

Universal Vehicle Controller

(Без зависимостей)

Документация

Содержание:

Содержание:	2
Сторонние ассеты.	4
!!! Инструкция по импорту ассета !!!	5
Введение.	6
Стиль кода.	7
Структура проекта.	9
Настройки проекта	10
VehicleController.cs.	12
CarController.cs.	14
CarController.cs.	14
<i>CarController.cs.</i>	14
<i>Engine.cs.</i>	14
<i>Steering.cs.</i>	17
<i>Transmission.cs.</i>	22
BikeController.cs.	23
<i>BikeConfig</i>	25
TrailerController.cs.	27
Wheel.cs.	28
<i>BikeWheel.cs</i>	28
<i>WheelCollider.</i>	29
Искусственный интеллект (ИИ).	31
<i>AIPath.</i>	31
<i>AITrigger.</i>	33
<i>AI.Spawner.</i>	33
<i>BaseAIControl.</i>	35
<i>BaseAIConfigAsset</i>	35
<i>Насстройки OffsetToTargetPoint и OffsetTurnPrediction для разных типов ИИ.</i>	36
<i>RaceAIControl.</i>	38
<i>RaceAIConfigAsset</i>	38
<i>DriftAIControl</i>	40
<i>DriftAIConfigAsset</i>	40
<i>PursuitAIControl</i>	40
<i>PursuitAIConfigAsset</i>	41
Световые сигналы, стекла.	42
<i>LightObject.</i>	42
Visual effects.	44
<i>VehicleVFX.</i>	44
<i>CarVFX.</i>	45

Sound effects.	46
VehicleSFX.	46
CarSFX.	47
GroundDetection:	49
GroundDetection.	49
GroundConfig.	49
ObjectGroundEntity.	50
TerrainGroundEntity.	50
CarDamage.	52
VehicleDamageController.	52
DamageableObject.	53
DetachableObject.	53
GlassDO.	54
MoveableDO.	54
GameController и PlayerController.	55
UserInput.	56
CameraController.	57
Создание автомобиля (CreateCarWindow)	60
Подготовка модели автомобиля.	60
Подготовка префаба (Объекта) для создания автомобиля.	61
Окно создания автомобиля.	63
Создание автомобиля:	63
Настройки создания автомобиля.	69
Создание байка (CreateBikeWindow)	71
Подготовка модели байка.	71
Подготовка префаба (Объекта) для создания байка.	72
Окно создания байка.	73
Создание байка:	73
LookAtComponent	76
Мобильная версия.	78
Транспортные средства в проекте.	79
Контакты.	80

Сторонние ассеты.



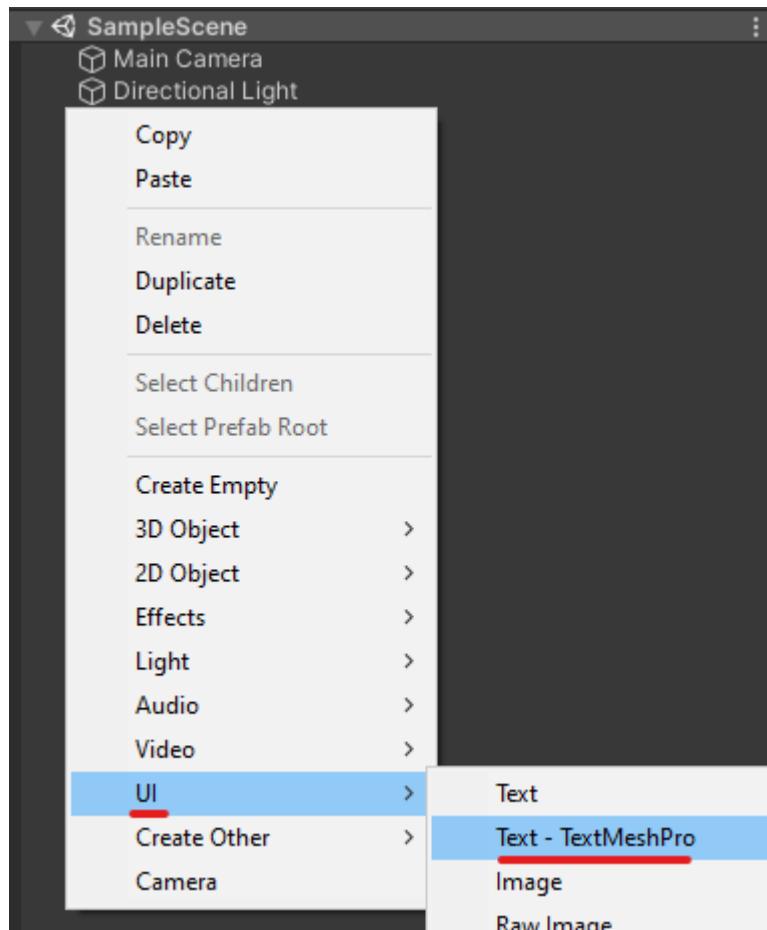
[Easy Roads Pro v3](#) - Для создания дорог и пропсов вдоль дороги, очень удобный и понятный инструмент. Для создания гоночной игры этот ассет незаменим.



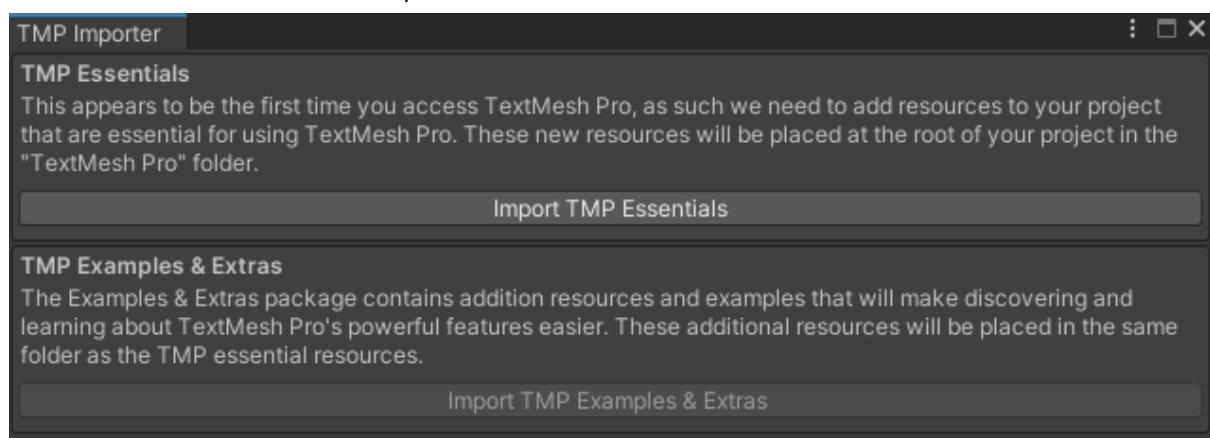
[Gaia 2 - Terrain & Scene Generator](#) - Для создания ландшафта, растительности и окружающей среды, очень мощный инструмент, с его помощью можно создать отличную локацию за несколько кликов.

!!! Инструкция по импорту ассета !!!

Перед импортом ассета нужно сначала импортировать TextMeshPro: Для этого достаточно создать любой объект TextMeshPro на сцене.



После этого возникнет сообщение от TextMeshPro:



Здесь нужно нажать кнопку “Import TMP Essentials”, после этого можете закрыть данное окно (Этот пункт нужен для предотвращения изменения размера шрифтов).

Введение.

Прежде всего хочу поблагодарить вас за приобретение этого ассета.

Весь код я старался писать максимально понятным и простым способом, старался не нагружать его лишними деталями, лишними комментариями затрудняющими читаемость кода, так же старался делать как можно меньше связей между классами, думаю он будет понятным даже для новичков в Unity. А если у вас возникнут какие либо проблемы, вы можете связаться со мной любым удобным для вас способом, все контакты указаны в конце этого документа, обычно я отвечаю на вопросы в течении 24-х часов (Иногда могут быть задержки).

Этот ассет создан с целью упростить создание физической модели поведения автомобилей разных типов, с немного аркадным типом управления. Ассет использует стандартные WheelCollider, по моему мнению они прекрасно справляются со своей задачей.

Стиль кода.

Во всем проекте я старался придерживаться одного стиля написания кода:
Все классы находятся в пространстве имен “PG”, для избежания совпадения имен классов.

Любые объявленные в классе переменные (Публичные, приватные), названия классов, названия методов начинаются с заглавных букв:

```
public Wheel[] Wheels;  
float MaxMotorTorque;  
float FixedUpdateBrakeLogic ()  
public class EngineConfig
```

За исключением переменных использующихся для проверки этих переменных в свойствах, такие переменные начинаются с “_” или “m_” и они приватные:

```
MeshFilter _MeshFilter;  
public MeshFilter MeshFilter  
{  
    get  
    {  
        if (!_MeshFilter)  
        {  
            _MeshFilter = GetComponent<MeshFilter> ();  
        }  
        return _MeshFilter;  
    }  
}
```

Все фигурные скобки находятся в отдельной строке (Как на предыдущем изображении), это намного увеличивает читаемость кода. Исключение составляют некоторые свойства которые отлично читаются в одной строке и частей кода где разделение по строкам выглядит неуместно:

```
public int CurrentGearIndex { get { return CurrentGear + 1; } }
```

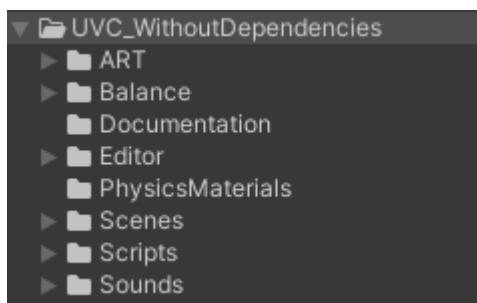
С маленькой буквы начинаются переменные используемые в качестве параметров или объявленные в теле метода.

```
public void SetDamageForce (float force)
```

```
...
bool forwardIsSlip = false;
bool anyWheelIsGrounded = false;
...
```

Все возможные неочевидные места в коде прокомментированы, я старался называть переменные, методы, свойства и т.д. максимально понятно и логично. Исключения могут составить некоторые формулы, описание которых будет такое же как сама формула.

Структура проекта.



Проект разделен на следующие директории:

- ART: В этой папке хранятся все возможные артовые файлы (Текстуры, спрайты, FBX файлы, материалы и т.п)
- Balance: В этой папке хранятся файлы баланса, сейчас таких немного (Только ассеты настроек).
- PhysicsMaterials: Все физические материалы.
- Prefabs - В этой папке хранятся различные префабы (UI, префабы различных GameObjects использующиеся в разных сценах)
- RendererSettings: Все настройки связанные с рендером.
- Scenes: Сцены.
- Scripts: Скрипты.
- Sounds: Звуки используемые ассетом.

Настройки проекта

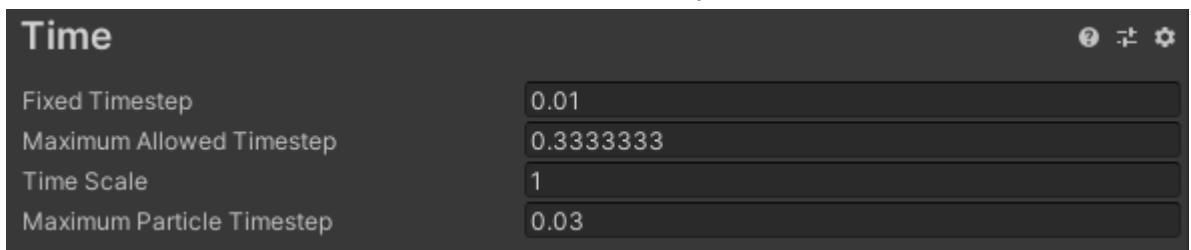
- Physics, в настройках физики изменена только LayerCollision Matrix, это облегчает расчет физики и дает возможность выбрать какие слои могут сталкиваться друг с другом.

	Default	TransparentFX	Ignore Raycast	Water	UI	Vehicle	Wheel	Road	Ground	Stone	Metal	ColliderOnlyForWheel	DamageObject	ConnectTrailer	Plastic	IgnoreVehicleCollision
Default	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓					
TransparentFX																
Ignore Raycast																
Water																
UI																
Vehicle	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓					
Wheel			✓			✓	✓									
Road	✓	✓		✓	✓	✓	✓	✓								
Ground	✓	✓			✓	✓	✓	✓								
Stone	✓	✓				✓	✓									
Metal	✓	✓					✓									
ColliderOnlyForWheel							✓									
DamageObject		✓														
ConnectTrailer			✓													
Plastic	✓	✓														
IgnoreVehicleCollision	✓															

- Player, основные настройки проекта, в этом пункте настройки индивидуальны для каждого проекта, вы можете оставить все как есть изменив имя продукта и название компании.
- Quality, все варианты качества как по умолчанию, изменены только “LOD bias”.
- ScriptExecutionOrder, добавлены некоторые компоненты для контроля порядка инициализации

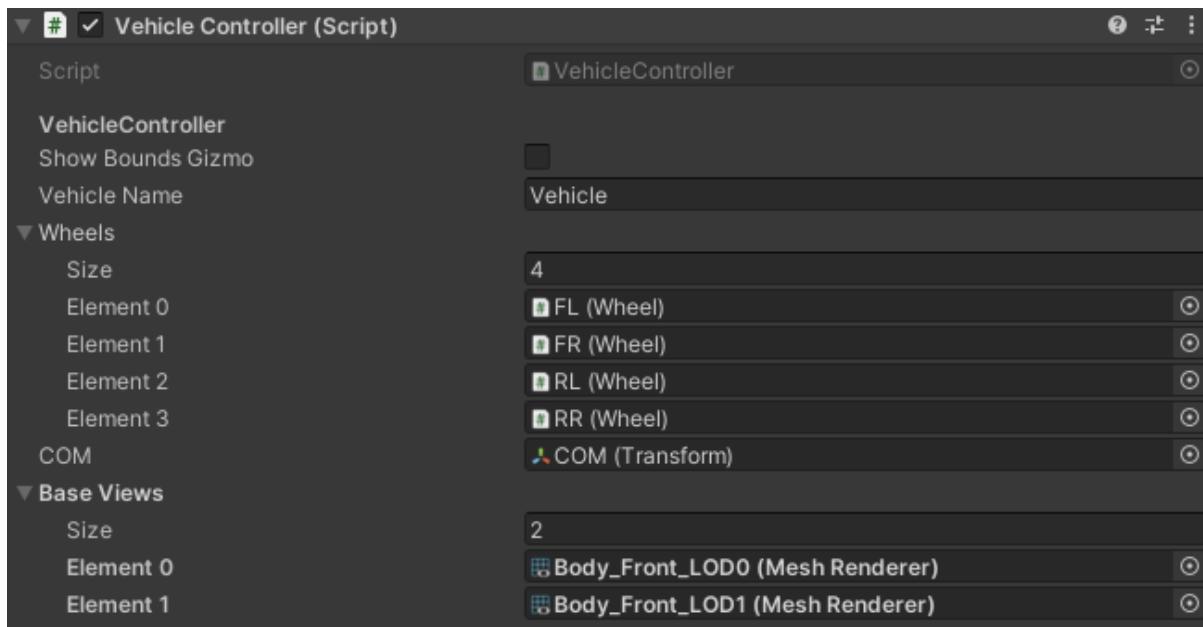
= PG.GameController	-25	-
= UnityEngine.Experimental.Rendering.Universal.CinemachineUniversalPixelPerfect	-1	-
Default Time		
= UnityEngine.UI.ToggleGroup	10	-
= FMODUnity.StudioListener	80	-
= FMODUnity.RuntimeManager	100	-
= Gaia.ScreenShotter	500	-
= PG.CarController	1000	-
= PG.VehicleController	1000	-
= PG.DamageableObject	1101	-
= PG.DetachableObject	1102	-
= PG.GlassDO	1103	-
= PG.MoveableDO	1104	-
= PG.Wheel	1200	-

- Time, изменен параметр “FixedTimestep” - чем ниже параметр, тем чаще будет просчитываться физика игры (Столкновения будут рассчитываться качественнее), но расчет физики станет более ресурсозатратным и наоборот.



VehicleController.cs.

