Dynamic Flight with Blueprints

Hello, and thank you for your purchase! This documentation covers the following topics:

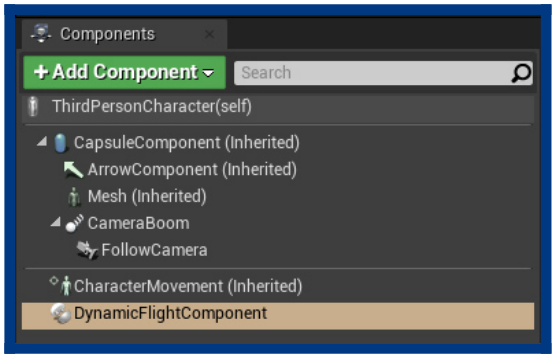
- Merging Into Your Project
- Integration with ALS
- Configuring Flight Behaviors
- Legacy Singleplayer Version
- Mixamo Retarget Crash

Merging Into Your Project

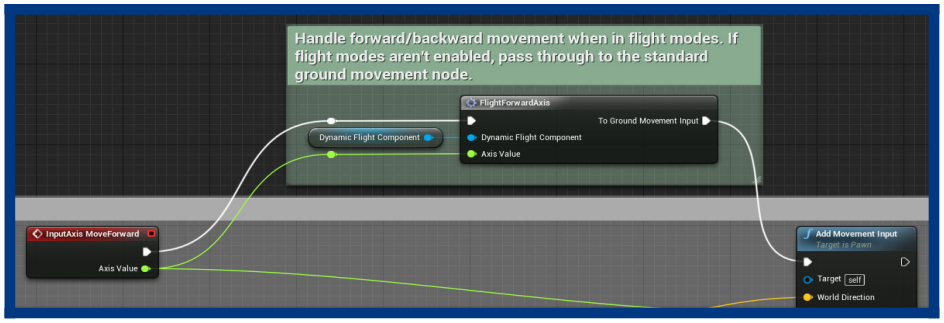
If you want to use the included character BP as a new base for your project, this can easily be done by migrating the assets into your project and swapping out the character BP you were using with the one provided. However, if you want to merge this system into an existing character BP and AnimBP, there are a few more steps required. Full instructions are below.

NOTE: Do not use any flight assets with the **LEGACY_SP** prefix unless you intend to use an older version that is no longer supported. Please see the Legacy section for more information.

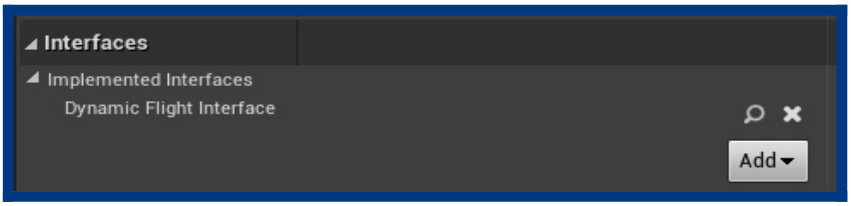
1. Open the provided **Dynamic Flight** project.
2. Right click on the **DynamicFlight** folder in the content browser and select **Migrate.** Select the **Content** folder of your project and migrate the assets.
3. Close the **Dynamic Flight** project and open your project.
4. Create **Action Mappings** in your input settings (**Edit** → **Project Settings** → **Input**) with the following names, and assign them to any keys/buttons you'd like:
    - **ToggleHover**
    - **ToggleFastFlight**
    - **DodgeLeft**
    - **DodgeRight**
5. Create an **Axis Mapping** named **MoveUp**, and assign two buttons to it – one with **Scale** set to **1.0** (this will move up) and one with **Scale** set to **-1.0** (this will move down).
6. Open your existing character BP and add the **DynamicFlightComponent** to it.



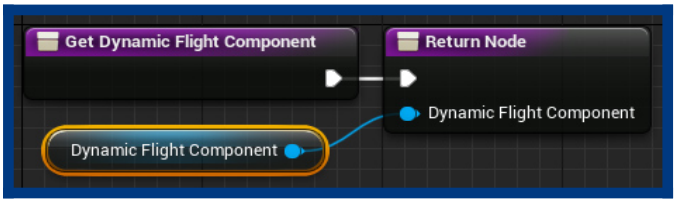
7. With the **DynamicFlightComponent** selected, enable the **Component Replicates** option in the **Details** panel.
8. Open the provided **ThirdPersonCharacter** blueprint (DynamicFlight\Blueprints\Characters) and copy the small sections of nodes surrounded by green-colored comment boxes. There should be 6 sections.
9. Open your existing character BP and paste these nodes into its Event Graph. Note that three of them should connect to pre-existing character logic – refer to the provided character BP and carefully recreate exactly how they connect. One section connects to jump logic, one connects to InputAxis MoveForward logic, and another connects to InputAxis MoveRight logic.



10. In the character BP's class settings, add the **DynamicFlightInterface** interface.

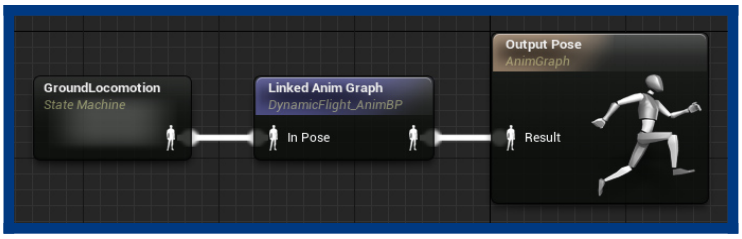


11. On the left side of the character BP, expand the **Interfaces** category and double-click on the **Get Dynamic Flight Component** function. Click on your **DynamicFlightComponent**, drag a reference into the graph, and plug it into the **Dynamic Flight Component** input on the **Return Node**.



12. Compile and close the character BP.
13. Make sure that your character's skeleton is ready for retargeting – open the skeleton, go to the **Retarget Manager**, and choose the **Humanoid** rig in the **Select Rig** dropdown. Save the skeleton.
14. Right-click on the included **DynamicFlight_AnimBP** (DynamicFlight\Animations) and select **Retarget Anim Blueprints → Duplicate Anim Blueprints And Retarget.** Choose where you want the retargeted assets to be saved and complete the retarget. **NOTE:** There is a potential issue here when retargeting to a Mixamo rig, and possibly other rigs that are not based on the UE4 skeleton. If you encounter a crash here, see the section about Mixamo for a work-around.

15. Open the AnimBP you're using for your character. In the main AnimGraph, create a **Linked Anim Graph** node (also called **Sub Anim Instance** in older engine versions). Set its **Instance Class** to the retargeted **DynamicFlight_AnimBP**.
16. Plug your existing state machine / pose result into the **In Pose** on the **Linked Anim Graph**, then connect its output to the **Output Pose**.

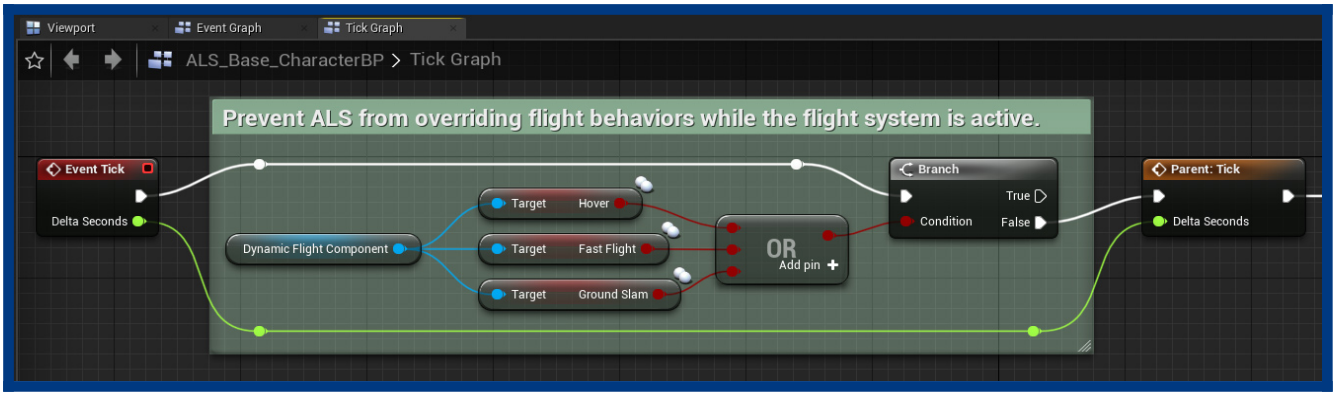


17. Compile and save the AnimBP, and you're done! However, if you're using ALS (Advanced Locomotion System), there are a couple additional steps to take for best results. See the section below!

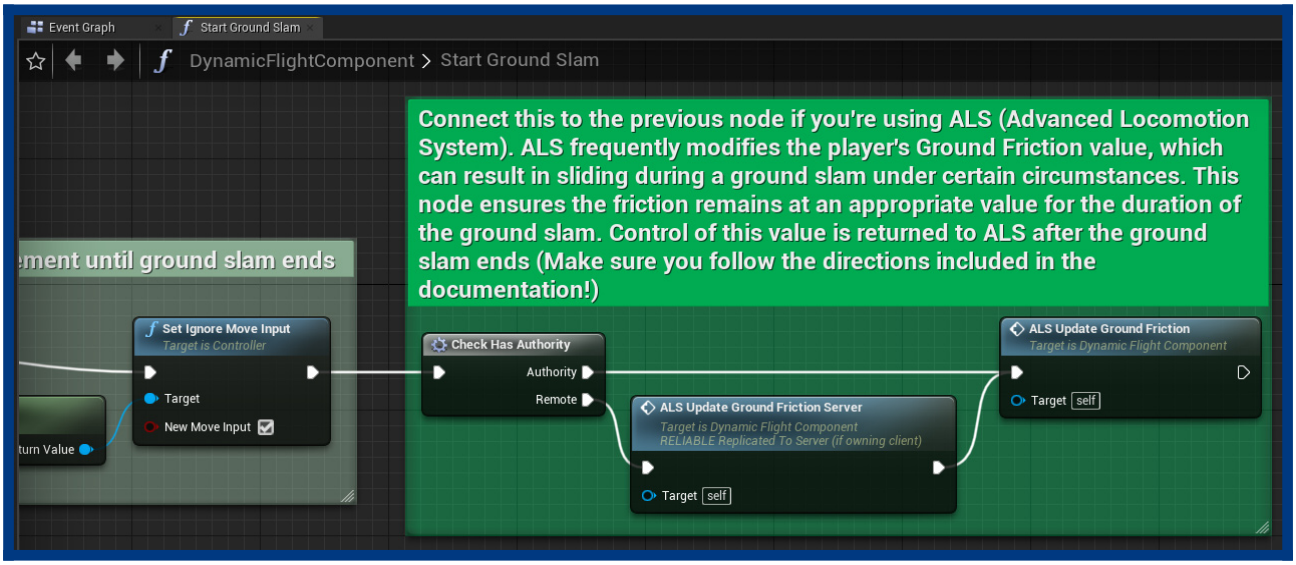
Integration with ALS

Due to how different and complex the Advanced Locomotion System is, there are a couple extra modifications we need to make to prevent the two systems from conflicting with each other. After completing the steps in the previous section, please perform these steps:

1. While the flight system is active, we need to prevent ALS from performing actions that will interrupt/conflict with what the flight system is trying to do. We can do this by opening the ALS character BP, then placing a branch at the beginning of its **Event Tick** node inside the **Tick Graph**. The branch should check if any of the **Hover**, **FastFlight**, or **GroundSlam** booleans are true – and only proceed if they are all false:



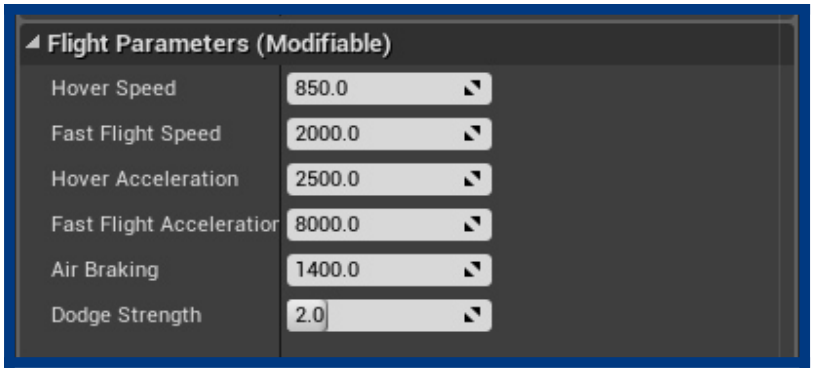
2. Next, we want to open the **DynamicFlightComponent** BP and enter its **StartGroundSlam** function. Connect the **Check Has Authority** node to the previous one so that it will execute. ALS sometimes sets the character's Ground Friction to be very low, which can sometimes cause bad behaviors with the ground slam – this node ensures the friction is at an appropriate level for the duration of the ground slam.



3. Compile and save the BPs, and you're done!

Configuring Flight Behaviors

Several flight characteristics can be easily changed via parameters on the **DynamicFlightComponent**. Open your character BP, select the flight component, and the following parameters can be found in the **Details** panel:



Legacy Singleplayer Version

The newest version of this system now includes multiplayer support! However, if you do not need multiplayer support and would prefer to use the older, simpler singleplayer blueprints, these are still available inside the **Legacy_Singleplayer** folder. But please keep in mind that this singleplayer version will no longer be updated moving forward. If you wish to use this version, the instructions above still apply, but you will need to use the versions of the assets found in the **Legacy_Singleplayer** folder rather than the ones pointed to in the original directions. For clarity, these specific steps require the use of the Legacy assets:

Step 6: Use LEGACY_SP_DynamicFlightComponent instead of DynamicFlightComponent
Step 8: Use LEGACY_SP_ThirdPersonCharacter instead of ThirdPersonCharacter
Step 10: Use LEGACY_SP_DynamicFlightInterface instead of DynamicFlightInterface
Step 14: Use LEGACY_SP_DynamicFlight_AnimBP instead of DynamicFlight_AnimBP
Step 15: Set the linked anim graph to use the retargeted LEGACY_SP_DynamicFlight_AnimBP

DO NOT mix and match the new and legacy assets – they are not compatible!

Mixamo Retarget Crash

For some reason, UE4 doesn't like retargeting to Mixamo rigs when the animations use certain additive settings. This results in an editor crash. If you're encountering this issue with a Mixamo rig (or possibly any other rig that isn't based on the default UE4 rig), follow these steps as a work-around.

1. Select all animation assets that have **Add** or **AdditiveBase** in the name (there should be 23 - ignore the Blendspaces).
2. Right click them and go to **Asset Actions** → **Bulk Edit via Property Matrix**.
3. Expand the **AdditiveSettings** category on the right side and change the **Base Pose Type** to **Skeleton Reference Pose**.
4. Close the **Property Matrix**, then retarget the AnimBP like normal.
5. Select the retargeted versions of those same 23 animations and re-open the **Property Matrix**.
6. Change the **Base Pose Type** back to **Selected Animation Frame**
7. Done!