ALPHA\_RCRacing / RCRacingPawn

Architecture/Design Document

**Table of Contents**

1 Introduction.. 3

2 Design Goals.. 3

3 System Behavior.. 3

4 Logical View... 3

4.1 High-Level Design (Architecture of the Entire system)… 4

4.2 Mid-Level Design of the RCRacingPawn Module... 5

4.3 Detailed Class Design of the RCRacingPawn Module… 6

5 Process View of the RCRacingPawn Module... 7

6 Use Case View... 16

Change History

**Version:** 0.1

**Modifier:** Antoine Plouffe

**Date:** 03 / 14 / 2022

**Description of Change:** Module Design Document started.

**Version:** 0.2

**Modifier:** Antoine Plouffe

**Date:** 03 / 16 / 2022

**Description of Change:** Logical and Process View updated.

**Version:** 0.3

**Modifier:** Antoine Plouffe

**Date:** 03 / 18 / 2022

**Description of Change:** Adding Use Case Section.

**Version:** 0.4

**Modifier:** Antoine Plouffe

**Date:** 03 / 18 / 2022

**Description of Change:** Adding order to the process view and updating Mid-Level Design.

**Version:** 0.5

**Modifier:** Antoine Plouffe

**Date:** 04 / 11 / 2022

**Description of Change:** Updating all Process View, Mid and Detailed-Level Design of all Power ups to reflect the newly added Multiplayer feature.

1. **Introduction**

This document describes the architecture and design for ALPHA\_RCRacing, a game being developed by Inertial Sketch. ALPHA\_RCRacing is a Multiplayer RC Car Game where player compete against each other in a competitive racing track filled with jumps, turns and Power Ups in a multiplayer setting.

The purpose of this document is to describe the architecture and design of the RCRacingPawn Module application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

* Developers;
* Project Manager.

1. **Design Goals**

The design priorities for the RCRacingPawn system are:

* The design should be dynamic enough to interact with different kind of power ups;
* The design should feel natural to the player;
* The design should allow the Designers to modify key variables within the Engine;
* The design should let the player reset the car if it is stuck;
* The design should work in a Multiplayer setting.

1. **System Behavior**

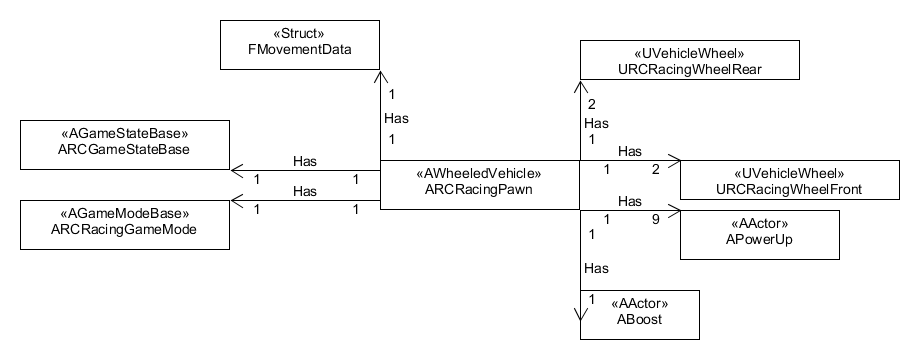
The RCRacingPawn module is built from a single RCRacingPawn and is linked to multiple class system like the UI/Menu, PowerUp, Boost and RCRacingWheels. This configuration will allow a centralization of the different system to create a cohesive game.

1. **Logical View**

The logical view describes the main shared components, attributes and switches of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed from a macro perspective and progressively goes to a micro perspective to view the detailed sequences and components.