Базовый класс для всех типов транспорта, содержит следующие поля:



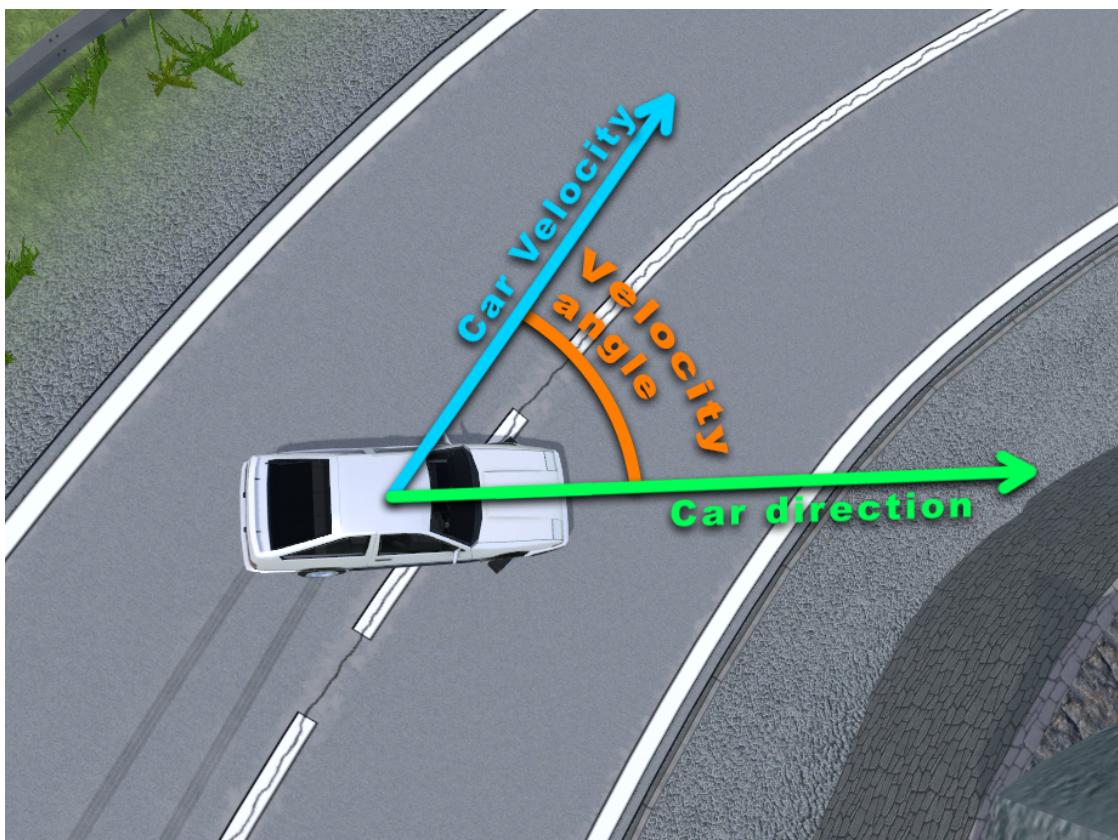
- VehicleName - Имя машины, поле нужно для поиска оригинального префаба машины при восстановлении машины.
- Wheels - Ссылки на колеса машины типа Wheel (О этом классе будет описано дальше).
- COM - Центр масс (Центр тяжести) транспортного средства, от положения этого объекта сильно зависит поведение транспортного средства (Чем ниже центр тяжести тем устойчивее транспортное средство и наоборот), лучше всего его устанавливать в ту же точку как и у реальных объектов.
- Base Views - Массив типа Renderer, нужен для определения видимости транспортного средства, свойство VehicleController.VehicleIsVisible (Это свойство нужно для проверки на воспроизведение звуков и визуализацию эффектов). Используется массив так как основные части могут состоять в LODgroup (Как в данном случае), в этот массив стоит добавить объекты тела автомобиля.

В этом классе есть логика сброса транспортного средства (сейчас сбрасывается в ту же точку в которой находятся колесами вниз, вы можете реализовать любую другую логику изменив метод "ResetVehicle". Есть логика столкновения транспортного средства, если нужно отслеживать столкновение в любом классе, достаточно подписаться на событие Action<VehicleController, Collision> CollisionAction;.

Рассчитываются следующие свойства:

- Свойство VehicleIsVisible - возвращает true если любой из камер отрисовывается любой Renderer из списка BaseViews.
- Свойство VehicleIsGrounded - возвращает true если хотя бы одно колесо касается земли.
- Свойство CurrentSpeed измеряется в юнитах(Метр) в секунду.
- Свойство SpeedInHour измеряется в км/ч или миль/ч в зависимости от выбранной системы замера скорости в ассете GameSettings.asset.

- VehicleDirection- Направление движения автомобиля.
- VelocityAngle - Угол вектора движения автомобиля относительно направления тела автомобиля.



CarController.cs.

CarController наследник класса VehicleController, в нем находится вся логика и физика поведения авто. Для удобства редактирования этот класс разделен на 4 partial класса:

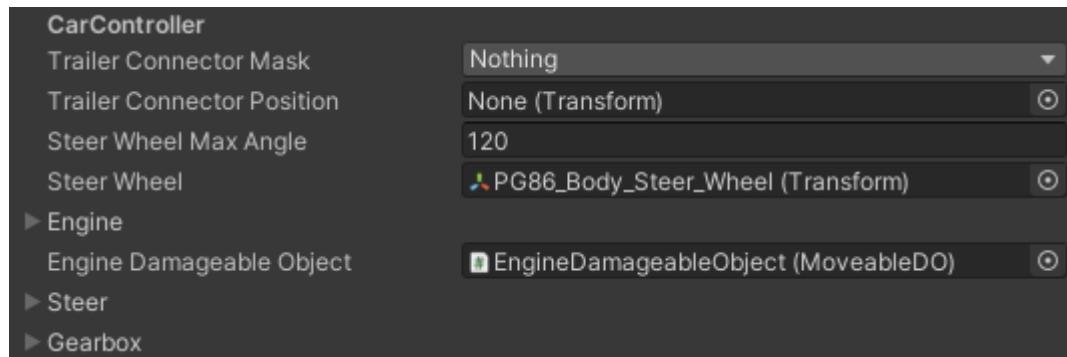
- CarController.cs - В этой части компонента находятся общие поля, свойства и методы, такие как информация о прицепе, руле и т.д.
- Engine.cs - Эта часть компонента отвечает за работу двигателя, рассчитываются текущие обороты двигателя, загруженность двигателя, отсечка, турбо, буст и т.д.
- Steering.cs - Эта часть компонента отвечает за управление автомобилем, поворотом рулевых колес, так же здесь происходит помощь в управлении за счет чего управление ощущается немного аркадным.
- Transmission.cs - В этой части компонента рассчитывается мощность передаваемая от двигателя на колеса, также здесь находится логика переключения передач (Автоматическая и ручная).

Далее будет подробное описание каждой части.

CarController.cs.

CarController.cs.

Этот компонент содержит следующие поля:



- TrailerConnectroMask - Маска соединения с прицепом.
- TrailerConnectorPosition - Позиция присоединения прицепа к машине.
- SteerWheelMaxAngle - Максимальный градус поворота рулевого колеса (Только визуально).
- SteerWheel - ссылка на рулевое колесо.
- EngineDamageablePbject - Повреждения двигателя зависят от расположения этого объекта. Если этого объекта нет, двигатель повреждаться не будет.

Engine.cs.

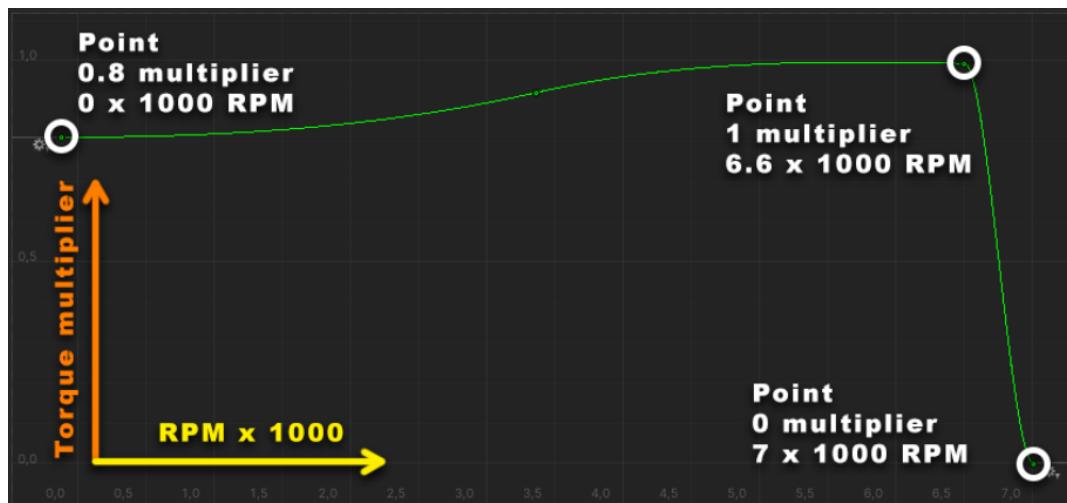
Этот компонент содержит только поле Engine типа EngineConfig, конфиг был создан для возможности подмены этого поля (Например при тюнинге автомобиля, смена двигателей).

Имеет следующие поля:

▼ Engine

Power	
Max Motor Torque	300
Motor Torque From Rpm Curve	
Max RPM	7000
Min RPM	700
RPM Engine To RPM Wheels Fast	8
RPM Engine To RPM Wheels Slow	3
Cut off	
Cut Off RPM	6600
Target Cut Off RPM	6000
Cut Off Time	0.05
Turbo	
Enable Turbo	<input checked="" type="checkbox"/>
Turbo Increase Speed	3
Turbo Decrease Speed	10
Turbo Additional Torque	0.3
Boost	
Enable Boost	<input checked="" type="checkbox"/>
Boost Amount	5
Boost Additional Power	1
Back fire	
Probability Backfire	0.3
Automatic change gear	
RPM To Next Gear	6500
RPM To Prev Gear Diff	500

- MaxMotorTorque - Мощность двигателя передаваемая трансмиссии (Крутящий момент), изменяя этот параметр можно легко изменять мощность автомобиля.
- MotorTorqueFromRpmCurve - Кривая мощности двигателя, рекомендуемые размеры кривой $X = \text{MaxRPM} / 1000$, $Y = 1$.



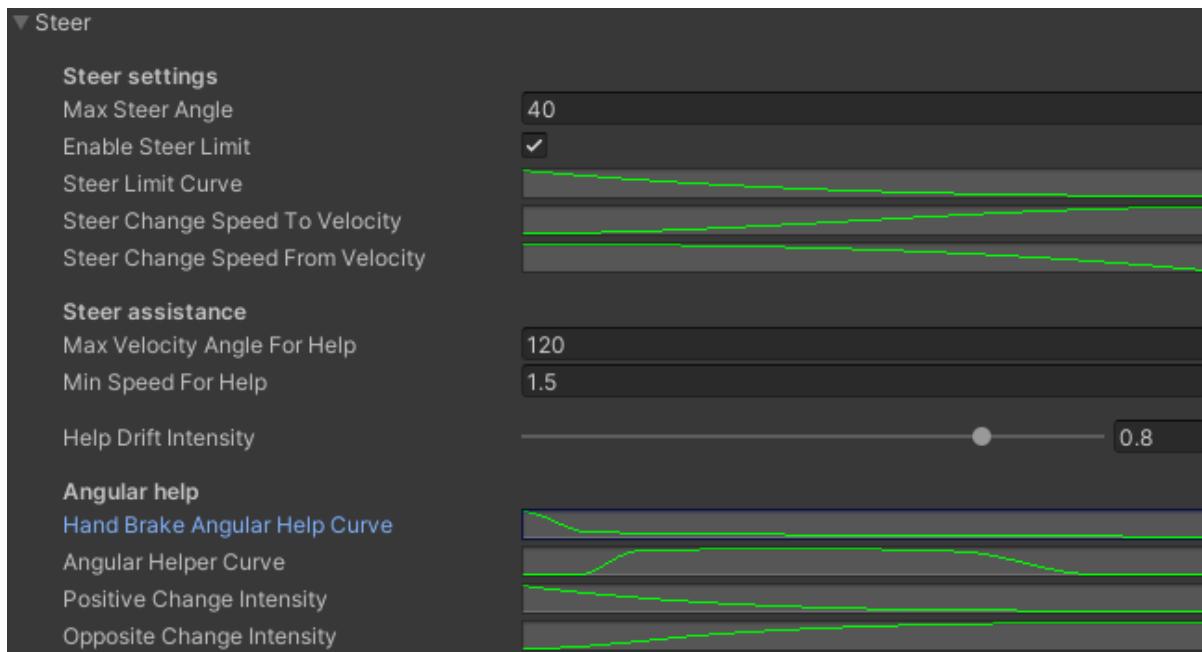
- MaxRPM - Максимальные обороты двигателя, нужен для определения максимального угла на тахометре.
- MinRPM - минимальные обороты двигателя, холостые обороты.
- RPMEngineToRPMWheelsFast - Скорость изменения оборотов двигателя, если скорость увеличивается (Машина разгоняется или происходит понижение передачи).

- RPMEngineToRPMWheelsSlow - Скорость изменения оборотов двигателя, если скорость уменьшается (Машина тормозит или происходит повышение передачи).
- CutOffRPM - Обороты на которых происходит отсечка, двигатель не может раскрутиться выше этих оборотов.
- TargetCutOffRPM - Обороты до которых снижаются обороты двигателя.
- CutOffTime - Время, за которое обороты двигателя доходят до TargetCutOffRPM при отсечке.
- EnableTurbo - Включает/Выключает турбо у автомобиля.
- TurboIncreaseSpeed - Скорость с которой увеличивается значение турбо.
- TurboDecreaseSpeed - Скорость с которой уменьшается значение турбо.
- TurboAdditionalTorque - Дополнительный множитель мощности двигателя при максимальном значении турбо.
- EnableBoost - Включает/Выключает буст (Нитро) у автомобиля.
- BoostAmount - Количество буста, измеряется в секундах.
- BoostAdditionalPower - Дополнительный множитель мощности двигателя при максимальном при использовании буста.
- ProbabilityBackfire - Вероятность возникновения огня из выхлопной трубы, 0 - не будет вообще, 1 - будет возникать всегда.
- MaxBrakeTorque - Сило торможения, передается на все работающие колеса.
- RPMToNextGear - Обороты при которых произойдет повышение передачи, если выбрана автоматическая коробка передач, и выполнены все условия в Transmission.cs.
- RPMToPrevGearDiff - Переключение на пониженную передачу если текущие обороты + RPMToPrevGearDiff меньше переключения предыдущей передачи. Обороты при которых произойдет понижение передачи, если выбрана автоматическая коробка передач, и выполнены все условия в Transmission.cs.

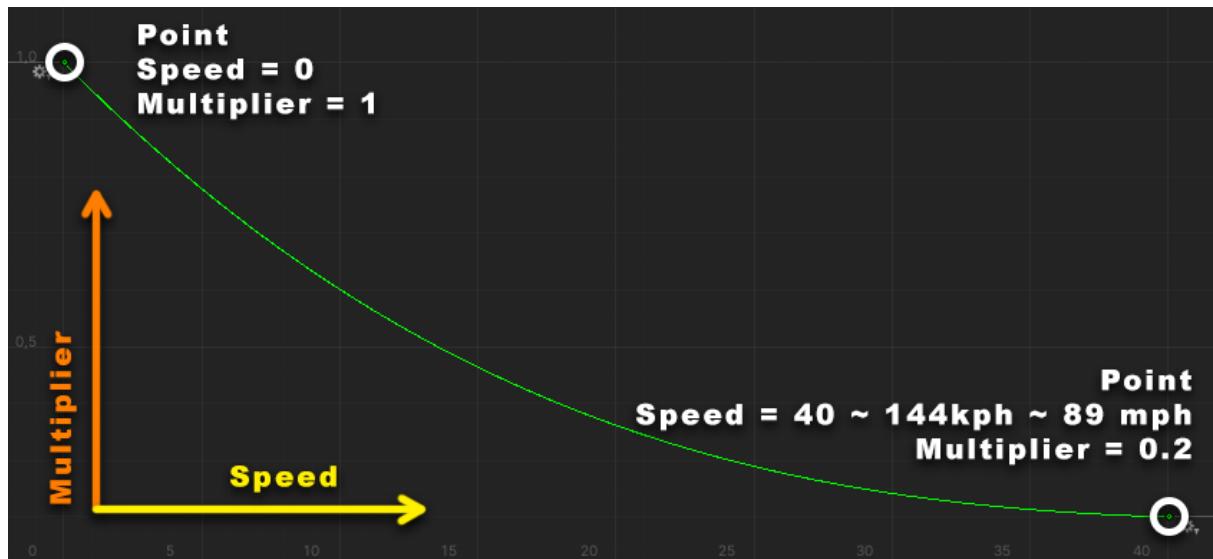
Steering.cs.

Этот компонент содержит только поле Steer типа SteerConfig, конфиг был создан для возможности подмены этого поля (Например при тюнинге автомобиля, смена подвески).

Имеет следующие поля:



- MaxSteerAngle - Максимальный угол поворота рулевых колес.
- EnableSteerLimit - Включает ограничение поворота колес в зависимости от скорости автомобиля.
- SteerLimitCurve - Кривая ограничения поворота колес если включен флаг EnableSteerLimit.



