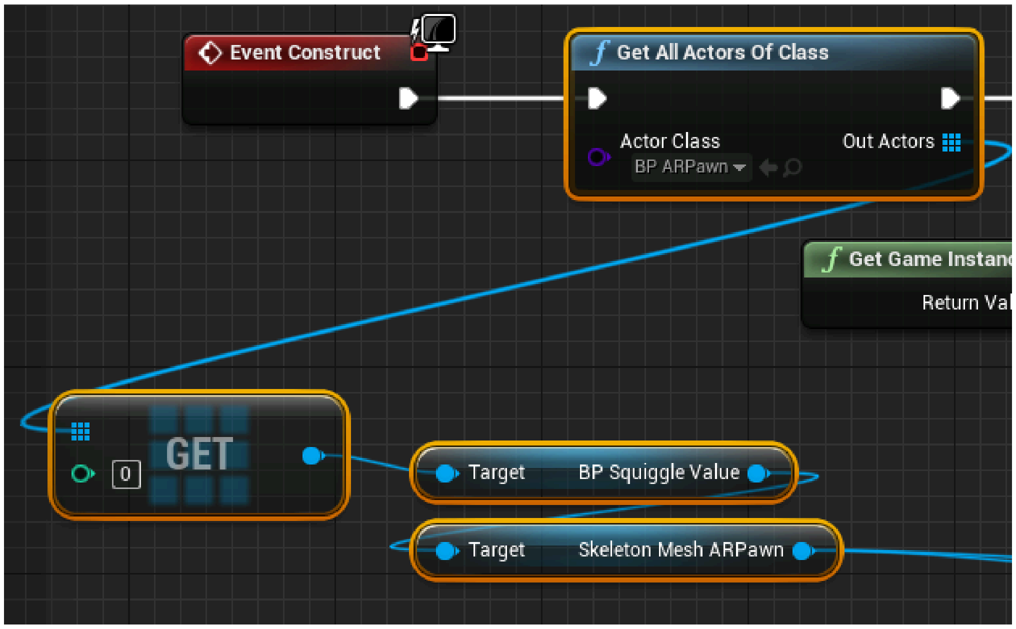
Global Variables

The global variables are the ongoing variables when using the game modes or globally playing the game. There are 2 ways to access them from separate actors and widgets:

- BP_GameInstance - an actor that is always played when the game is launched. It stores important global variables such as Emotion Variables of Aliens, Warning Message, Energy, Timer and all additional variables to be added in the future

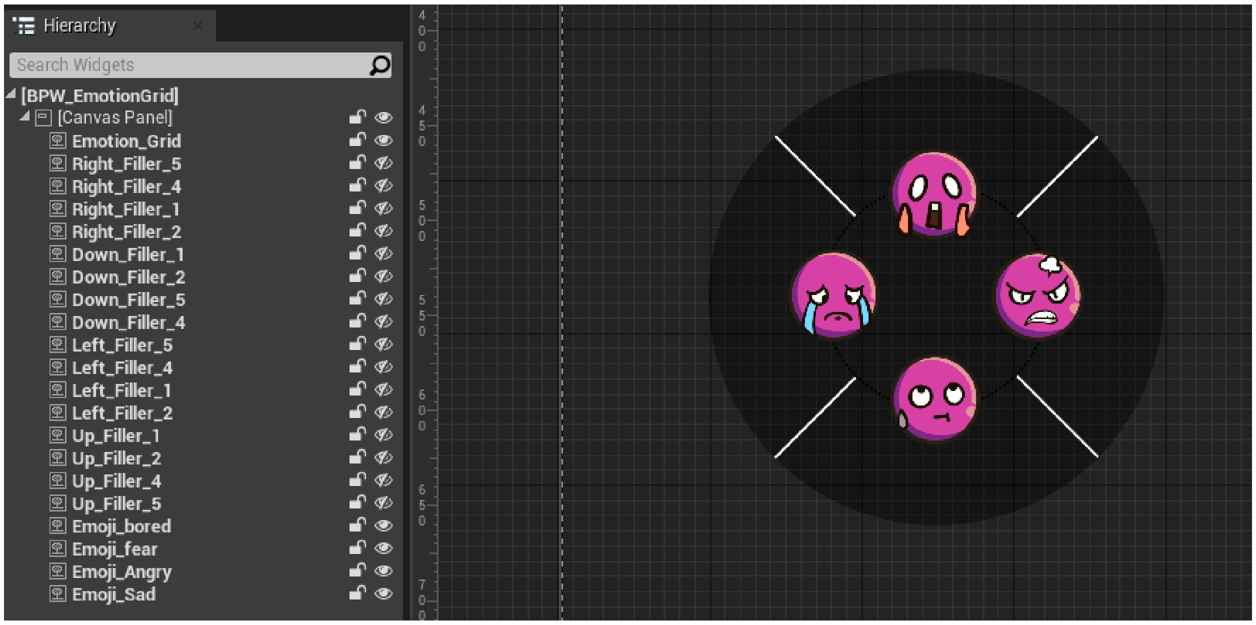


- Another method of extracting variables is by using **Get All Actors** from a Pawn. It functions in the following way:
  - Create Get All Actors blueprint
  - Use Get (a copy) to get all variables from the blueprint
  - Get the variables you need (don't forget to disable Context Sensitive)
  - The example below shows how to access Skeleton Mesh from ARPawn



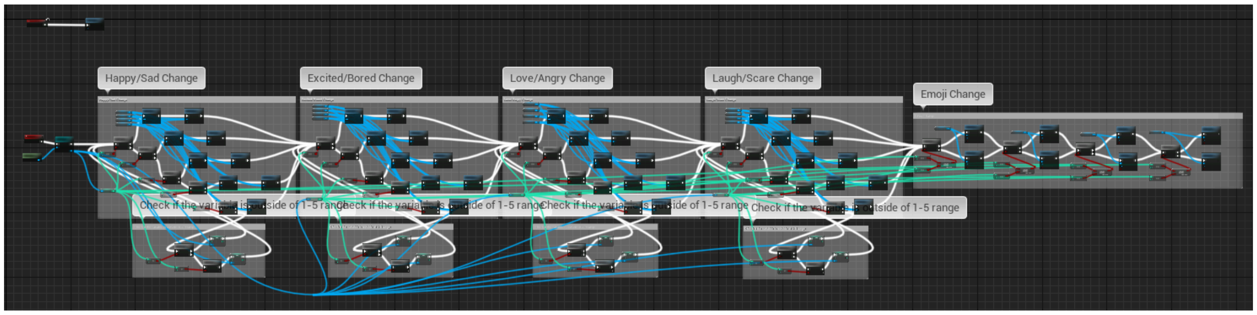
Emotion Grid

The emotion grid is stored in BPW_EmotionGrid. It consists of images for emojis, background and Fillers, which are hidden and only opened depending on the value of emotion variables.



The complicated graph for emotion grid consists of 3 main sections:

1. Check the BP_GameInstance for 4 emotion variables
2. Hide/Unhide Fillers based on the Variables
3. If variables are outside the 1-5 range, change it to the closest value. If the variable = 3, hide all fillers of the section
4. Change Emoji based on the current variables:
    a. If 1-3 then Negative Emoji
    b. If 4-5 then Positive Emoji
5. This sequence of actions is created into the Custom Event - EmotionGrid. Everytime EmotionGrid is called, it runs all of the code and showcases the results.


Happy/Sad Change
Excited/Bored Change
Love/Angry Change
Laugh/Scare Change
Emoji Change
Check if the variable is outside of 1-5 range
Check if the variable is outside of 1-5 range
Check if the variable is outside of 1-5 range
Check if the variable is outside of 1-5 range

AR Camera-On Mode

AR Camera-On mode represents interaction with the alien in the real world.

Activating Camera

For consistency, turning on the AR-Camera is done on the Player Pawn for each level. There is a bug with Android and Unreal 4.26.1 that prevents the camera from being turned off, but an Event End Play should be added with a Stop AR-Camera to turn off the camera when the player leaves the level. This will save battery life.

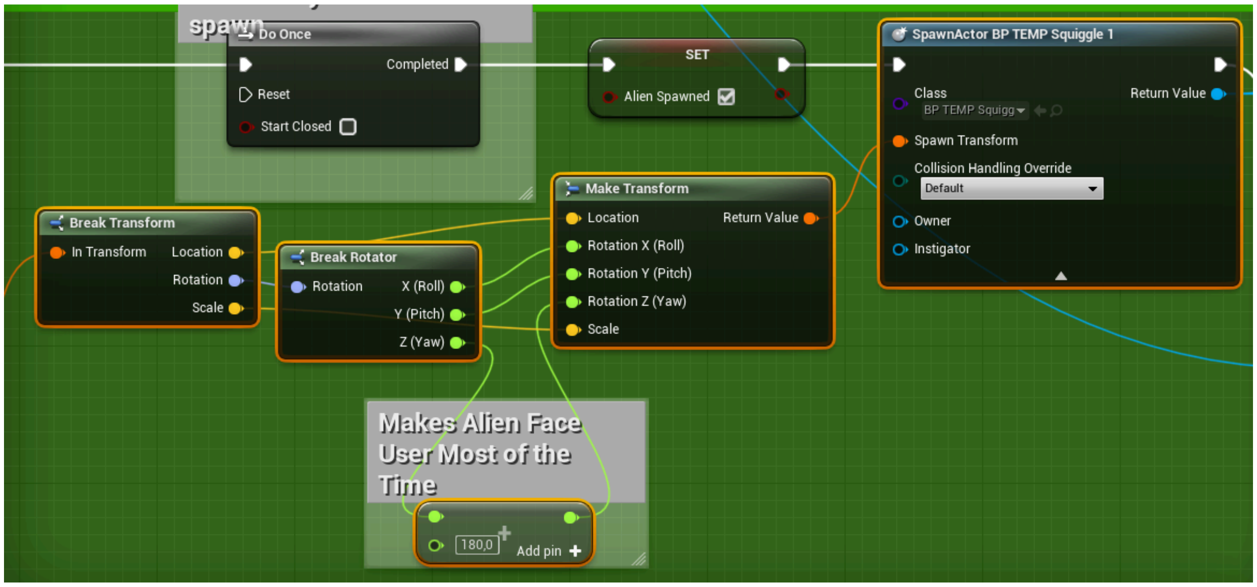
Plane Detection

Plane detection is done through the code created by the Handheld AR template. No further alterations were done to this code.

