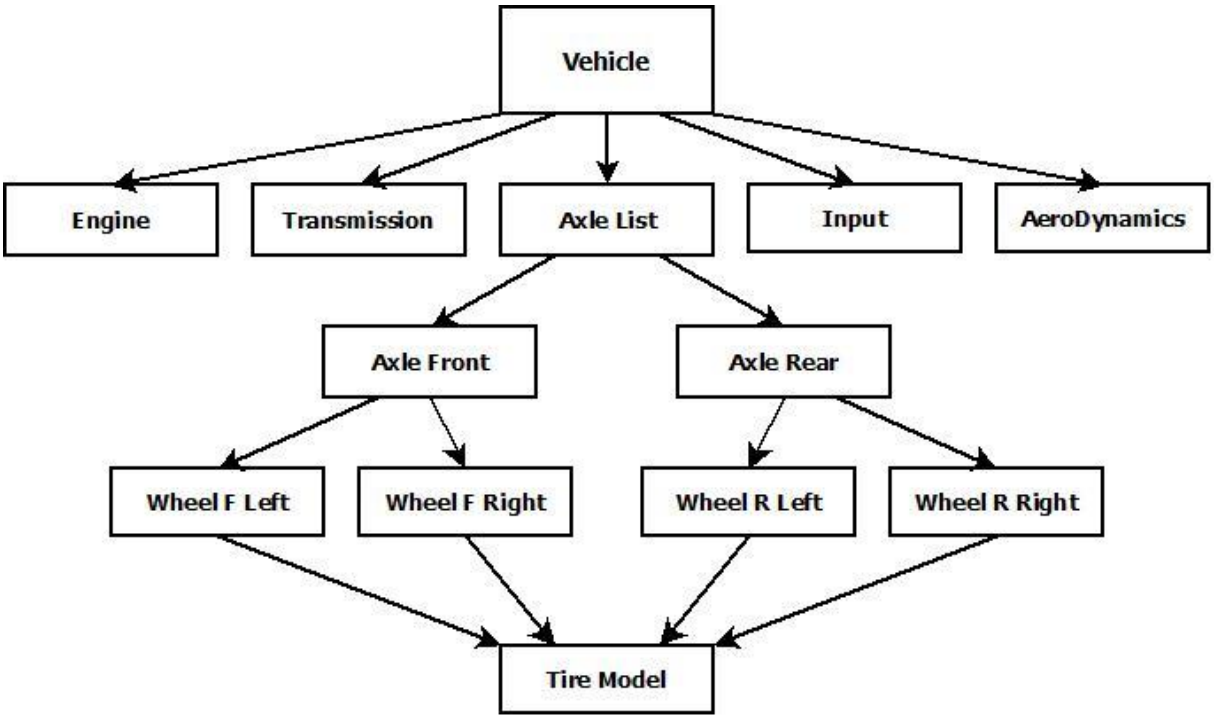
FGear Overview

FGear is a custom vehicle physics solution for unity that helps you build arcade/semi-arcade style racing games. The package consists of a core module and some extra features that were used for making the demo scenes. The package is mostly focused on 4 wheel cars but multi axle setups are also possible.

Core features:

- Desktop or mobile.
- Single script component for a driveable vehicle.
- Full source code.
- Adjustable integration steps.
- Simplified pacejka, pacejka96 and MF6.1 tire model options.
- 3D wheel support(multiple raycasts/ sphere/ convex cast).
- Simple engine model with a torque curve.
- Configurable auto/sequential/manual(h-pattern) transmissions.
- Open, locked and lsd differential options.
- Ackerman steering, toe and camber.
- Adjustable torque and brake distribution of axles.
- Simple spring/damper suspensions with preload option.
- AeroDynamics component for drag and downforce effects.
- Standard input manager with keyboard,joystick and wheel+shifter support.
- Driving aids like abs, asr, esp and anti roll bars.
- Detailed telemetry ui.
- Basic save/load system for fast prototyping.
- Various sample setups and demos.