- SteerChangeSpeedToVelocity - Скорость поворота колеса в сторону силы движения машины (В сторону силы движения, руль поворачивается легче и быстрее), чем дальше колесо повернуто от вектора скорости, тем быстрее

колесо будет поворачиваться к вектору скорости.

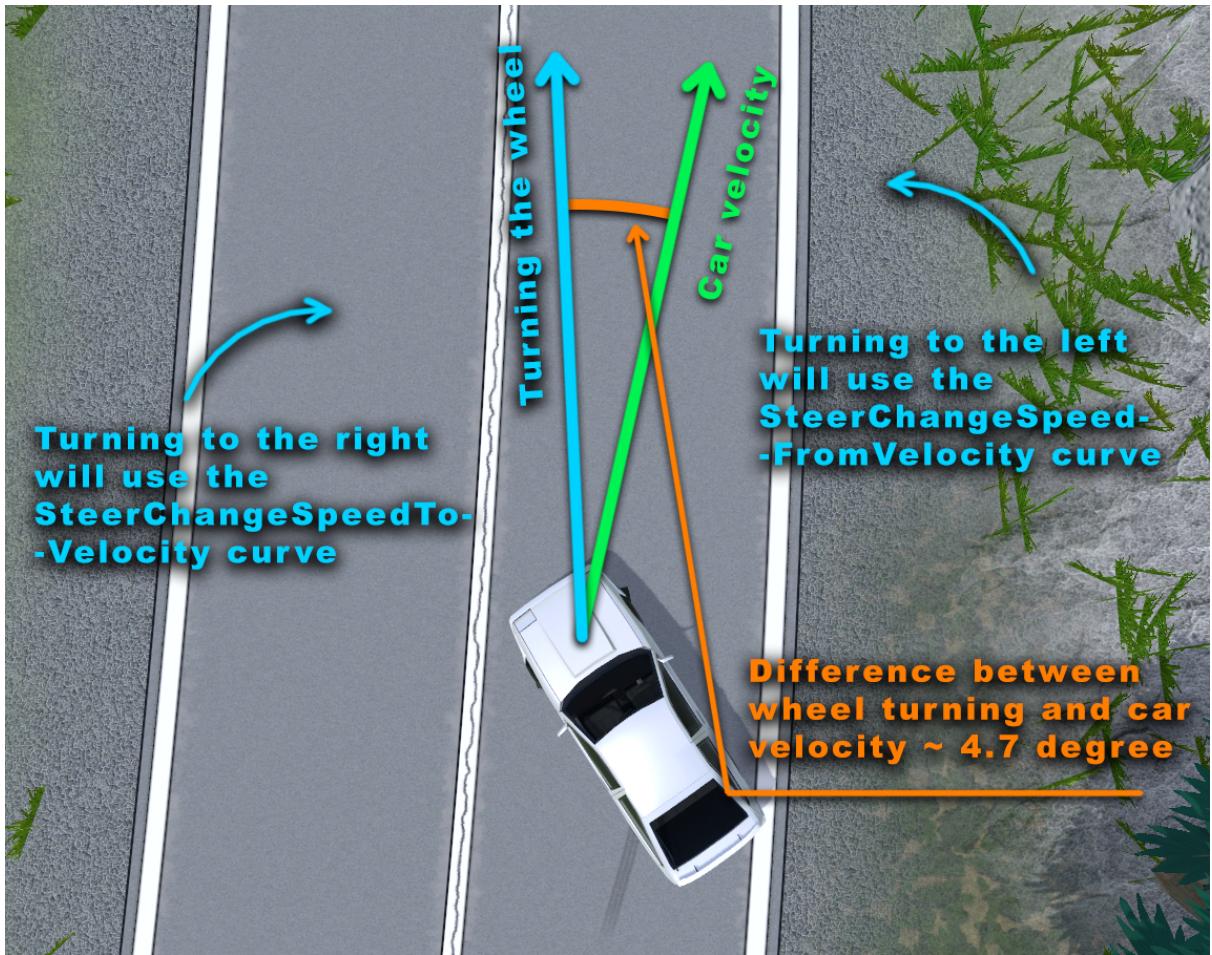
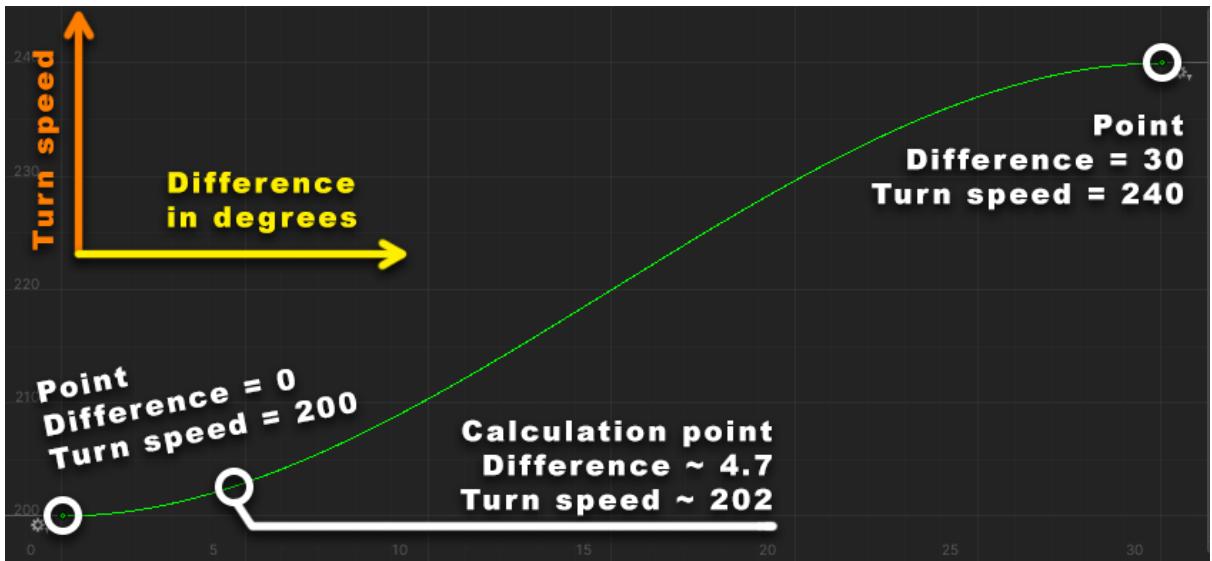


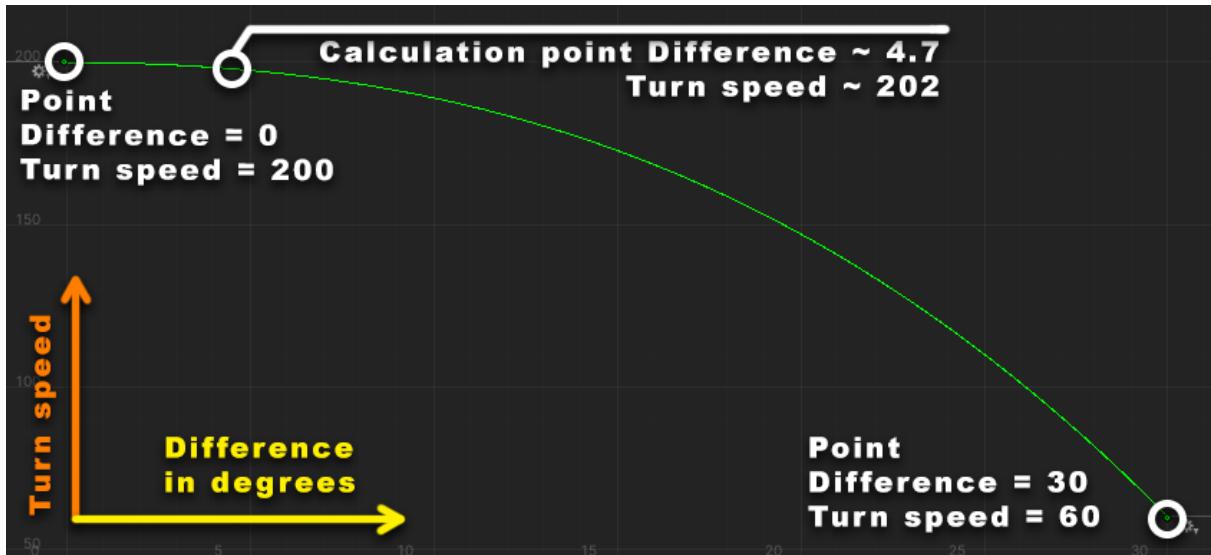
Рисунок используется для SteerChangeSpeedToVelocity и SteerChangeSpeedFromVelocity.

Кривая с данными из рисунка при повороте направо:



- SteerChangeSpeedFromVelocity - Скорость поворота колеса в противоположную сторону силы движения машины (В противоположную сторону силы движения, руль поворачивается тяжелее и медленнее), чем дальше колесо повернуто от вектора скорости, тем медленнее колесо будет поворачиваться от вектора скорости.

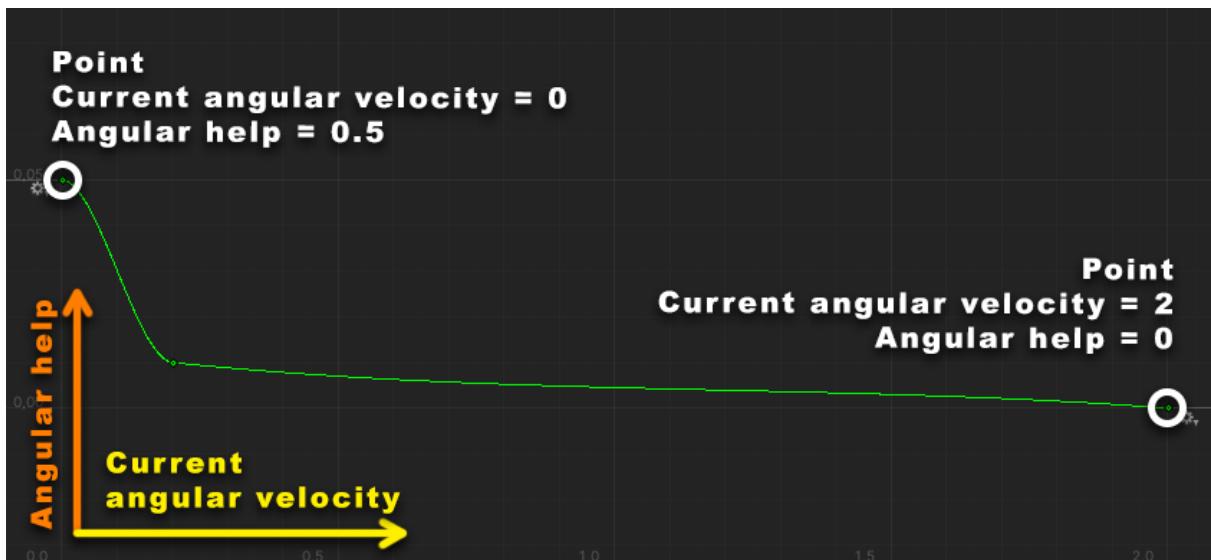
Кривая с данными из рисунка при повороте налево:



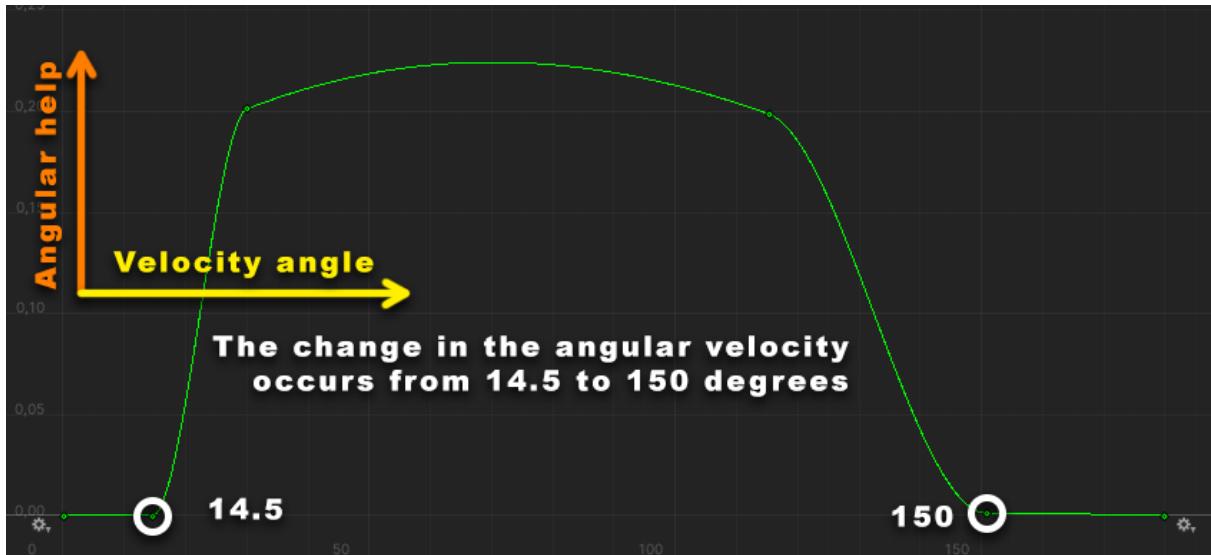
- HelpDriftIntensity - Интенсивность подруливания в сторону дрифта, 0 - подруливания нет, 1 - максимальное подруливание. Переменная влияет только на поворот колес.
- MaxVelocityAngleForHelp - Максимальный угол поворота машины при котором будет работать помощь в управлении (Если угол больше, то машину скорее всего закрутит).
- MinSpeedForHelp - Минимальная скорость для помощи, если скорость ниже то помощь в управлении работать не будет.

Далее переменные и кривые которые влияют на изменение угловой скорости (Сила которая поворачивает автомобиль не смотря на то в какую сторону повернуты колеса), за счет этого возникает ощущение аркадного управления.

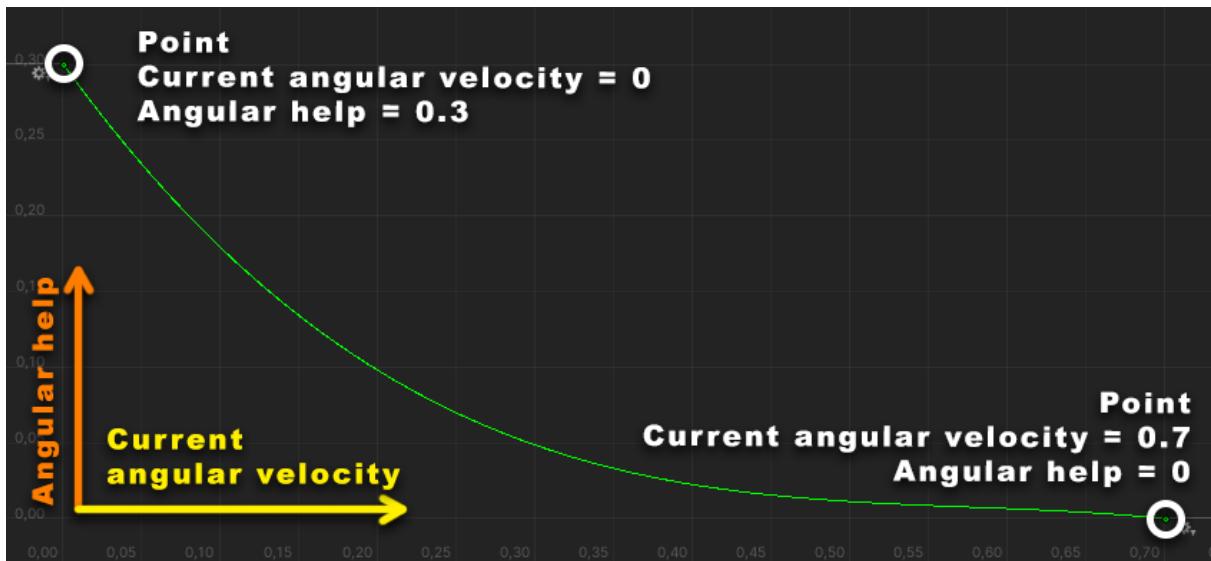
- HandBrakeAngularHelpCurve - Кривая, изменяющая угловую силу (Силу вращения) автомобиля, при использовании ручного тормоза. Нужно для задания начального импульса вращения (для имитации заноса на машинах колеса которых с сильным сцеплением с дорогой) или для помощи при отрицательной силе вращения.



- AngularHelperCurve - Кривая, изменяющая угловую силу (Силу вращения) автомобиля, зависит от текущего угла скорости.



