Доп. информация

Эти компоненты на самом деле не являются частью физики автомобиля, а представляют собой дополнительные инструменты, использованные при создании демонстрационных сцен. Их можно использовать для создания прототипов или обучения, но они далеки от надежных решений и не рекомендуются для использования в производстве.

AIController : Это простой компонент ai драйвера. На входе системы находится сплайн-кривая, по которой движется автомобиль, и кривая скорости/угла, ограничивающая скорость автомобиля на поворотах. Если контрольные точки сплайна расположены правильно, а кривая скорость/угол настроена, ai может достичь хорошего времени прохождения круга. Посмотрите демо-версию "Гонка", чтобы увидеть ее в действии.

- Сначала находится целевая точка на сплайне. Эта точка удаляется по мере увеличения скорости движения автомобиля.
- После этого запускаются 5 лучей для поиска потенциальных целей столкновения. Если рядом находится противник, то точка цели смещается влево или вправо, чтобы избежать столкновения. Эта точка цели определяет входной сигнал рулевого управления.
- Для входов дросселя/тормоза используется тангенс угла наклона сплайна и кривой "скорость/угол". По оси x кривой "скорость/угол" откладывается угол поворота в градусах, а по оси y - максимальное значение скорости в кмч. Максимально допустимая скорость определяется по кривой "скорость/угол", и входы дросселя/тормоза настраиваются соответствующим образом.
- Наконец, если автомобиль застрял на 5 секунд, он возвращается в положение на шлицевой линии.

CarManager : Этот класс имеет несколько функциональных возможностей:

- Прежде всего, он хранит список автомобилей, которые должны быть предоставлены пользователем. Используя его ui, можно переключаться между этими автомобилями, запускать/останавливать двигатель и активировать/деактивировать компонент ai, если он имеется.
- Если указано значение *CountDown*, то все автомобили остаются замороженными на старте, а на экране отображается текст с обратным отсчетом.
- Если задан сплайн и установлено значение *ShowRankings*, то рассчитывается рейтинг автомобилей, который отображается в правой верхней части экрана. Значение *ShowRankings* должно быть задано в самом начале, иначе программа может работать некорректно.

- Наконец, если пользователь нажимает кнопку R, то автомобиль возвращается в положение на линии.

Эффекты : В этом классе собраны все визуальные и звуковые эффекты. Звуки двигателя, эффекты частиц, заносы, тормозные огни, дульные эффекты... все это обрабатывается здесь.

- Сначала в сцене создается игровой объект под названием "Effects". Он находится в папке resources и содержит эффекты частиц и заноса. Для эффекта дула геймобъект должен быть предоставлен пользователем.

- Для звуковых эффектов пользователь должен предоставить AudioClips. Если клипы предоставлены, то AudioSources создаются автоматически.
- Для создания простого эффекта стоп-сигнала пользователь должен указать рендерер(ы), индекс материала и цвет. Если целевой материал найден, то при торможении автомобиля устанавливается его цвет излучения.
- Заносы, частицы дыма и звуковые эффекты заноса вызываются при использовании значений скольжения колес.
- Звук двигателя зависит от заданных параметров и оборотов двигателя.
- Дульный эффект и звук вызываются базовым расчетом, основанным на ускорении оборотов.
- При столкновении воспроизводится звуковой эффект аварии.
- Вы можете определять типы поверхностей на основе физического материала и изменять поведение автомобиля и эффекты в зависимости от текущей поверхности столкновения. В функции *physicsSurfaceUpdate для* каждого коллайдера, с которым столкнулось колесо, проверяется его физический материал. Затем в соответствии с материалом изменяются значения трения колес. Также на величину сопротивления жесткого тела влияет значение Roughness (заданное пользователем), которое призвано ограничить скорость, если автомобиль выезжает за пределы дороги.
- Если *TerrainBasedSurface* имеет значение true, то система, основанная на физическом материале, отключается, вместо нее используется текущая текстура пораженного рельефа (функция *TerrainSurfaceUpdate*). Вы должны определить типы поверхностей, но задавать физические материалы для поверхностей не нужно. Индекс текстуры местности соответствует индексу поверхности. Демонстрацию можно посмотреть в файле *varing_grip.scene*.

