

# KartGame Template

## Reference Documentation

This document serves as an introduction and overview to the different systems included in the kart game template.

Throughout the kart game template there are references to interfaces rather than objects. This is achieved using an attribute called `RequireInterface` which is implemented in the project. This can be applied to `UnityEngine.Object` serialized fields and then rejects any objects which do not implement the interface.

The kart itself is separated into several systems. The following is a summary of each.

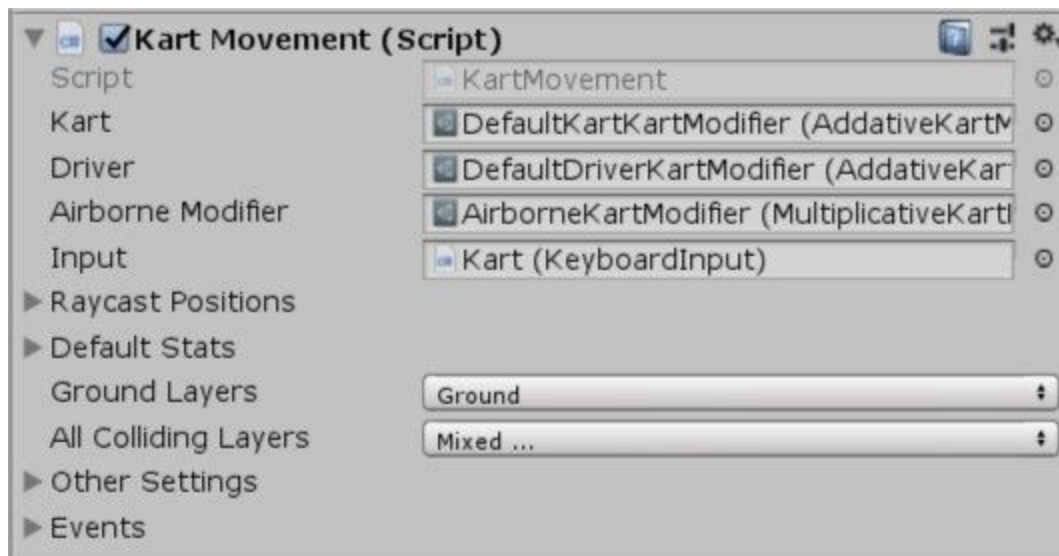
## Kart Systems

### Input

Input is represented by any object implementing the `IInput` interface. The examples given in the template are `MonoBehaviours` which allow for gamepad and keyboard user input with both examples present on the kart prefab. The keyboard input is enabled and referenced by the other scripts by default. Although the only example implementations for `IInput` are user inputs, it could also be implemented by an AI system for other karts.

