Documentation changes in Pouring Simulation

Requirements

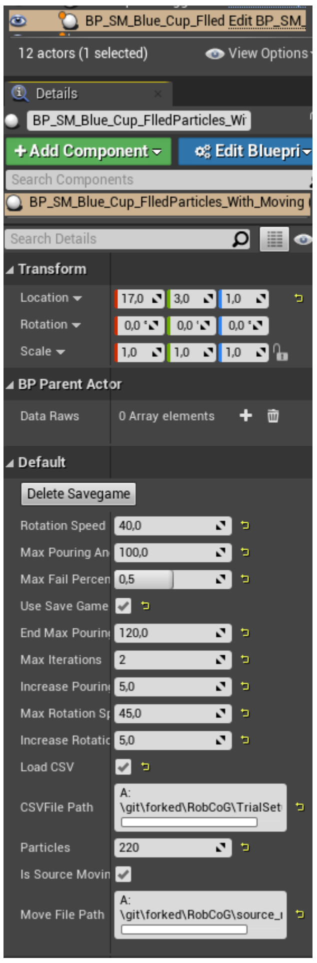
This simulation requires Unreal Engine version 4.27.

How to use

Parameters and Files

The most advanced version is in the Maps/Simulation/SM_Blue_cup/DestSM_BigBowl folder. The name of the blueprint is "BP_SM_Blue_Cup_FillParticles_With_Moving".

This version does fill the source cup within the first trial in any SaveGame. If the cup should be refilled, because a failure appears, stop the simulation before the first trial ended or delete the Savegame by hitting the "Delete Savegame" button within the Default parameters. Here a screenshot of these parameters can be found:


BP_SM_Blue_Cup_Flled Edit BP_SM_

12 actors (1 selected)

BP_SM_Blue_Cup_FlledParticles_With_Moving

Transform	
Location	17,0 / 3,0 / 1,0
Rotation	0,0° / 0,0° / 0,0°
Scale	1,0 / 1,0 / 1,0
BP Parent Actor	
Data Raws	0 Array elements
Default	
Delete Savegame	
Rotation Speed	40,0
Max Pouring An	100,0
Max Fail Percen	0,5
Use Save Game	✓
End Max Pourin	120,0
Max Iterations	2
Increase Pourin	5,0
Max Rotation S	45,0
Increase Rotatic	5,0
Load CSV	✓
CSVFile Path	A:\git\forked\RobCoG\TrialSet
Particles	220
Is Source Movin	✓
Move File Path	A:\git\forked\RobCoG\source_

In case the simulation should run regularly, only with the first three parameters as set up, the Use Save Game File and all other check marks (LoadCSV, Is Source Moving) should be checked off. In all other cases this must be activated.

In case it is not activated, make sure to turn off "Load CSV" and "Is Source Moving" as well. If that is the case, the script will work in an infinite loop until the user stops the simulation. The Parameters

“Rotation Speed”, “Max Pouring Angle” and “Max Fail Percentage” will be the only used parameters in this case. Explanation of these parameters:

-Rotation Speed describes the Rotation Speed in Degree per second.

-Max Pouring Angle is how far the source container will be turned.

-Max Fail Percentage needs a number between 0.0, which is 0%, and 1.0, which is 100%. This are the amount of partilces that could drop on the table, without counting as failure.

The following parameters are used with the “Use Save Game Script” only, so “LoadCSV” and “Is Source Moving” must be turned off. These are the parameters:

-End Max Pouring Angle: Since the script starts with the very first “Max Pouring Angle” it will increase the Pouring Angle by the Amount set in “Increase Pouring Angle By” until the “End Max Pouring Angle” is reached.

-Max Iterations is the amount of trials that will be performed for each possible trial configuration. This means the amount of Trials could be calculated by: ((EndMaxPouringAngle-MaxPouringAngle)/Increase Pouring Angle by) * ((MaxRotationSpeed-RotationSpeed)/IncreaseRotationSpeedby) * Number of iterations.

-“Increase Pouring Angle by” contains the amount of Degrees that pouring Angle should be increased, when the trials iterations are done. If Max Pouring Angle is set to 100, the increase angle to 5 and the iteration to 3, after 3 iterations with 100 it will increase to 105 and so on, until the End Max Pouring Angle is reached.

-“Increase Rotation Speed” by and “Max Rotation Speed” behave the same.