ForceFeedback : *Для использования этого класса необходимо получить последнюю версию игрового пакета logitech gaming sdk, включить в проект c# класс-обертку LogitechGSDK и поместить в него LogitechSteeringWheelEnginesWrapper.dll. Чтобы проверить его в race.scene, сначала установите в настройках ввода режим контроллера "WHEEL", затем активируйте скрипт ForceFeedback (он уже должен быть подключен) на машине игрока.*

Эта программа использует logitech gaming sdk для создания эффектов обратной связи на колесах. В настоящее время тестируется только с logitech G29. При запуске инициализируется logitech steering wheel sdk. Он находит доступные устройства, получает идентификатор первого подключенного колеса и устанавливает его рабочий диапазон. По умолчанию максимальный рабочий диапазон для G29 составляет 900 градусов, но для данного сценария он установлен на 450 градусов. Используется 8 типов эффектов, и некоторые из них не работают вместе с другими, например, нужно отключить эффект пружины, когда активен эффект поверхности. Некоторые эффекты используют для расчетов AnimationCurves. Если вы отметили опцию *UseDefaultCurves* (включена по умолчанию), то вам не нужно определять эти кривые. Если вы хотите изменить кривые, снимите флажок *UseDefaultCurves* и настройте кривые вручную. Здесь также показан дополнительный интерфейс, отображающий мощность каждого эффекта в процентах.

- **Led Effect**: Анимация светодиодных лампочек на руле в зависимости от оборотов двигателя.
- **Эффект пружины** : Использует *кривую AlignCurve* для расчета силы пружины, которая пытается выровнять колесо по направлению движения автомобиля. Масштабируйте этот эффект с помощью коэффициента *AlignForce*(0-100).
- **Constant Effect**: Прикладывает к колесу постоянную противоположную силу,

основанную на *кривой ConstantCurve*. Входным параметром *ConstantCurve* по оси x является скорость автомобиля. Например, когда автомобиль не движется, постоянная сила должна быть высокой, и она ослабевает по мере движения автомобиля. Масштабируйте этот эффект с помощью коэффициента *ConstantForce*(0-100).

- **Эффект демпфера**: эффект демпфера затрудняет вращение колеса. Сила эффекта рассчитывается на основе *DamperCurve* и нагрузки на колесо. Масштабируйте этот эффект с помощью коэффициента *DamperForce*(0- 100).
- **Airborne Effect** : Этот эффект воспроизводится, когда передние колеса не касаются земли. Он не имеет параметров.

- **Эффект поверхности**: Этот эффект создает вибрацию на колесе для создания ощущения неровности дороги. Масштабировать этот эффект можно с помощью коэффициента *RoughnessForce*(0-100). Для активации эффекта вызывается функция getSurfaceRoughness. Этот метод проверяет тип физического материала поверхности и возвращает положительные числа, если обнаружена плохая дорога. Рекомендуется написать собственную версию этого метода, исходя из своих потребностей.
- **Эффект лобового/бокового столкновения**: При столкновении мы используем размер и направление удара для расчета коэффициента силы в функции OnCollisionEnter для лобового и бокового столкновений. Затем эти коэффициенты применяются к sdk колеса в качестве параметра эффекта и за короткое время интерполируются до нуля.

JoystickVibration : Это базовый пример, который вибрирует джойстик при наезде автомобиля на препятствие. Сначала он пытается найти первое подключенное устройство и сохраняет его id. При столкновении активируются двигатели геймпада через api метод XInputs. *Для использования данного класса необходимо получить XInput.NET с сайта https://github.com/speps/XInputDotNet/releases (проверено на версии v2017.04-2).*

GaugeUI: отображает обороты, скорость, передачу, рулевое управление, состояние педалей, тягу и состояние подвески. Объект *UICanvas* из папки Resources добавляется в сцену, и компоненты пользовательского интерфейса обновляются в соответствии со значениями, полученными от автомобиля.

MinimapTool : Этот инструмент помогает создать на экране карту. Прикрепите его к игровому объекту, и автоматически будет создана камера и рендерер линий. Для отображения транспортных средств на карте необходимо также предоставить список транспортных средств. Посмотрите демонстрацию "Гонка", чтобы увидеть это в действии.

- Сначала для каждого автомобиля создается объект *Resources/minimapQuad* и прикрепляется к нему.
- Если задан сплайн, то из него генерируется линия. Она будет представлять собой трассу, видимую с камеры карты.
- Для того чтобы это работало, необходимо добавить новый слой minimap. В этом слое должен находиться родительский геймобъект рендера линий и minimapQuads. После этого удалите слой minimap из маски выбраковки основных камер и установите маску выбраковки камер minimap только на слой minimap. Наконец, расположите камеру minimap над трассой, установите тип ее проекции на ортографический. На этом этапе можно играть в игру и смотреть, как она заполняет экран, менять прямоугольник окна просмотра, размер проекции и т.д. в соответствии с вашими потребностями, и все готово. *Предупреждение: в демонстрационных сценах используется слой "minimap", и если у вас нет этого слоя, то, скорее всего, будет использоваться 8-й слой, который по умолчанию является слоем постпроцесса, это может быть не проблемой, но рекомендуется добавить этот слой, если вы хотите использовать эту функцию.*

MobileInput : Показывает основные элементы управления ui для управления автомобилем. Сначала на сцену добавляется объект MobileUICanvas и устанавливаются обратные вызовы событий компонентов ui. Входы автомобиля устанавливаются в соответствии с состояниями кнопок ui. Посмотрите демонстрацию "mobile_race", чтобы увидеть ее в действии.

