Spider pawn

Movement

Create a new blueprint class of type pawn called SpiderBP. Add the character mesh and set it as the default scene root. Add a spring arm component and set it as the child of the character. Add a camera and set it as the child of the spring arm. Set the offset to 0 if that’s what you want. Make sure to enable the pawn control rotation option for the camera. Open the third person character blueprint and disable the auto-possess player option. This prevents the mannequin form being possessed. Now on simulation the camera should show the spider bot. Open the 3rd person character BP and copy the mouse input portion and paste it in the SpiderBP. Now on simulation you will be able to control camera movement.

Create a new event/function to do line trace and determine the surface called TraceSurface. We want to get the direction perpendicular to character position i.e., if he is on the ground the direction will be downwards and if he is on his left-side the direction will be sideways to the right. Extend it and add a line trace by channel node and set the camera as trace channel. Add a get actor location node. Add a get actor up vector node and extend the return value and add a vector \* float node and set the other value to 200 this value was set based on the character model we need to tweak it according to the model being used. Extend the return value and add a vector – vector node and to the A pin set the get actor location node and to the B pin set the return value of the vector \* float node. Extend the return value again and add a vector + vector node and to the other pin set the get actor location node. Hook the return value of the vector + vector node to the start pin and the return value of the vector – vector node to the end pin. Set the debug type to for duration for testing purposes.

We don’t need to call TraceSurface at every tick instead we can call it periodically so extend the event begin play node and add a set timer by event/function extend the event pin or function pin and if it’s an event pin add a create event node and set the signature to TraceSurface and set the time to 0.5 and check the looping option. Now it will be called every 0.5s instead of every tick.

Now on simulation the line trace can be seen directly downwards i.e., perpendicular to the current position. It will also be called periodically in the BP.

How this works is we know how much distance the character should be above ground ideally. But in the game, we calculate it and if it’s different from the ideal distance we tweak it accordingly.

Extend the return value of the line trace by channel node and add a branch node. This returns true if a hit is obtained. Make a vector variable called SurfaceLocation. Extend the out hit pin of the line trace by channel node and add a break hit result node. Extend the location pin and add a set SurfaceLocation node and hook the true pin of the branch node to the input exec pin of the set SurfaceLocation node. Extend the trace end pin and add a set SurfaceLocation node and hook its input exec pin to the false pin of the branch node. This surface location should be the new actor location right i.e. if the player is tilted to the right then the left wall is below him and his position should change to being on the left wall. So you can extend the return value of the SurfaceLocation and promote it to a new variable called TargetLocation (from both branches) or use the SurfaceLocation variable itself.

Now with every event tick, / try doing this at the end of the TraceSurface itself, we need to update actor location. In the viewport add a floating pawn movement component to make things easier. Make a new function/macro called movement input and extend the exec or input pin and add an add movement input node. Add a get actor location node and extend the return value and add a get unit direction vector node and extend the to pin and add a reference to TargetLocation. Extend the return value and hook it to the world direction pin.

Now on simulation the character lands on the ground at the appropriate height (make sure you have gravity turned off if you don’t want that force to work). The character might vibrate and suddenly change position.

The position change is because the Movement Input function/macro is being called at every event tick i.e. it’s being called before the first execution of TraceSurface so the TargetLocation’s default value is going to be used causing the sudden change in position. A quick fix is to extend the event play node and add a set target location node and extend the input value and add a get actor location node. The vibration is due to the change in position even for minor changes in the TargetLocation to fix this we use a clamp so as to make the changes accordingly. Go back to the Movement Input function/macro and extend the get actor location pin and add a distance node and hook the return value of the TargetLocation node to the To pin. Extend the return value and add a vector / float node and set the other value to 100 extend the return value and clamp it and hook the result pin of the clamp node to the scale value of the add movement input node pin. How it works is the / by 100 makes all values less than 100 into decimal digits and the clamp makes all values above 100 into 1 so for high as well as low values the result will be between 0-1.

Now on simulation the character will be in the right position and will not vibrate. Next we implement movement.

