Multicopter Blueprint Model Import Experiment Based on RflySim3D Interface

1. Experiment Objective

Familiarize and master the process of importing blueprint models into RflySim3D.

2. Experiment Requirements

- Software requirements: Windows 10 or higher; RflySim toolchain[1]; UE4.27.2; UE5.2.
- Hardware requirements: 1 laptop/desktop computer[2].

3. Experiment Location

Example directory:
[Installation Directory]\RflySimAPIs\3.RflySim3DUE\2.AdvExps\e2_BlueprintModel\1.BlueprintBuild

- ./Droneyee_WithBP_cooked_UE4: The Droneyee_WithBP folder inside contains baked copters (with XML, can be directly imported into RflySim3D)
- ./Droneyee_WithBP_content_UE4: Blueprint models edited in UE4.27.2, can be imported and opened in projects using UE4.27.2 or higher
- ./Droneyee_WithBP_cooked_UE5: The Droneyee_WithBP folder inside contains baked copters (with XML, can be directly imported into RflySimUE5, **full version of RflySim only**)
- ./Droneyee_WithBP_content_UE5: Blueprint models edited in UE5.1.1, can be imported and opened in projects using UE5.1.1 or higher (must be baked with UE5.2!, **can only be imported to full version of RflySim**)
- ./DroneyeeX680Body.FBX: Airframe model
- ./DroneyeeX680Prop.FBX: Propeller model

4. Experimental Content or Steps

This experiment uses the RflySim3D blueprint interface to control the rotation of propellers during simulation.

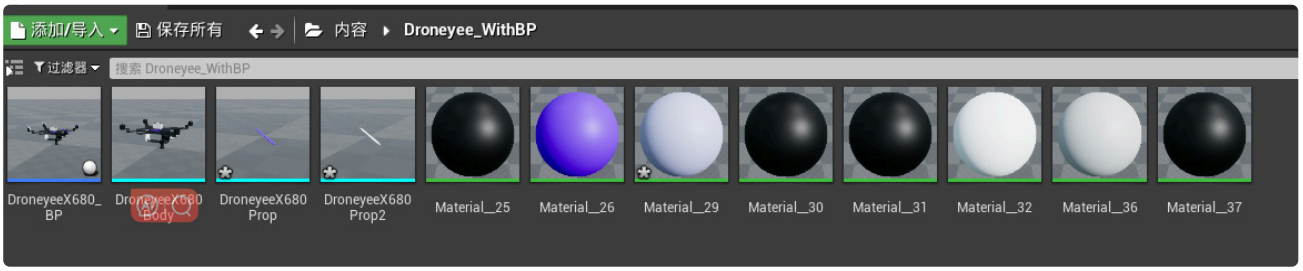
4.1 Blueprint Model Import Experiment (Optional)

Step 1: Create Blueprint Class in UE

UE4

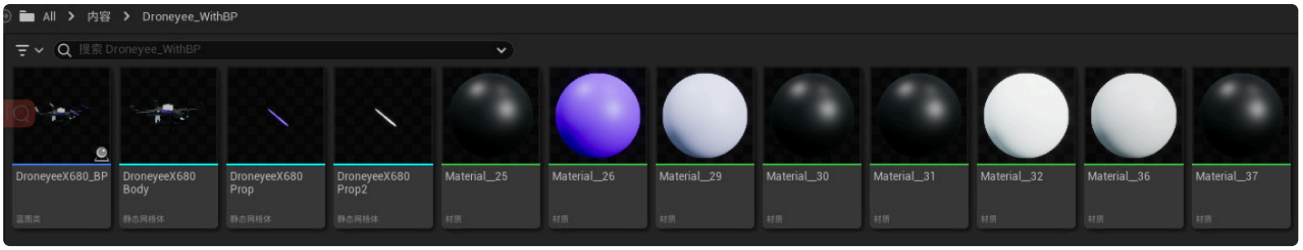
We open UE4 and create an empty project. Specific steps can refer to ../../../1.BasicExps/e0_StarterContent/1.UE4StarterContent/Readme.pdf