Spawning & Transforming Actors

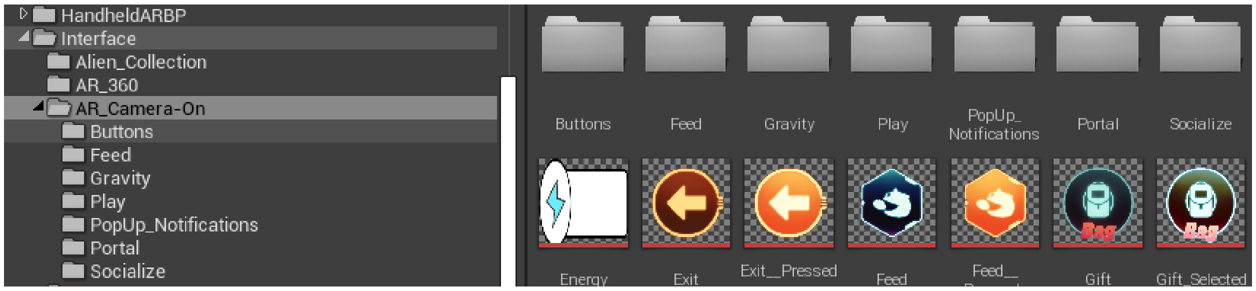
The alien is spawned with the SpawnActor blueprint. It has 2 mandatory requirements: Select Class and get Spawn Transform input. To spawn the alien in front of the camera we use the following options:

1. Get Local to World transform (Trace of your camera)
2. Break the position into location, rotation and scale to change the relative variables of the alien
3. In our case we rotate the Alien on Z axis by 180 degrees to make the Alien face the player
4. With Make Transform we change variables back into one stream
5. We connect the variables to the SpawnActor, and now the alien will be spawn relatively to the player



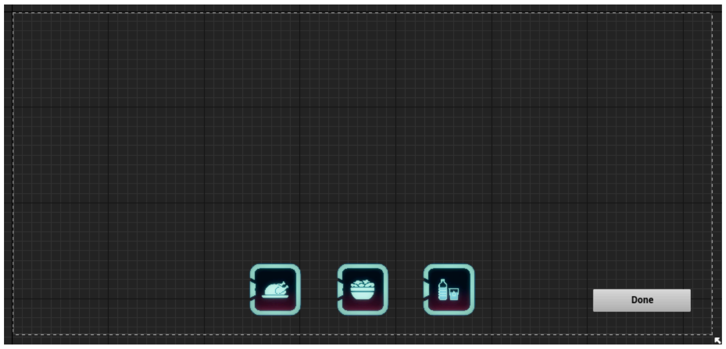
Basic Interactions

There are 5 basic interactions in the game, stored in Interfaces -> AR_Camera-On folder: feed, gravity, play, portal & socialize.

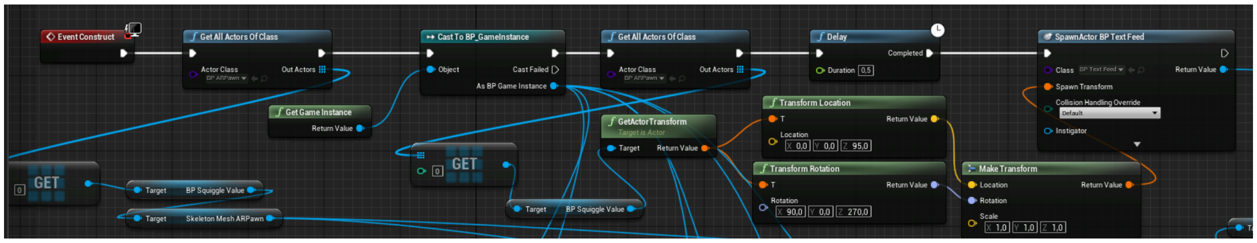


A typical interaction uses the following structure (Feed is used as an example):

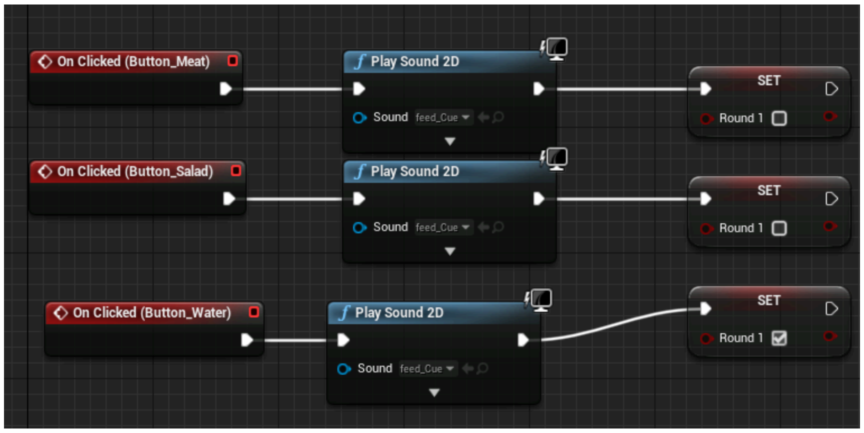
1. Remove all Widgets and add Feed Widget to viewport



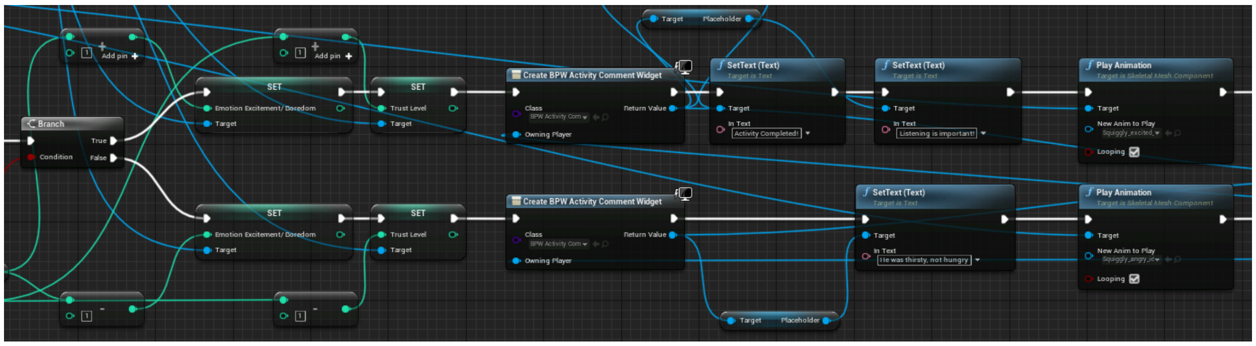
2. When the Widget is constructed, get actors from BP_ARPawn and cast to GameInstance to get all variables, and spawn the cloud on top of the alien relatively to its position in ARPawn.



3. When the button is clicked, play a sound and assign a True/False value to Round 1, depending if the player chose the correct answer



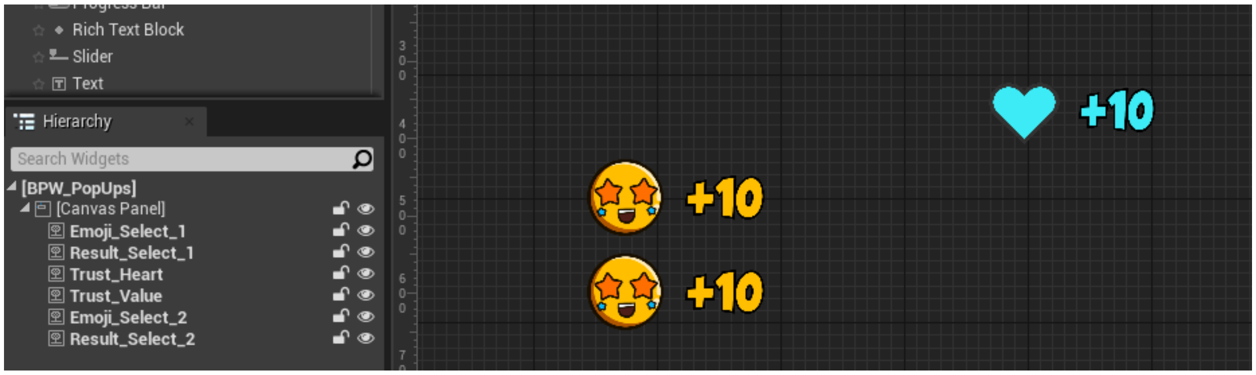
4. When the Done button is clicked, IF branch checks if the answer is correct or not. If the answer is wrong, the player gets an Activity Failed feedback and -10 values. If true, the opposite.



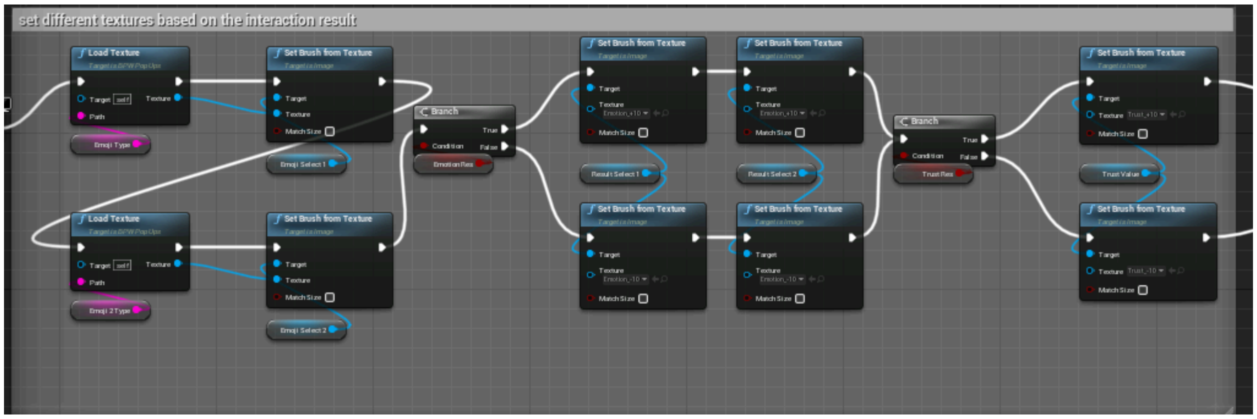
5. Initiate post-interaction feedback with relevant information about the activity.