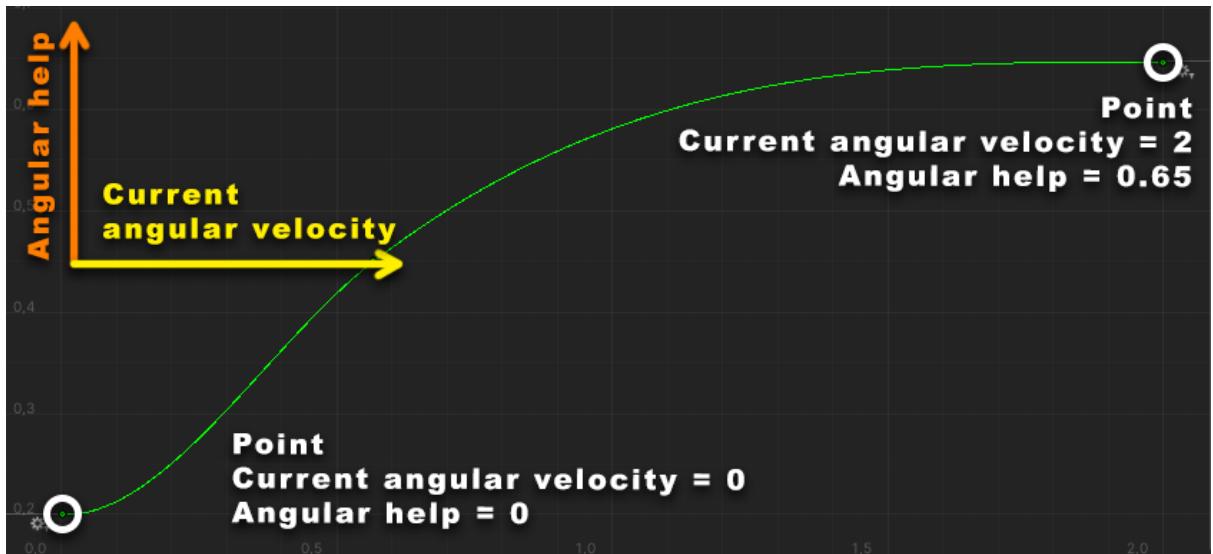
- PositiveChangeIntensity - Множитель изменения угловой скорости, если есть ввод поворота в сторону действующей угловой скорости, зависит от текущей угловой скорости.



Сейчас кривая настроена таким образом: Чем больше текущая угловая скорость в сторону желаемого поворота (Горизонтальный ввод), тем меньше угловая скорость изменяется.

-

- OppositeChangeIntensity - Множитель изменения угловой скорости, если есть ввод поворота от стороны действующей угловой скорости, зависит от текущей угловой скорости.

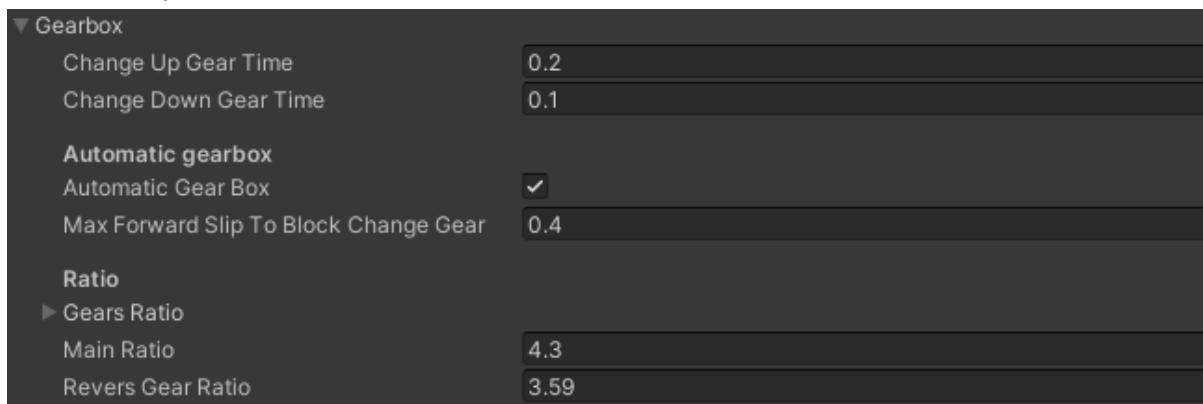


Сейчас кривая настроена таким образом: Чем больше текущая угловая скорость от стороны желаемого поворота (Горизонтальный ввод), тем больше угловая скорость изменяется.

Transmission.cs.

Этот компонент содержит только поле Gearbox типа GearboxConfig, конфиг был создан для возможности подмены этого поля (Например при тюнинге автомобиля, смена коробки передач).

Имеет следующие поля:

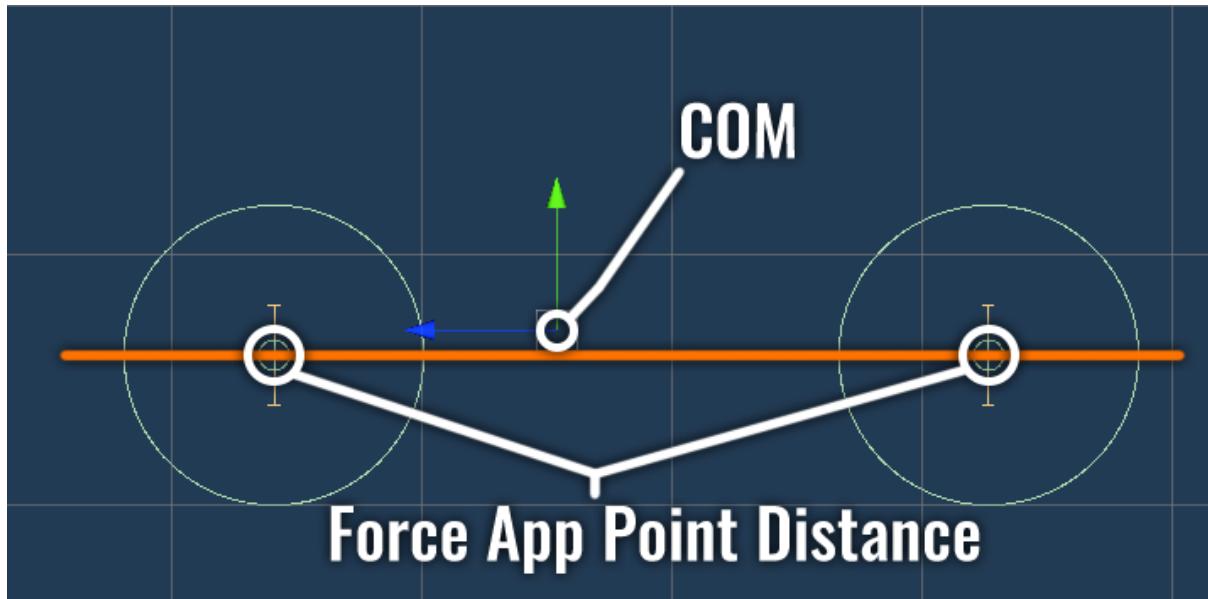


- ChangeUpGearTime - Время повышение передачи.
- ChangeDownGearTime - Время понижения передачи.
- AutomaticGearBox - Автоматическая коробка передач.
- MaxForwardSlipToBlockChangeGear - Максимальное скольжение колес до которого автоматическая коробка передач будет переключаться (Например, если колеса сильно прокручиваются, то переключения не будет).
- GearsRatio - Передаточные числа каждой передачи (От количество элементов массива зависит количество передач), этот массив желательно заполнять реальными техническими данными (Например из Википедии).
- MainRatio - Главное передаточное число.
- ReversGearRatio - Передаточное число задней скорости.

BikeController.cs.

BikeController наследник класса CarController, в классе реализована дополнительная логика удержания равновесия на 2-х колесах, логика аварий, визуальное отображение подвижных частей (Руль, передняя вилка, задняя вилка).

!!!Внимание!!! Позиция COM сильно влияет на поведение байка. Чем ближе позиция COM к линии между ForceAppPointDistance колес, тем стабильнее ведет себя байк при изменении RB.Velocity (Ускорение, торможение, повороты).



Этот компонент содержит следующие поля:

Bike Settings	
► Bike	
Handlebar	Handlebar (Transform)
Front Fork	ForkFront (Transform)
Rear Fork Parent	ForkRear_Parent (Transform)
Rear Fork	ForkRear (Transform)
Front Com Position	FrontCOM (Transform)
Rear Com Position	RearCOM (Transform)

- Bike - Поле типа BikeConfig (Подробное описание дальше, в этом разделе).
- Handlebar - Мотоциклетный руль, поворачивается со всеми дочерними объектами. От наклона этого объекта зависит в какой плоскости будет поворачиваться руль (Только визуальное изменение, на физику байка не влияет), по поводу настройки руля можете посмотреть в разделе CreateBikeWindow.
- FrontFork - Передняя вилка, важно чтобы pivot (Опорная точка) вилки был повернут так же как pivot руля (Параллельно), для правильного отображения и анимации амортизации.
- RearForkParent - Родитель для задней вилки, должен находиться в той же точке что и задняя вилка, но с Vector.zero локальным вращением (Нужно для определения точки из которой нужно следить за задним колесом). Создается автоматически если не был настроен до PlayMode, .
- RearFork - Задняя вилка, следит за задним колесом для имитации амортизации задней подвески байка.

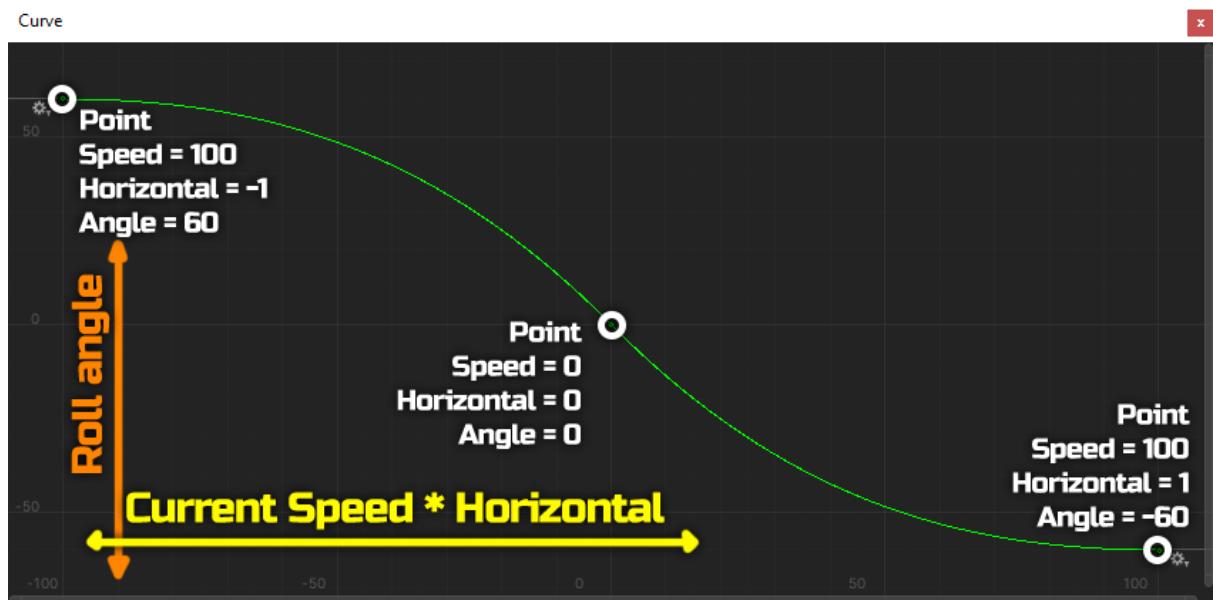
- FrontComPosition - Центр масс байка смещается в эту позицию (Упор на переднее колесо) при положительном тангаже (Если байк находится на земле).
- RearComPosition - Центр масс байка смещается в эту позицию (Упор на заднее колесо) при отрицательном тангаже (Если байк находится на земле).

BikeConfig

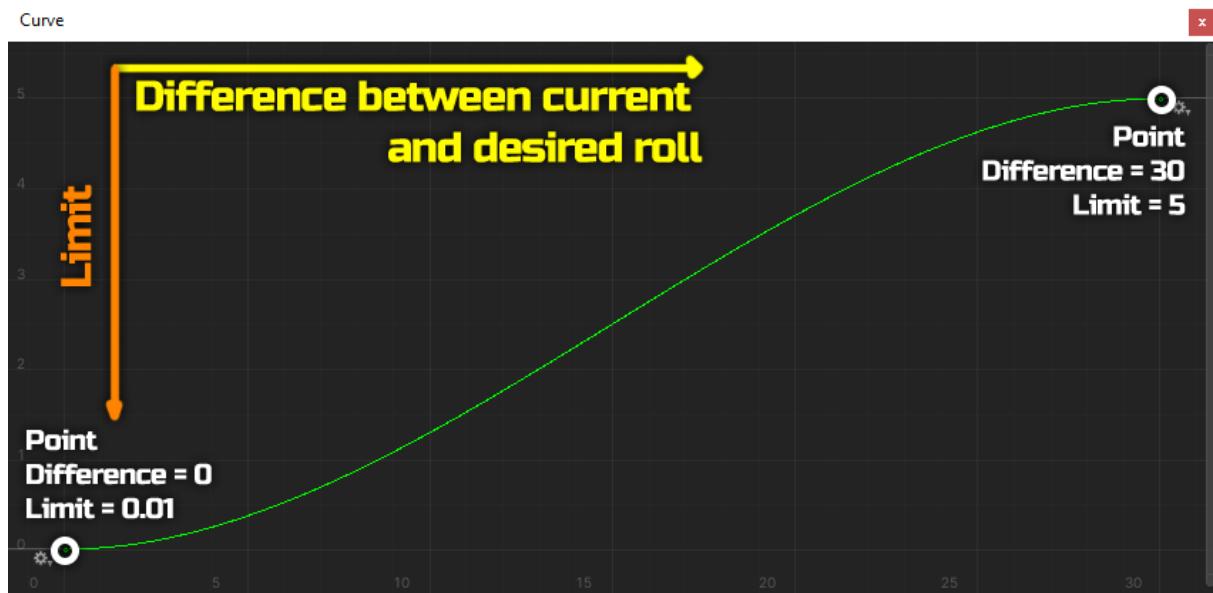
Имеет следующие поля:

▼ Bike	
Speed Roll Angle	
Roll Angular Limit	
Max Height Diff Wheels For Change Com	1
Max Pitch Angular In Air	0.02
Max Yaw Angular In Air	0.02
Max Reverse Speed For Crash	5
Max Sqr G Force For Crash	100
Target Reverse Speed	3

- SpeedRollAngle - Кривая крена байка, зависит от текущей скорости * ControlHorizontal. В примере настроено: чем выше скорость тем сильнее наклон байка.



- RollAngularLimit - Ограничение силы крена. В примере настроено: Чем текущий крен ближе к нужному крену тем сильнее ограничение (Для того чтобы избежать раскачивание байка).



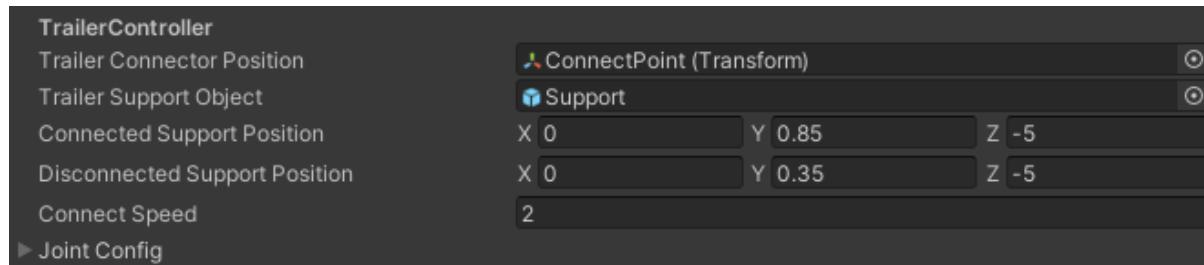
- MaxHeightDiffWheelsForChangeCom - Максимальная разница в высоте между колес для изменения Центра Масс. Чем ближе разница к этому значению, тем СОМ (Центр масс) ближе к стандартной позиции (Для эффекта баланса байка, чтобы байк не переворачивался назад или вперед). Например: Байк едет прямо, разница высоты == 0, при нажатии на NegativePitch СОМ переносится максимально в RearComPosition, байк при ускорении начнет вставать на дыбы, при увеличении разницы высоты колес СОМ постепенно будет смещаться к стандартной позиции.
- MaxPitchAngularInAir - Скорость изменения силы Тангажа в воздухе.
- MaxYawAngularInAir - Скорость изменения силы Рысканья в воздухе.
- MaxReverseSpeedForCrash - Скорость заднего хода при которой произойдет авария байка.
- MaxSqrGForceForCrash - Максимальная перегрузка при которой произойдет авария (Например при сильном ударе или приземлении).
- TargetReverseSpeed - Скорость заднего хода.