The below diagram shows how the core module is organized:



- **Vehicle** : This is the host to all components of a vehicle and the only script that you need to add to your gameobject for a driveable vehicle. Apart from managing components, this also contains implementations of most of the driving assists.
- **Engine** : Uses a torque curve and rpm values to generate power to the vehicle.
- **Transmission** : This is a transmission that can be manually controlled or uses configurable values to automatically change gears for you. H-pattern or sequential shifting is possible and there is also a simple clutch implementation which is automatically managed in sequential mode.
- **AeroDynamics** : This component has two jobs. First a drag force is generated as the vehicle goes faster and secondly a down force is applied. These are calculated according to the parameters and a constant air density value.
- **Input** : The first duty of this component is to read unity input axis values and apply them to the vehicle like throttle or brakes. Keyboard, joystick or wheel+shifter inputs can be used. Optionally this modifies the steering inputs according to some parameters to provide a better control.
- **Axles** : The main function of an axle is the power transfer. Engine torque is tranferred to the axle and the axle transfers torque to the wheels via a differential. A vehicle has 2 or more axles and an axle has 2 wheels attached.
- **Wheels** : The wheel component both generates the tire forces and the suspension force. One or more raycasts are used to determine the suspension compression and the required force is calculated to keep the vehicle away from ground. To generate lateral and longitudinal forces, a tire model is used. The generated forces are applied to the vehicle body and also the reaction torques are fed back to the engine.

- **Tire** : Wheels use tire model(s) to calculate traction forces. The default tire model is pacejka96. Simplified pacejka is also decent and computationaly cheaper. MF6.1 tire model is the most advanced one. Some features like camber effect is not available with the simple model. Which one to use depends on your requirements. You can read more about tire models in CoreModule documentation.
- **Extra Features** : There are some additional content that are used for the sample scenes. You can use them for prototyping and learning purposes but they are not recommended for production use. Here are the extras:
- **AIController** : This is the ai script that drives the cars in the racing sample. This script needs a SplineTool instance in the scene which is a racing line representation.
- **ArcadeAssists** : This script contains a couple of fake but effective driving assists. These can be useful for arcade style games.
- **Car Manager** : This is a utility that enables the user to easily switch between vehicles in the scene. Additionally this uses the SplineTool script to calculate the positioning of the vehicles and shows the standings on the screen.
- **Effects** : First of all this script adds engine and skidding sound effects along with a TireSkid mesh effect and smoke particles. Additionally this contains a sample system that detects each wheels hit-surface physic material or terrain texture and alters friction values according to provided parameters.
- **ForceFeedback** : This is a sample script that uses LogitechSteeringWheelSDK to add some simple forcefeedback effects. You need to install the sdk to use this script.
- **Gauge UI** : A set of assets and a script that displays a cool gauge hud showing speed, rpm and current gear.
- **JoystickVibration** : This is a sample script that uses XInput to add vibration effect to the joysticks. You need to install the XInput library to use this one.
- **MinimapTool** : Uses the SplineTool and a LineRenderer to generate a visual representation of the track. Additionally an orthographic camera is used to display the minimap and the cars on the screen.
- **MobileInput** : This is the script that manages the mobile inputs in the mobile sample.
- **OrbitCamera** : A simple configurable camera that orbits around a given transform. It also displays a simple ui so that you can alter its settings during play mode.
- **SplineTool** : This script constructs a spline out of the children transforms of the attached gameobject. Constructed curves are drawn as gizmos in the editor.
- **Statistics** : Constantly measures performance stats like 0-100, 0-200, 100-0 and also if a SplineTool is provided, it calculates and displays the current and the last lap times.
- **Suspension Constraint** : Adds a configurable joint to each wheel that stops suspension compression at its hard limit.
- **RuntimeSample** : Creates a simple vehicle from code, check runtime.scene to see it in action.
- **NetworkSample** : Uses photon unity network to establish a connection and instantiates a network enabled vehicle. You need to install the PUN2 library to use this one.
- **NetworkSync** : This is attached to a vehicle to make it synchronized accross network. You need to install the PUN2 library to use this one.

For a quick start go to **1-Quickstart** or see https://youtu.be/WzAV3pWq70U

For a detailed explanation of core module go to **2-CoreModule**.

For a detailed explanation of extras go to **3-Extras**.

Get api document* from https://www.dropbox.com/s/nn7m1zshcygui57/api.chm?dl=0

For any kind of support contacts us at lazybitgames@gmail.com

*if api documents content is not visible, right click the .chm file and unblock the file from properties: