Dingye Wang

wangding1 300422014

Lead Programmer

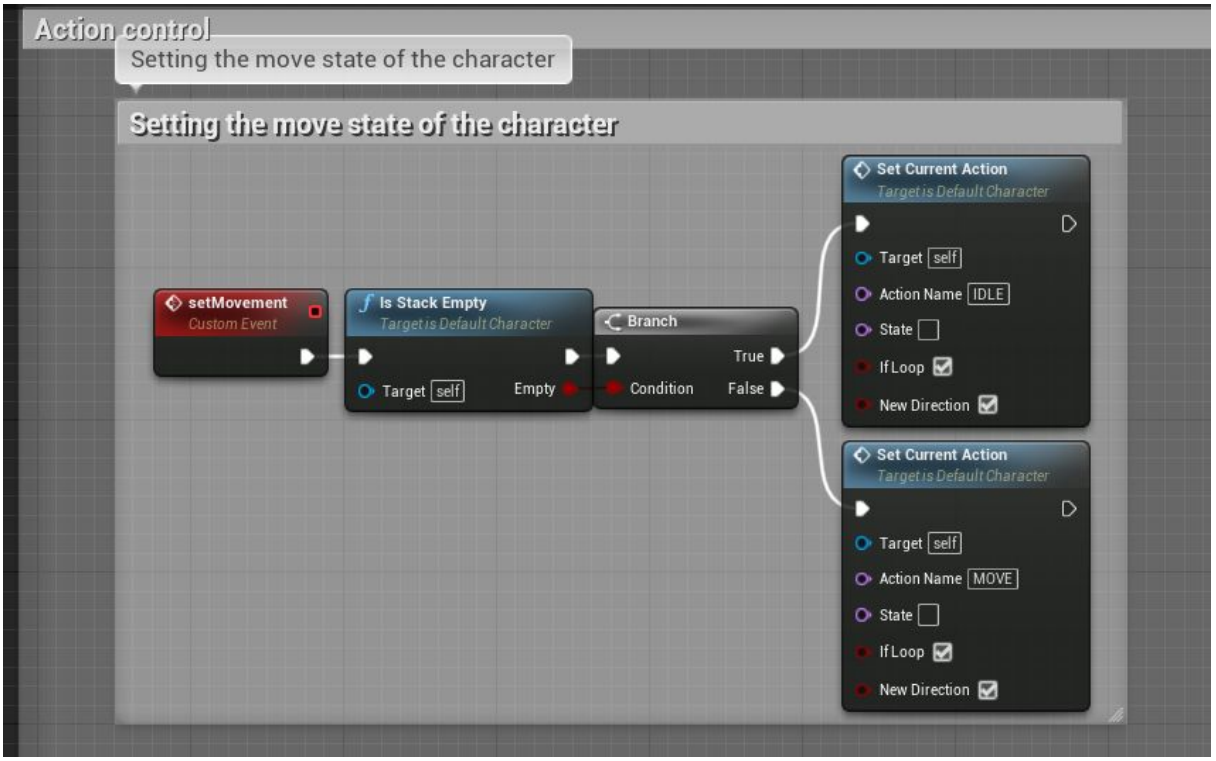
Code Discussion

I was the only programmer in our group, and the only person who worked on the Unreal engine, thus I will be claiming all [All] of the blueprints and code work.