В проекте есть настроенный мотоцикл, вы можете использовать его как RefBikeController и изменять настройки по вашему желанию.

TrailerController.cs.

TrailerController наследник класса VehicleController, в нем находится вся логика связанная с использованием прицепа, возможность прицепить прицеп, создание соединения, авто расцепление при разрушении соединения, логика фар, логика торможения.

Имеет следующие поля:



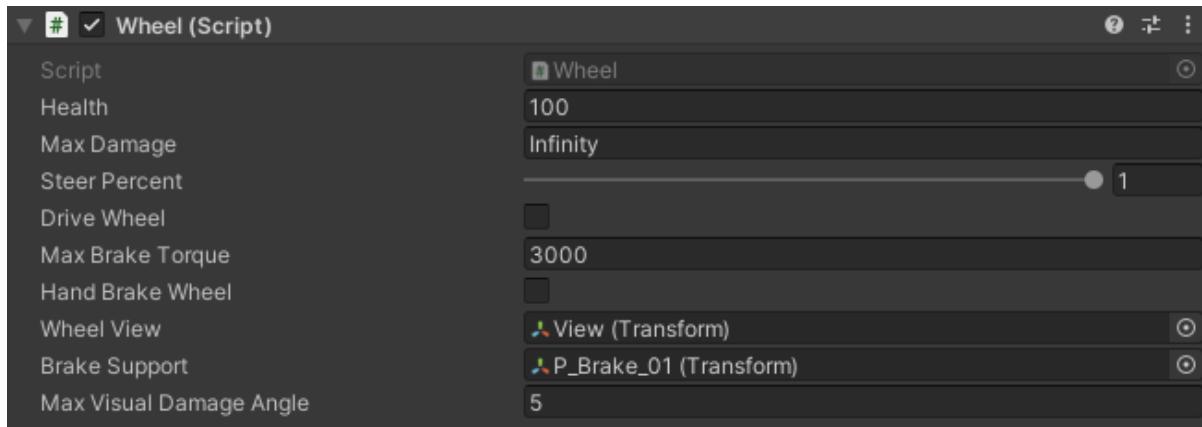
- TrailerConnectorPosition - Позиция присоединения прицепа к машине.
- TrailerSupportObject - Опоры прицепа.
- ConnectedSupportPosition - Позиция опоры при подключенном прицепе.
- DisconnectedSupportPosition - Позиция опоры при отключенном прицепе.
- ConnectSpeed - Скорость изменения позиции опоры.
- JointConfig - Часть параметров для создания ConfigurableJoint при присоединении прицепа.

Wheel.cs.

Этот класс наследник MoveableDO (Для смещения колеса в сторону удара).

Обертка для стандартного WheelCollider, в этом классе расчитываются такие данные как текущее скольжение, температура (Пока нужна только для визуализации дыма от покрышек), информация о контактируемой поверхности.

Класс имеет следующие поля:



- Health - Прочность детали.
- MaxDamage - Максимально наносимый удар за одно столкновение.
- SteerPercent - Процент поворота колеса от MaxSteerAngle, можно выставить значение в диапазоне от -1 до 1: 1 - максимальный поворот, 0 - колесо не рулевое, -1 - отрицательный максимальный поворот колеса.
- DriveWheel - При значении равным true на это колесо будет передаваться крутящий момент от трансмиссии. Весь крутящий момент распределяется поровну на все колеса, например: если одно ведущее колесо то на него будет передаваться 100% мощности, если колеса 2 то на них будет передаваться по 50% мощности и т.д.
- MaxBrakeTorque - Сила торможения колеса.
- HandBrakeWheel - При значении равным true колесо будет тормозить при использовании ручного тормоза.
- WheelView - Объект колеса который принимает позицию и вращение колеса, если нужно настроить вылет колеса, то нужно изменять положение вложенных во View объектов по оси X.
- BrakeSupport - Объект тормозной колодки, принимает позицию колеса, и поворачивается вместе с колесом в сторону поворота руля.
- MaxVisualDamageAngle - Все вложенные объекты во View рандомно поворачиваются до этого значения, зависит от степени повреждения. Это только визуальное повреждение, из за того что колесо немного смещается после повреждения, машина начинает ехать не совсем ровно, поэтому технического повреждения пока нет.

BikeWheel.cs

BikeWheel наследник класса Wheel, нужен только для правильного визуального отображения колес. BikeWheel не изменяет позицию WheelView, анимация подвески байка реализована в BikeController.

WheelCollider.

Для работы Wheel компонента необходим WheelCollider на объекте.

WheelCollider имеет очень важное значение в поведении управления автомобилем.

Очень многие люди не понимают как правильно настраивать WheelCollider, если у вас тоже с этим проблемы, постарайтесь внимательно прочитать документацию

[WheelCollider](#), думаю что у большинства возникают трудности с настройкой

Forward/Sideways Friction

Forward Friction - трение по направлению колеса, возникает при разгоне(Подачи на колесо крутящего момента)/торможении.

Sideways Friction - трение перпендикулярно направлению колеса, возникает при сносе/заносе оси, например при дрифте.

Далее будет несколько примеров с параметрами для Forward/Sideways Friction. изменяя значения между этими параметрами вы можете добиться желаемого результата, важно: масса автомобиля/колеса влияет на данные параметры (Например если автомобиль будет слишком легкий, эти параметры не будут соответствовать названию).

Слабое трение (Подходит для Sideways при дрифте):

Extremum Slip - 0.3

Extremum Value - 0.65

Asymptote Slip - 0.6

Asymptote Value - 0.2

Среднее трение :

Extremum Slip - 0.4

Extremum Value - 1

Asymptote Slip - 0.8

Asymptote Value - 0.5

Высокое трение :

Extremum Slip - 0.4

Extremum Value - 2

Asymptote Slip - 0.8

Asymptote Value - 1.2

Очень высокое трение :

Extremum Slip - 0.4

Extremum Value - 4

Asymptote Slip - 0.8

Asymptote Value - 2

Запредельное трение :

Extremum Slip - 0.4

Extremum Value - 10

Asymptote Slip - 0.4

Asymptote Value - 10

Stiffness - Множитель значений **Extremum Value** и **Asymptote Value** (по умолчанию 1).
Меняет жёсткость трения. Если установить в ноль, трение колеса полностью отключится.
Показатель меняется в зависимости от дорожного покрытия (Смотрите пункт
GroundDetection).

Искусственный интеллект (ИИ).

В ассете есть три типа ИИ: Для гонок, для дрифта и для погони. Для каждого типа ИИ есть ассет с настройками. Важно, для разных конфигураций машин (Отличается скорость, управляемость и т.п.) могут понадобиться разные настройки ИИ.

AIPath.

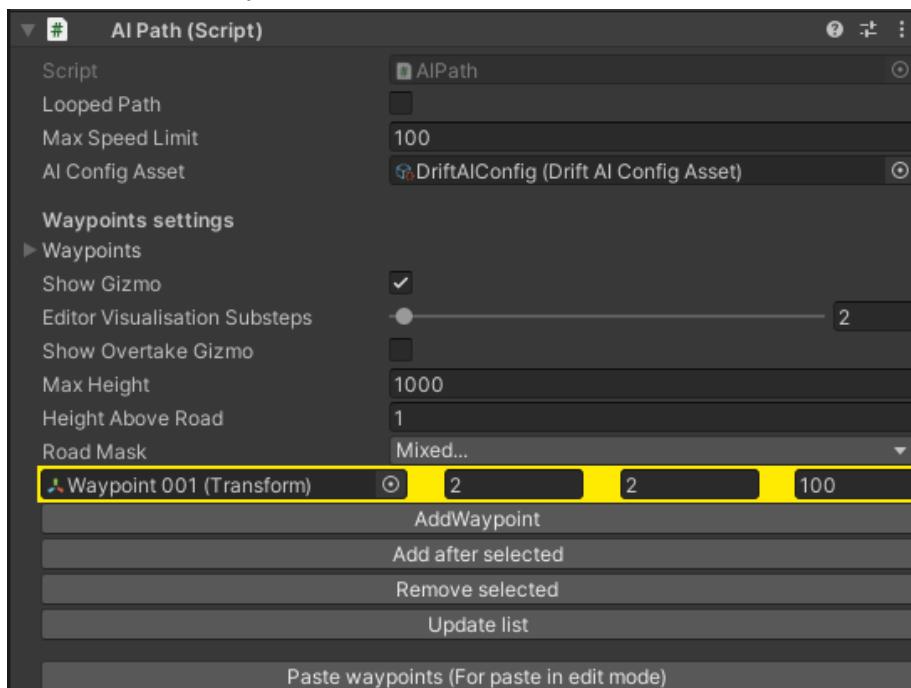
Этот класс нужен для создания путей по которым будет следовать ИИ в таких режимах как гонки и дрифт.

Класс разделен на 2 части:

AIPath.cs - в нем находится все данные и логика определения точек маршрута.

AIPathEditor.cs - в нем находится логика редактирования данных и отображения Gizmo, работает только в редакторе, на игровую логику не влияет.

Класс имеет следующие поля:



- LoopedPath - Зацикленность пути, вкл - круг, выкл - отрезок от точки А до Б.
- MaxSpeedLimit - Максимальное ограничение скорости.
- AIConfigAsset - Ассет с конфигурацией ИИ, все ИИ не имеющие ссылки на ассет будут обращаться к ассету из этого поля.
- Waypoints - Список точек типа WaypointData.
- ShowGizmo - Отображение Gizmo.
- EditorVisualisationSubsteps - Размер отрезков из которых состоит Gizmo всего пути, параметр только для визуализации пути в окне редактора, на логику не влияет.
- ShowOvertakeGizmo - Показывает Gizmo линии обгона.
- MaxHeight - Максимальная высота путевых точек (С этой высоты будет пускаться луч вниз, для поиска нужной высоты).
- HeightAboveRoad - Высота над поверхностью в которую попадает луч при обновлении путевых точек.
- RoadMask - Маска с которой взаимодействует луч.

- SelectedWaypoint - Вспомогательное поле для редактирования выбранной точки.

Кнопки:

- AddWaypoint - Добавляет точку в конец списка, с параметрами последней точки.
- AddAfterSelected - Добавляет точку после выбранной точки, с параметрами выбранной точки.
- RemoveSelected - Удаляет выбранную точку, выделяет следующую после удаленной.
- UpdateList - Обновляет высоту всех точек, переименовывает все точки по порядку в списке.
- CopyWaypoints - Копирует все настройки точек в Playmode (Для возможности редактирования пути в PlayMode).
- PasteWaypoints - Вставляет скопированные точки из Playmode вEditMode.

WaypointData - Данные путевой точки, все поля находятся в одной линии для удобства редактирования и нормального восприятия данных, особенно в списках. У данных есть всплывающие подсказки при наведении.

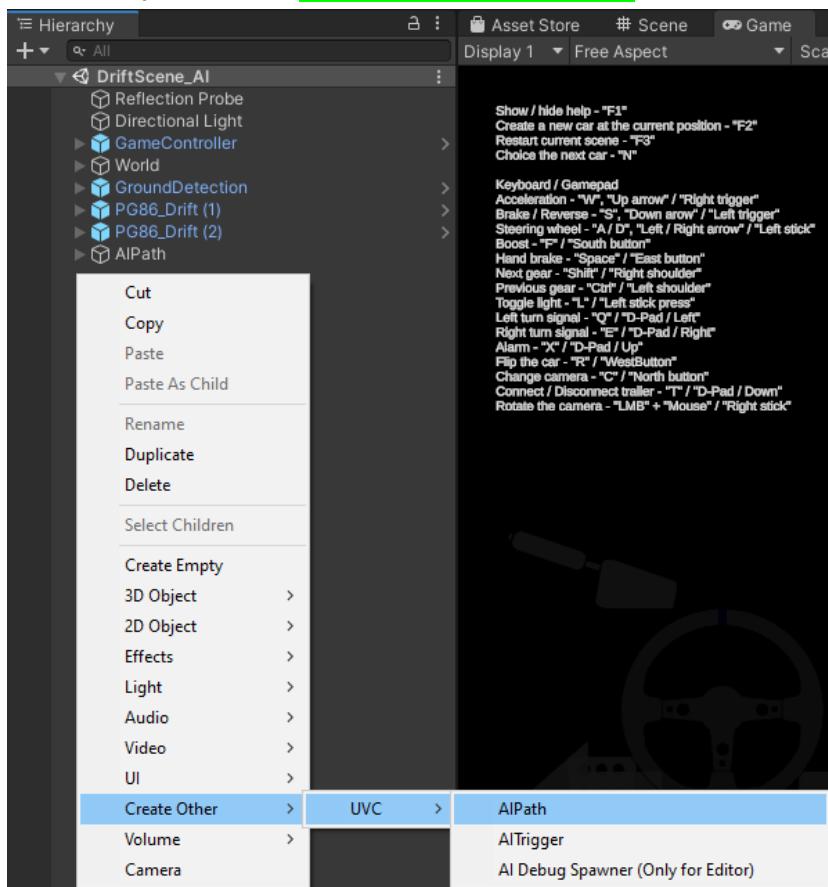
Содержит следующие поля:



- Point - Transform точки, для определения позиции.
- OvertakeZoneLeft - Размер зоны для обгона слева по движению пути.
- OvertakeZoneRight - Размер зоны для обгона справа по движению пути.
- SpeedLimit - Ограничение скорости на данном участке пути.

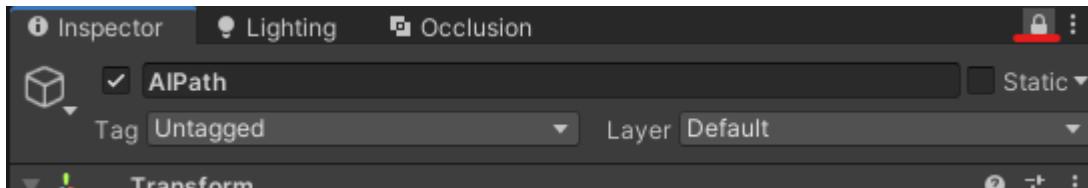
Создание AIPath:

Для создания объекта с компонентом AIPath нужно: В окне Hierarchy щелкнуть ПКМ и выбрать пункт меню “Create Other/UVC/AIPath”.



Редактирование AIPath:

Для удобства редактирования нужно заблокировать окно Inspector с выделенным объектом AIPath, для того чтобы фокус не смешался с этого объекта.



После этого вы можете создать новую точку (Если это новый AIPath, без созданных точек), первая точка создается в нулевой точке, все последующие точки создаются в позиции предыдущей точки и с параметрами предыдущей точки.

Для редактирования можно выставить ортографический вид сверху (**Щелкнуть СКМ по оси Y**)



После создания первой точки, вам нужно установить ее в место где на вашем треке находится начало пути. Далее нажимая на кнопку AddWaypoint, добавляйте нужное количество точек с нужными параметрами. После добавления точки высота всех точек обновляется.

Для редактирования уже созданной точки вы можете навести на нее курсор и щелкнуть любой кнопкой мыши, после этого точка выделяется и ее можно перемещать или менять любые параметры.

AITrigger.

Триггер влияющий на поведение ИИ. Сейчас с помощью триггера можно только настраивать зоны использования буста для ИИ, с возможностью выбора вероятности использования буста.

Для создания объекта с компонентом AITrigger нужно: В окне Hierarchy щелкнуть ПКМ и выбрать пункт меню “**Create Other/UVC/AITrigger**”.

Имеет только поле BoostProbability: 0 - буст не будет включаться, 1 - буст будет включаться всегда.

При выходе из триггера если буст был включен, то он прекращается.

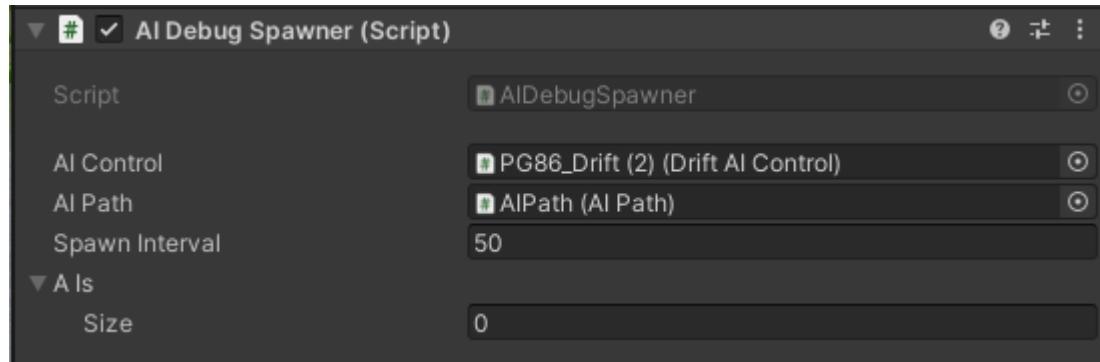
При желании вы можете любую другую логику добавить в AITrigger.cs.