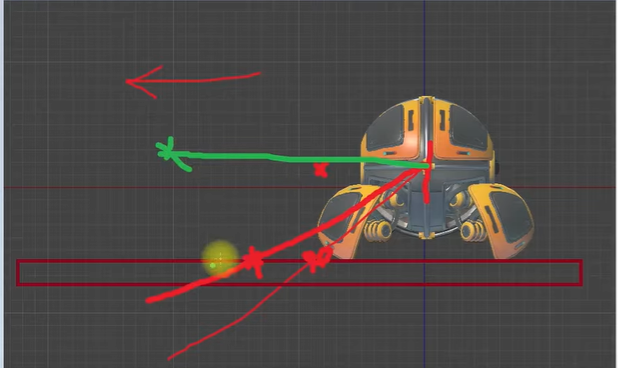
The following tutorial makes the character walk on surfaces that are bent inwards and not outwards.

Create an event/function called Trace. Extend the Event BeginPlay node and add a set timer by event/function node and if you’re using Trace as an event extend the event pin and add a create event node and set the signature to Trace and the time to 2s. Check the looping option.

Create a new function called TraceMovement and extend it and add a line trace by channel node. Extend the start pin and add a get actor location node.

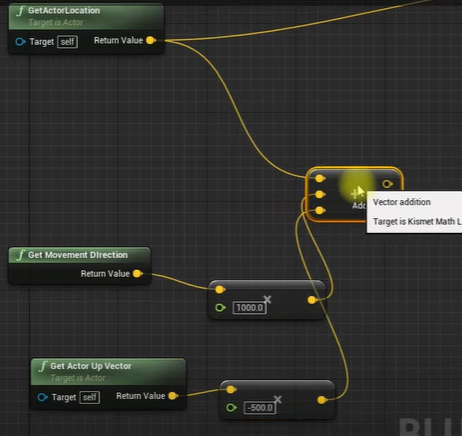
Add a get actor forward vector node and extend the return value and add a vector \* float node. extend the float pin and add a get move forward node. Add a get actor right vector, extend it and add a vector \* float node and extend the float pin and add a get move right node. extend the return value of the vector \* float node and add a vector + vector node and to the other in hook the return value of the other vector \* float node. Extend the return value of the vector + vector node and add a normalize node so as to get a unit direction vector. Select all the nodes included in this paragraph and make it into a function by right-clicking it and selecting the make function option and name it as GetMovementDirection so that we can reuse this later on. In the function definition check the pure function option on the right-hand window so that we can use this without an execution pin. To know if this works the exec pin disappears and the node changes from blue to green. To get a return value pin for the pure function you might have to extend the last node’s exec pin and make a reroute node and then select all the nodes before the reroute node and then collapse them into a function.

Now we have a GetMovementDirection node in place of the other node and extend the return value and add a vector \* float node and set the other value to 1000 so that we can get a point as marked by the green dot.



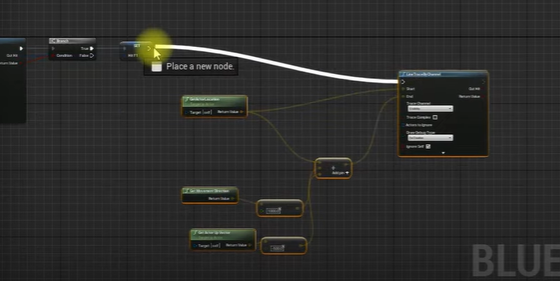
(length of the green line would be 1000)

Add a get actor up vector and extend the return value and add a vector \* float node and set the other value to -500. Extend the return value and add a vector + vector node and to the other pin set the return value of the other vector \* float node. Add another pin to the vector + vector node and to the third pin set the value of the get actor location node like so:



Hook the return value of the vector + vector node to the end pin. Setting the trace channel to visibility might be alright and set the debug type to for duration for debugging purposes.

If we get a hit means that we have space in front of the direction we want to move and that means we can move to it else we just move to where the trace ends. Extend the return value of the line trace by channel node and add a branch node. Extend the out hit pin and promote it to a local variable called hitF1 and set the input exec pin to the true pin of the branch node. Now copy the line trace by channel functionality and paste it after the set hitF1 node and hook the input exec pin of the line trace by channel node to the output exec pin of the set hitF1 node like so:



Set the float value in the vector \* float node to which get movement direction node is connected to 1100 so as to get 2 traces infront of the character.