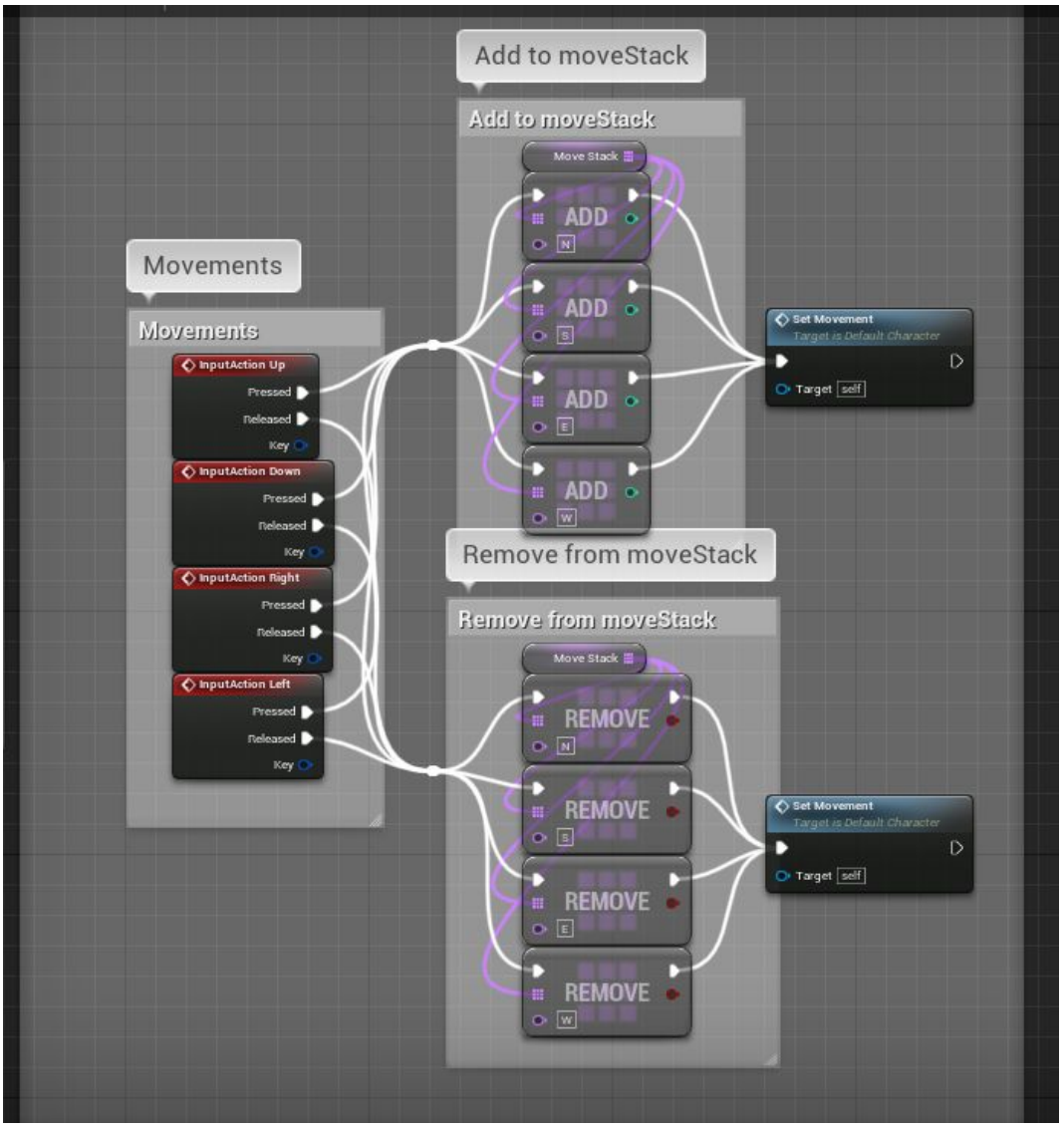
Video
https://youtu.be/RWpJx8iKXs8

Most Interesting
The most interesting part of my blueprints that I think, will be the 2D top-down character movement stack. Instead of using the unreal axis mapping, I used my own default movement system. As the character is moving, if the user gives it a new direction, It will start walking in the new direction.

When the user releases a movement, it removes that movement from the stack, the code will go back to the stack and obtain the latest movement input that's in the stack and perform it. This gives the game a very smooth feeling, as in some games, the player has to release all the movement keys and give it a new input for the game to register. The movement stack will keep on going even if the character is in some other animations. When the animation finishes players can still release keys and it will perform the latest movement key that was held.
#1 setMovement called from Movement stack



#2 Movement stack logic



Most proud code

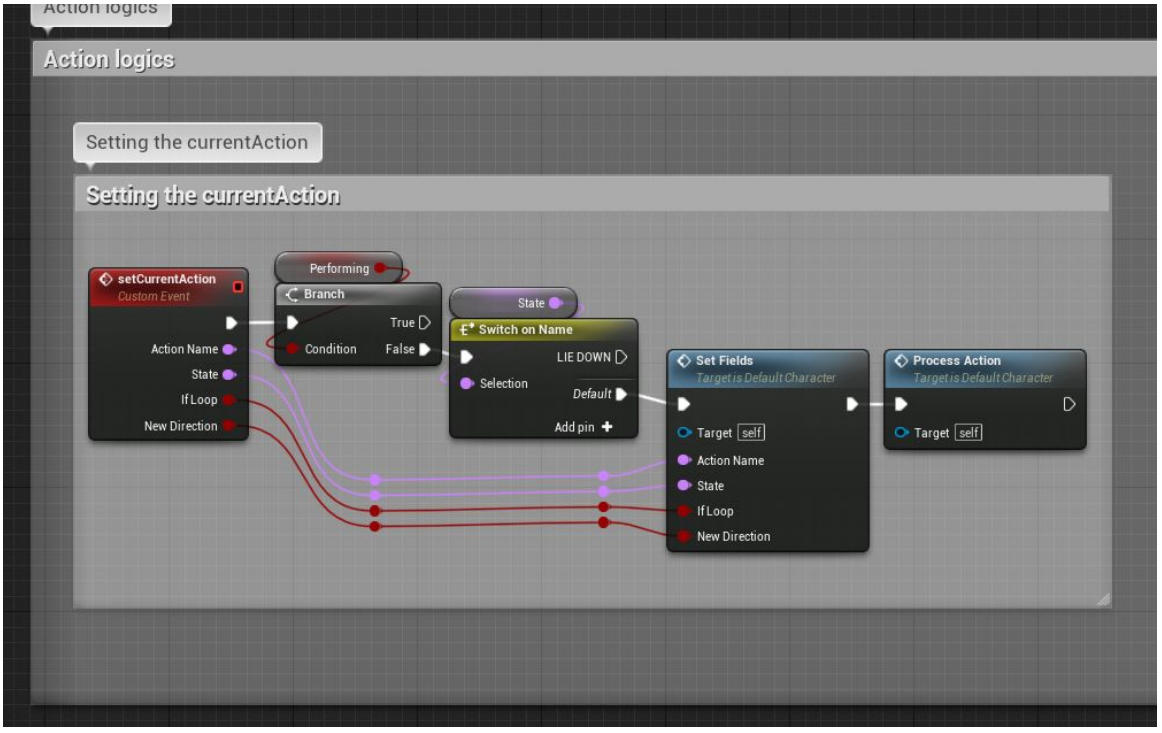
The part that I'm most proud of is the character action management system. It might not seem like alot of code. But it did require a lot of back and forth decision to come up with. Each action is stored as an enum or “Name”, when the user inputs a new Action, the game then has to process several steps to check if the action can be performed. What's good about using “Names” is that I can use a switch case to find the corresponding flipbook with its current direction since all my flipbook is all stored inside a map.

For example, If the player inputs a punch. The game will check though, if the character is in a current animation, if he is still performing, then the punch signal will not respond. Then the game checks for state, if the character is dead, he also should not receive any action. Of course there are animations that can be interrupted, so i've separated them into 2 parts. One is checking for action like if the player has inputted, the other one is set fields, which instantly perform this without caring for other constraints for actions that will interrupt others.