AISpawner.

Нужен для быстрого тестирования и редактирования пути.

Для создания объекта с компонентом AISpawner нужно: В окне Hierarchy щелкнуть ПКМ и выбрать пункт меню “**Create Other/UVC/AISpawner**”.

Имеет следующие поля:



AIControl - Машина, которая будет дублироваться на протяжении всего пути.

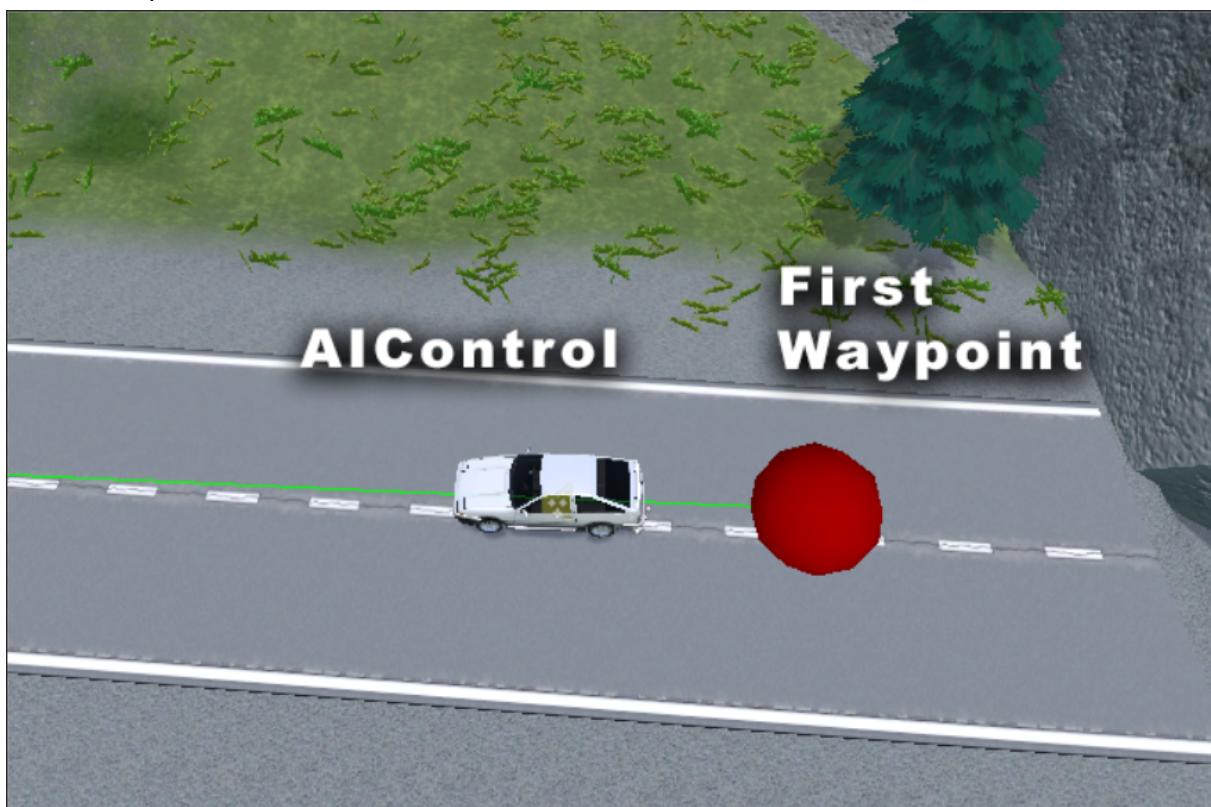
AIPath - Путь по которому будет дублироваться машина.

SpawnInterval - Расстояние между дублирующимися машинами.

AIs - Список с дублирующимися машинами (Машины туда добавляются при создании дубликатов).

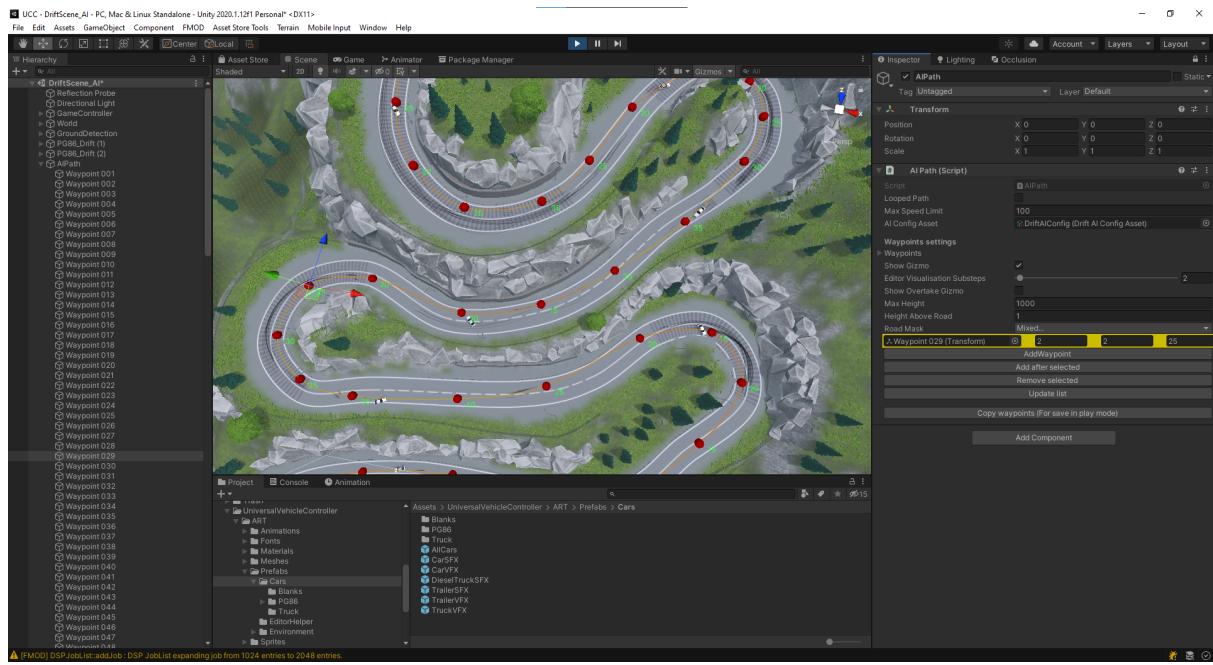
Пример использования:

1. Создаем AISpawner, указываем нужную машину и нужный путь (Машину нужно переместить в точку возле начала пути, так чтобы она была немного после первой точки по направлению пути). Также указать SpawnInterval, чем больше интервал тем меньше будет дубликатов, также количество дубликатов зависит от длины трека.



2. Запускаем Playmode, переключаемся на AIPath, блокируем Inspector и приступаем к редактированию пути (Можно изменять позиции точек, ограничения скорости и размер зон для обгона), на все изменения все ИИ будут менять свое поведение, это намного удобнее и быстрее чем запускать

Playmode после каждого редактирования пути.



- После завершения редактирования, важно **НЕ ВЫХОДИТЬ ИЗ PLAYMODE** нужно скопировать все изменения в Clipboard нажатием на кнопку “CopyWaypoints”, только после этого можно выйти из Playmode.
- В Editmode можно нажать на кнопку “PasteWaypoints”, после чего будут вставлены изменения скопированные из Playmode.
- Можно удалить или скрыть AISpawner.

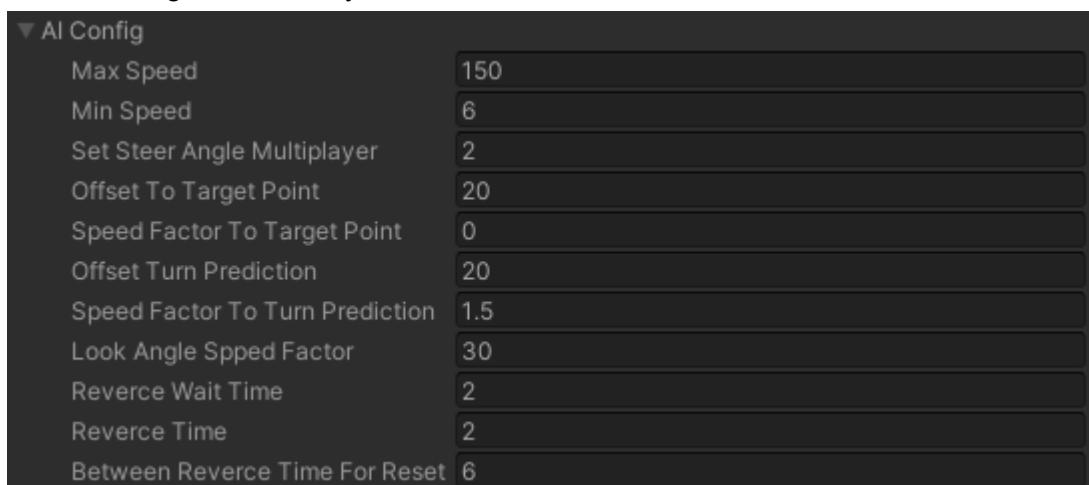
BaseAIControl.

Базовый класс для ИИ, все ИИ наследники этого класса, этот класс implements интерфейс ICarControl, в нем находится ссылка на AIConfigAsset и логика входа и выхода в AIRtrigger.

BaseAIConfigAsset

Асет с настройками ИИ типа BaseAIConfig, эти настройки применимы ко всем типам ИИ представленным в UVC.

BaseAIConfig имеет следующие поля:



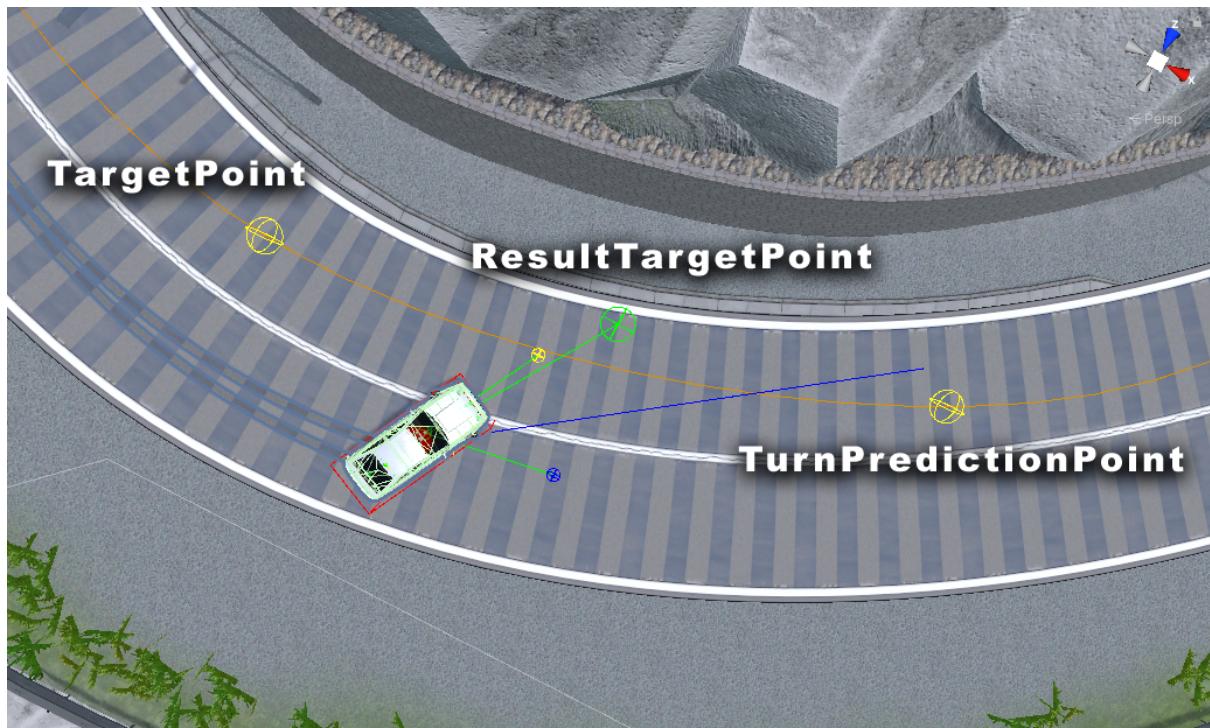
- MaxSpeed - Максимальная скорость (Unit в секунду) к которой будет стремится ИИ.

- MinSpeed - Минимальная скорость к которой будет стремится ИИ.
- SetSteerAngleMultiplayer - множитель поворота руля к TargetPoint, чем выше этот параметр, тем быстрее ИИ будет поворачивать к нужной позиции (Оптимальное значение от 1 до 4, зависит от скорости и типа ИИ).
- OffsetToTargetPoint - Постоянное смещение TargetPoint. ниже будут примеры для всех типов ИИ.
 - Для RaceAIControl и DriftAIControl, смещение от текущей позиции на пути.
 - Для PursuitAIControl, смещение от позиции преследуемой машины в сторону перемещения (Rigidbody.velocity) преследуемой машины.
- SpeedFactorToTargetPoint - Динамическое смещение TargetPoint. ниже будут примеры для всех типов ИИ.
 - Для RaceAIControl и DriftAIControl, добавочное смещение, зависит от скорости машины.
 - Для PursuitAIControl добавочное смещение, зависит от скорости преследуемой машины.
- OffsetTurnPrediction - Смещение точки для контроля скорости на поворотах, для разных ИИ применяется так же как OffsetToTargetPoint . ниже будут примеры для всех типов ИИ.
- SpeedFactorToTurnPrediction - Динамическое смещение TurnPredictionPoint, для разных ИИ применяется так же как SpeedFactorToTargetPoint . ниже будут примеры для всех типов ИИ.
- LookAngleSppedFactor - Максимальный угол между направлением машины ИИ и точкой (Зависит от типа ИИ), чем больше угол этого параметра, тем сильнее машина ИИ будет тормозить.
 - Для DriftAIControl, угол между направлением машины ИИ и рассчитанной точкой следования ИИ.
 - Для RaceAIControl и PursuitAIControl, угол между направлением машины ИИ и TurnPredictionPoint.
- ReverseWaitTime - Время ожидания до включения заднего хода, если машина ИИ застряла или уперлась в препятствие, то по истечении этого времени включится задний ход и колеса повернуться в противоположную сторону.
- ReverseTime - Время на которое включается задний ход.
- BetweenReverseTimeForReset - Если в течении этого времени включится задний ход повторно, то машина будет сброшена (Логику сброса нужно сделать исходя из требований вашей игры, сейчас сброс происходит в туже позицию).

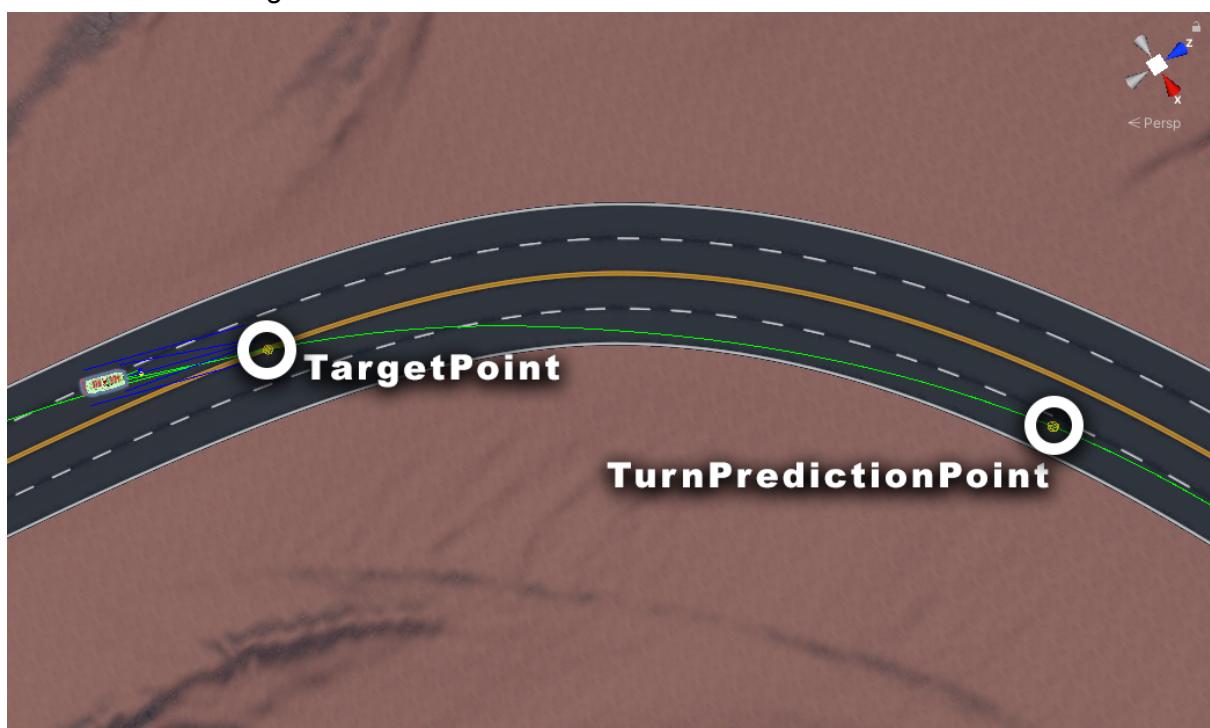
Настройки OffsetToTargetPoint и OffsetTurnPrediction для разных типов ИИ.