OrbitCamera : Простой скрипт камеры, которая вращается вокруг заданного целевого преобразования. Камеру можно вращать, удерживая правую кнопку мыши, и увеличивать/уменьшать масштаб с помощью прокрутки мыши. Если параметр spring не равен нулю, то вращение по рысканию интерполируется по рысканию цели. Если включен режим DriftMode, то вращение по рысканию интерполируется по вектору движения цели.

RewindReplay : Этот сценарий представляет собой пример реализации перемотки назад (см. rewind_replay.scene). Вы определяете временной диапазон для хранения истории транспортных средств, и при вызове функции *перемотки* все транспортные средства, которые были отобраны, будут анимированы назад во времени. Вызов функции *replay* приведет к воспроизведению состояния от прошлого к настоящему, а вызов функции *play* - к отмене состояния.

SplineTool : Формирует список точек из дочерних игровых объектов и создает из них объект кривой. В качестве реализации кривой используется CubicBezierCurve от Tristan Grimmer. Этот сплайн используется в различных местах, таких как ai, генерация minimap и расчет рейтинга. Построенную кривую можно увидеть в сцене, если вы добавили хотя бы 2 дочерних игровых объекта. Посмотрите демонстрацию "Гонка", чтобы увидеть ее в действии.

Статистика: Рассчитывает и отображает некоторые статистические показатели. Рассчитываются и отображаются показатели 0-100, 0-200, 100-0 и дрифт. Если задан сплайн, то также рассчитывается и отображается время последнего круга.

RuntimeSample : Прикрепите это к пустому игровому объекту, и он создаст простое транспортное средство из кода, проверьте runtime.scene, чтобы увидеть его в действии. Здесь также содержится несколько примеров модификации транспортных средств во время выполнения. Предупреждение: Как и в обучающей сцене, вы можете получить ошибку "Input Axis is not setup", если вы изменили значения по умолчанию в списке осей ввода unity.

Suspension Constraint : Этот скрипт может использоваться для предотвращения столкновений шины с землей или шины с автомобилем. При запуске добавляет настраиваемый шарнир к каждому колесу и затем обновляет их параметры в реальном времени, чтобы кузов автомобиля оставался в приемлемой трансформации. Предупреждение: этот компонент устанавливает минимальное значение "solverVelocityIterations" жесткого тела автомобиля равным 2, чтобы лучше вести себя при резком нарушении ограничений. Посмотрите на *constraint.scene,* чтобы увидеть его в действии.

ArcadeAssists : Помощники вождения в основном модуле используют реалистичный подход, но их эффективность иногда ограничена. Этот скрипт содержит несколько эффективных помощников, которые могут быть использованы для получения более управляемых автомобилей. Демонстрацию можно посмотреть в сцене "easydrive".

- **Восстановление недостаточной поворачиваемости: прикладывает** внешний крутящий момент к кузову автомобиля для предотвращения недостаточной поворачиваемости. Это дешевый способ борьбы с недостаточной поворачиваемостью, но в сочетании с функцией восстановления избыточной поворачиваемости можно легко добиться аркадного стиля вождения.
- **Восстановление избыточной** поворачиваемости**:** прикладывает внешний крутящий момент к кузову автомобиля для предотвращения избыточной поворачиваемости.
- **Антипробуксовочная система (передняя/задняя) :** Это традиционная система антипробуксовочной системы, которая прикладывает внешние усилия для предотвращения кренов кузова при прохождении быстрых поворотов. Передняя антипробуксовочная система помогает уменьшить избыточную поворачиваемость, а задняя - недостаточную. Это работает только для полноприводных автомобилей.
- **Drift Assist Front** : При боковом движении автомобиля это значение влияет на трение передних колес. Положительные значения приводят к улучшению сцепления с дорогой при увеличении угла избыточной поворачиваемости, а отрицательные - к уменьшению сцепления с дорогой при увеличении угла избыточной

поворачиваемости. Если задать отрицательные значения для передних и задних колес, то автомобиль будет легко уходить в сторону, но при сильном превышении угла поворота может потребоваться положительное значение для задних колес. Настройте его так, чтобы получить желаемое поведение. Это работает только для четырехколесных автомобилей.

- **Drift Assist Rear** : Аналогичное соотношение эффектов для задних колес. Это работает только для 4-колесных автомобилей.