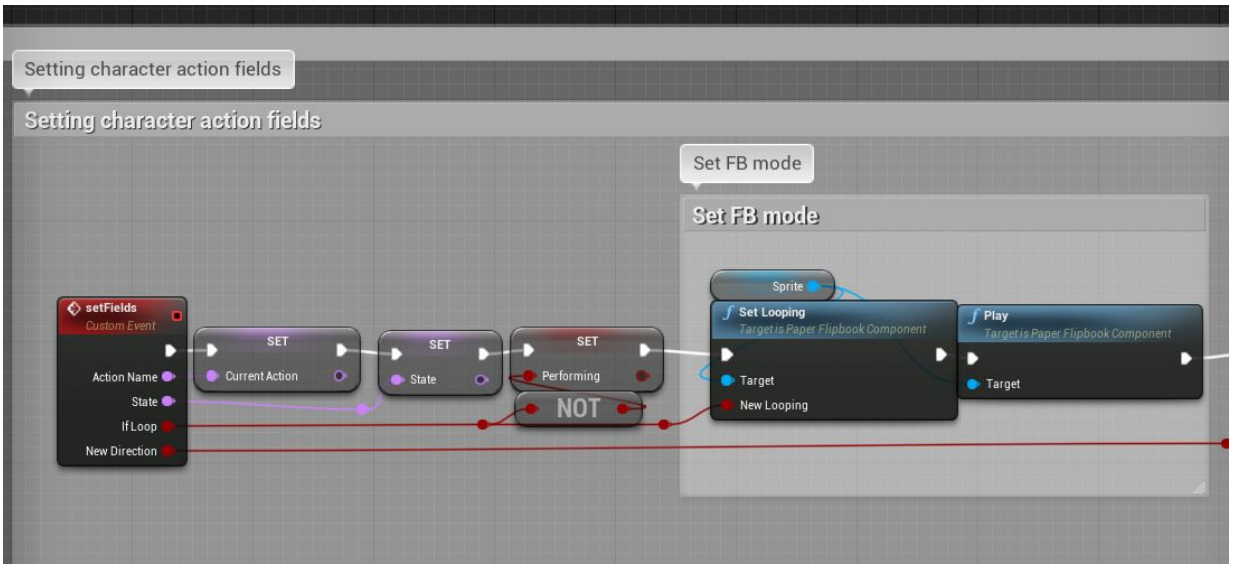
This is all based on the parameters that each action sends out. If it should be loopable, if the character can change directions in it, what state will it be in and what action is it. If these checks pass, it will then be setted and perform those actions.

#3 InputActions, different actions should have different properties. (also #1 and #2)

#4 SetCurrentAction which acts like branch checking if the place can perform such action.