* 1. **High-Level Design (Architecture of the Entire system)**
* RacingPawn System is the main system and handles the car movements, current power up and various effects.
* PowerUp System knows its mesh and collision sphere as well as its cooldown and despawn timer. Its children have unique effects.
* UI System is used to indicate crucial information like the player’s current power up.
* Menu System handles the selection of car, map and handles the race creation.
* InputComponents system handles the controls for PC, Console and VR.
  1. **Mid-Level Design of the RacingPawn Module**



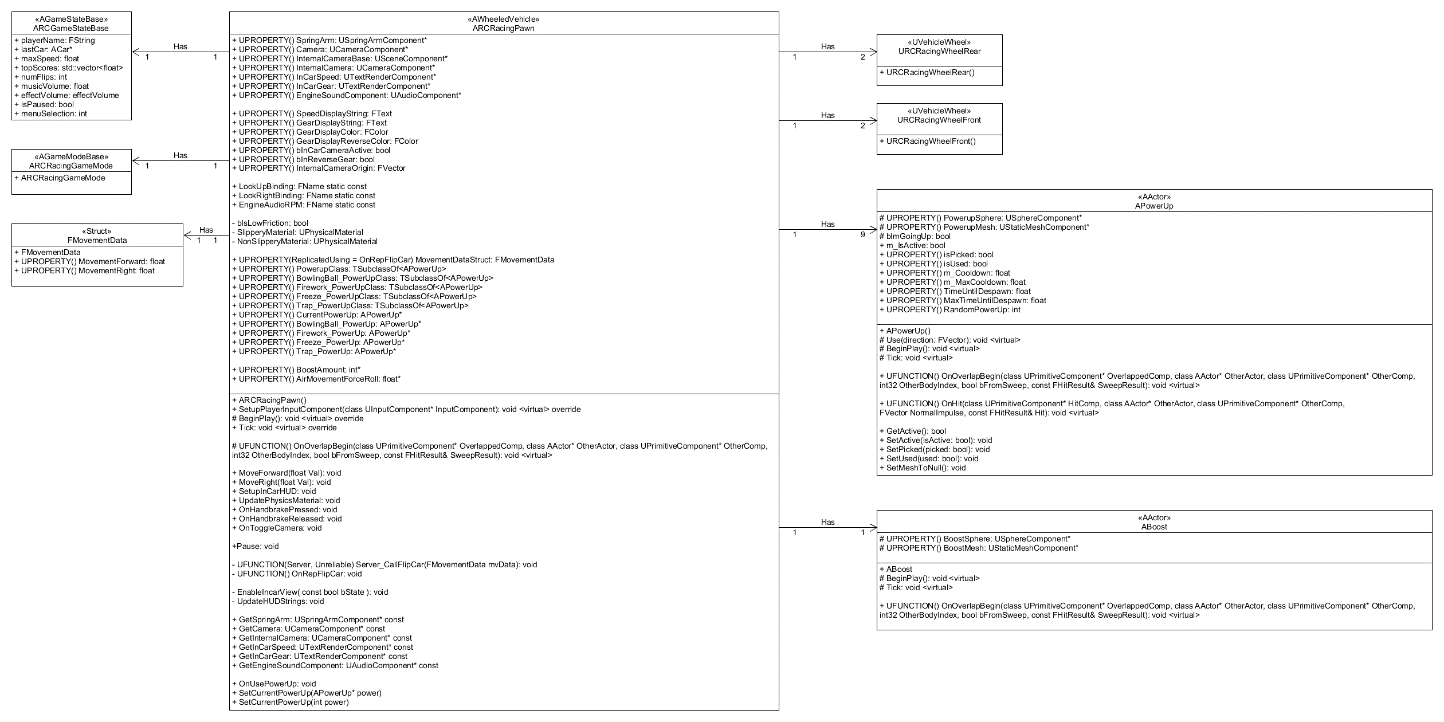
The RCRacingPawn is composed of different classes. Besides the normal GameState and GameMode, the RCRacingPawn has two WheelRear and two WheelFront, nine power ups including setters. It also has access to a boost pad.

ALPHA 2 UPDATES

The RCRacingPawn has been updated to support Multiplayer interactions. This includes the creations of Server Functions for all power ups, updating the OnUse Function and the Server\_CallFlipCar function.

The RCRacingPawn also have a new Struct Class FMovementData that holds the forward and right Key Input data. This has been put in place to remove the double call of the data. It used and called in the Server\_CallFlipCar function.