- **Traction Assist**: Модель шин имеет недостаток, заключающийся в том, что на низких скоростях шины ведут себя не так, как ожидалось, и могут терять сцепление с дорогой. Эта функция помогает увеличить сцепление до %200 на низких скоростях. Следует учитывать, что при этом изменяется параметр поперечного/продольного трения колес.
- **Traction Assist Max Speed** : Усиление тяги активируется ниже этой скорости в кмч. Усиление тяги масштабируется по мере снижения скорости автомобиля и достигает максимального значения при его остановке.
- **Torque Splitter Ratio**: Если этот параметр активирован, то распределение крутящего момента между передней и задней осями происходит динамически в зависимости от скорости их вращения (по-прежнему основываясь на начальной доле крутящего момента). Более быстро вращающаяся ось будет получать меньший крутящий момент, а более медленная - больший, даже если начальная доля крутящего момента равна нулю. Это соотношение - процент передачи доли крутящего момента. Если это %100 и автомобиль является заднеприводным, то в крайнем случае весь крутящий момент может быть передан на переднюю ось.
- **Время срабатывания делителя крутящего момента**: Нежелательная передача доли крутящего момента может быть нежелательной. Это время передачи в милисекундах. Передача не произойдет мгновенно, а будет интерполирована на это время.
- **Метод setCruiseSpeed**: Это не настройка, а функция, которую можно вызвать для установки мгновенной скорости автомобиля. Это может быть полезно при старте с места.

BikeHelper : Вспомогательный скрипт, который используется для имитации мотоцикла (см. bikes.scene). Обычно в fgear нет возможности создать двухколесный транспорт, но мы можем поместить левое и правое колеса в центр и использовать этот скрипт для предотвращения падения. Это может привести к флуктуациям при низкой частоте обновления физики, поэтому рекомендуется использовать частоту не менее 100 Гц.

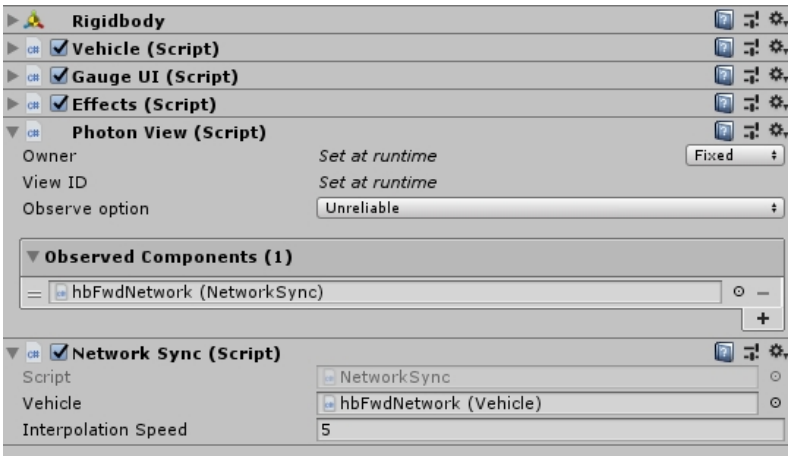
BikeHelper2 : Альтернативный скрипт-помощник для подделки велосипедов. В некоторых случаях он ведет себя лучше.

NetworkSample : Этот скрипт используется только в "multiplayer.scene", он использует сетевую библиотеку photon unity для установления соединения с сервером photon и после подключения инстанцирует транспортное средство с поддержкой сети. Для запуска этого примера необходимо установить PUN2.

- **SendRate** : Определяет, сколько раз в секунду PhotonNetwork должна отправлять пакет. Непосредственно влияет на качество синхронизации и использование полосы пропускания.
- **VehiclePrefabName** : Имя префаба, который инстанцируется при подключении и должен быть доступен в папке Resources. В папке resources находится пример префаба транспортного средства с поддержкой сети под названием "hbFwdNetwork".

NetworkSync : Этот скрипт прикрепляется к игровому объекту транспортного средства, чтобы обеспечить его синхронизацию по сети. Существует множество способов сделать это, данный скрипт реализует базовую клиентскую авторизованную сетевую модель.
Прежде всего, этот скрипт требует наличия компонента PhotonView, и необходимо добавить

этот компонент в список "наблюдаемых компонентов" PhotonView.



Сценарий начинается с отключения обновления входных данных для автомобиля. Ваши собственные входные данные вычисляются, применяются и отправляются другим клиентам. На приемной стороне к транспортным средствам применяются последние полученные входные данные. После настройки параметров ввода к транспортному средству прикрепляется компонент OrbitCamera, а также добавляется компонент TextMesh с 3d-именем. Внимание: данный компонент предназначен для работы только в "multiplayer.scene", для использования в других сценариях могут потребоваться некоторые изменения.
Оптимизация по пропускной способности также отсутствует.

InterpolationSpeed : Это скорость интерполяции по отношению к последней известной информации о преобразовании. Более высокие значения могут привести к появлению джиттера, а более низкие - к увеличению видимой задержки. В настоящее время это постоянное значение, но динамическое значение, зависящее от времени пинга, также может быть полезным.