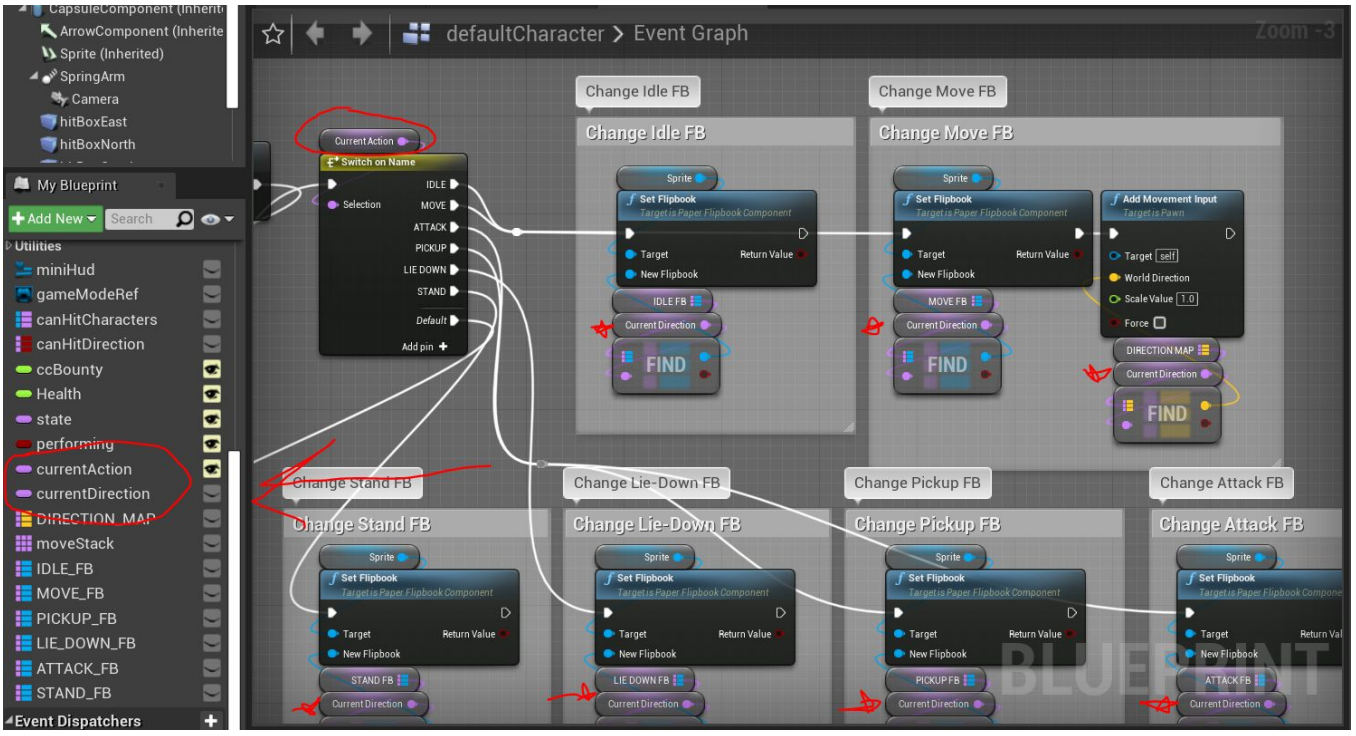


#5 SetFields which is an instant response to an action. Called by setCurrentAction or other instat effect.



#6 continue to #5

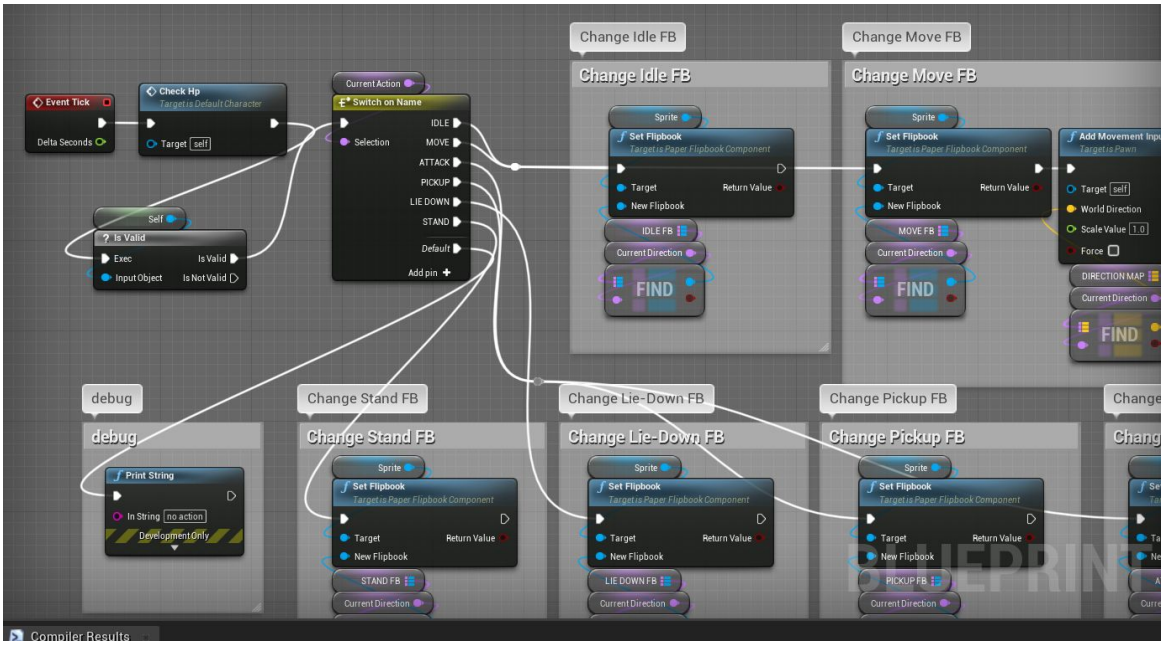
#7 enum/Name action switch case and direction mapping for flipbooks