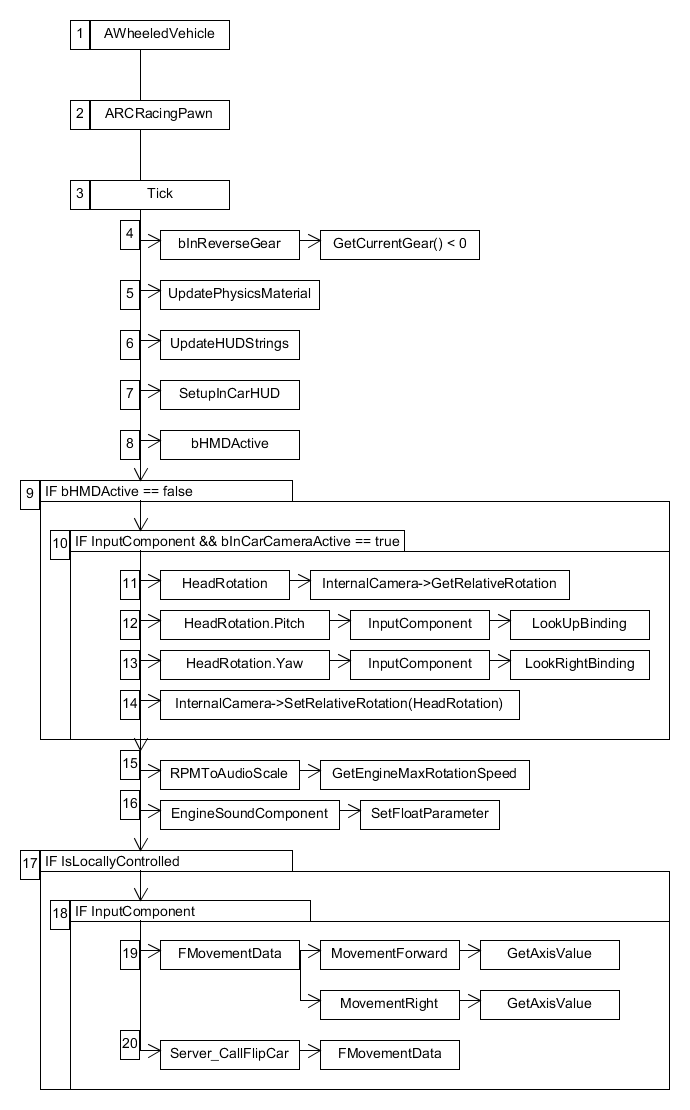
* 1. **Detailed Class Design of the RacingPawn Module**

****

1. **Process View of the RacingPawn Module**

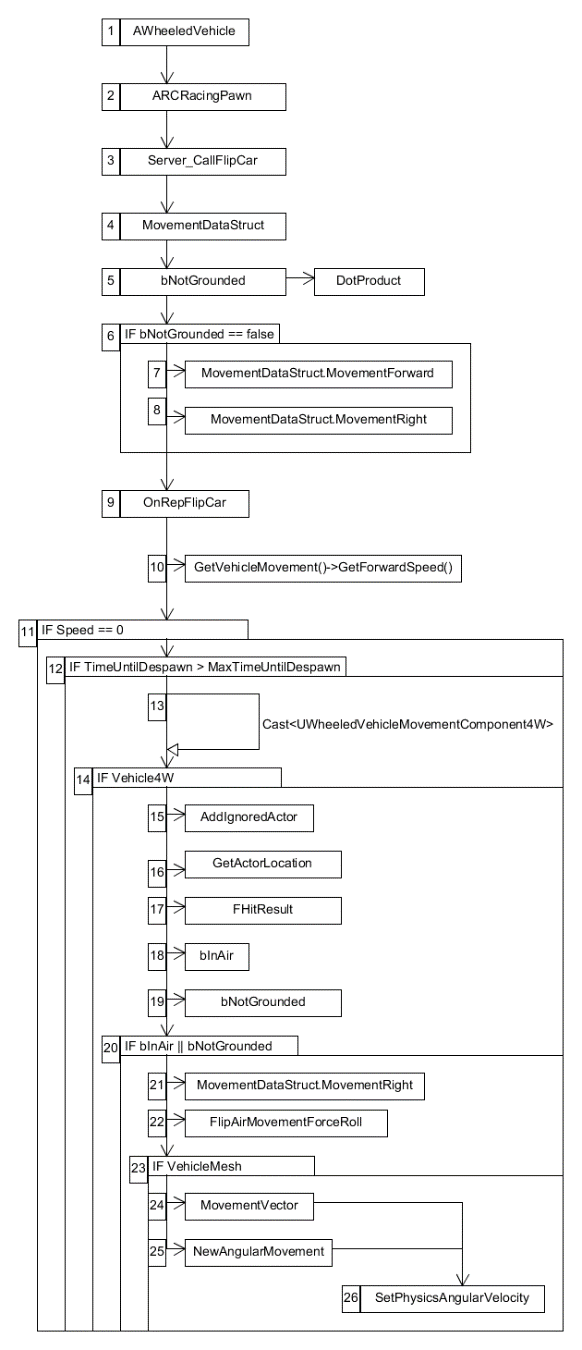
The operation related to the RacingPawn are dependant of factors like isPicked and isUsed. Some power ups are also dependant on their cooldown to activate. The power up isn’t active until it gets assigned when PowerUp OnOverlapBegin is triggered.