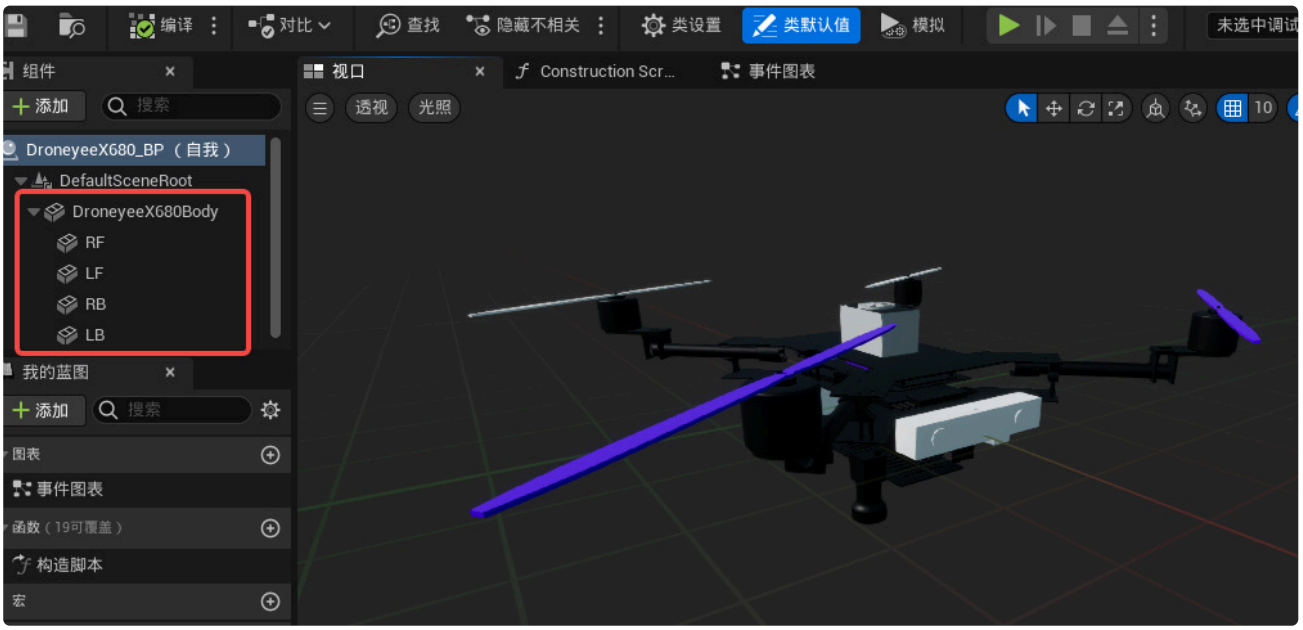
Create a folder named "Droneyee_WithBP" under its Content, then import "DroneyeeX680Body.FBX" and "DroneyeeX680Prop.FBX" from this document into UE4. Next, create a Blueprint Actor and assemble the drone body and propellers together, using different colored propellers to mark the nose orientation:



UE5

Same configuration as UE4

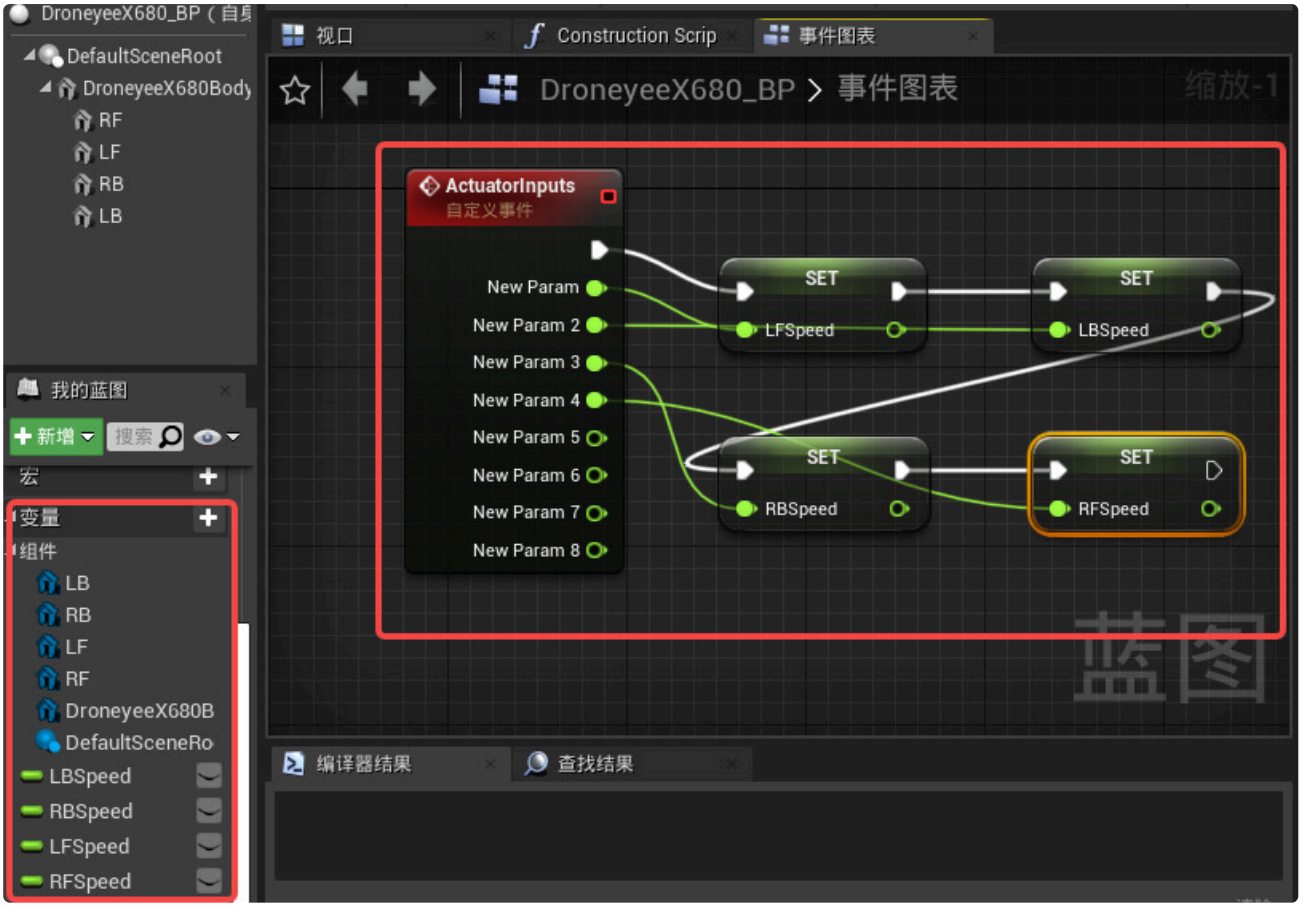




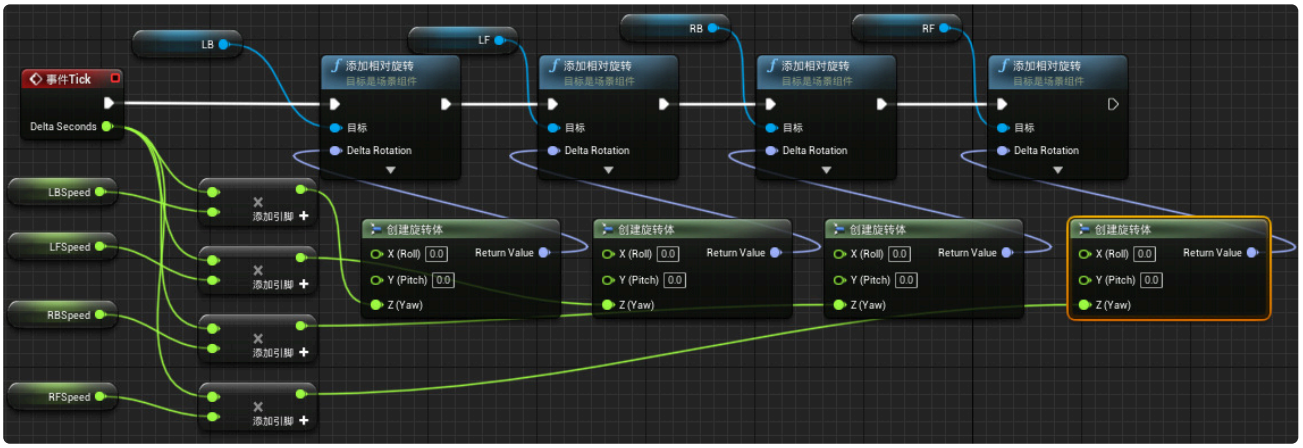
Step 2: Define Interface Functions and Write Features

UE4

Next, enter the event graph interface of the blueprint actor, right-click to add a Custom Event node, create a custom event named "ActuatorInputs" (it must be this name), and add 8 float parameters to it. When this interface is triggered, this function will be activated. Here we receive its first 4 values, store them respectively as the rotation speeds of 4 propellers.

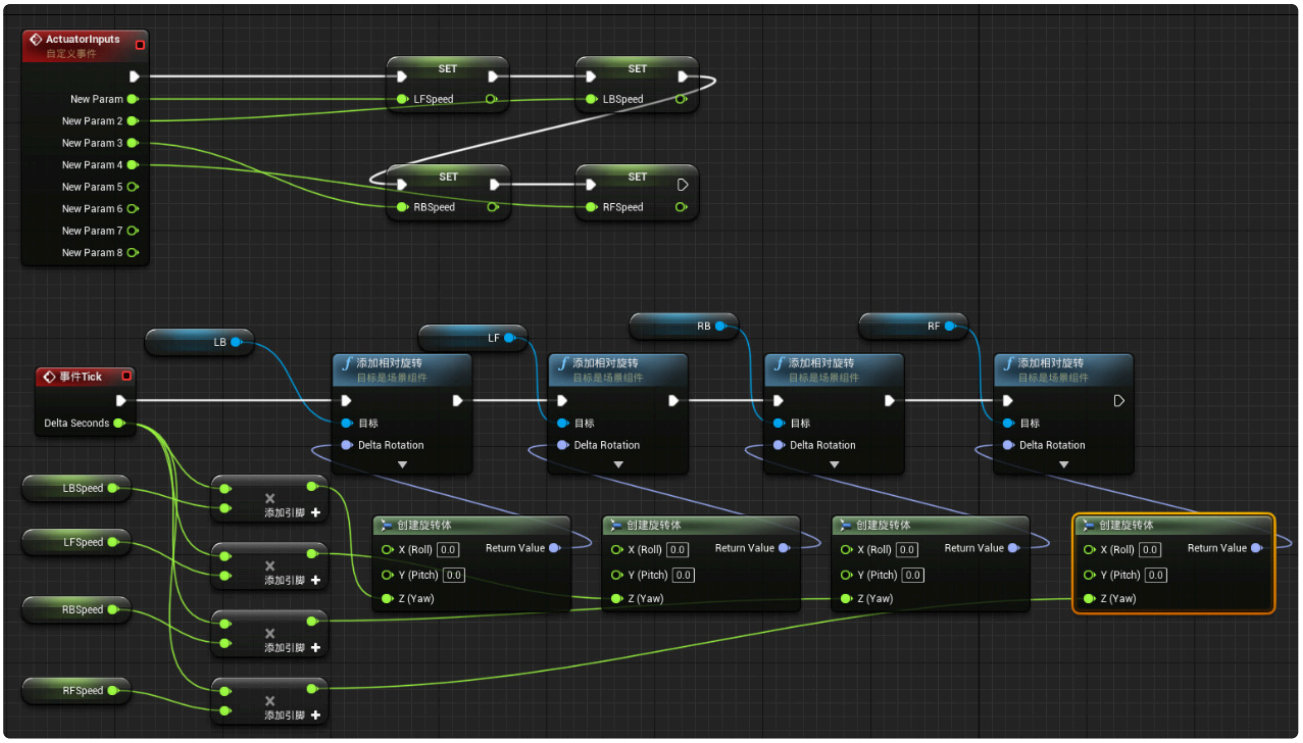


Then add an EventTick node to rotate these propellers in the EventTick. EventTick is an event that is called every frame in the scene. If we want the propellers to appear rotating, we need to modify their poses every frame. When multiple frames are continuously connected, they appear to be rotating.