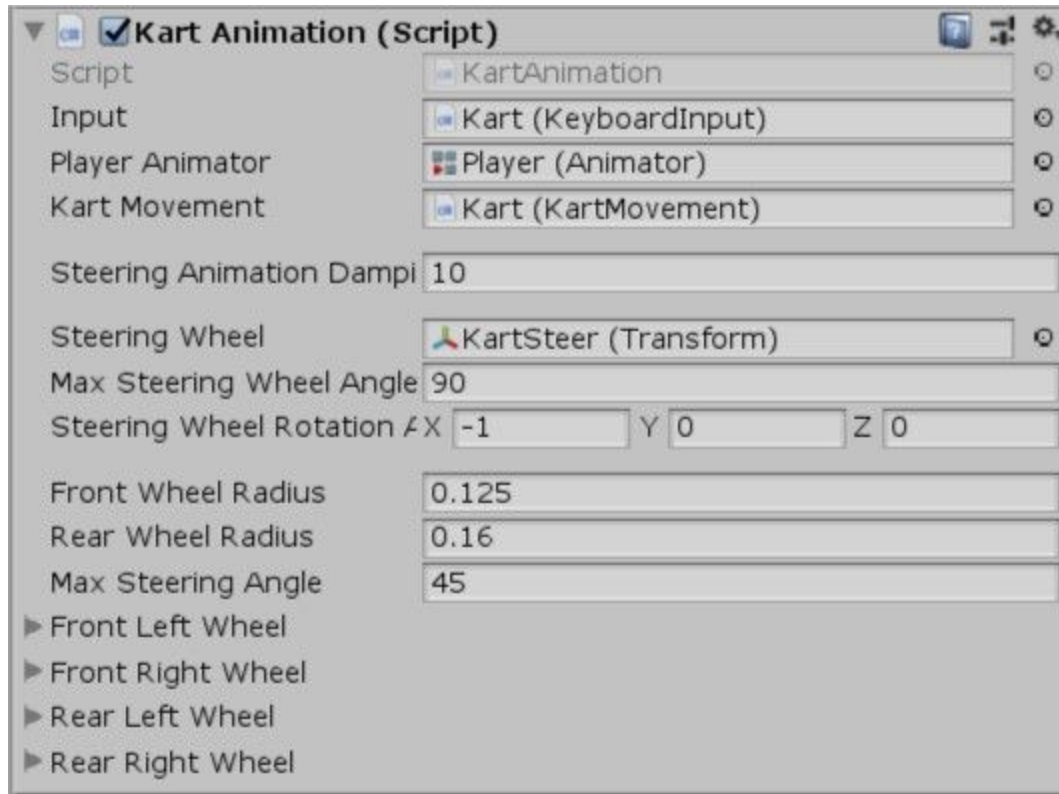
## Movement

Movement is represented by a MonoBehaviour called KartMovement. It takes information from its IInput reference and its own physics implementation to allow the kart to move. These are combined with the kart's stats to define how the kart moves. The kart prefab has a set of default stats, these should function as a default across all karts and should only be tweaked to affect all karts. The stats for an individual kart can be modified using implementations of the IKartModifier interface. Each kart has a collection of objects implementing the IKartModifier interface to modify its default stats to get the stats it uses for velocity calculations. These are deliberately agnostic of concrete implementations so that they can be used for things such as the kart driver, the kart type, powerups, etc. The kart uses a kinematic rigidbody and its own physics implementation and so needs to know what to collide with. It has layer masks to define what it should treat as the ground and all the layers it can collide with. The latter should include the ground layer.