**TICK**



The Tick sequence most important part starts at 17. If the vehicle is being locally controlled and if there’s an InputComponent, it will update the Struct class called FMovementData. It takes the AxisValue of the players input and stores them inside two variables called MovementForward and MovementRight. Then, it will be passed to Server\_CallFlipCar as an argument to be used later on.

**ONREPFLIPCAR**



The OnRepFlipCar sequence goal is to prevent the player getting their vehicle upside down and/or stuck in a Multiplayer setting.

The function is first being called by the Server Function Server\_CallFlipCar. This function is server based so both parties can use, see its interactions on screen. It looks if the vehicle is grounded with the boolean bNotGrounded. bNotGrounded uses a DotProduct to detect if the player’s up vector is parallel to an UpVector. If the DotProduct result is under 0.1f, then the vehicle isn’t upside down. If that’s the case, it will take the data from MovementDataStruct and call OnRepFlipCar which replicate the desire movement to the server.

The data from MovementDataStruct is also called inside the GetLifetimeReplicatedProps function which allows the struct to be replicated on the server until the host stops.

The OnRepFlipCar function first looks to see if the car is moving and if the vehicle has a MovementComponent4W. Upon succeeding both checks, CollisionQueryParams will ignore this actor, registers this actor’s location and create two FVector:

1. TraceStart: where the LineTrace starts in the Z Axis;
2. TraceEnd: where the LineTrace ends in the Z Axis.

And one booleans:

1. bInAir uses LineTraceSingleByChannel to detect if the player is currently in the air;

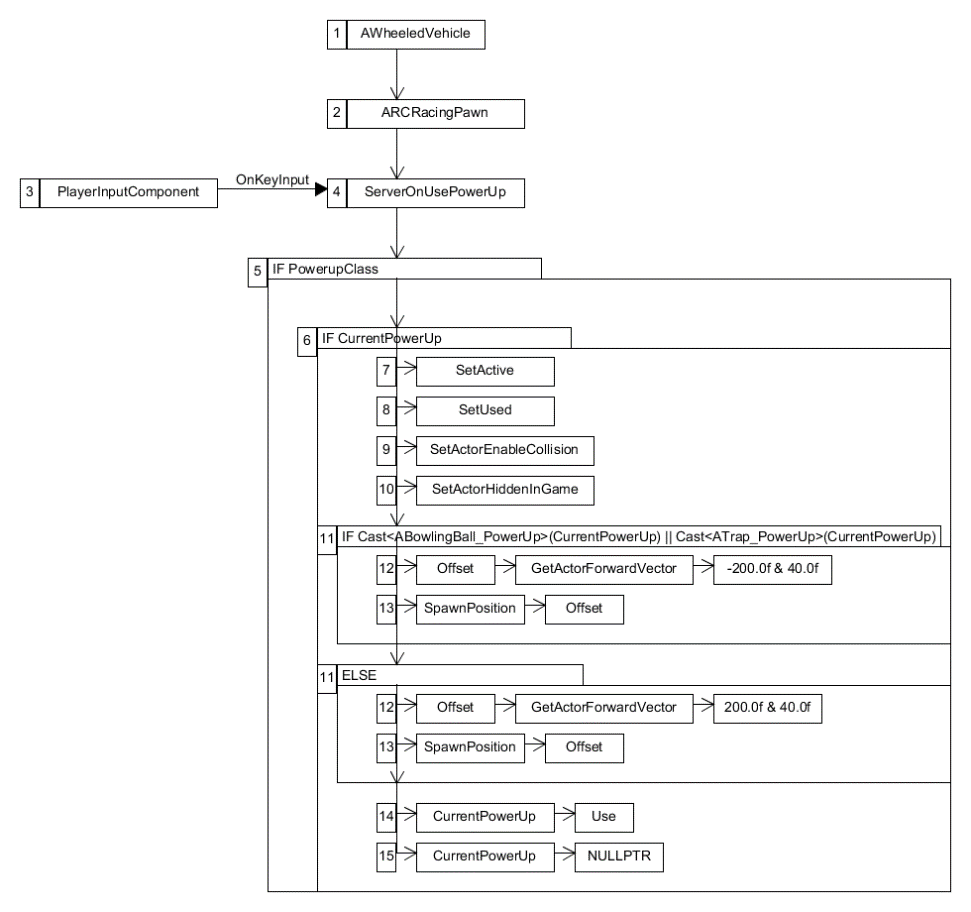
If bNotGrounded return true, it will store the MovementRigh local variables with the axis values of the MoveForward’s input and MoveRight’s input (A value between 1 and -1) located inside MovementDataStruct. It will also store the pitch and roll force that we want to apply.