Для DriftAIControl настройки находятся в ассете DriftAIConfig. Точка к которой следует автомобиль рассчитывается динамически по формуле: (TargetPoint + TurnPredictionPoint) * 0.5f, ResultTargetPoint находится ровно между точками TargetPoint

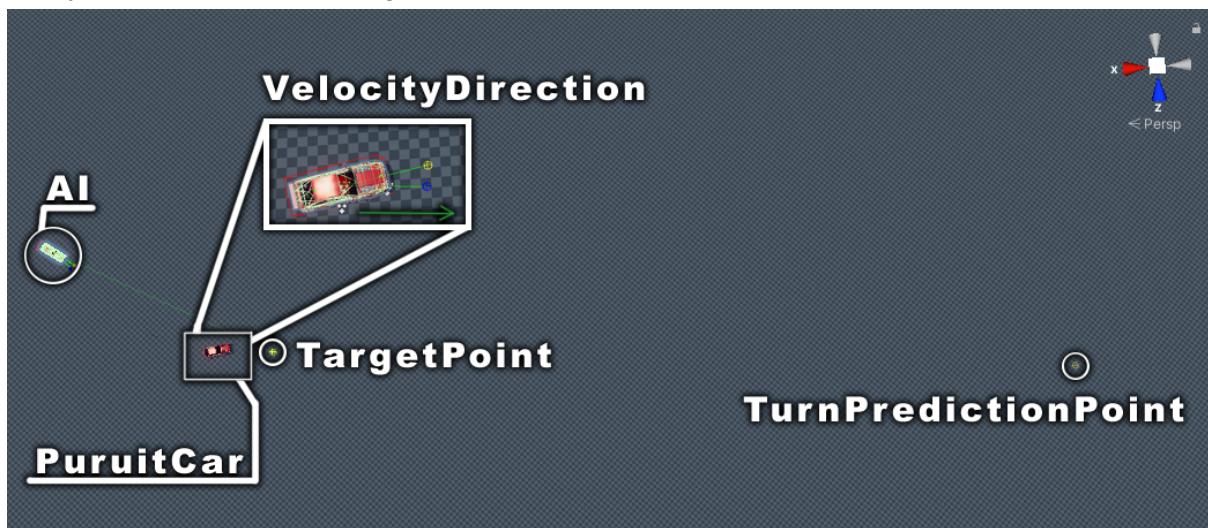
и TurnPredictionPoint, для возможности направлять машину ИИ во внутрь поворота.



Для RaceAIControl настройки находятся в ассете RaceAIConfig для сцены RaceScene и в ассете RaceAIConfig_ForRing для сцены MobileScene. Точка к которой следует автомобиль это TargetPoint.



Для PursuitAIControl настройки находятся в ассете PursuitAIConfig. Точка к которой следует автомобиль это TargetPoint.



RaceAIControl.

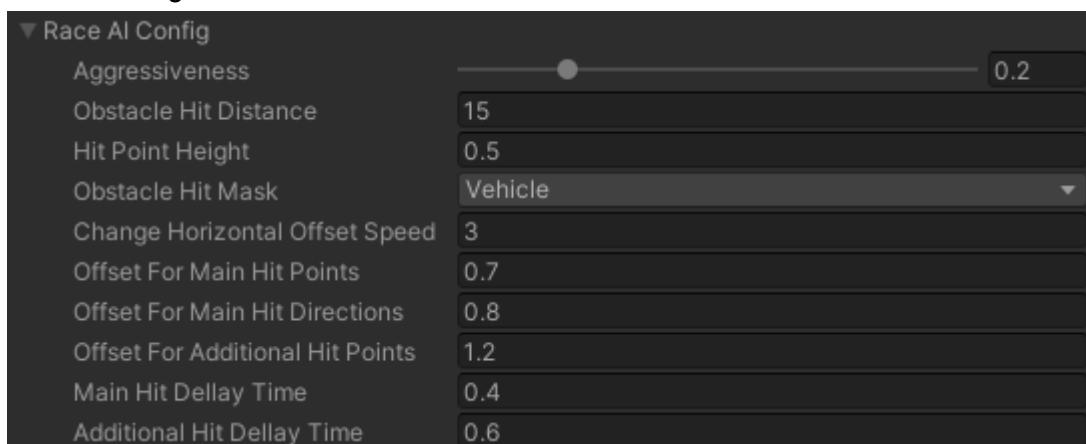
Класс ИИ для скоростных гонок, следует по пути, имеет логику обгона соперников. В ассете все трассы с длинными и плавными поворотами, если в вашей игре более резкие повороты и ИИ будет с ними плохо справляться, то вам может помочь ограничение скорости с помощью AIPATH.

Алгоритм расчета точек слежения смотрите в пункте “Настройки OffsetToTargetPoint и OffsetTurnPrediction для разных типов ИИ”.

RaceAIConfigAsset

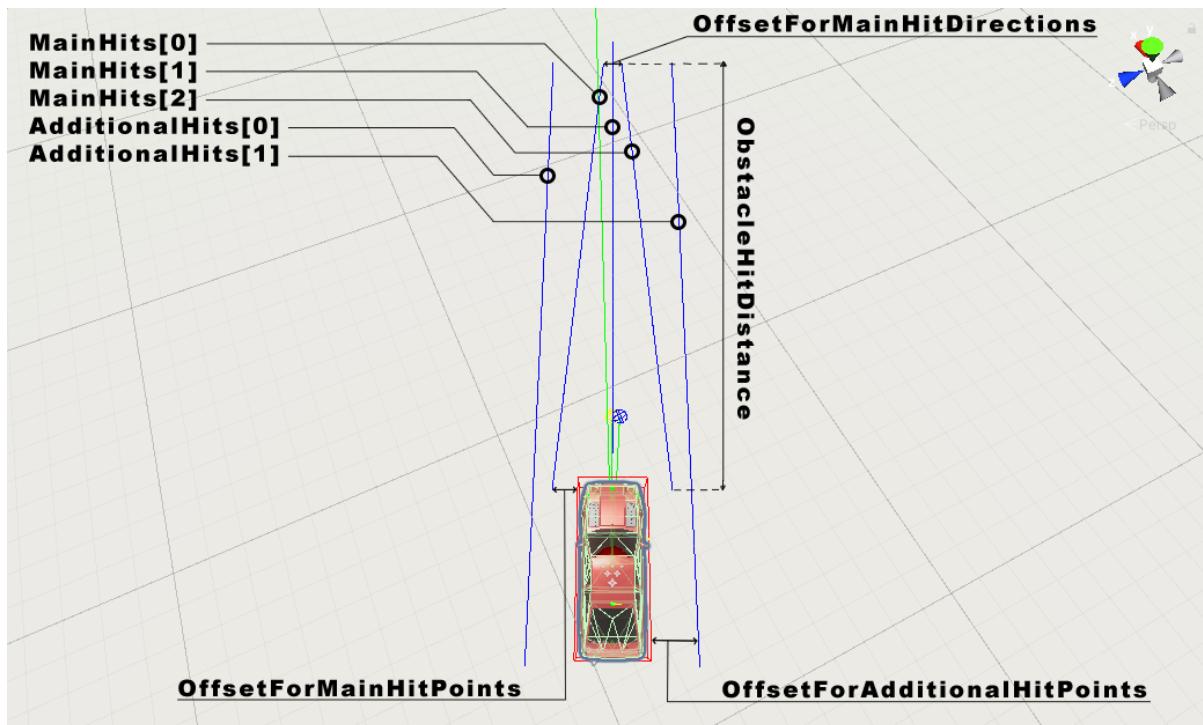
Наследник BaseAIConfigAsset, содержит класс RaceAIConfig, нужен для конфигурации RaceAiControl.

RaceAIConfig имеет поля:



- Aggressiveness - Агрессивность ИИ, 0 - ИИ будет притормаживать перед впереди идущей машиной, 1 - ИИ не будет сбавлять скорость, будет таранить впереди идущую машину.
- ObstacleHitDistance - Дистанция проверки препятствий.
- HitPointHeight - Высота луча для проверки препятствий.
- ObstacleHitMask - Мaska препятствий, по умолчанию содержит только слой 8, у вас номера слоев могут отличаться.

- ChangeHorizontalOffsetSpeed - Скорость изменения горизонтального смещения, для обгона соперников.
- OffsetForMainHitPoints - Смещение позиции главных лучей относительно боковых границ автомобиля, 0 - позиция лучей будет находиться на границе автомобиля.
- OffsetForMainHitDirections - Смещение направления главных лучей, 0 - лучи будут параллельны направлению машины, 1 лучи будут направлены к центру машины.
- OffsetForAdditionalHitPoints - Смещение позиции дополнительных лучей относительно боковых границ автомобиля, 0 - позиция лучей будет находиться на границе автомобиля.
- MainHitDelayTime - Интервал использования главных лучей, для оптимизации.
- AdditionalHitDelayTime - Интервал использования дополнительных лучей, для оптимизации.



Алгоритм работы обгона.

Если все лучи не имеют препятствий, то CurrentHorizontalOffset стремится к 0 со скоростью ChangeHorizontalOffsetSpeed * 0.2f.

Если главный луч имеет препятствие, то CurrentHorizontalOffset стремится в противоположную сторону до границы обгона со скоростью ChangeHorizontalOffsetSpeed * (1 - процент до препятствия от длины луча), чем ближе препятствие тем быстрее изменяется смещение.

Если MainHits[0] (левый луч) имеет препятствие то смещение происходит в правую сторону.

Если MainHits[1] (правый луч) имеет препятствие то смещение происходит в левую сторону.

Если оба луча имеют препятствия, то логика включается для того луча у которого дистанция меньше.

Если только MainHits[2] (центральный луч) имеет препятствие, то смещение продолжает стремиться в сторону в которую уже есть смещение.

Если дополнительный луч имеет препятствие а главные лучи не имеют препятствия, то CurrentHorizontalOffset остается на месте.

DriftAIControl

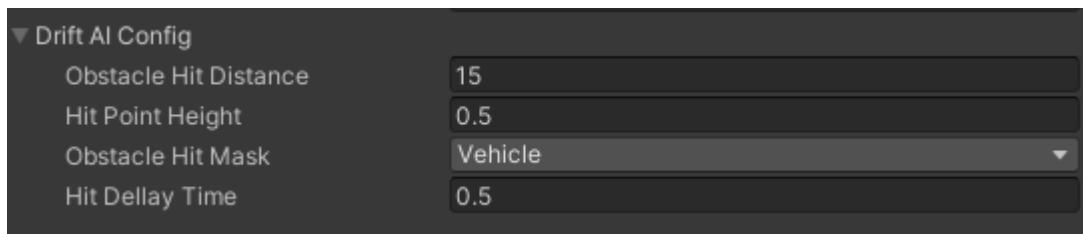
Класс ИИ для дрифта, следует по пути, очень чувствителен к настройкам и к скорости, в ассете все пути для DriftAIControl имеют тонкую настройку по ограничению скорости, из за невозможности определить незначительные повороты при высоких скоростях, ограничение также дает больше контроля над ИИ и предсказуемости его поведения. DriftAIControl пускает один луч в сторону между направлением машины и силой ее перемещения, для обнаружения впереди едущей машины и притормаживания при необходимости.

Алгоритм расчета точек слежения смотрите в пункте “Настройки OffsetToTargetPoint и OffsetTurnPrediction для разных типов ИИ”.

DriftAIConfigAsset

Наследник BaseAIConfigAsset, содержит класс DriftAIConfig, нужен для конфигурации DriftAiControl.

DriftAIConfig имеет поля:



- ObstacleHitDistance - Дистанция проверки препятствий.
- HitPointHeight - Высота луча для проверки препятствий.
- ObstacleHitMask - Мaska препятствий, по умолчанию содержит только слой 8, у вас номера слоев могут отличаться.
- HitDelayTime - Интервал использования луча, для оптимизации.

PursuitAIControl

Класс ИИ для преследования машины. Класс имеет дополнительное поле “TargetRB” - это ссылка на преследуемую машину, при желании вы можете добавить логику выбора преследуемой машины. Это примитивное преследование, без каких либо хитростей.

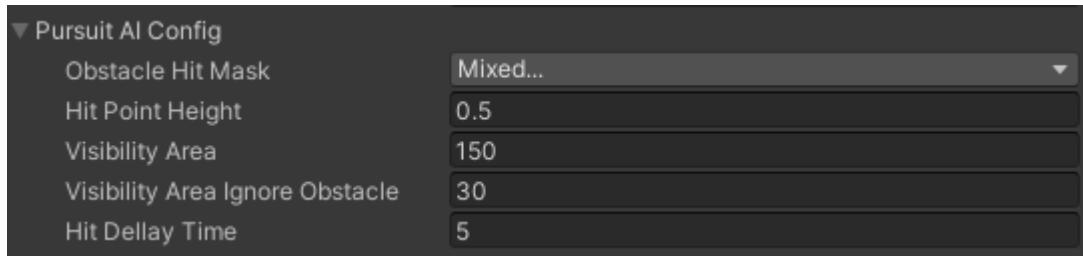
Сейчас ИИ может находиться в покое пока преследуемая машина не появится в поле зрения, после того как машина попадает в поле зрения, ИИ начинает преследовать машину (Преследование не выключается, но вы можете добавить логику выключения). Сейчас ИИ преследует машину не беря во внимание окружение, при резких поворотах (Например городские улицы) он может упереться в колайдер или упасть в пропасть. Вы можете улучшить логику преследования на свой вкус или комбинировать ее с другими типами ИИ. К сожалению для всех типов игр я не могу создать универсальный ИИ преследования, это сильно зависит от левелдизайна и типа игры.

Алгоритм расчета точек слежения смотрите в пункте “Настройки OffsetToTargetPoint и OffsetTurnPrediction для разных типов ИИ”.

PursuitAIConfigAsset

Наследник BaseAIConfigAsset, содержит класс PursuitAIConfig, нужен для конфигурации PursuitAiControl.

PursuitAIConfig имеет поля:



- ObstacleHitMask - Мaska препятствий, по умолчанию содержит только слой 8, у вас номера слоев могут отличаться.
- HitPointHeight - Высота луча для проверки препятствий.
- VisibilityArea - Зона видимости, 180 градусов перед машиной, препятствия не игнорируются. Если преследуемая машина приблизится с передней стороны и преследуемая машина не будет скрыта препятствием, то ИИ начнет преследование.
- VisibilityArealgnoreObstacle - Зона видимости вокруг машины, препятствия игнорируются. Если преследуемая машина приблизится с любой стороны на это расстояние, то ИИ начнет преследование.
- HitDelayTime - Интервал использования луча, для оптимизации.

Световые сигналы, стекла.

Главный компонент логики световых сигналов автомобиля CarLighting, для его работы требуется компонент CarController на объекте.