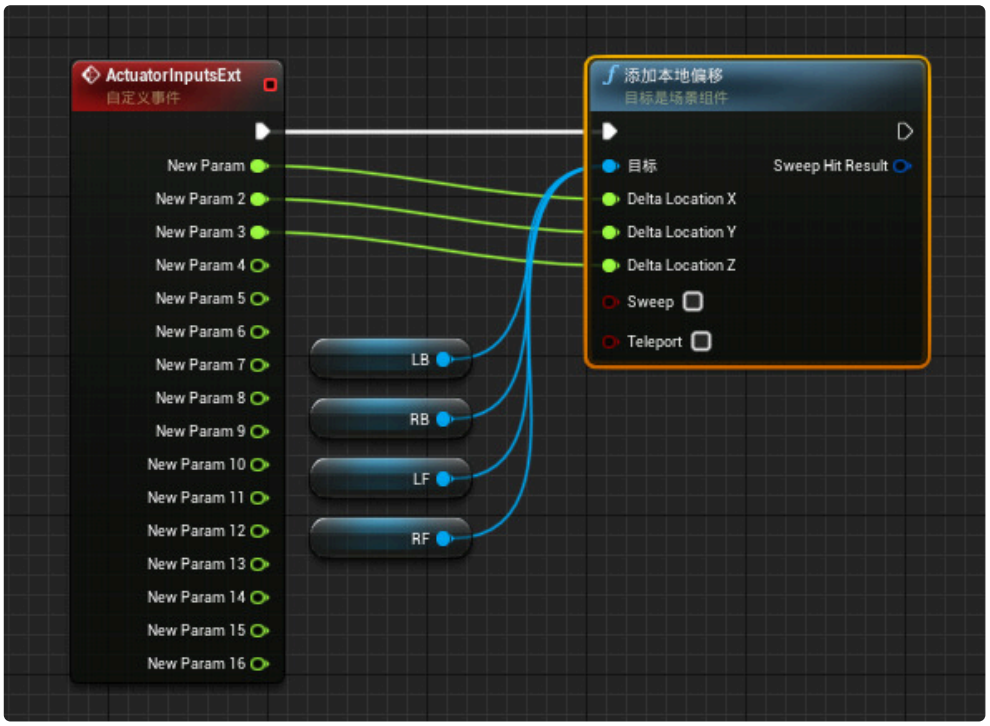


DeltaSeconds represents the time interval from the previous frame to the current frame. We know that [Time × Rotation Speed = Rotation Angle], so we know the angle the propeller needs to rotate between the previous frame and the current frame. The rotation axis is the z-axis of the propeller, so we can use the AddRelativeRotation node to increase the Yaw angle accordingly.

Finally, complete the blueprint event graph as shown in the figure below

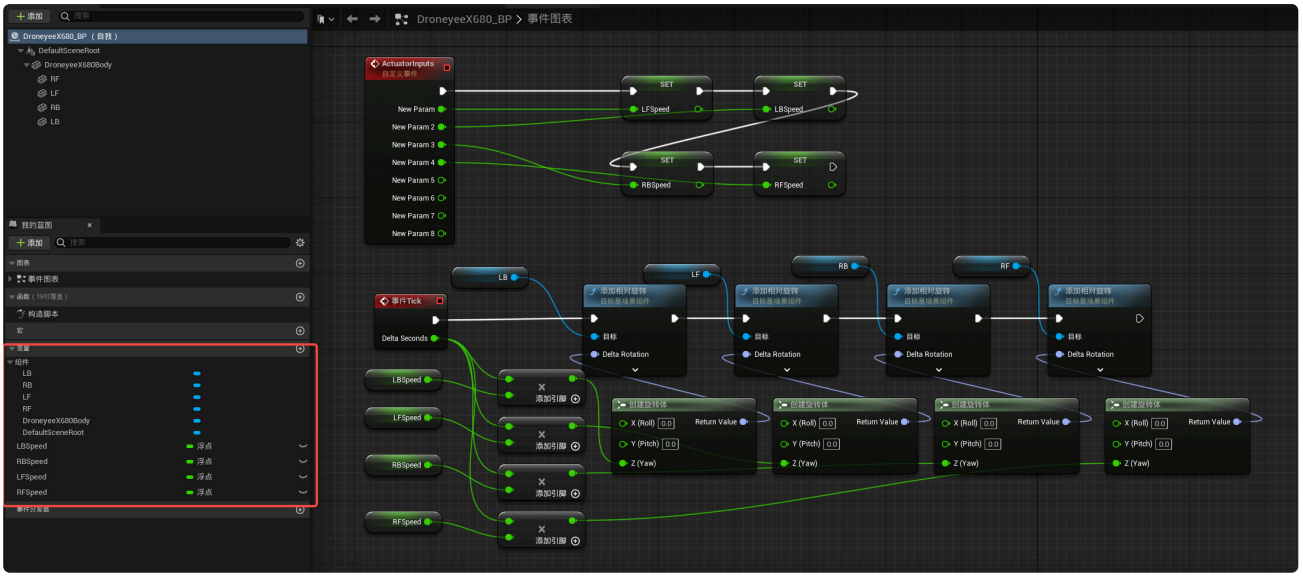


We can also create another custom event "ActuatorInputsExt", which is a custom blueprint interface. Here we use the first 3-dimensional data received as position offsets for four propellers in xyz directions.



UE5

Configuration steps are the same as UE4

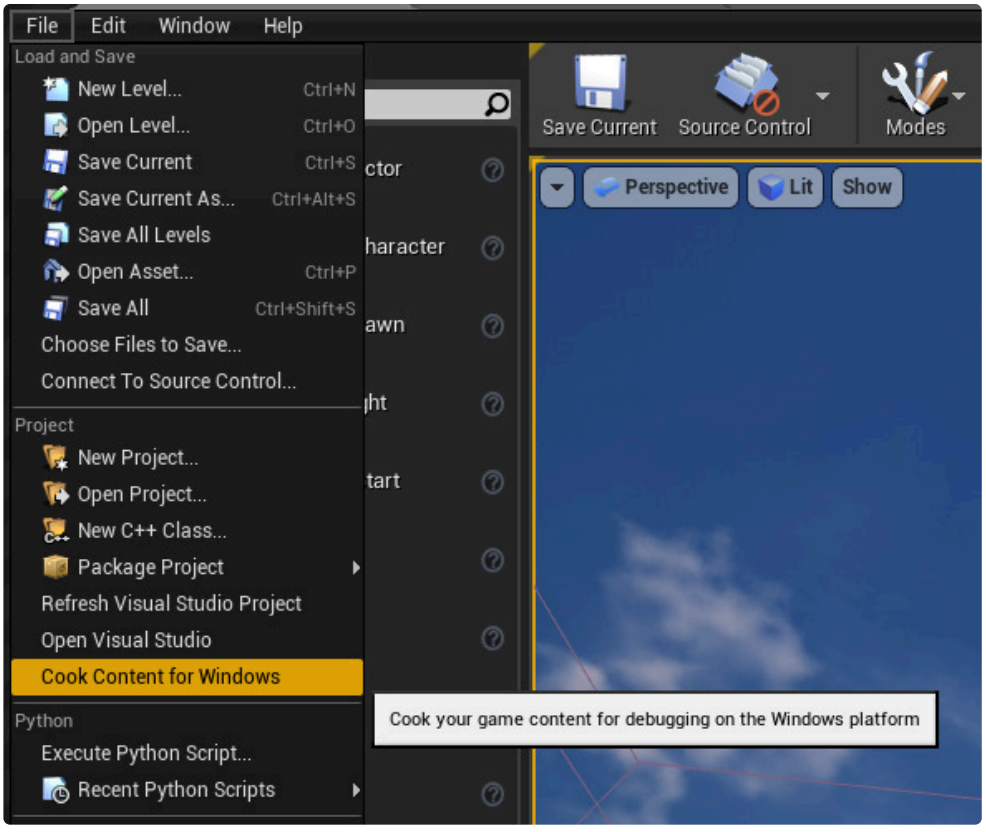


Step 3: Import to RflySim3D/RflySimUE5

UE4

Next, two operations are needed to import it into RflySim3D so that we can create it in RflySim3D:

First is **baking**, click `File->Cook Content for Windows`, after baking completes, go to the project directory and find the `"\\[Project Name]\\Saved\\Cooked\\WindowsNoEditor\\[Project Name]\\Content\\DroneyeeX680_WithBP"` folder, copy it out for backup use.



Second step is **writing XML file**, usually you can slightly modify other drone XML files. There is already a "DroneyeeX680_WithBP.xml" file written in this document directory, which was modified from `"\\PX4PSP\\RflySim3D\\RflySim3D\\Plugins\\Rfly3DSimPlugin\\Content\\XML\\F450_Default.xml"`.

Main modifications include:
The value of the tag was changed to 2, indicating that the aircraft model is declared as a blueprint model.

The path in the tag under the tag was modified to indicate the path of this blueprint model.

Removed the tag because blueprint models don't require XML files to configure actuators.

Place the XML file into the previously baked "DroneyeeX680_WithBP" folder.

Copy the "DroneyeeX680_WithBP" folder to the `"\\PX4PSP\\RflySim3D\\RflySim3D\\Content"` path.

UE5

Baking

XML file writing is the same as UE4

Copy this "DroneyeeX680_WithBP" folder to the `"\\PX4PSP\\RflySimUE5\\RflySim3D\\Content"` path.

Step 4: Call Blueprint Interface

UE4

Open RflySim3D, double-click the ground + letter O + number 3 to create a quadcopter drone, then press C to switch to the recently baked "DroneyeeX680_WithBP" drone.

The drone created this way has IDs incrementally starting from 1000 (you can also press S to view its ID).

Then use the command: "RflySetActuatorPWMs 1000 60 80 100 120 0 0 0 0", you can see four propellers begin to rotate, at 60°, 80°, 100°, 120° per second respectively.

Phenomenon explanation: When using the RflySetActuatorPWMs command, if the target drone is a blueprint class instance, then it will call its ActuatorInputs event and pass all 8 parameters to it. In our ActuatorInputs event, we programmed the first 4 values to control the rotation speed of 4 propellers respectively.

We also wrote another "ActuatorInputsExt" event. When using the RflySetActuatorPWMsExt command, if the target drone is a blueprint class instance and **the RflySim is the full version**, then this command is effective. If we input the command "RflySetActuatorPWMsExt 1000 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0":

We can see all propellers move upward by 20cm. This is because in the "ActuatorInputsExt" event we wrote, the first 3 of these 16 parameters can add an offset to the positions of these propellers, and the third float controls its z-axis offset, so its z-value increased by 20 (this coordinate is the UE4 coordinate, because we did not transform the parameters).

UE5

Open RflySimUE5 and follow the steps in UE4

5. Key Knowledge Points

Key Knowledge Point 1: Control Methods for Objects in 3D Scenes

Every object displayed in a 3D scene is an individual object with member variables and member functions. The drones we control are the same way. These objects are defined in UnrealEngine, and if they inherit from UE blueprint classes, the platform provides interfaces to call blueprint functions. The interfaces that RflySim3D provides for blueprints mainly include: "ActuatorInputs" and "ActuatorInputsExt". These two interfaces can transmit a set of data to a specified drone in the scene, which can be used to display some custom effects (these effects need to be written using the blueprint system themselves), such as opening hatches, explosions, displaying text, switching aircraft materials, rotating wings, or anything else the UE blueprint system can do. It is a higher-order interface, therefore requiring basic knowledge of UE blueprint usage as well.

So-called blueprint interfaces are calling blueprint functions from external sources, thereby triggering various functions written in the blueprint. This call can be achieved through two RflySim3D console commands introduced earlier: "RflySetActuatorPWMs" and "RflySetActuatorPWMsExt", or by using other programs (such as Python, Simulink) to send specified structures via UDP transmission.

6. References

1. RflySim3D Model Import Overview
2. RflySim3D Shortcut Keys Interface Overview
3. RflySim3D Console Commands Interface Overview

7. Frequently Asked Questions

Q1: How does the blueprint interface work?

A1: The blueprint interface allows calling blueprint functions from external sources, thereby triggering various functions written in the blueprint. This can be achieved through RflySim3D console commands such as "RflySetActuatorPWMs" and "RflySetActuatorPWMsExt", or by using other programs (such as Python, Simulink) to send specified structures via UDP transmission.

Q2: How to ensure the blueprint model is correctly imported into RflySim3D?

A2: Baking is required and the generated folder needs to be copied to the correct path (for UE4 it's "\PX4PSP\RflySim3D\RflySim3D\Content", for UE5 it's "\PX4PSP\RflySimUE5\RflySim3D\Content"). At the same time, the XML file needs to be modified, changing the tag value to 2, updating the path, and removing the tag.

Q3: How to verify if the blueprint interface is working properly?

A3: After creating a blueprint-type drone, you can use commands such as "RflySetActuatorPWMs 1000 60 80 100 120 0 0 0 0" to test interface functionality. If the target drone is a blueprint class instance, the command will call its ActuatorInputs event and pass the parameters to it.

---

1. https://rflysim.com/ ↩
2. For recommended configurations, please see: https://rflysim.com/ ↩