The total amount of iterations that are going to be done multiply, because each different setting is tried out. So the script does each rotation angle with each speed the amount of rotations.

The Save Game gets automatically deleted when all iterations are done. A restart of the Engine does not delete the Save Game, click the “Delete Save Game” button if needed.

In case the “Load CSV” is checked on (**the path must be set correctly, otherwise it crashes**) the above parameters are ignored, but make sure “Use Save Game Script” is turned on too. This parameters loads the CSV File containing the parameters. Each line in the CSV File is one trial set up. The lines should look like this:

MaxPouringAngle,RotationSpeedInDegreePerSecond,NumberOfIterations

Example:
110,45,2

140,30,2

110,90,2

This file would first use a max pouring Angle with 110 degrees with the rotation speed of 45 degrees per second for two iterations and moves on to the next line. When all lines are done the Save Game is automatically deleted like above.

If the “Source Moving” is turned on, it will Load the Source Moving CSV file and ignore all above. The Source Moving simply contains the relative movements the source container should make. Each line stands for one movement that is executed each 40 ms. A movement line can look like this:

X,Y,Z,Roll,Pitch,Yaw
Example:

0.0,0.1,0.1,1.2,-1.0,0.0

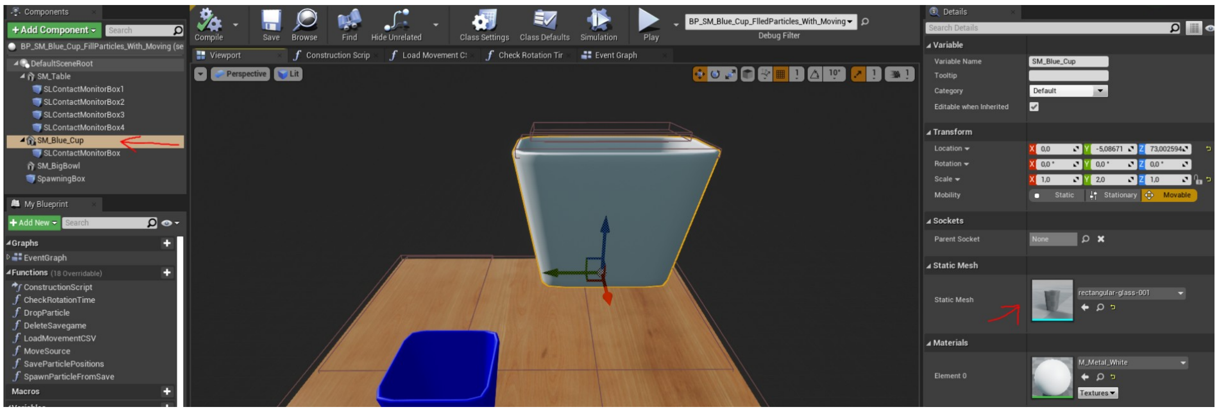
This line would execute a movement of 0.1 in x direction, 0.1 in y direction and 1.2 Degrees in the Roll direction as well as -1.0 to the Yaw direction. This will be executed, afterwards a 40 ms break appears and the next line is going to be executed.

The last parameter is "Particles" which is the amount of particles that are filled into the container. The particles position after the first filling, shortly before the movement begins, is saved, the particles will spawn exact at the position they have been on the next trial, until the Save Game is deleted. In case of a failure while filling the filling should be restarted, make sure that the Save Game has not saved the positions yet or delete the Save Game File with the button just in case.

Exchanging Source or Destination Container

It is recommended to work on a copy of the blueprint with this setup before exchanging the containers or scale them. So the old setups are always available.

To exchange the source or destination container, select the container that should be exchanged on the left site, in the Screenshot below the source container is selected. The red arrow on the left sides shows where to select the container. On the right site, where static mesh is marked with another red arrow, the mesh for the selected container can be exchanged, as well as the scale.