If a primitive component of type mesh is set on the vehicle, it will calculate a MovementVector by injecting:

1. RightInput and Roll Force;
2. ForwardInput and Pitch force.

Afterwards, it will calculate NewAngularMovement by getting the actor rotation and rotating it by MovementVector. Then the vehicle mesh physics linear velocity is set to NewAngularMovement to flip the car until it is grounded.

**SERVERONUSEPOWERUP**

****

The OnUsePowerUp sequence is dependant of the player’s input (Space bar). Its main goal is to switch a few settings, is to call the virtual Use function of the PowerUp class and is to set its position based on the type of PowerUp. First, it looks to see if the car has a PowerUp class and if it has a CurrentPowerUp active. Secondly and upon succeeding both checks, a series a function calls will be made:

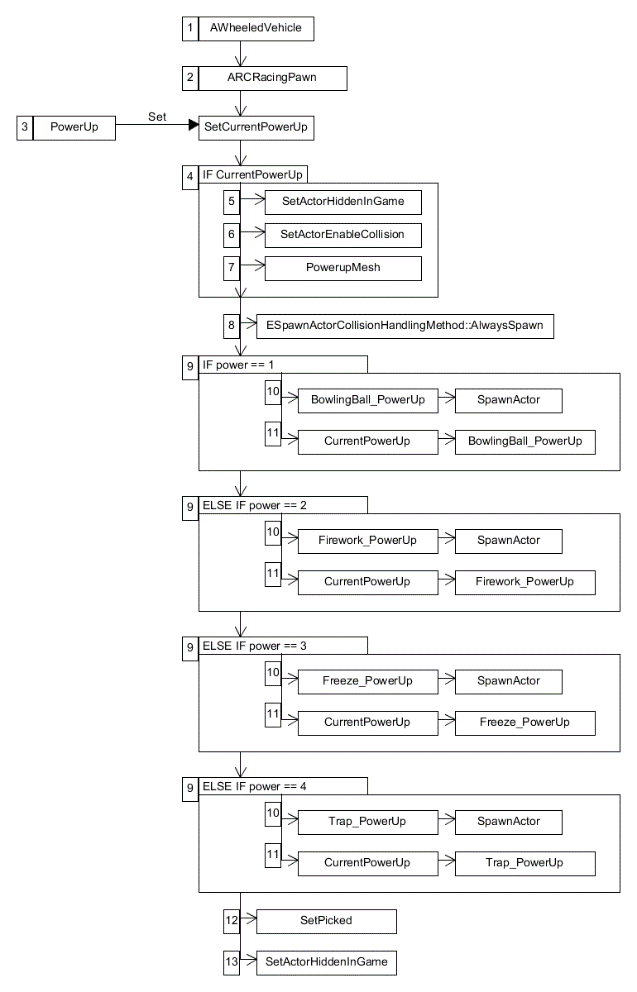
1. The CurrentPowerUp will be set to active;
2. The CurrentPowerUp will be set to Used;
3. The CurrentPowerUp collision’s sphere will be enable;
4. The CurrentPowerUp is set to not be hidden in game;

Then it will cast to see if CurrentPowerUp is either ABowlingBall\_PowerUp or ATrap\_PowerUp. If true, it will set the PowerUp offset to spawn behind the vehicle as they are useful to slow down a player who’s trying to catch up.

If CurrentPowerUp is either AFirework\_PowerUp or AFreeze\_PowerUp, it will PowerUp offset to spawn in front of the vehicle as they are useful to slow down a player who’s currently winning.

Then, the CurrentPowerUp call its function Use by giving it the actor forward vector and the custom offset. Finally, the CurrentPowerUp will be set to nullptr.

**SETCURRENTPOWERUP**



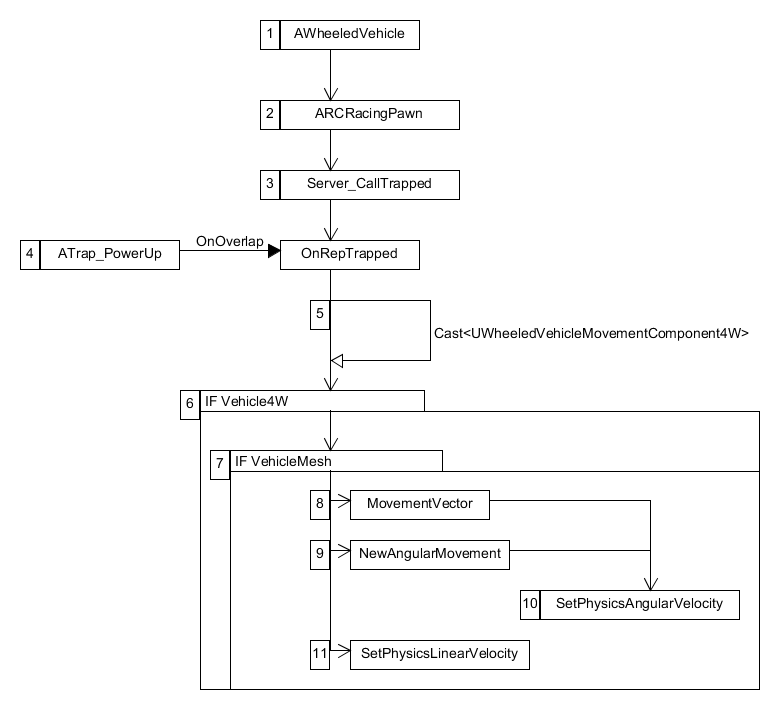
The SetCurrentPowerUp sequence is being called when the player’s overlaps with a power up who’s isPicked Boolean returns false. As the function names suggest, its main goal is to set the player’s current power up to the randomly chosen one inside the PowerUp class.

First, if CurrentPowerUp isn’t nullptr, CurrentPowerUp will set the itself to hidden, its collision to none and its mesh to nullptr. The server will notice that no one is calling the power up and will destroy it automatically.

Then depending on the result of the random number generation (1 to 4) done inside PowerUp, the function will give the corresponding PowerUp and set the CurrentPowerUp to that PowerUp. Finally, a series a function calls will be made:

1. The CurrentPowerUp will be set to Picked;
2. The CurrentPowerUp will be set to be hidden in game.

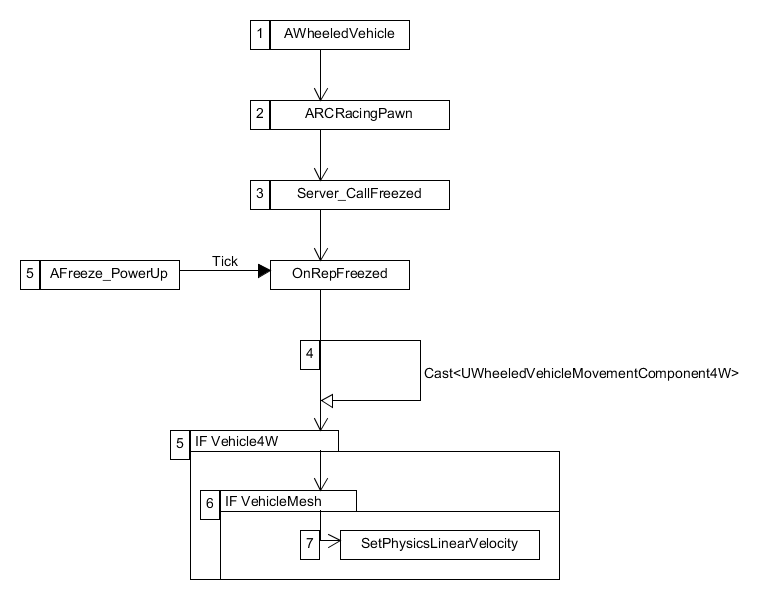
**ONREPTRAPPED**



The OnRepTrapped sequence starts when the player’s overlaps with the collision sphere of a ATrap\_PowerUp. The main goal if this function is to create a wow moment when the player overlaps a trap by moving upwards and rotating their vehicle mesh.

The function first is being called on the server with the function Server\_CallTrapped and looks to see if the vehicle has a MovementComponent4W and a primitive component of type mesh. Upon succeeding both checks, it will calculate a MovementVector by adding AirMovementForceRoll to the X Axis and NewAngularMovement by getting the actor rotation and rotating it by MovementVector. Then the vehicle mesh physics linear velocity is set to 1000.0f in the Z axis to simulate a jump and the vehicle physics angular velocity is set to that NewAngularMovement variable.

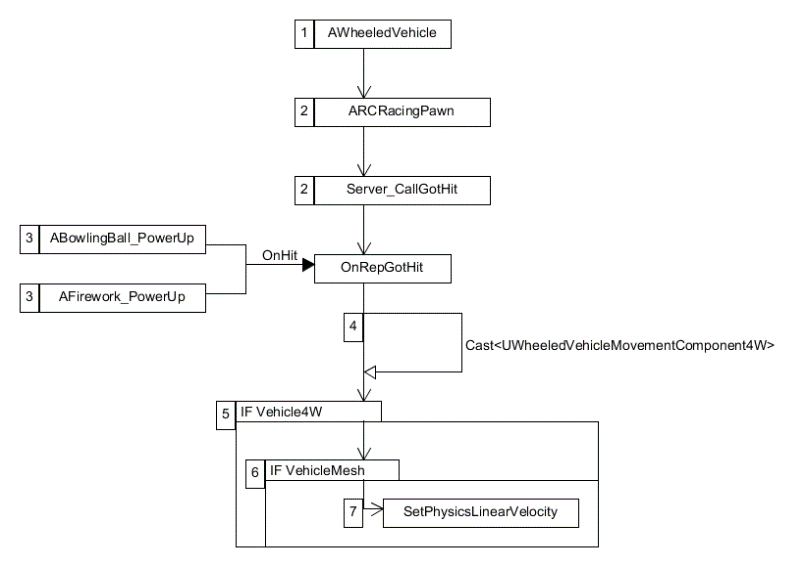
**ONREPFREEZED**



The Freezed sequence starts if the player’s overlaps with the blast radius of a AFreeze\_PowerUp. The main goal if this function is to apply the freeze effect on the vehicle mesh.

The function first is being called on the server with the function Server\_CallFreezed and looks to see if the vehicle has a MovementComponent4W and a primitive component of type mesh. Upon succeeding both checks, it will reduce vehicle physical linear velocity to zero.