Pop Up System

Pop Ups are stored in BPW_PopUps. They use images that can be changed with the use of Set Brush from Texture.

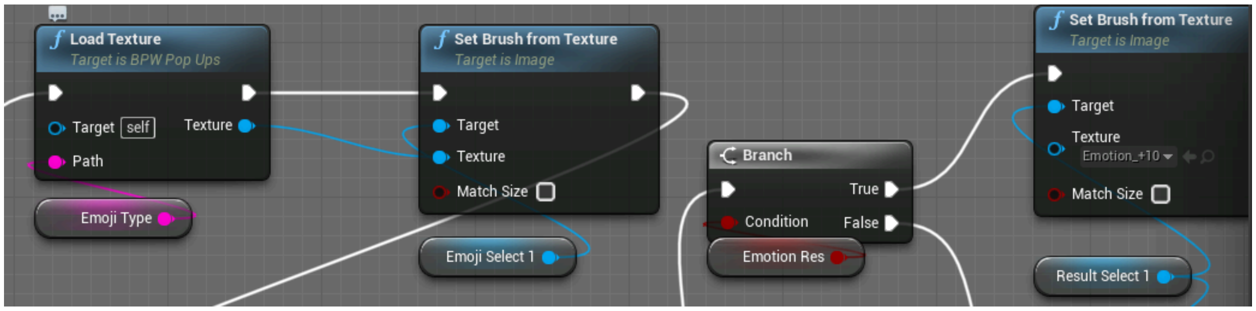


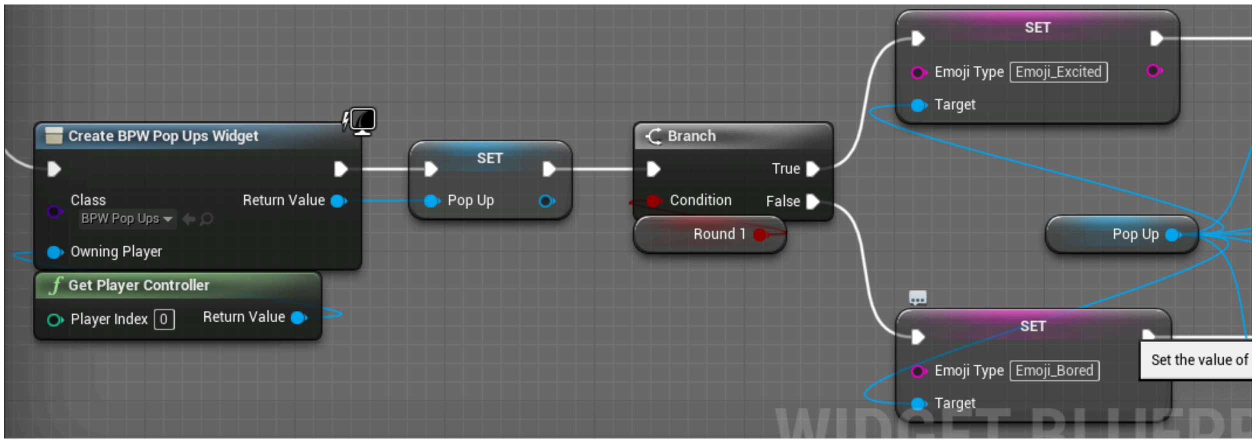
The blueprints load the correct texture, corresponding to the emotion that was changed, uses IF branch to determine if +10 or -10 needs to be shown and adds animation with the use of Play Animation.



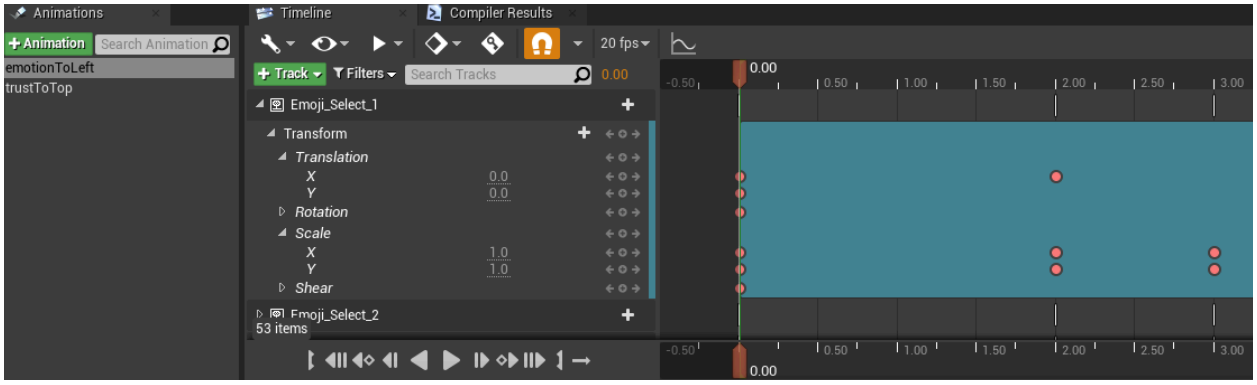
1. Feedback page
2. Pop-up symbols
   The pop-up symbols are in order to make the emotion/trust changes made by the interaction more intuitive and obvious to the player. Once it returns to the camera after the feedback page, pop-up symbols for specific emotions and trust along with the value based on that interaction result will be displayed.
   - Those pop-ups are created as widgets. Each interaction is associated with different emotions, so the type of the emotion symbol has to be reset to the correct texture. The variable "emoji_type" in ***BPW_PopUps*** blueprint is used to find the correct texture and then the texture will be applied to the emoji widget by "**SetBrushFromTexture**". It's the same idea with the value widget. Whether

there is an increase or decrease in the trust value is set in the interaction blueprint by the variable "trust_res" and the texture is changed respectively.





○ There is some animation added to the pop-up widget. By using the timeline feature of the widget blueprint, it is able to make the emoji symbols fly to the emotion grid and the trust symbol fly to the trust bar or even animate the size of the widget. This widget animation feature is really flexible and useful!



○ Involved blueprints: ***BPW_PopUps, BPW_Feed_Interface, BPW_Gravity_Interface, BPW_Play_Interface, BPW_Portal_Interface, BPW_Social_Interface.***

AR 360

AR 360 mode represents the player being able to walk around and view a 3D environment in 360 degrees.

Creating 3D Map

When laying out assets for the AR360 mode, care should be made for the purpose of the level. If the player is supposed to be able to move around the environment and look behind many 3D objects to find hidden things, then the environment should be made within a smaller space. Unreal works in centimeters, so a 200 unit wide room would allow for easy movement within a typical play area for a child. If instead, you don't want the player to be able to explore too much of the 3D environment, making the room bigger would prevent them from accidentally running through 3D walls or outside of the intended zone. Once the room is laid out, be sure to assign a Game Mode Override on the World Settings page so that your correct pawns spawn into the map.

Spawning player in AR 360

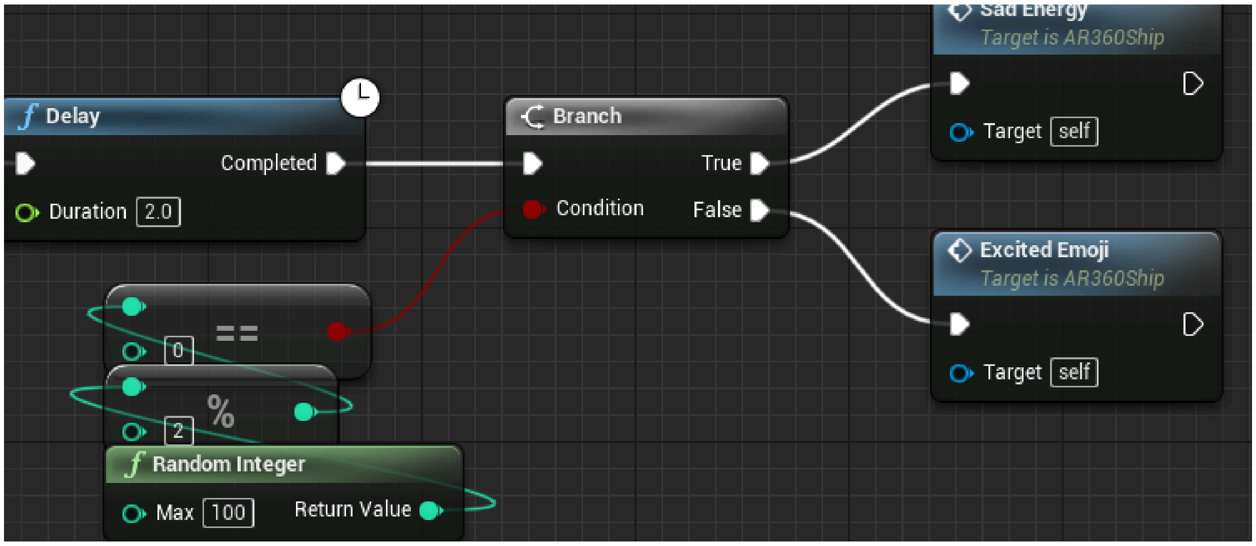
The player will spawn at the Player Start of the map. Position the arrow in the direction that you want the Pawn to be facing when loaded into the level. Placing the Player Start at a position between 100-140 units in +Z provides a good "eye level" for the player. The pawn that gets spawned in is define in the Game Mode assigned in the Game Mode Override in World Settings.

Detecting Players's Position

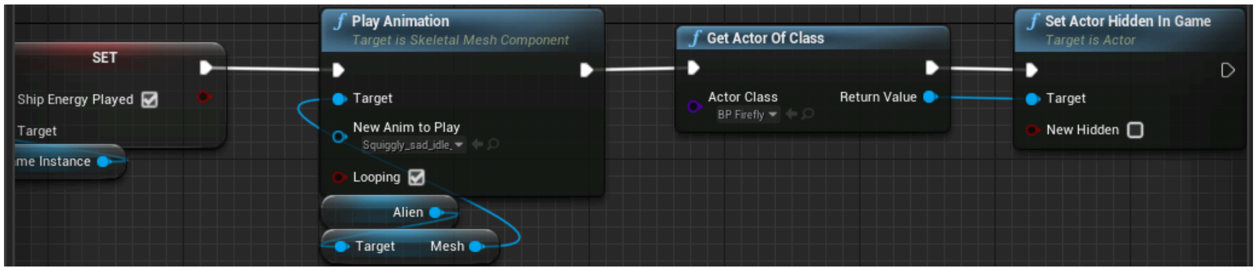
When the level loads, the game will need a few moments to detect where the ground plane and walls are. Unless the game has this information, shifting and swaying of the room might occur. This is the main reason for why we use the Scanner in order to find the aliens. It gives the game some time to scan the room and detect where the floor is. For developing other games, it is recommended to find a way to have the player scan the room before spawning anything in such as turning on the AR Camera during an intro scene or tutorial screen before the gameplay begins so it can scan the room in the background. It also helps to have the camera move towards and away from the player in order for the app to have a good sense of the depth of the room.

Random Triggers for Mini-Games

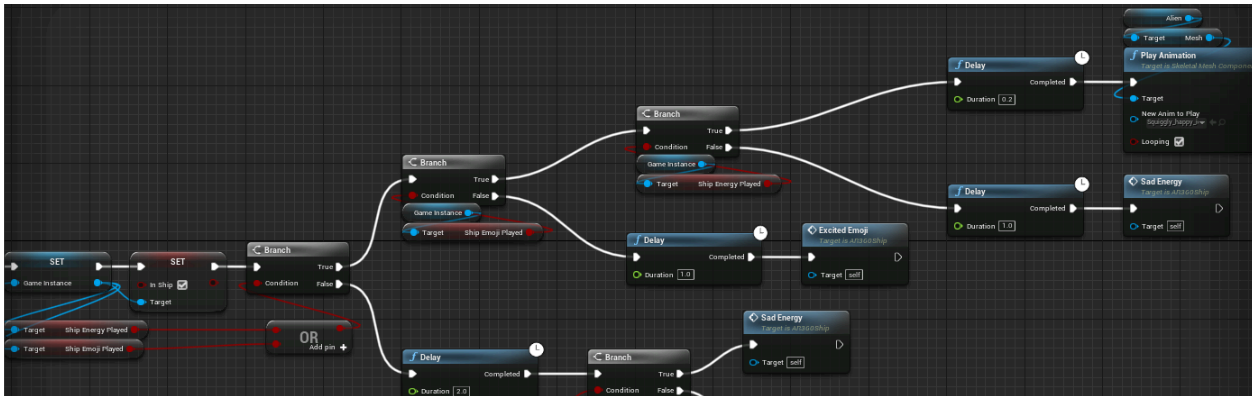
Our goal is to make the game flow more clear and natural for the player. Therefore, we added this random mini-game trigger in the AR 360 maps. When the player enters the AR 360 map, either the energy game or the emoji game will be spawned with a 50/50 chance which is decided by a random integer in the blueprint.



To make things simple, the way to spawn the game here is not by calling spawnActor. Instead, it is done by just setting the game actor in the map to unhidden.



There is a boolean for both games to indicate if this game has been played yet (“shipEnergyPlayed” & “shipEmojiPlayed”). When the first random game is triggered, the corresponding boolean will be set to true. Then when it returns to this map from that game, we know now it’s time to play the other game. Or if both games are already played, a happy animation will be played on the alien and both game widgets are unhidden.



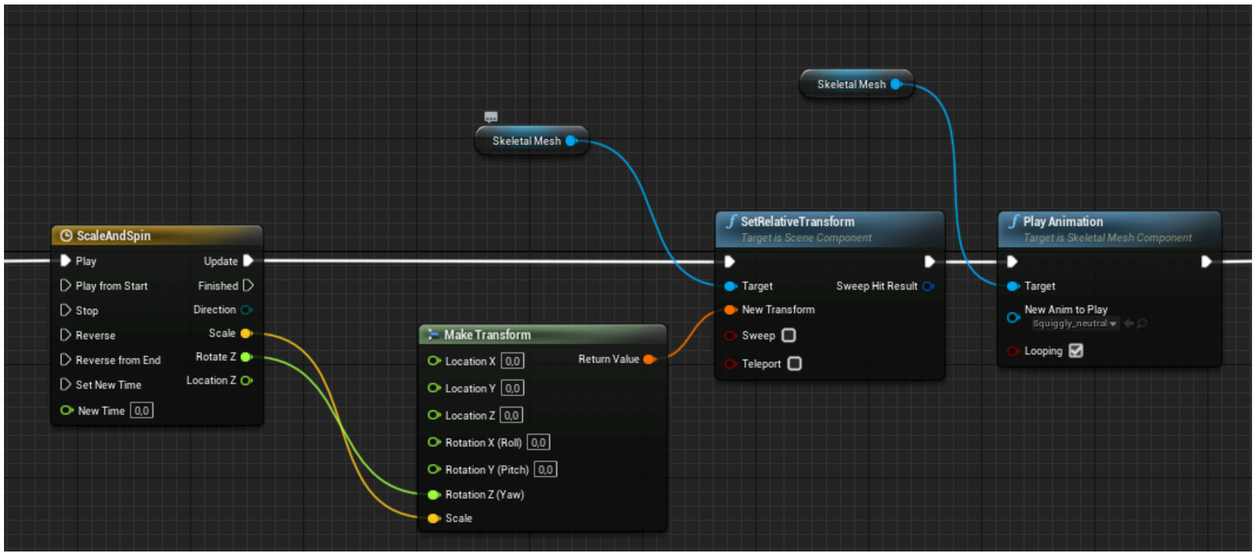
Mini-Games

Mini-Games are initiated in AR 360 environments.

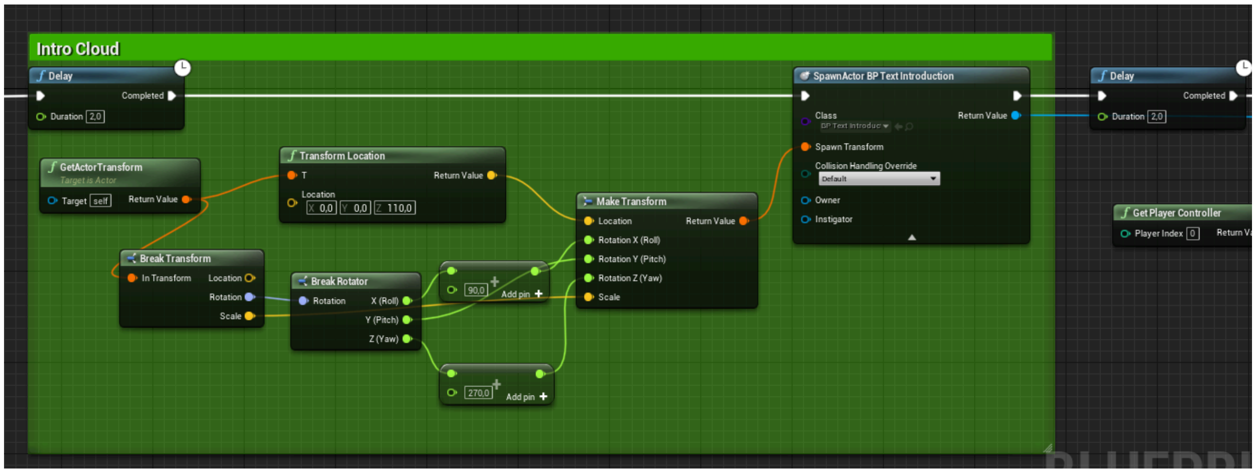
Emoji Game

Emoji Game is initiated and contains all of the blueprints in BP_Squiggle_Emoji_Game. It has the following structure:

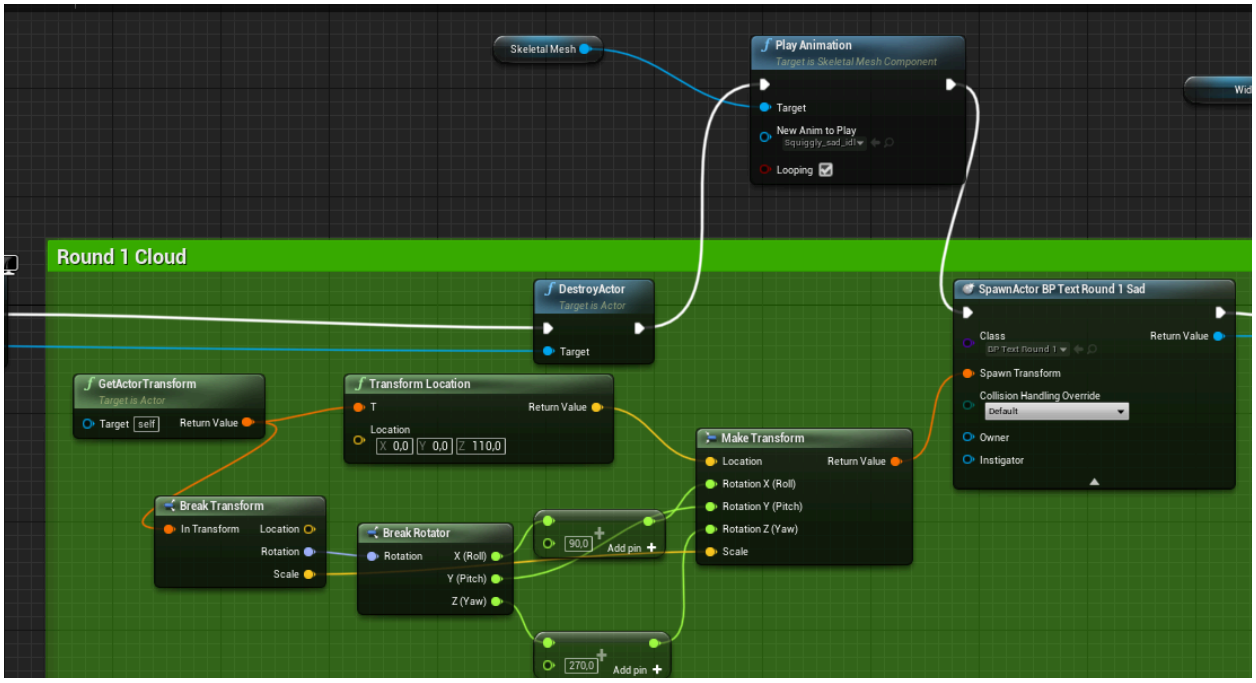
1. Spawn the Alien with Scale and Spin Animation & play a neutral animation



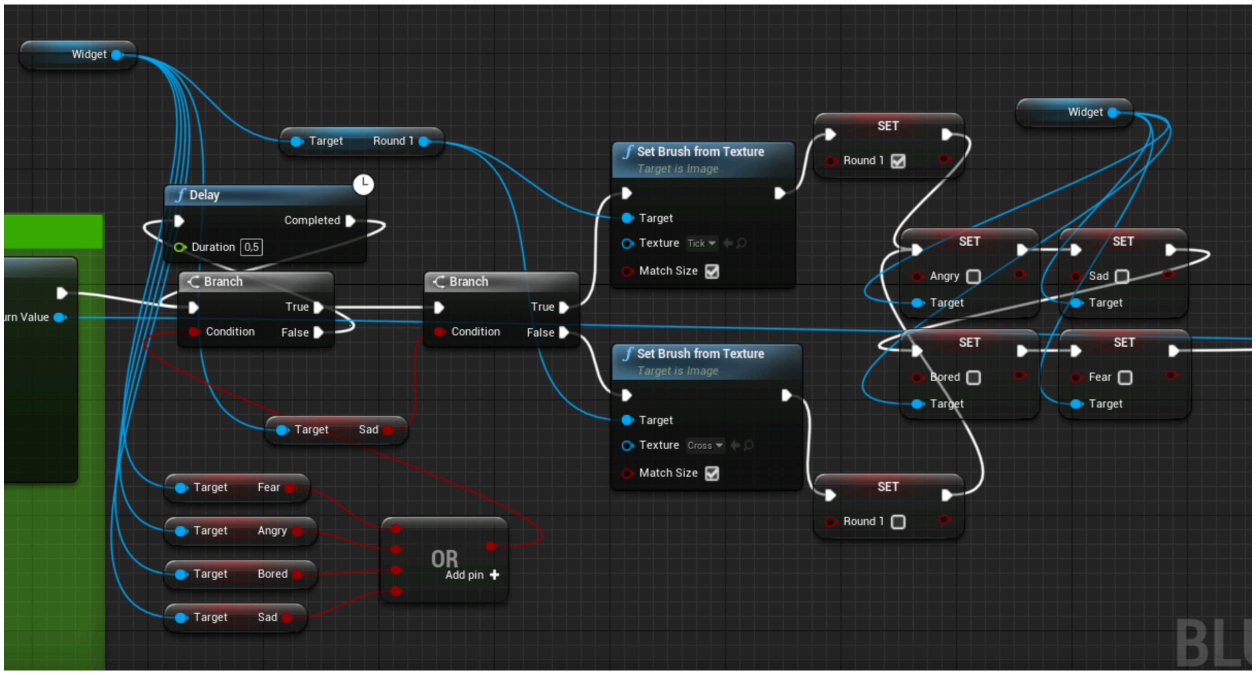
2. Spawn a text cloud on top of the alien, relative to its position and wait for the player to read the instructions



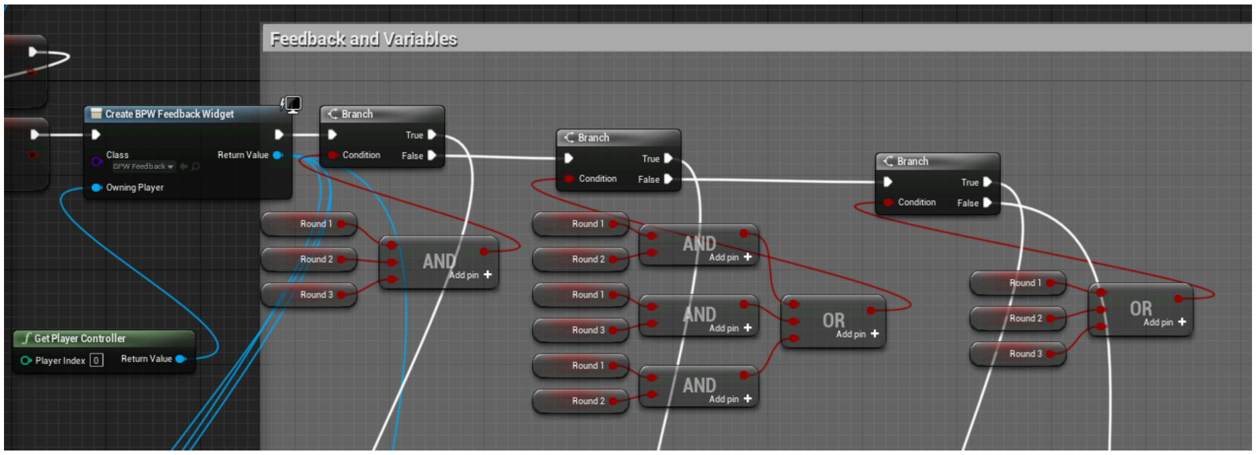
3. Spawn a new text cloud with a small story and play the animation corresponding to the cloud



4. Spawn a text cloud on top of the alien, relative to its position and wait for the player to read the instructions. Loop the IF branch until the player makes a selection, keep checking if something is clicked with Boolean OR check. When something is clicked, check if the correct emoji is clicked with IF branch. Use Set Brush from Texture to show a Tick or a Cross, depending if the answer is correct or now. Reset the Boolean of emoji selection and repeat Steps 3&4 three times.



5. Check how many rounds were correct and display relevant information if TRUE.

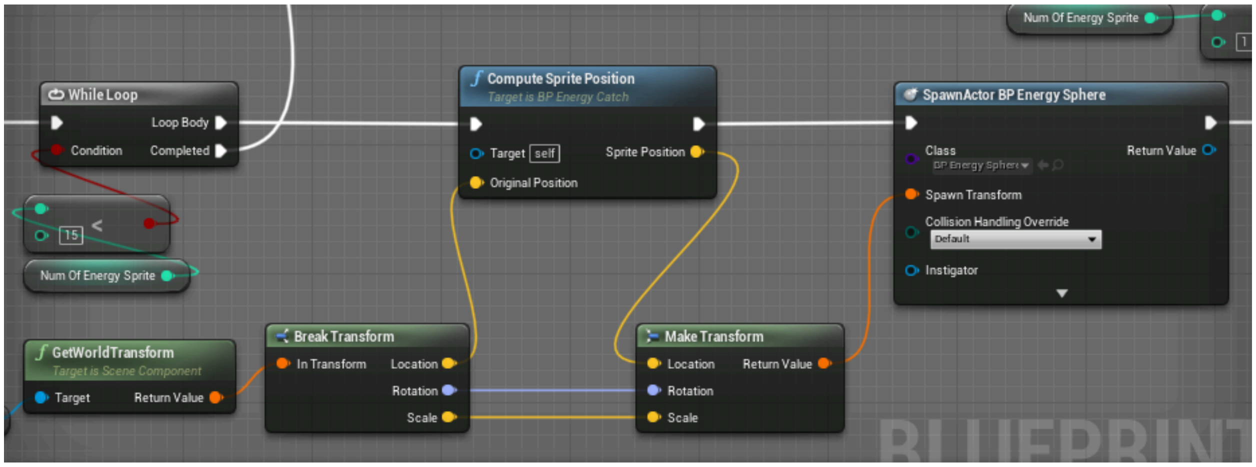


6. Get feedback when you finish the activity.

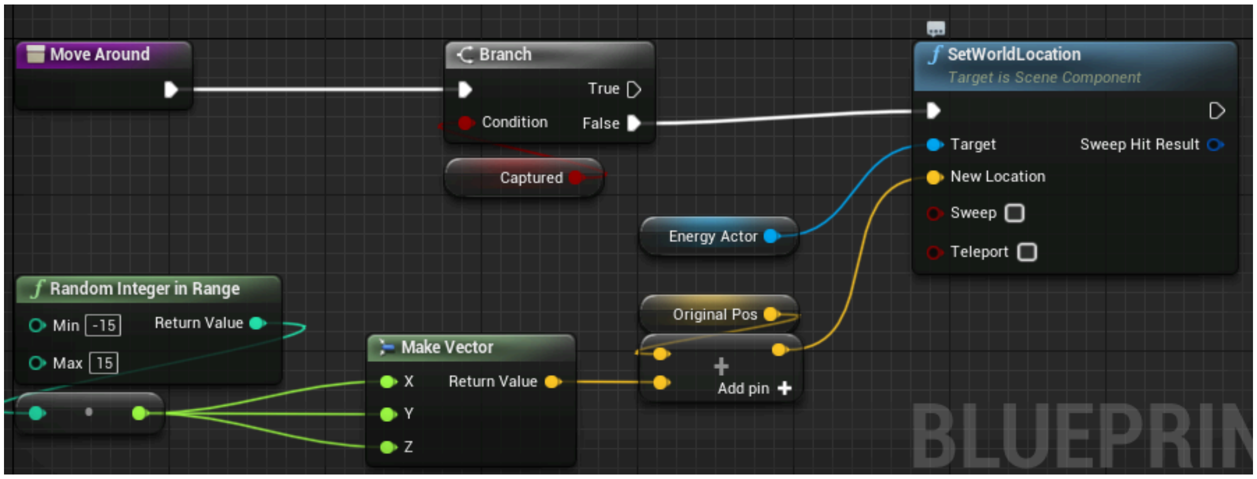
Energy Game

There are several key features of this mini game that are implemented:

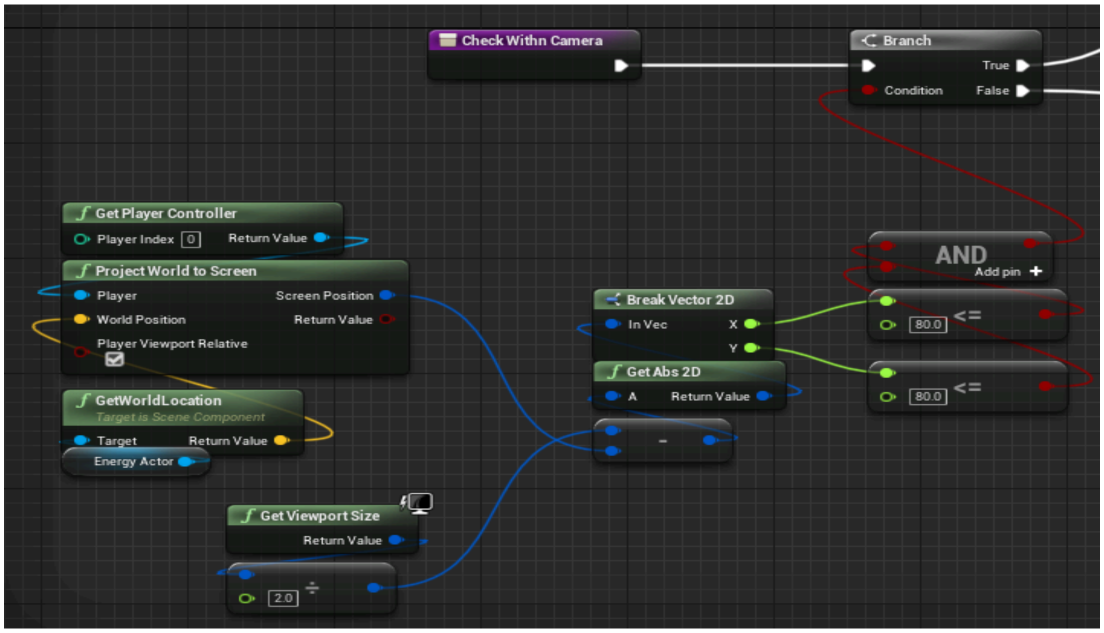
1. Energy spheres are spawned around the alien in space (of the desert or ship AR 360 map). A WhileLoop is used to spawn 15 energy actors. This number can be adjusted by setting the condition of the loop. For the location of each spawned energy, it is calculated by a function "**computeSpritePosition**". In order to put the energies in a random location, the function adds a random offset to the xyz coordinates of the energy actor. The range of the random offset is set according to the space range of the AR360 environment so that the energies will not be spawned outside of the desert or ship. Hence, this range can also be adjusted if the size of the environment is changed. *The related code is in the **BP_EnergyCatch** blueprint.*



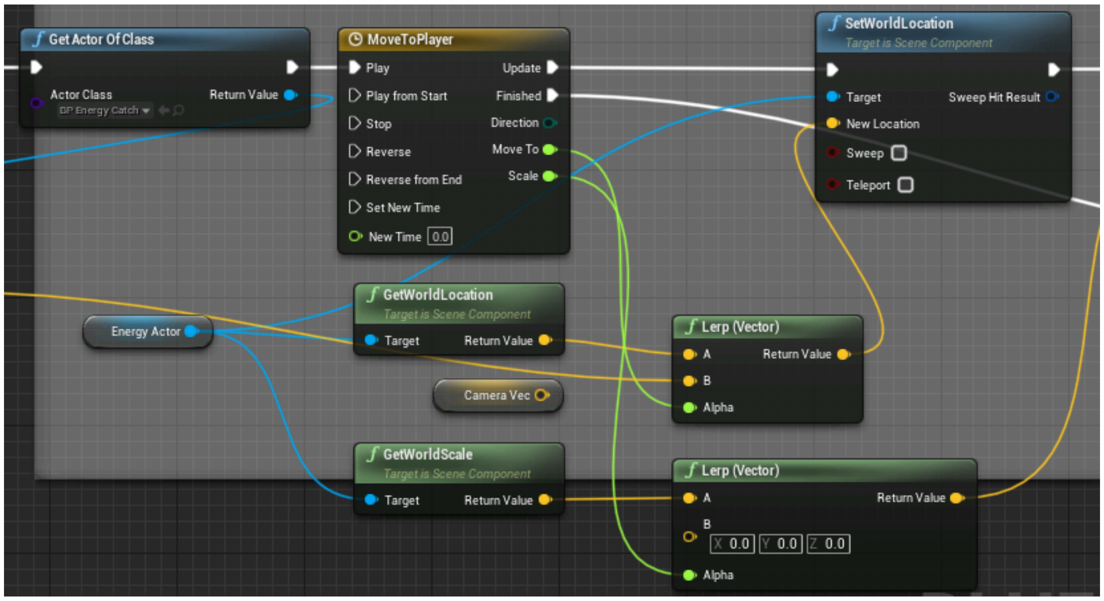
2. After an energy is spawned, it will have a simple moving-around animation. This is achieved by the function “**moveAround**” (*). This function just updates the energy to a new position by adding some offset to the original position. How large this movement is can be controlled by setting the offset which is generated by the RandomInteger node. *The related code is in the **BP_EnergySphere** blueprint.*

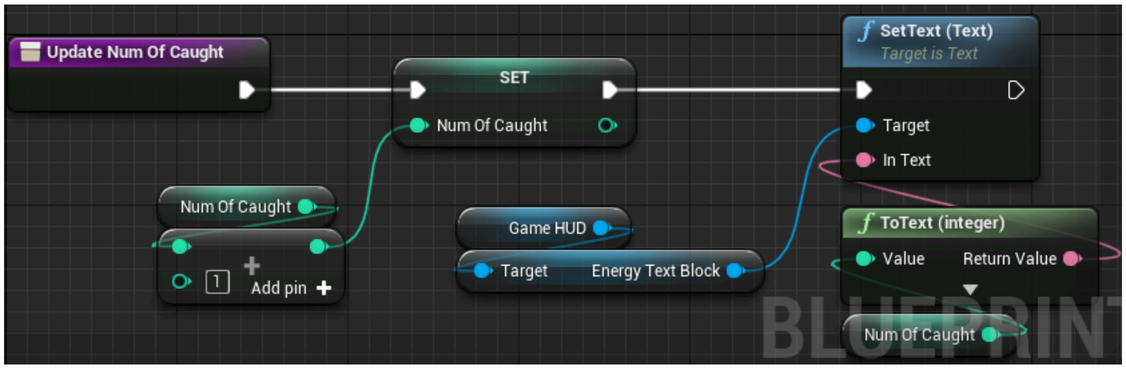


3. When the energy actor is at the center of the screen as the player moves the camera around, the color of the energy will be changed into blue. Or else it will be the original color. This is achieved by a function “**checkWithnCamera**” (*). The node “**projectWorldToScreen**” returns the screen location of the energy actor which can be used to check if that’s around the screen center (i.e., if the energy is within range to change its color). *The related code is in the **BP_EnergySphere** blueprint.*

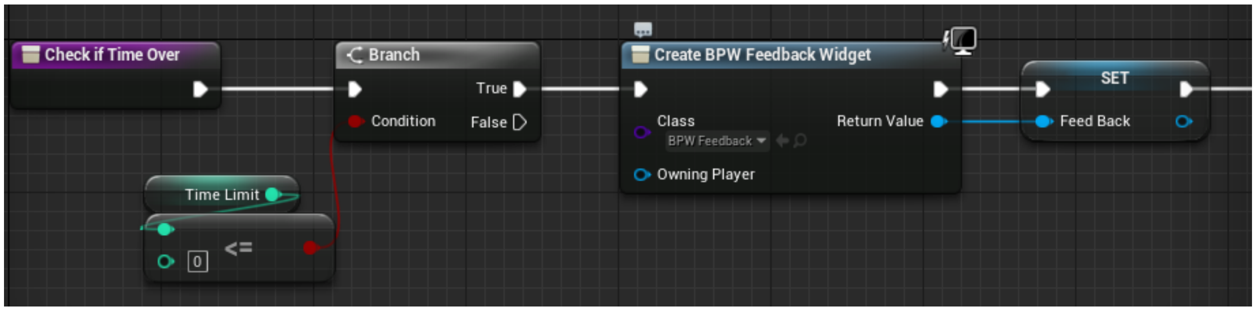


4. When an energy is caught, the variable "numOfCaught" of the BP_EnergyCatch class will be updated and this will be reflected on the "Energy Caught" text by the "updateNumOfCaught" function. Moreover, an animation will be played on the energy: the size is blown up first and then minimized to disappear and the energy will fly towards the screen first and then fly to the alien. This is done by the timeline "**MoveToPlayer**" which controls the path by "MoveTo" component and size of the energy by "Scale" component. *The related code is in the **BP_EnergySphere** and **BPW_EnergyGame** blueprint.*

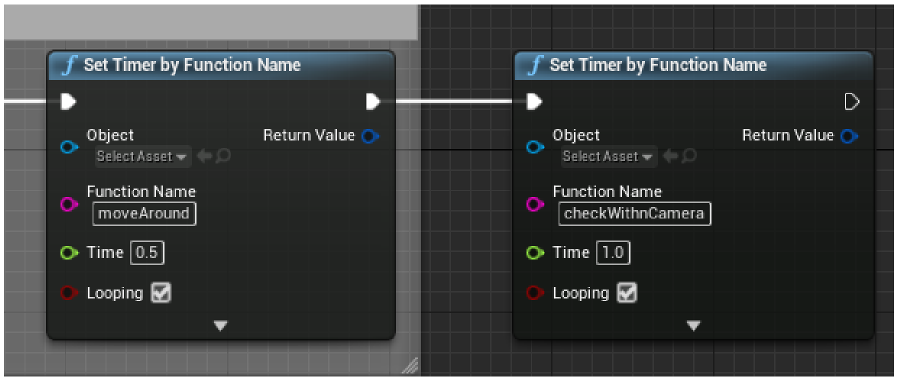




5. For this mini game, there is a time limit which can be adjusted by the developer. The left time is shown on the screen. When the time is up, the game will end. And then the different results (feedback) will be displayed based on whether you win or lose the game. The function "**checkIfTimeOver**" (*) will check if it reaches the time limit. *The related code is in the **BP_EnergyCatch** and **BPW_EnergyGame** blueprint.*

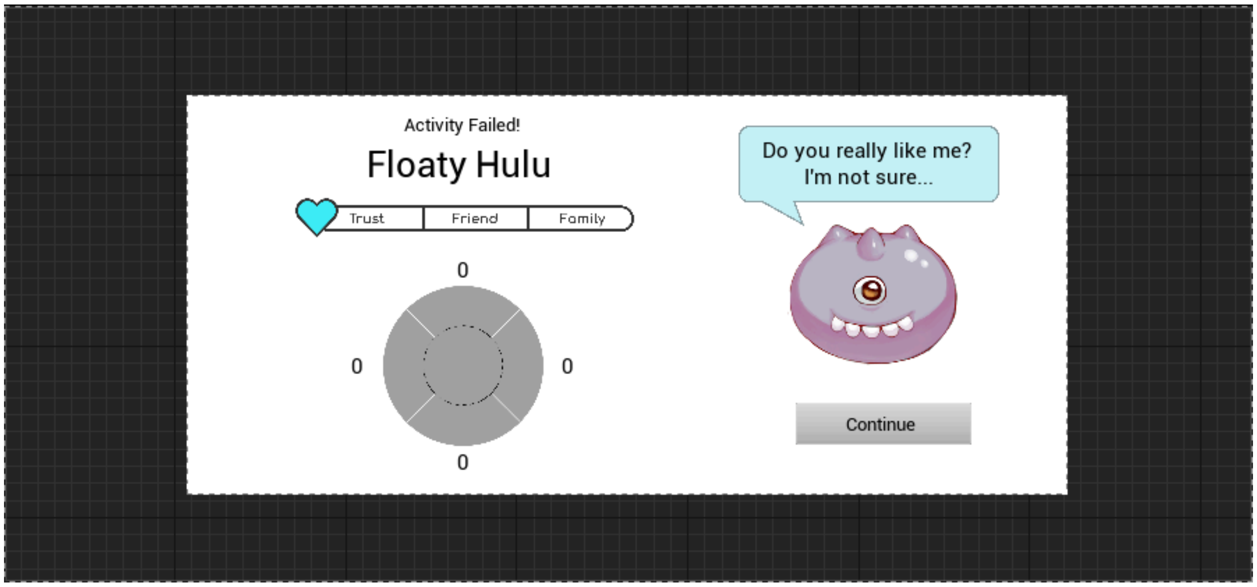


(*) Remark: Those functions are called every second by the "SetTimerByFunctionName" because the corresponding actions are expected to run every second.

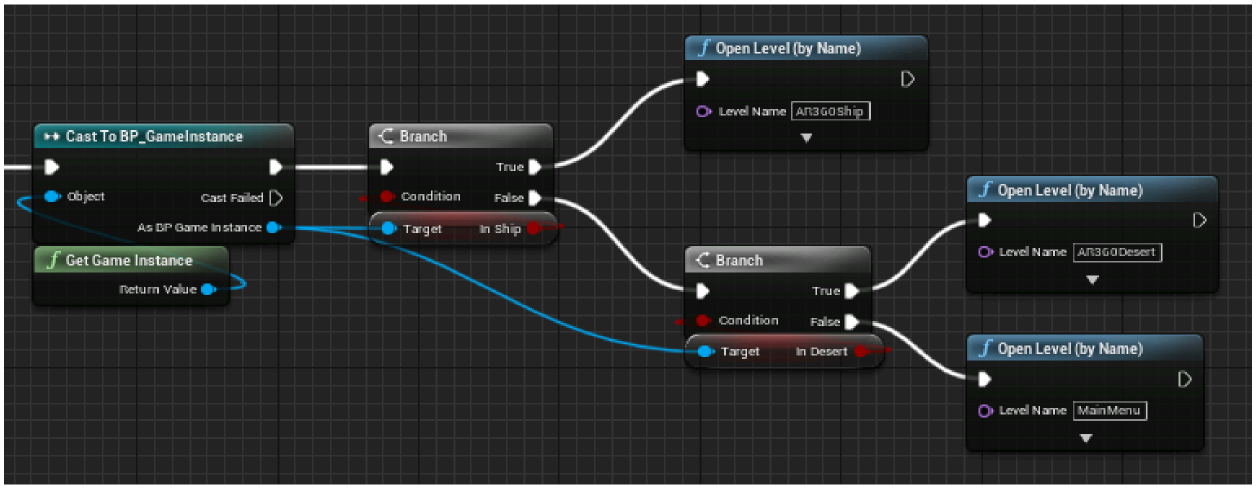


Post-Game Feedback

When the game is completed an interactive feedback page is displayed. It uses EmotionGrid widget and changes zeros around the graph to correct variables.



When clicking the Continue button, the player will be sent back to the level, where the player came from, using global instances. If the game was initiated from the Ship, the boolean In Ship will be true, same applies for the rest of the variables. If all are false, then the player is returned to the main menu (if the game was accessed through debug in the main menu).

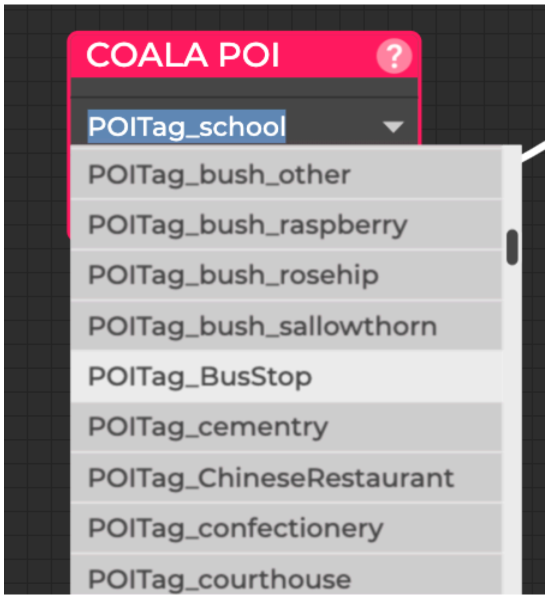


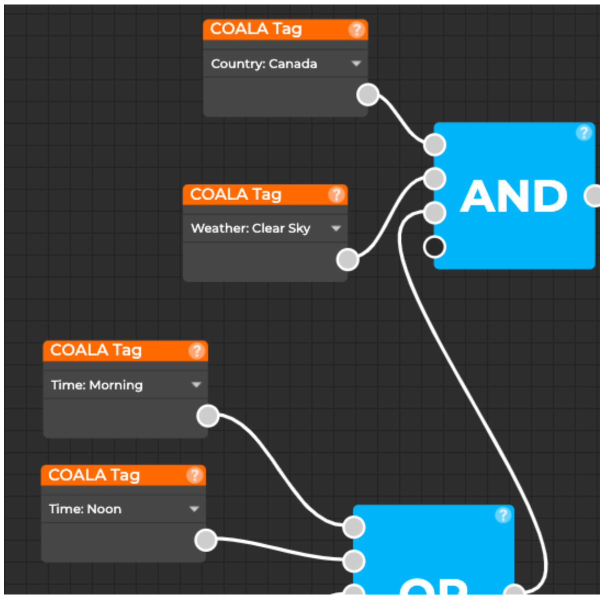
AR Explore

This section breaks down the usage of Coala Plugin and required blueprints.

Coala Back-End

In the back-end, the developer can add and design any type of POIs. Under the “Points of Interest” tab, you can modify the normal POIs of your Coala map. By setting the Coala POI tag, the landmark of this POI can be defined. There are also other properties of the POI that can be self-defined like country, weather and time so that this POI will only appear when the conditions are met.



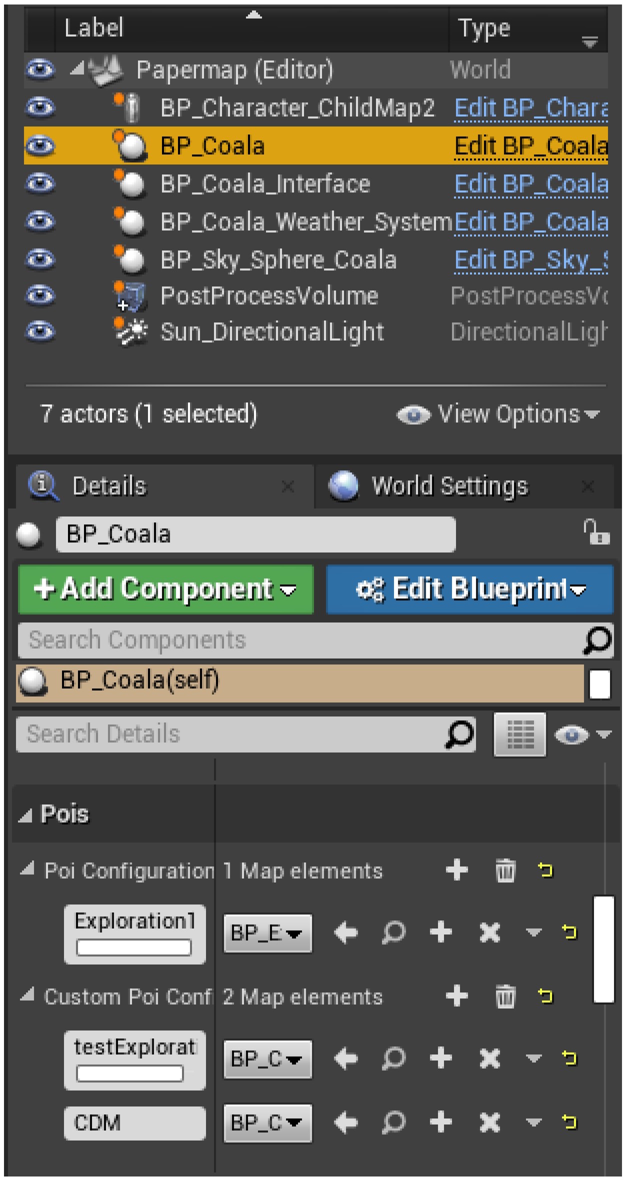


Under the “Custom Data” tab, the developer can define any custom POI which has its own geo-position. Then that POI will be displayed at that location on the Coala map.