Variable
Variable Name: SM_Blue_Cup
Tooltip:
Category: Default
Editable when Inherited: ✓

Transform
Location: X 0,0 Y -5,08671 Z 73,002594
Rotation: X 0,0° Y 0,0° Z 0,0°
Scale: X 1,0 Y 2,0 Z 1,0
Mobility: Static | Stationary | [✓ Movable]

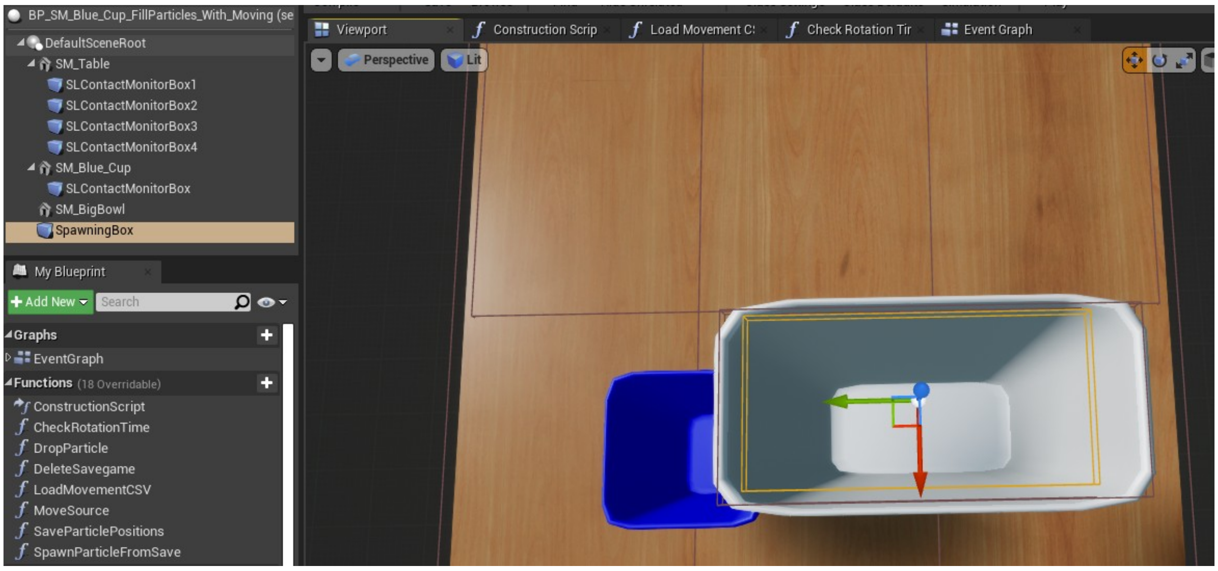
Sockets
Parent Socket: None

Static Mesh
Static Mesh: rectangular-glass-001

Materials
Element 0: M_Metal_White
Textures

Make sure that the SLContactMonitorBox is right above the opening of the source container.

Also make sure that the spawning box is right above the opening, within the extend of this box the particles will spawn. It is okay if this box is smaller than the opening, just has less random places where to spawn the current particle during the dropping phase. If its larger the particles could spawn and drop outside of the container. In the next screenshot the spawning box is marked.

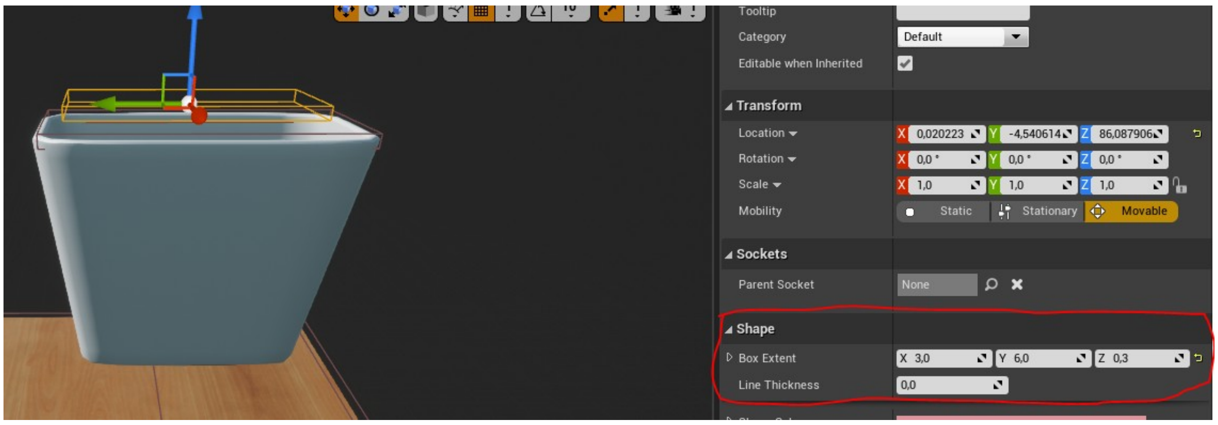

BP_SM_Blue_Cup_FillParticles_With_Moving (se
- DefaultSceneRoot
  - SM_Table
    - SLContactMonitorBox1
    - SLContactMonitorBox2
    - SLContactMonitorBox3
    - SLContactMonitorBox4
  - SM_Blue_Cup
    - SLContactMonitorBox
  - SM_BigBowl
  - SpawningBox

My Blueprint

Graphs
- EventGraph

Functions (18 Overridable)
- ConstructionScript
- CheckRotationTime
- DropParticle
- DeleteSavegame
- LoadMovementCSV
- MoveSource
- SaveParticlePositions
- SpawnParticleFromSave

If this box needs to be scaled to fit the new size or the new mesh, make sure to change only the box extant, not the scale. Since the dropping uses the local position and size of the box extant to calculate the dropping position. The Box extant values are located in the details.


Tooltip
Category: Default
Editable when Inherited: ✓

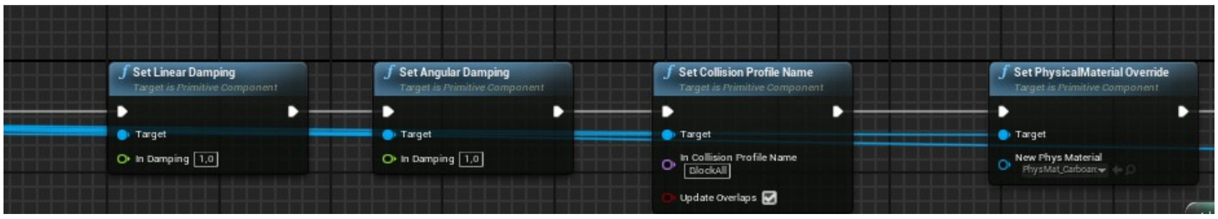
Transform
	X	Y	Z
Location	0,020223	-4,540614	86,087906
Rotation	0,0 °	0,0 °	0,0 °
Scale	1,0	1,0	1,0

Mobility: Static / Stationary / [✓ Movable]

Sockets
Parent Socket: None

Shape
Box Extent: X 3,0 Y 6,0 Z 0,3
Line Thickness: 0,0

If the particles should be exchanged or have different attributes / behaviors, select the “ParticleAttributes” function. This function is used in the “DropParticle” and “SpawnParticleFromSave” functions and sets all particle settings. For example exchange the “SetLinearDamping” value. Also other attributes could be changed by calling new functions if needed.


Set Linear Damping
Target is Primitive Component
Target
In Damping 1,0

Set Angular Damping
Target is Primitive Component
Target
In Damping 1,0

Set Collision Profile Name
Target is Primitive Component
Target
In Collision Profile Name BlockAll
Update Overlaps ✓

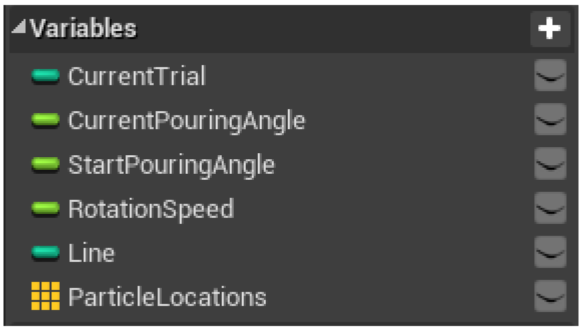
Set PhysicalMaterial Override
Target is Primitive Component
Target
New Phys Material PhysMat_Carboard

Explanation

Save Game File

The Savegame File can be found in /Maps/Simulation/MySaveGame

This file contains the variables that are saved in the save game used during the simulation. Since all variables of the regular blueprints are deleted when executing the “Restart Level” command, that resets the trial and starts it again, this file contains all parameters that needs to be saved. Here are the variables contained within this file:



Variables
- CurrentTrial
- CurrentPouringAngle
- StartPouringAngle
- RotationSpeed
- Line
- ParticleLocations

-CurrentTrial contains the Trial that was previously executed. It gets increased each time a Trial finishes.

-CurrentPouringAngle Contains the PouringAngle it has right now, that is for the first parameters, like EndMaxPouringAngle, to set the correct Pouring Angle for each trial during the script is executed.

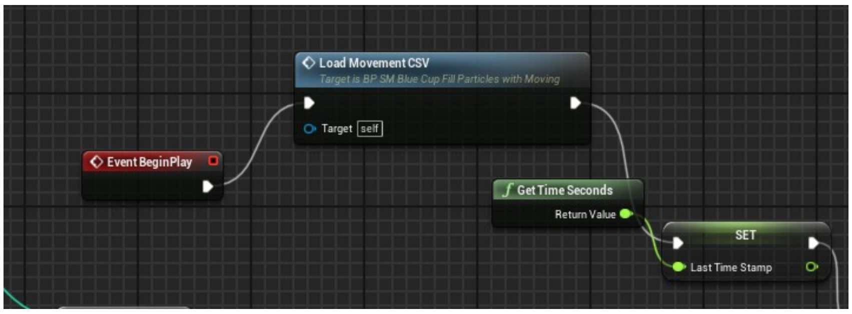
-Rotation Speed behaves the same as above.

-Line contains the current Line number that is executed in the LoadCSV case.

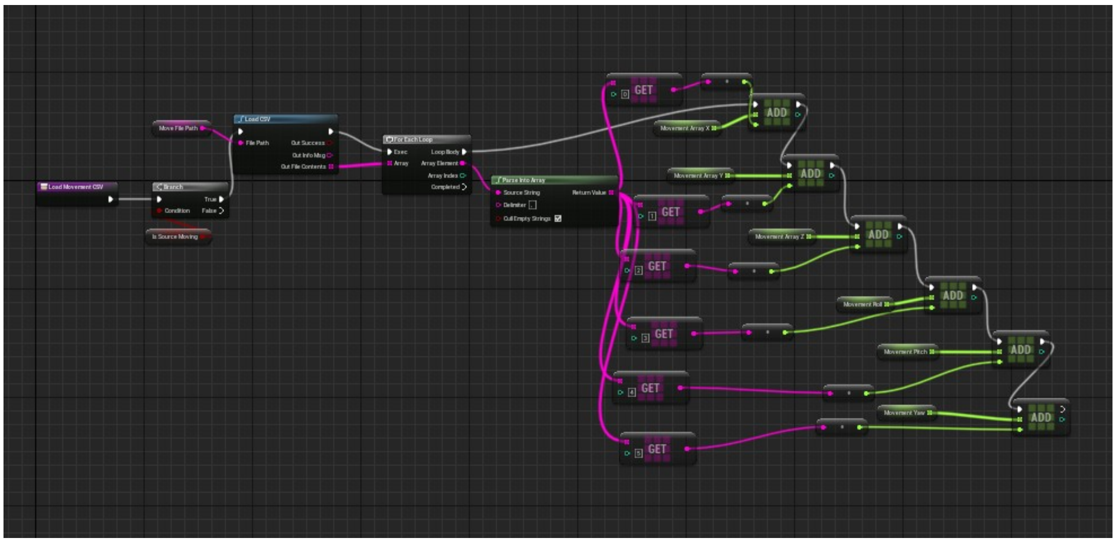
-Particles Locations is a list of Locations, each particle that has been spawned has its positions saved in that list. This is used for the second iteration, so it does not have to spawn and drop each particle again to save time.

Begin Play

Begin play is the part of the unreal script that executes first, when the play button is pressed, before the first ticks appear. That is the time to set up what is needed, also this function is executing when the level is restarted after a trial. It is on the top of the blueprint.



The first thing that is executed is the Load Movement CSV, in this function the CSV File is loaded line by line and each value is stored in a list of floats. Each movement (X, Y, Z, Roll,Pitch,Yaw) has its own list. Maybe here is room for improvement by storing LocationAndRotation directly in a list.