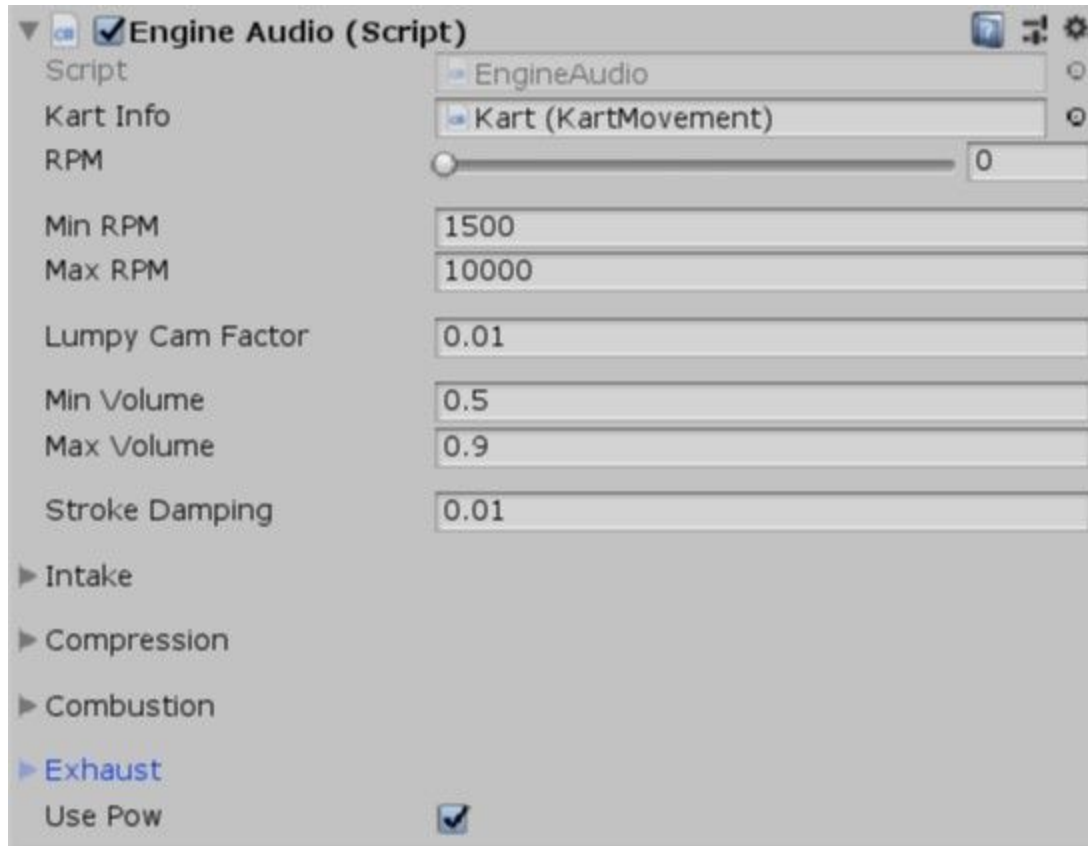
## Animation

The kart has three sources of animation: the kart model is animated, the driver/player is animated and the kart wheels and steering wheel are procedurally animated. The KartAnimation MonoBehaviour manages these different animations.



## Audio

The kart plays various audio clips, the audio sources for which are together under the Audio child gameobject of the kart prefab. Also included there is the EngineAudio MonoBehaviour, which adjusts the sound of the kart's engine procedurally.



## Timing

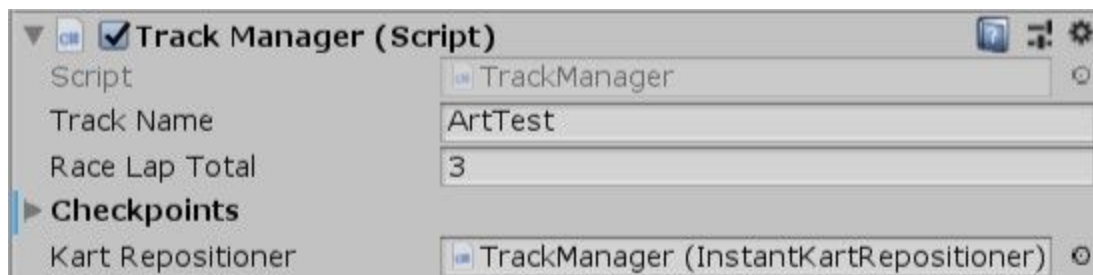
The timing for a track is handled by a separate prefab called TrackManager and the TrackManager MonoBehaviour. It records timing information for all objects in the scene implementing the IRacer interface. The Racer MonoBehaviour is the default implementation of this and exists on the kart prefab.

The other important collection of systems in the kart game template is the track. This works as follows:

## Track Systems

### Manager

All the systems of the track flow through the central point of the TrackManager. It is responsible for dealing with the timings and positions of each of the racers. It uses Checkpoints and implementations of the IRacer interface to achieve this. The TrackManager is also responsible for starting and stopping the race. There are public methods called StartRace and StopRace to do this. These will enable and disable control of all the racers.



## Checkpoints

A Checkpoint represents a part of the track that each racer must pass through in order to complete a lap. It works using a BoxCollider as a trigger and reporting when objects implementing the IRacer interface pass through it. When karts become stuck, fall off the track or are out of bounds they can be repositioned. The last checkpoint a racer goes through is the place they will be repositioned. One checkpoint must be the start-finish line of the track. It is recommended that a track has a minimum of 3 checkpoints roughly equidistant around the track. This will ensure that driving half way around the track and driving back does not complete a lap.

## Time Display

The TimeDisplayCanvas prefab has a TimeDisplay script on. This is responsible for displaying information about the lap/race time of a specific racer. It contains a collection of display options. Adding elements to this collection will allow you to see additional pieces of information. Setting different display options at runtime is possible using the RebindDisplayOptions method. This creates garbage and should be used sparingly. It should be noted that this class deals with strings and so creates a small amount of garbage each frame.

