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

5.1 Process View of the ABowlingBall\_PowerUp Submodule… 9

5.2 Process View of the AFirework\_PowerUp Submodule… 12

5.3 Process View of the AFreeze\_PowerUp Submodule… 15

5.3 Process View of the ATrap\_PowerUp Submodule… 16

6 Use Case View... 17

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.

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.

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.

1. **System Behavior**

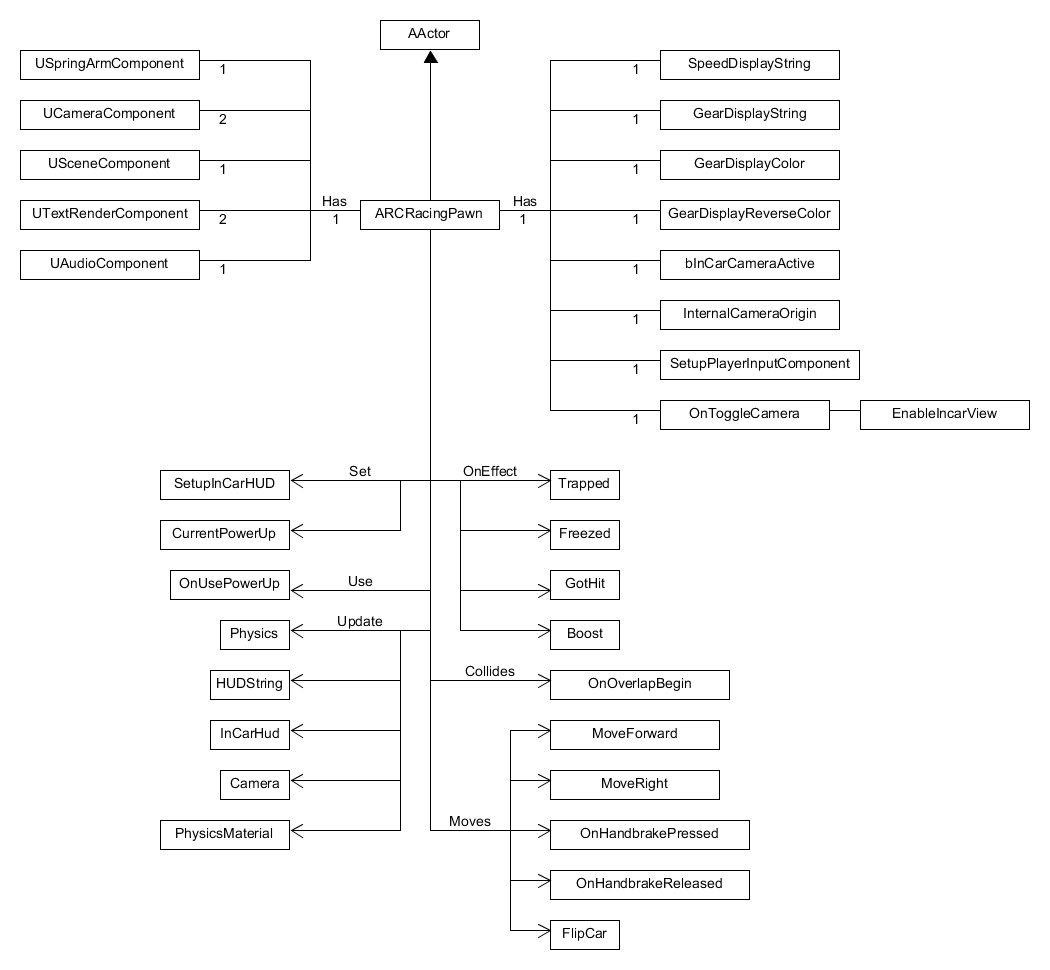
The RCRacingPawn module is built from a single RCRacingPawn and is linked to multiple class system like the UI/Menu, PowerUp, Boost and RCRacingWheels. This configuration will allow a centralization of the different system to create a cohesive game.