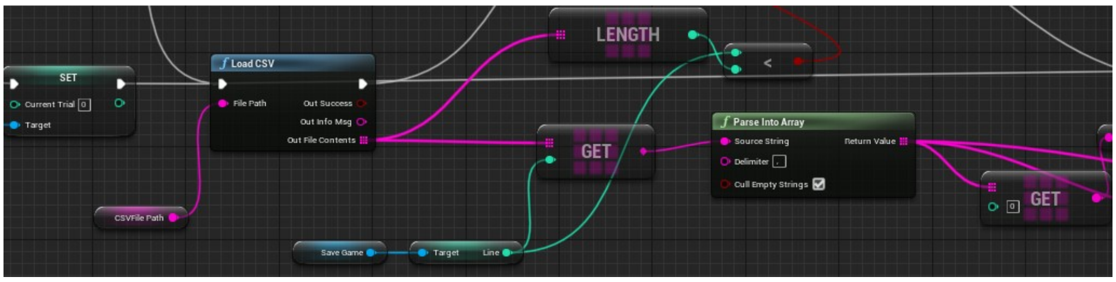


After this the Latest Timestamp and other values will be initialized. Some of those values are the flags like "isStartTimeSet", thats needed because the "EveryTick" that will be executed later on, is called every tick. So not to do something twice and have some kind of mutal execution to set the start time only once at the very beginning, the flag is set after the first moving iteration.

Also if the SaveGame is activated and a Savegame exists, its made sure that the next iteration still should be executed and the settings are going to be applied that should be used on the upcoming iteration. The SaveGame that is loaded will be saved into a local variable for an easier access. In case it does not exist it gets created.

In case the savegame is used for something new, make sure to not just modify the one at the variables by saving the save game later to the used save game slot. The Savegame slots do have string names, the current used one is saved in the string "SaveGameName".

In case that the LoadCSV option is turned on (the SourceMoving needs to be turned off other wise it would overwrite this option), the csv file is loaded within the "Begin Play" function as well. The current line will be read out from the save game and used as index for the string array to mark the line that is currently used. The screenshot below gives a quick overview of this part:

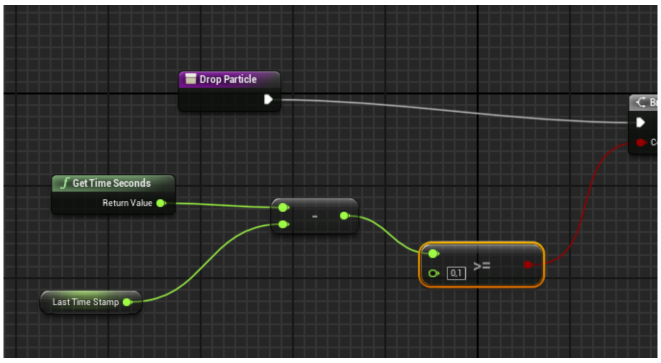


Event Tick

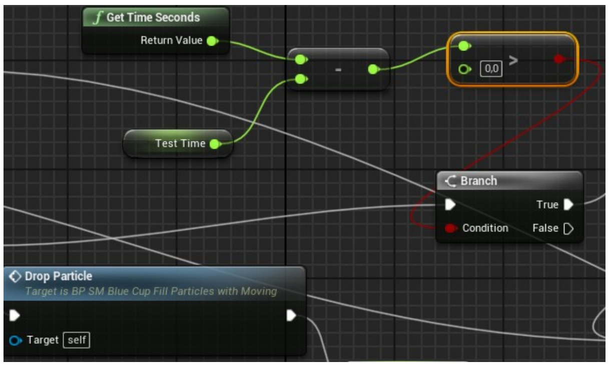
Each event tick starts with a few checks. The first check is if the savegame contains saved positions of the particles. Right after it the next check, in both cases, is if it is filled yet. If its not filled, depending on if it has positions, it will either spawn the particles at the saved positions or fills the cup.

When its getting filled it always saves after a dropped particle the current timestamp at "TestTime", this is to make sure that each particle has time to drop down before spawning the next particle. To change this time that is required to wait, go into the "DropParticle" function.

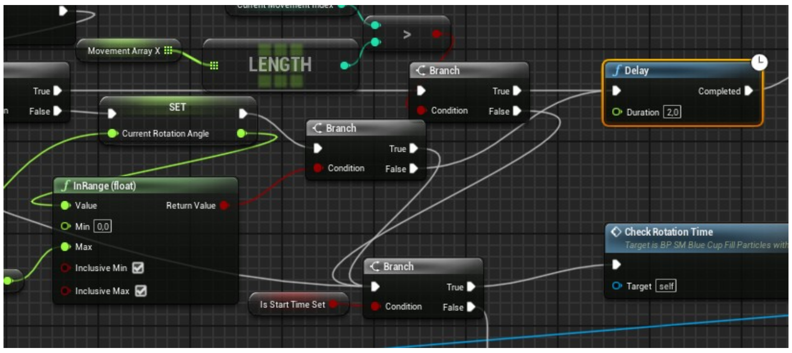
In the marked cell the 0.1 means 100 ms that has at least to be waited until the next particle is allowed to drop.



Afterwards theres a short checkup, that is not really used currently. The tiny check in the screenshot below is an option to have a small break before the moving starts, which has been made for debugging, it could be remove. In case a break time is needed to watch how the particles behave before start of the regular motion, type in any amount of seconds into the marked field. This is only executed after the particles are dropped.

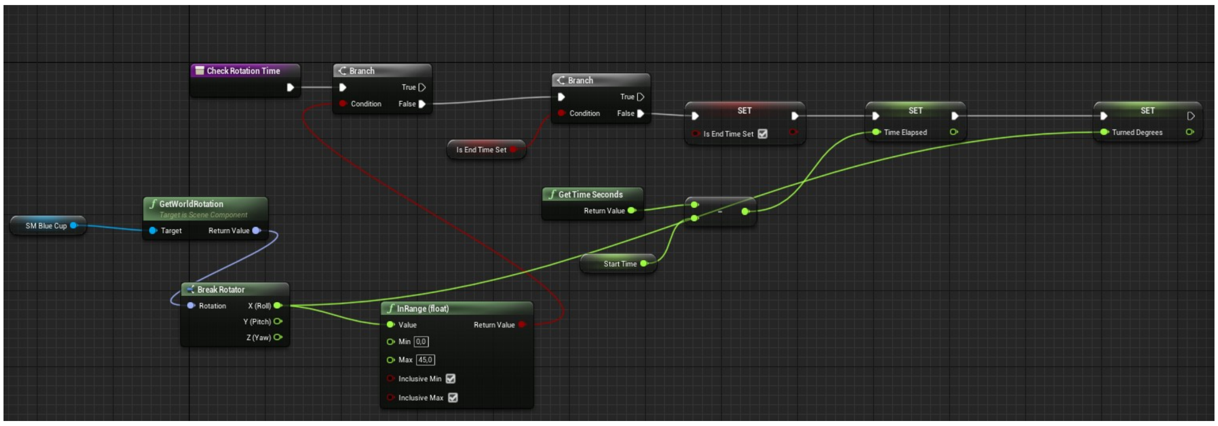


Next the checks are performed if the movement is finished, if this happens a small amount of time is waited to make sure each particle that fell reaches the table. When the waiting time is the Level will be restarted for a new trial. The waiting time is marked in the following screenshot.

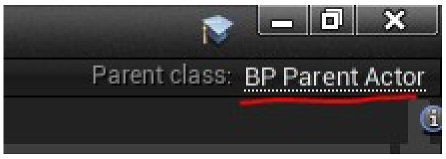


In the following step it checks if the “isSourceMoving” flag from the settings is set, to make sure that movement from the CSV is executed instead of any trial set up. If not the regular movement is executed. In case a CSV that provides trial set ups has been loaded and the SourceMoving flag is turned off, the setup settings have been set previously already in the “begin play” section. The rest of the movement would be same as if the settings were set with the parameters.

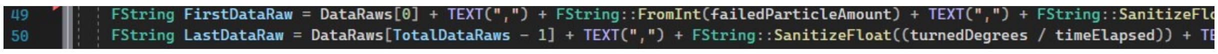
After the movement applied the “CheckRotationTime” is executed, checking how long it took to reach at least 45 degrees. In the following screenshot shows the “CheckRotationTime” function:



This checks at each tick of the simulation if the source container reached 45 degrees to provide an estimation of the rotation speed. The moment the 45 degrees are passed for the first time, the flag for the EndTime is set to make sure this is only executed right after it passed the 45 degrees. The calculation of the rotation speed happens in the Parent_Actor.cpp file. This can be easily accessed by clicking on the top right of the blueprint window “Parent Class: BP_ParentActor” or using the visual studio solution file of the project.