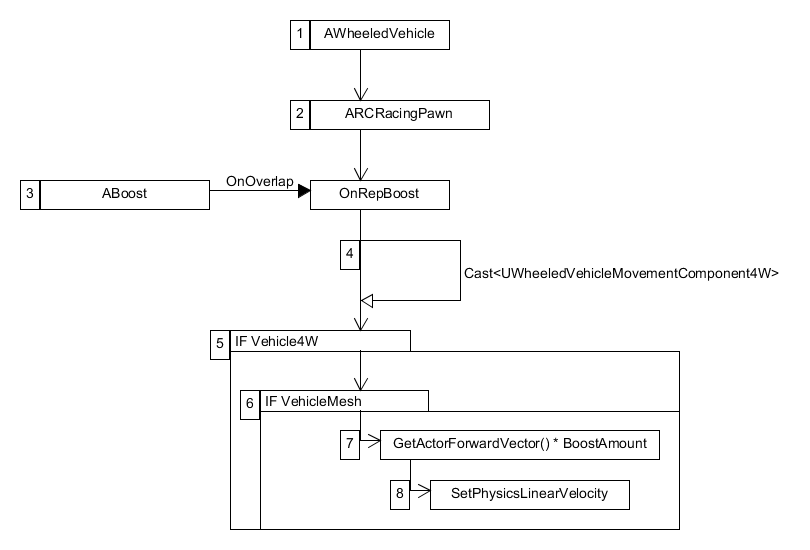
**ONREPGOTHIT**



The OnRepGotHit sequence starts when the player’s hit with the collision sphere of either the ABowlingBall\_PowerUp or the AFirework\_PowerUp. The main goal if this function is to temporarily paralyzed the player’s vehicle OnHit.

The function first is being called on the server with the function Server\_CallGotHit and upon succeeding both checks, it will set the vehicle mesh physics linear velocity on the Z axis to 500.0f and will briefly stop moving.

**ONREPBOOST**

****

The Boost sequence starts when the player’s overlaps with the collision sphere of ABoost. The main goal if this function is to temporarily boost the player’s vehicle speed.

The function first is being called on the server with the function Server\_CallBoost and upon succeeding both checks, it will create a local variable of type FVector. Then, it will set its value to the ForwardVector of the actor times its BoostAmount. Finally, it will add to the vehicle mesh physics linear velocity this boost force.

1. **Use Case View**

For this release, all power ups indicators are displayed via debug messages. In normal gameplay, the type of power up will be indicated by an icon.

The RCRacingPawn has multiple variables related to its design – all editable either within code or inside the Unreal Editor. They can be divided in two sections.

Functionalities from UE4 Vehicle Base Class:

* SpringArm
* Camera
* InCarSpeed
* InCarGear
* EngineSoundComponent
* HMDFunctions
* bInReverseGear

Functionalities for PowerUp:

* PowerupClass: Holds the parent PowerUp BP.
* CurrentPowerUp: Is the current power up the player can use.
* BowlingBall\_PowerUpClass: Hold the BowlingBall BP.
* Firework\_PowerUpClass: Holds the Firework BP.
* Freeze\_PowerUpClass: Holds the Freeze BP.
* Trap\_PowerUpClass: Holds the Trap BP.
* BowlingBall\_PowerUp: Sets the CurrentPowerUp to itself.
* Firework\_PowerUp: Sets the CurrentPowerUp to itself.
* Freeze\_PowerUp: Sets the CurrentPowerUp to itself.
* Trap\_PowerUp: Sets the CurrentPowerUp to itself.

The RCRacingPawn is set upon choosing the map you want to race in. It will interact with the environment, power ups, gravity and other players.

**ONREPFLIPCAR**

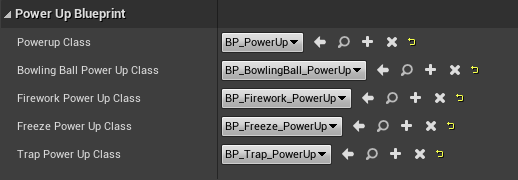
If the player’s car is stuck upside down, the player can simply use the MoveRight’s input key to rotate vehicle either left or right to continue the race.

**ONUSEPOWERUP**

For this release, all power ups will impact the player who activated them for testing purposes. In normal gameplay, this won’t be the case.

Once the CurrentPowerUp has been set, the player can choose to use it using the InputKey: Space. The results will differs based on which power up has been used.

**POWERUPS**



The power ups are set inside the RCRacingPawn blueprint. Inside you will find the PowerUp Blueprint category which will hold the different power ups characteristics and set ups, including their parent.