Now on simulation the 2 line traces will appear one in front of the other in the direction we want the character to move. Eg: if we press W the line traces will appear in front of the bot, A it will appear to the right of the bot and while stationary the line trace will happen at the same spot as the character.

Make sure you are calling the TraceMovement function using the Trace function/event though.

Extend the return value of the 2nd line trace by channel node and add a branch node. Extend the out hit pin and promote it to a variable called hitF2 and hook its input exec pin to the true pin of the branch node. Copy the line trace by channel functionality again and paste it after the set hitF2 node. Make sure that the float value in the vector \* float node connecting to get movement direction node is 1000. Extend the get movement direction node’s return value and add a rotate vector around axis node and hook the return value of the get actor up vector node to the axis pin. Set the Angle Deg value to 90. Extend the return value of the rotate vector around axis node and add a vector \* float node and set the float value to 100. Add another pin to the vector + vector node and to the 4th pin hook the value of the vector \* float node to which the rotate vector around axis node is connected.

Now on simulation we’ll have 3-line traces. 2 in the direction of motion and one to the right of the direction of motion.

Extend the return value of the 3rd line trace by channel node and add a branch node. Extend the out hit pin and promote it to a variable called hitF3. Hook its input exec pin to the true pin of the branch node.

If you want you can select all the nodes and collapse it and name it as Trace Float like so:



This is because we are going to calculate the target location next and the calculation of the out hit is done so this will make the code easier to read.

Add a reference to the hitF1 pin and split it. Extend the hit location pin and add a vector + vector node. Extend the hit normal pin and add a vector \* float node and set the float value to 60(distance between the floor to the middle of the character) and hook the result to the 2nd pin of the vector + vector node extends the return value and promote it to a variable called target location. Hook its input exec pin to the output exec pin of the macro (if you collapsed it) / the set hitF3 pin (if you didn’t collapse.)

Extend the hit location of the F1 node and add a get unit direction vector node. Add a reference to hitF2 pin and split it, extend the hit f2 location pin to the to pin of the get unit direction node. Extend the return value of the get unit direction vector node and add a make rotation from axes node. Make sure it’s hooked to the forward pin.

Extend the hit location of the F1 node and add a get unit direction vector node. Add a reference to the hitF3 node and split it. Extend the Hit location pin and hook it to the To pin of the Get Unit Direction vector node. Extend the return value and hook it to the right pin of the make rotation from axes node.

Extend the hit F1 normal pin and hook it to the up pin of the make rotation from axes node. Extend the return value and promote it to a new variable called Target Rotation.

If you made the macro trace float then when you try to compile it you might get an error at the set HitF1, set HitF2 nodes etc. this is because the macro is outside the function but it still uses the local variables of the function to which it doesn’t have access to. To solve this make the variables global by doing so: right-click on the node say hitF1 node > replace with > hitF1. Recompile this and the error disappears. Do so for the rest of them. Else if you’re not making it into a macro you wont have this problem and can continue using the local variables.

Go back to the event graph and between the event begin play node and the set timer by event/function node add a set TargetLocation node and extend the input pin and add a get actor location node. Extend the output exec pin and add a set TargetRotation node and extend the input pin and add a get actor rotation node. This is to prevent the character’s location and rotation to being set to the default values of both the variables.

Make an event/function called move extend it and add a branch node. We need to execute this only if the character is moving so add a get move forward node extend it and add a get absolute value node and extend it and add a float + float node. Add a get move right node, extend it and add a get absolute value node and hook the return value to the B pin of the float + float node. extend the return value of the float + float node and add a float>float node and set the other value to 0. Hook the return value to the condition pin of the branch node. Extend the true pin and add a add movement input node. Extend the return value of the get actor location node and add a get unit direction node. Extend the Target Location node’s return value and hook it to the To pin of the get unit direction node. Hook the return value to the world direction pin of the add movement input node. The scale can be left at 1 itself as we are already checking the movement input but you can tweak it accordingly.

Now on simulation the character moves on the surface but it won’t rotate to align itself on the surface. Make sure to call the move function/event in the Trace function/event.

Extend the output exec pin of the add movement input node and add a set actor rotation node. Add a get actor rotation node, extend the return value and add a lerp node and extend the B pin and add a get target rotation node. Set the alpha to say 0.01 and check the shortest path option. Hook the return value of the lerp to the new rotation pin of the set actor rotation node.