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…
  1. **Mid-Level Design of the RacingPawn Module**



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

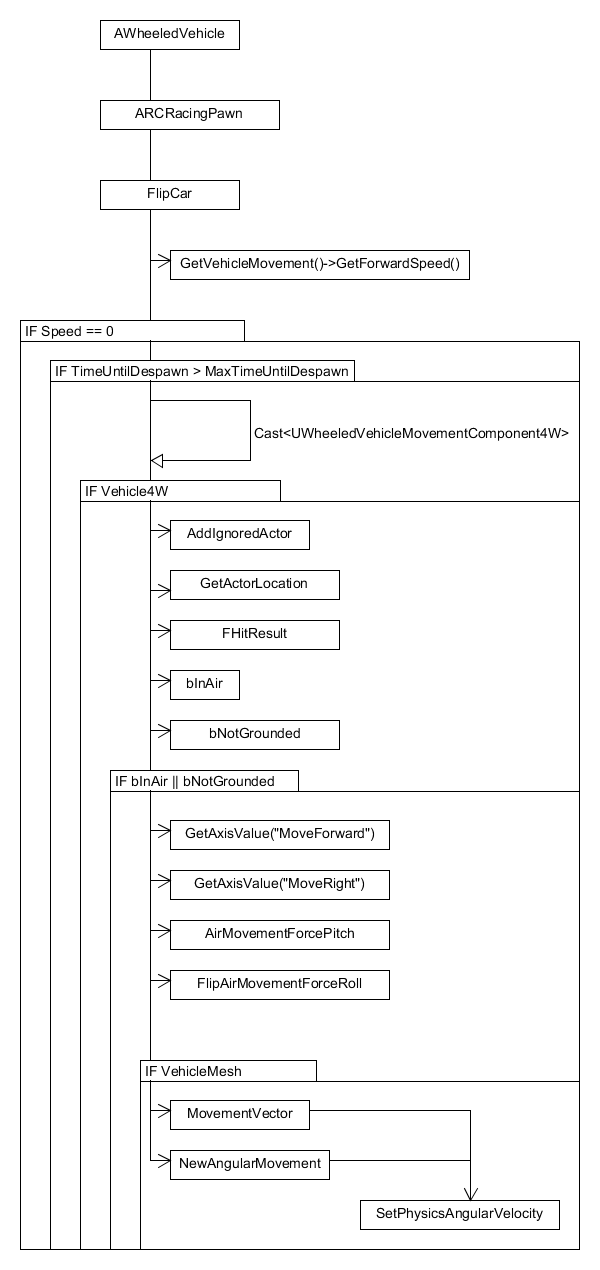
1. **Process View of the RacingPawn Module**

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.