For more explanation and details, please check the documentation provided by Coala: https://sites.google.com/a/thoughtfish.de/coalapluginunreal/coalabackend

Points of Interest (PoI)

Both the normal POIs and custom POIs can be controlled in the ***papermap*** under the ***BP_Coala*** component of the Coala plugin.

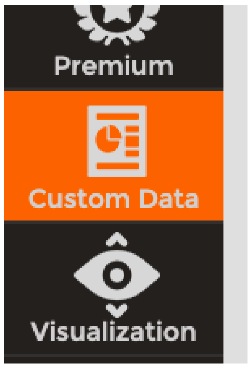


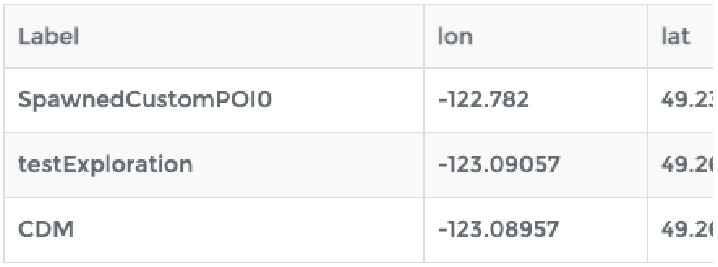
Label	Type
Papermap (Editor)	World
BP_Character_ChildMap2	Edit BP_Chara
BP_Coala	Edit BP_Coala
BP_Coala_Interface	Edit BP_Coala
BP_Coala_Weather_System	Edit BP_Coala
BP_Sky_Sphere_Coala	Edit BP_Sky_S
PostProcessVolume	PostProcessV
Sun_DirectionalLight	DirectionalLigh

7 actors (1 selected)

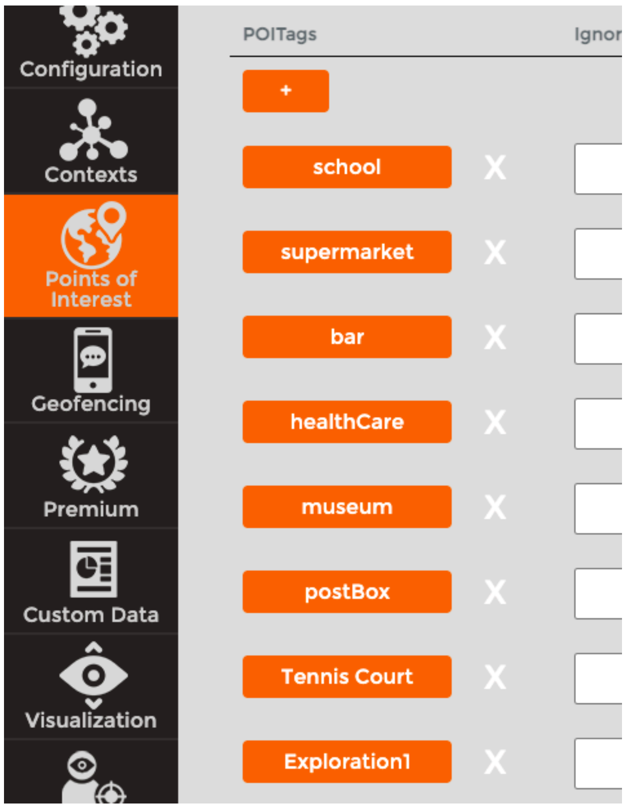
Poi Configuration	1 Map elements
Exploration1	BP_E
Custom Poi Conf	2 Map elements
testExplorat	BP_C
CDM	BP_C

Here you can use the POIs from the Coala backend. To add more Pois, click the plus sign besides the “**Map elements**”. Then the Pois will be added to the map automatically. One reminder is the name of the Poi configuration should match the one you want to use from the backend: for custom Pois, it should match the label name; for normal Pois, it should match the POITag name. Then the plugin is able to get the correct data of that Poi. And you can also set any blueprint class to be the blueprint of normal Pois. However, you could only use the BP_Custom_Poi (or its child blueprint) for the custom Pois. This is a known limitation of Coala.

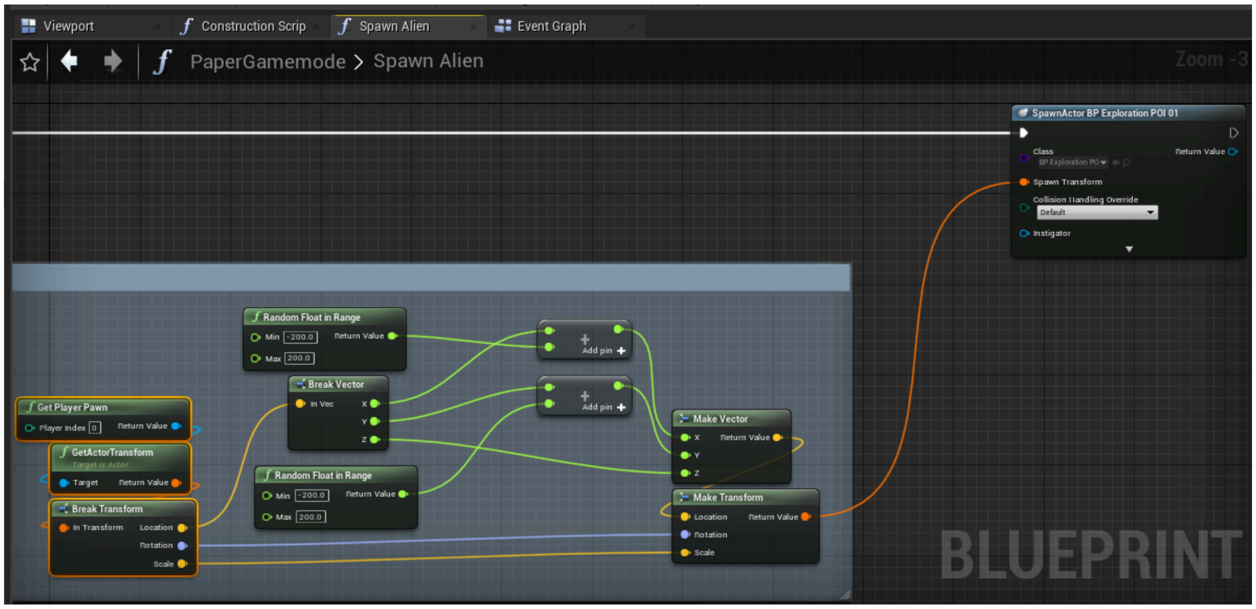




Label	lon	lat
SpawnedCustomPOI0	-122.782	49.2
testExploration	-123.09057	49.2
CDM	-123.08957	49.2

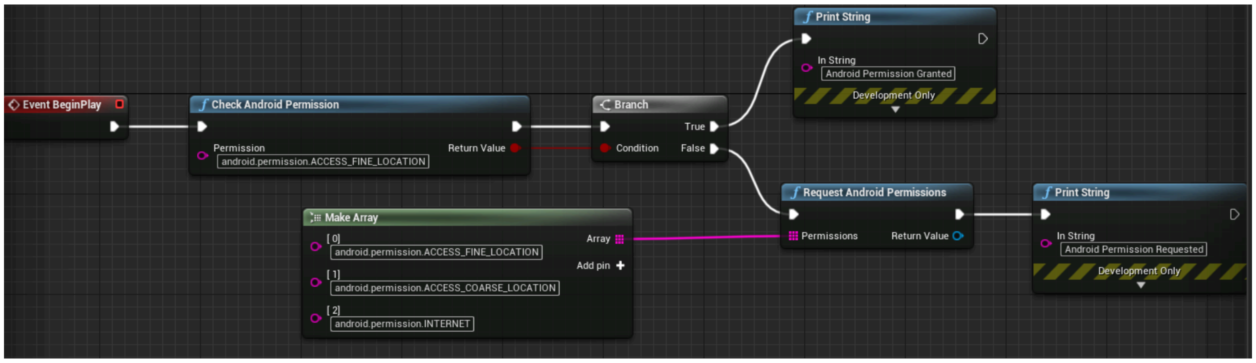


In our papermap, there will be a normal Poi that’s spawned at some random interval and within a random range around the player’s location. This is handled by the function “**SpawnAlien**” in the ***PaperGameMode*** blueprint.



GPS Detection & GPS Map

The GPS technology is handled by the Coala plugin. The only thing we need to do here is to ask GPS permission from Android devices in the level blueprint of ***Papermap***. And then the plugin will use the GPS information from the device to adjust the map.


Event BeginPlay
Check Android Permission
Permission
android.permission.ACCESS_FINE_LOCATION
Return Value
Branch
Condition
True
False
Print String
In String
Android Permission Granted
Development Only
Make Array
[0] android.permission.ACCESS_FINE_LOCATION
[1] android.permission.ACCESS_COARSE_LOCATION
[2] android.permission.INTERNET
Array
Add pin
Request Android Permissions
Permissions
Return Value
Print String
In String
Android Permission Requested
Development Only