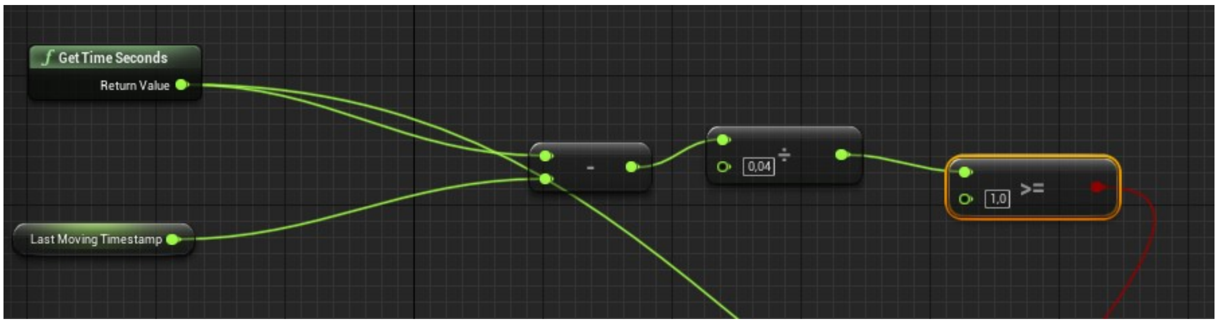


The calculation of the actual speed is made here in line 50.

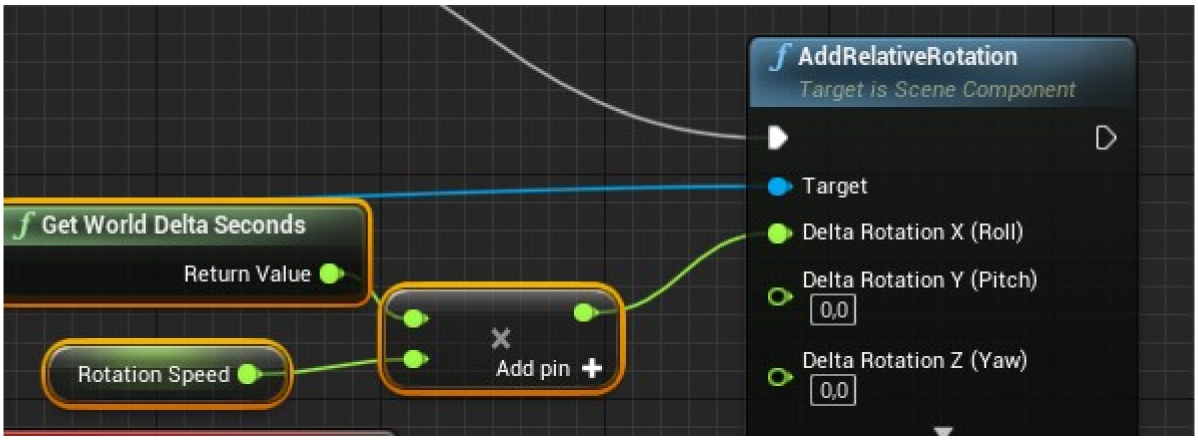


```
49      FString FirstDataRaw = DataRaws[0] + TEXT(",") + FString::FromInt(failedParticleAmount) + TEXT(",") + FString::SanitizeFl
50      FString LastDataRaw = DataRaws[TotalDataRaws - 1] + TEXT(",") + FString::SanitizeFloat((turnedDegrees / timeElapsed)) + TE
```

If the source Moving is activated, the “MoveSource” function is handling the movement of the source container. To change the time that should at least be waited before executing the next line, go into the “MoveSource” function and change the marked value. Currently its set to 0,04 which is 40ms.



In case the regular movement is activated, the calculation on how far to move at each tick is made in this section inside the EventGraph:



GetWorldDeltaSeconds returns how much seconds a tick takes, by multiplying the length of one tick in seconds with the rotation speed in seconds the exact rotation is calculated that needs to be done within this tick, to keep the rotation speed in degree movement.

At the Overlapped function region, those functions are going to be activated whenever an object that has “GenerateOverlapEvents” activated is overlapping with the marked SLContactMonitorBox, in this cases only particles would cause that, the object that hits generated this event is checked if it already is contained in the list of all objects that hit the table already. This could happen if a particle hits one box and rolls over to another one, to make sure it is not counted twice, it is only added when it is not contained in this list already.

In the end, the List “ParticlesHitTheTable” contains a distinct list of all particle references that hit the table. By getting the amount of particles that are within this list, the amount of particles that dropped beside the destination container. By the knowledge of this and the amount of particles that are actually in the scene the calculations of the fail percentage can be made.