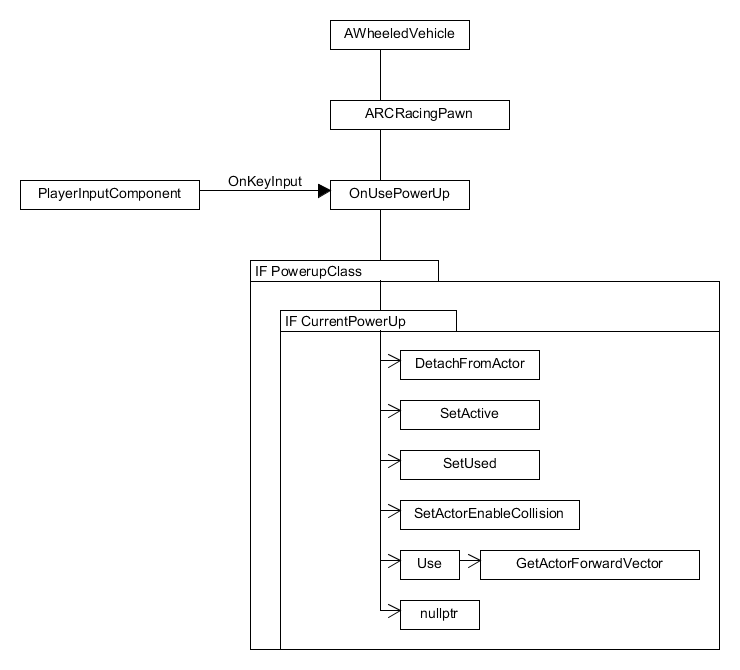
**ONOVERLAP**

**TICK**

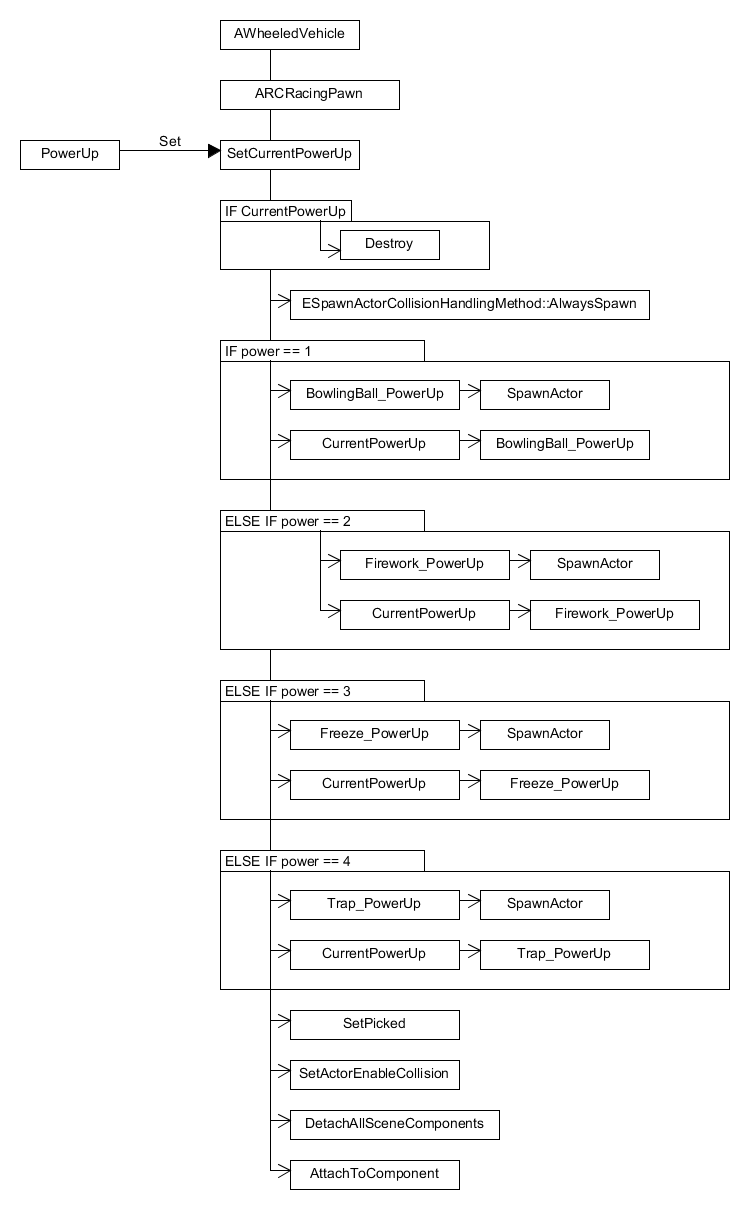
**FLIPCAR**



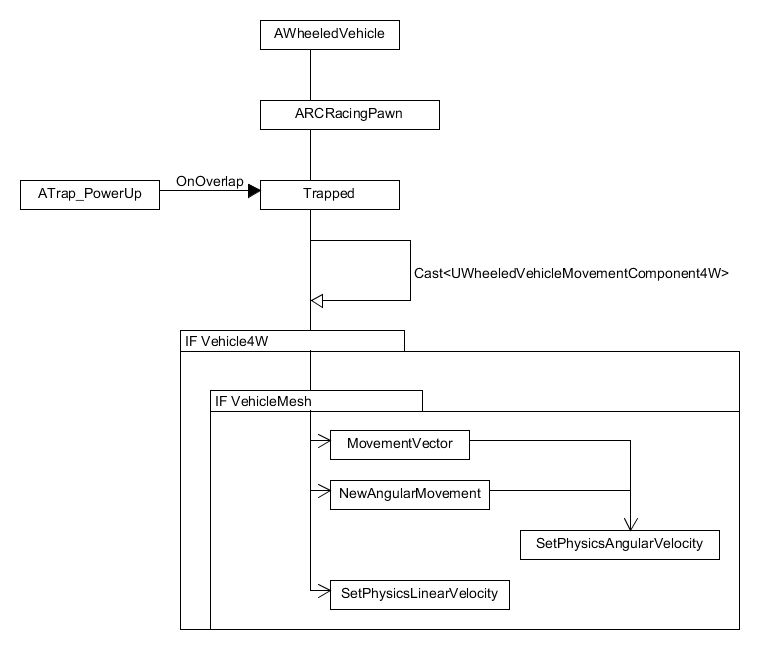
**ONUSEPOWERUP**

****

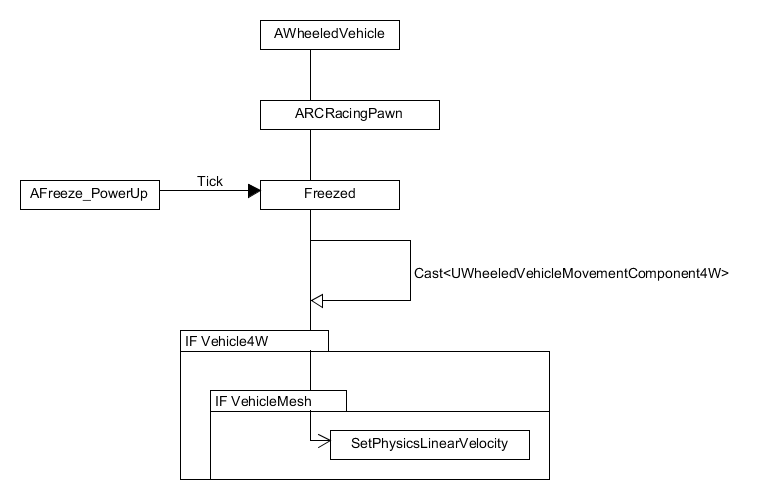
**SETCURRENTPOWERUP**



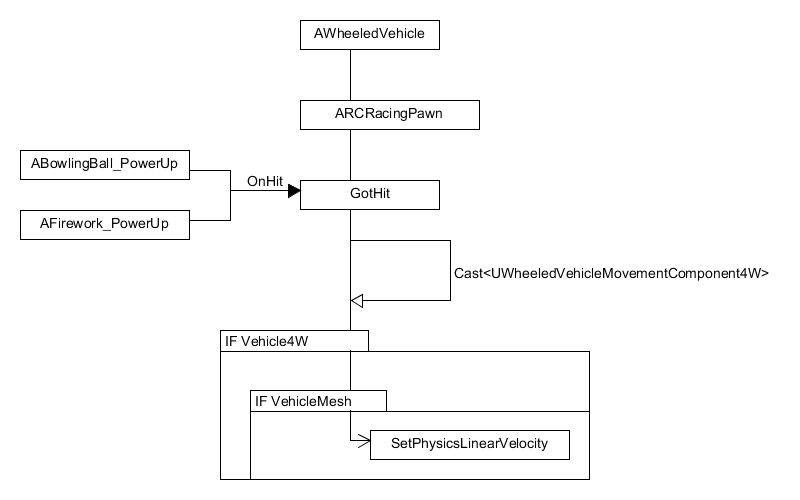
**TRAPPED**



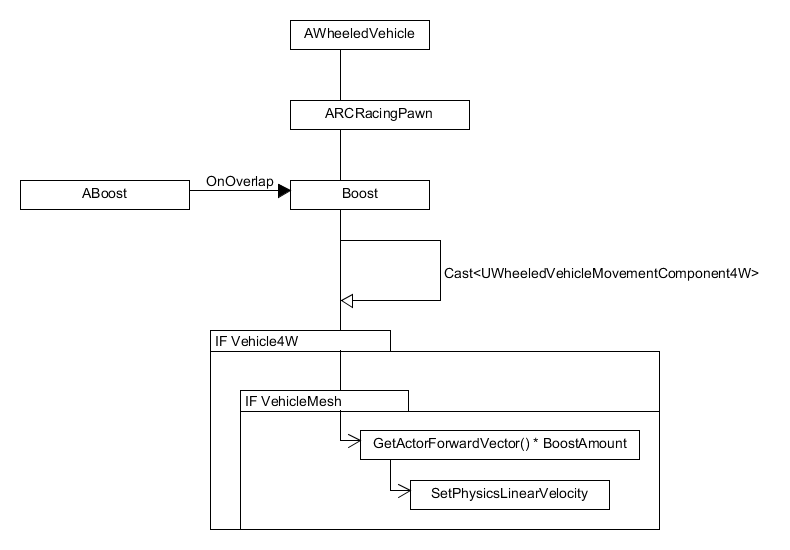
**FREEZED**



**GOTHIT**



**BOOST**

****

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 PowerUp has multiple variables related to its design – all editable either within code or inside the Unreal Editor.

* isPicked: This switch indicates if the power up has been picked up. Default: false.
* isUsed: This switch indicates if the power up has been used. Default: false.
* m\_MaxCooldown: This timer indicates the maximum cooldown of the power up. Default: 2.0f.
* MaxTimeUntilDespawn: This timer indicates the maximum time before a projectile is despawn. This timer is used on both AFirework\_PowerUp and ABowlingBall\_PowerUp. Default: 5.0f.
* RandomPowerUp: This integer indicates the value associated with the latest given power up to the player. Default: nullptr;

The PowerUp is dragged into the scene where needed and will interact OnOverlap with a vehicle.

**ONOVERLAP**

When a power up overlaps with a vehicle, it will selfcheck if it was picked before. If not, it will continue and cast the OtherActor to ARCRacingPawn. If succeeded, the power up will generate a random FMath::RandRange between 1 and 4 corresponding to a power up and sets the player’s current power up to that random one and destroy itself.

1 = ABowlingBall\_PowerUp

2 = AFirework\_PowerUp

3 = AFreeze\_PowerUp

4 = ATrap\_PowerUp