Bad code

For the bad part of my code, I think it would be the event tick for my flipbook animations. Now thinking back. Since i've already have the parameters for checking if the flipbook should loop or not. It's unnecessary for me to use event ticks to manage the flip books based on an Action name(Enum). It processes way more information that it needs and should just manage the movements only. From putting more effort into the design concept, this can be avoided by using all the key release functionality for movement to stop/ interrupt certain flipbook loops in order for the game to run more efficiently.

#8 Similar to #7 but the concept of using event ticks to manage fb is bad.



Reflections

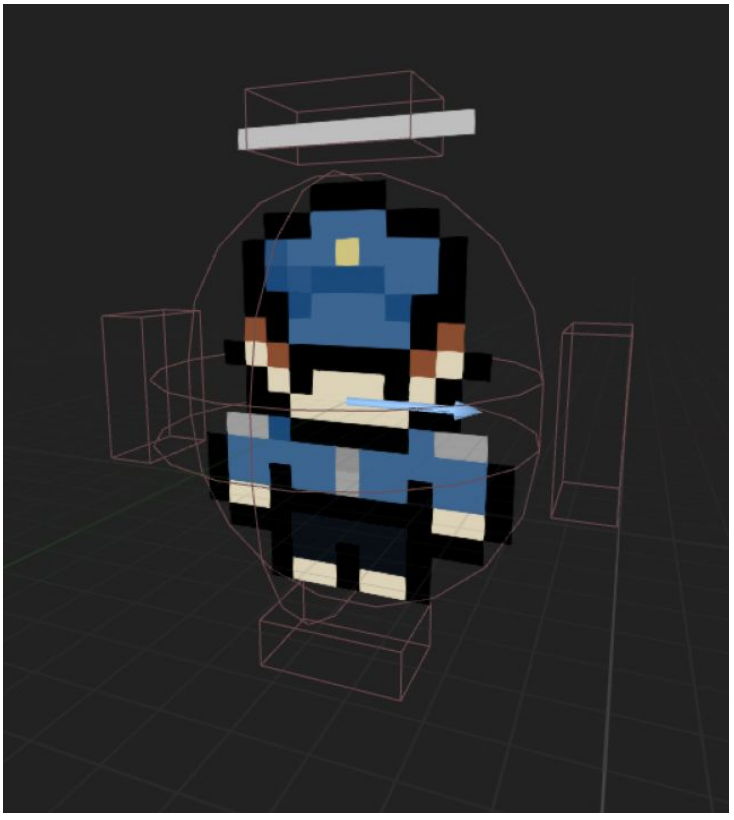
Honestly this may have to be the most rewarding and challenging course I've done. Thinking as a programmer, learning this Unreal thing **is** like learning a whole new language, even though it's in C++. Some ideas I've learned are to really adapt to changes. Most of the time when I'm stuck I just think, it would be so much easier if this was done in … languages, but the main problem is that I'm not familiar enough with Unreal blueprints.

Once i've got the hang of the logic thing, it became so much easier. Things like getting player character, getting gamemode, and casting to a specific class to get its variables was the main important thing I learned to program in Unreal. The style of blueprint programming kind of led me to another way of thinking. It allows me to be more creative and try out dodgy things to make improvements for run speed and file size.

If I was to use something from this project in my future projects, it definitely would be the whole 2D top down code concept. Since Unreal isn't really made for 2D Top-down style games but rather 3D. A Lot of the videos came up around making 2D side scrollers or 3D third person. But 2D top down was pretty rare.

Other than character generation there weren't much tutorials. For some systems of the game I had to be proactive and create my own kind of way to solve the problems, such as hit boxes for my paper2d character in a top down world. So continuing off this 2D top-down game concept for my future projects would definitely help me build up new problem solving strategies and there will always room for more improvements.

#9 hit box example



#10 my take on 2D top-down hit register system

#11 continue to 10