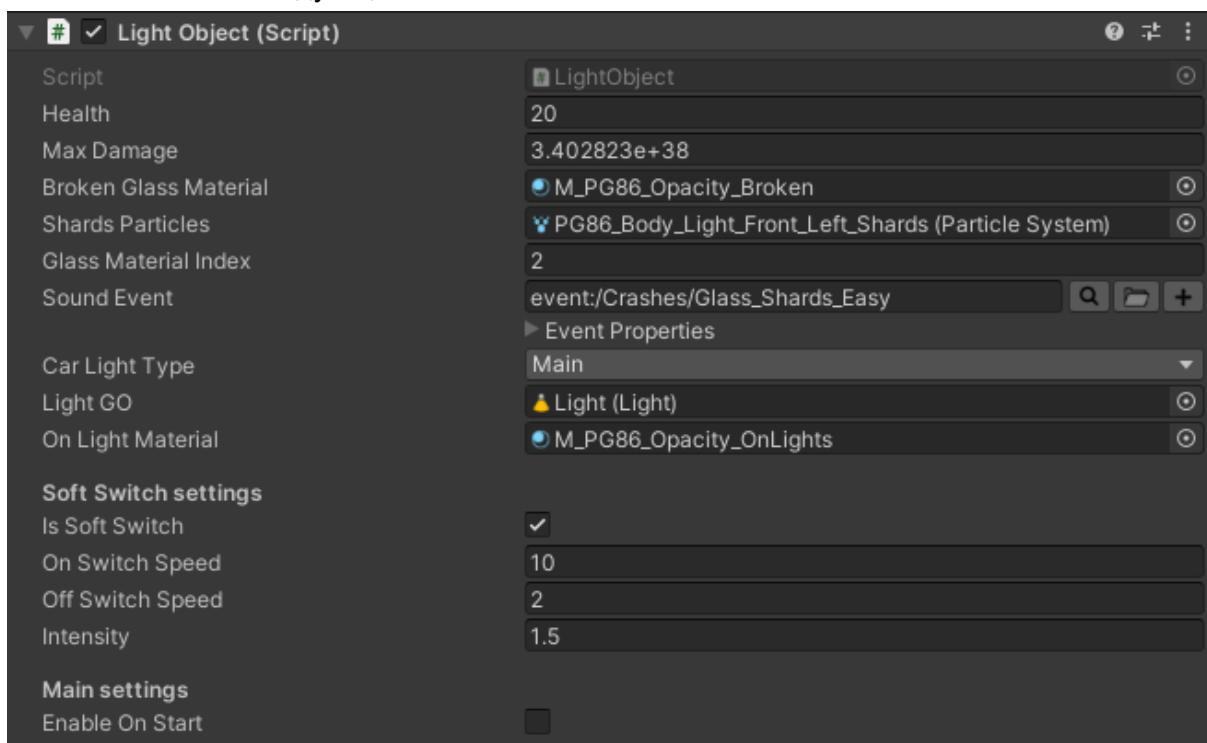
Имеет только одно настраиваемое поле:

TurnsSwitchHalfRepeatTime - Время половины цикла работы поворотника
(Используется для всех поворотников и аварийной сигнализации).

Все подписки на события (Задний ход, торможение, включение/выключение света), управление светом находящимся в дочерних объектах происходит в теле этого(CarLighting) класса.

LightObject.

Это настраиваемый объект, в нем настраивается тип источника света и другие параметры. LightObject наследник GlassDO (GlassDO используется для стекол и имеет ту же логику разрушения). Количество однотипных источников света не ограничено. Компонент имеет следующие поля:



Поля из GlassDO:

- BrokenGlassMaterial - Сломанный материал, если нужно отобразить осколки после разрушения, если это поле null то после разрушения объект будет невидим.
- ShardsParticles Система частич запускаемая при разрушении объекта
- GlassMaterialIndex - Индекс материала, на случай если Mesh имеет более одного SubMesh.
- SoundEvent - звук воспроизводимый при разрушении объекта.

Поля из LightObject:

- CarLightType - Тип света, от него зависит в какой момент будет включаться свет.
- LightGO - Объект источника света, просто включается/выключается при переключении.
- OnLightMaterial - материал применяемый к Renderer(Если этот компонент есть на объекте), у материала флаг Emmision должен быть включен. Один и тот же материал может использоваться со всеми источниками света (Как в этом ассете), так как используется MaterialPropertyBlock.
- IsSoftSwitch - Флаг мягкого включения, для имитации ламп накаливания (Постепенное включение/выключение). Если флаг выключен, то включение/выключение будет происходить в один кадр, просто подменой материалов OnLightMaterial и материалом по умолчанию (Материал по умолчанию присвоен в Renderer.materials).
- OnSwitchSpeed - Скорость включение при включенном флаге IsSoftSwitch.
- OffSwitchSpeed - Скорость выключения при включенном флаге IsSoftSwitch.
- Intensity - Интенсивность свечения при включенном свете, при включенном флаге IsSoftSwitch.
- EnableOnStart - Флаг если нужно включить свет на старте (Например, если гонка происходит ночью).

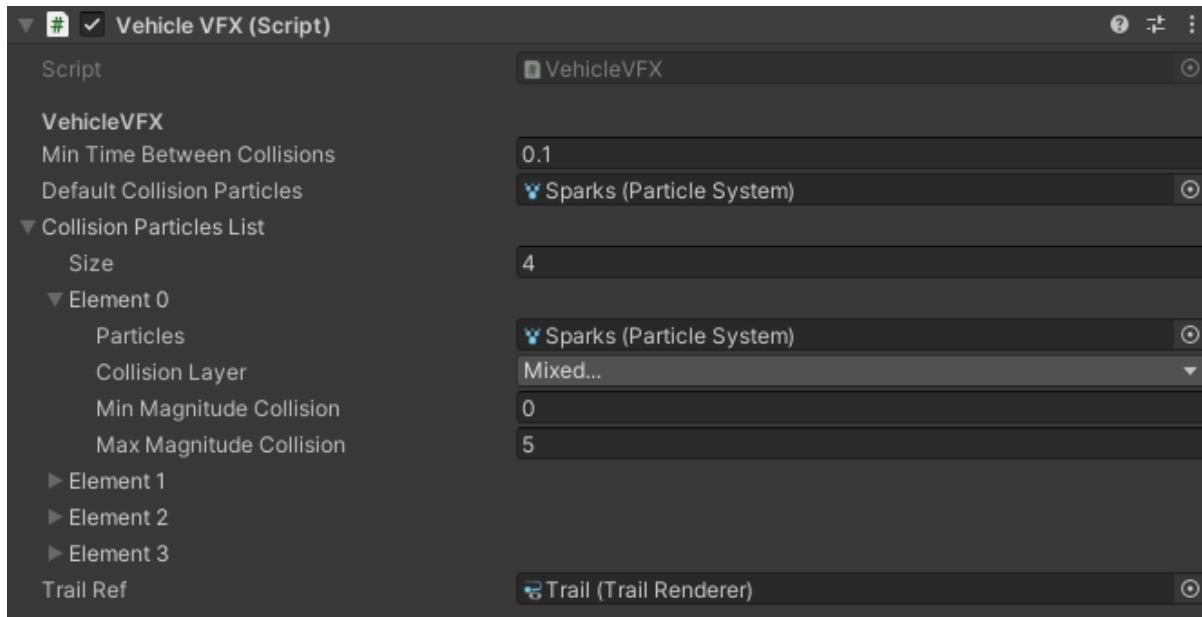
Для удобства добавления осколков стекол и световых сигналов есть два метода CreateGlassShards и CreateLightShards вызвать которые можно через контекстное меню компонента. После создания осколков нужно выбрать правильное направление системы частиц и настроить размер если в этом есть необходимость.

Visual effects.

VehicleVFX.

Базовый компонент логики обработки визуальных эффектов VehicleVFX, объект с этим компонентом должен быть дочерним объекту с компонентом VehicleController.

VehicleVFX создает такие эффекты как дым от покрышек (В зависимости от поверхности по которому едет колесо), следы от шин, искры и дым от столкновений. Компонент имеет следующие поля:



- MinTimeBetweenCollisions - Если за это время произойдет более одного столкновения, то частицы выпускаться не будут.
- DefaultCollisionParticles - Система частиц использующаяся если не найдена система частиц из списка CollisionsParticles.
- CollisionParticlesList - список типа CollisionParticles, при ударе происходит поиск подходящей системы частиц из этого списка, структура CollisionParticles имеет следующие поля:
 - Particles - Ссылка на систему частиц.
 - CollisionLayer - Система частиц выбирается если слой объекта столкновения принадлежит этой маске.
 - MinMagnitudeCollision - Система частиц выбирается если сила столкновения больше этого значения.
 - public float MaxMagnitudeCollision - Система частиц выбирается если сила столкновения меньше этого значения.
- TrailRef - при скольжении колеса, создается копия этого Trail который оставляет след в пятне контакта с поверхностью.
- Пыль/Дым из под колес воспроизводится из системы частиц GroundConfig.SlipParticles или GroundConfig.IdleParticles, в зависимости от текущего скольжения колеса. Информация о текущем GroundConfig находится в классе Wheel.

CarVFX.

CarVFX наследник класса VehicleVFX, объект с этим компонентом должен быть дочерним объекту с компонентом CarController. Нужен для воспроизведения эффектов машин таких как дым/огонь из выхлопной трубы.

Компонент имеет следующие поля:



ExhaustParticles - Системы частиц выпускаемые во время нагрузки на двигатель (Подходит для дизельных грузовиков).

BackFireParticles - Вспышки огня из выхлопной трубы, во время отсечки или переключения передач.

BoostParticles - Огонь из выхлопной трубы во время использования буста.

EngineHealth75Particles - Система частиц включаемая при здоровье двигателя 75%.

EngineHealth50Particles - Система частиц включаемая при здоровье двигателя 50%.

EngineHealth25Particles - Система частиц включаемая при здоровье двигателя 25%.

При уровне здоровья двигателя <25% система частиц EngineHealth50Particles тоже активна.

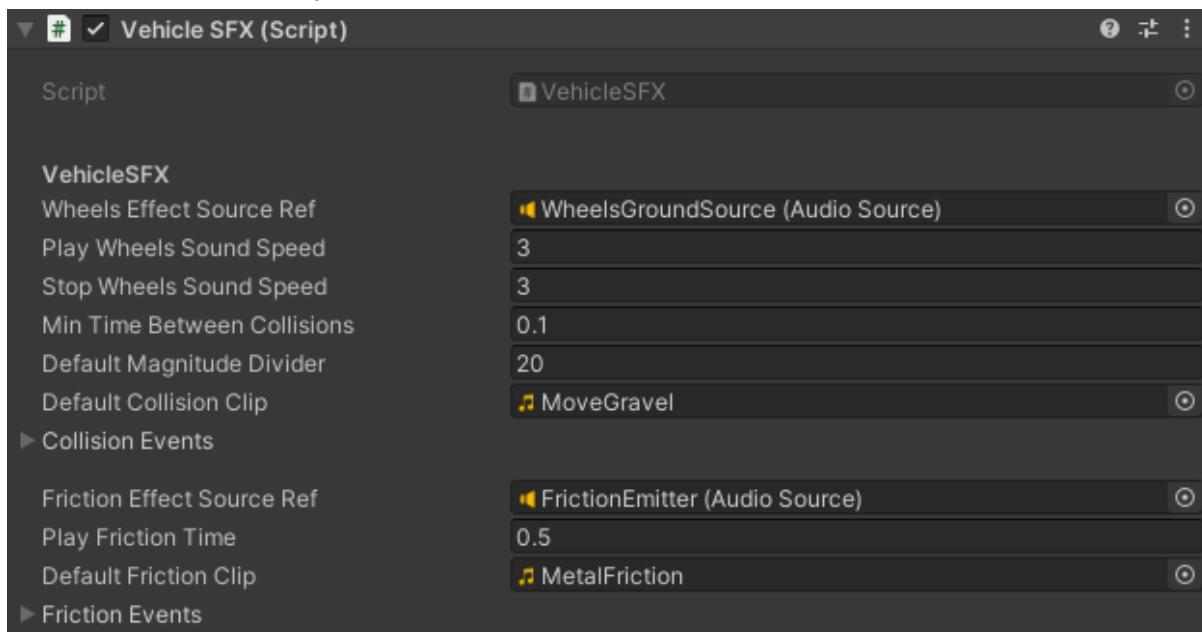
Sound effects.

На машине игрока должен быть компонент AudioListener.

VehicleSFX.

Базовый компонент логики обработки звуков VehicleSFX, объект с этим компонентом должен быть дочерним объекту с компонентом VehicleController. Некоторые звуки воспроизводятся не этим компонентом (Например звук осколков стекла).

Компонент имеет следующие поля:



- **WheelsEffectSourceRef** - ссылка на AudioSource, копия создается для звуков колеса (Машина может ехать по разным поверхностям одновременно), имя события находится в GroundConfig.Event. Информация о текущем GroundConfig находится в классе Wheel. В событие передается текущее скольжение колеса (Максимальное если колеса несколько с одним событием) и текущая скорость машины.
- **MinTimeBetweenCollisions** - Минимальный промежуток времени между воспроизведением звуков столкновения, для предотвращения звуковых артефактов при трении колпайдеров.
- **DefaultMagnitudeDivider** - максимальный делитель силы удара, для вычисления громкости звука удара (Если не найден ColissionEvent при столкновении).
- **DefaultCollisionClip** - Звук удара воспроизводимый если не найден ColissionEvent при столкновении.
- **CollisionEvents** - Массив типа CollisionEvent. Этот массив определяет звуки воспроизводимые при столкновениях, для одного слоя можно настроить несколько звуков в зависимости от силы столкновения.
- **FrictionEffectEmotterRef** - ссылка на Studio Event Emitter, копия создается для звуков скольжения по различным поверхностям (Звуки настраиваются в списке FrictionEvents),
- **PlayFrictionTime** - Время проигрывания скольжения, если контакт чаще чпроисходит чаще чем PlayFrictionTime, то звук проигрывается по кругу.
- **DefaultFrictionClip** - Звук удара воспроизводимый если не найден ColissionEvent при скольжении.

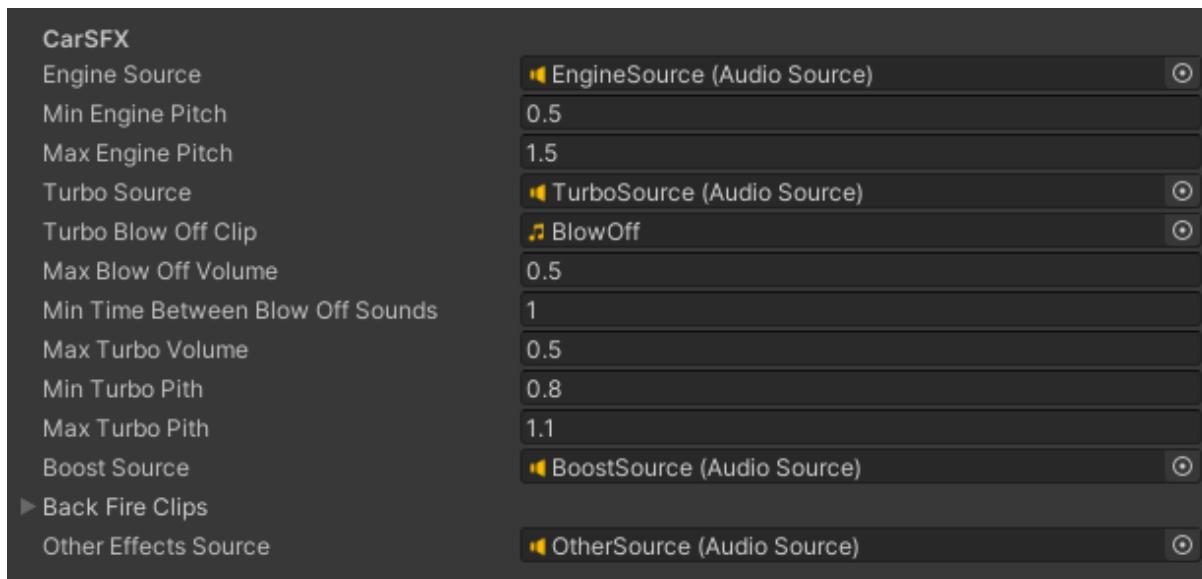
- FrictionEvents - Массив типа CollisionEvent. Этот массив определяет звуки воспроизведимые при скольжениях, для одного слоя можно настроить несколько звуков в зависимости от силы скольжения.

CollisionEvent Имеет следующие поля:

- CollisionEvent.AudioClip - Звук.
- CollisionEvent.CollisionLayer - Слой, при столкновении с которым будет воспроизведится звук.
- CollisionEvent.MinMagnitudeCollision - Минимальная сила удара для воспроизведения этого звука.
- CollisionEvent.MaxMagnitudeCollision - Максимальная сила удара для воспроизведения этого звука. Эта же переменная используется в качестве делителя, для определения громкости звука.

CarSFX.

CarSFX CarVFX наследник класса VehicleSFX, объект с этим компонентом должен быть дочерним объекту с компонентом CarController. Нужен для воспроизведения звуков двигателя. Имеет поля:



- EngineSource - Source который постоянно проигрывает звук двигателя.
- MinEnginePitch - Pitch на минимальных оборотах двигателя.
- MaxEnginePitch - Pitch на максимальных оборотах двигателя.
- TurboSource - Source для проигрывания звука турбины.
- TurboBlowOffClip - Звук сброса давления турбины.
- MaxBlowOffVolume - Максимальная громкость звука сброса давления.
- MinTimeBetweenBlowOffSounds - Минимальное время между звуками сброса давления (Чтобы звук сброса не надоедал).
- MaxTurboVolume - Максимальная громкость звука турбины.
- MinTurboPith - Pitch на минимальных оборотах турбины.
- MaxTurboPith - Pitch на максимальных оборотах турбины.
- BoostSource - Source нитро ускорения, просто включается/выключается при использовании.
- BackFireClips - Список звуков огня из выхлопной трубы.
- OtherEffectsSource - Source в котором проигрываются дополнительные звуки (Звук огня из выхлопной трубы, звук сброса давления).

GroundDetection:

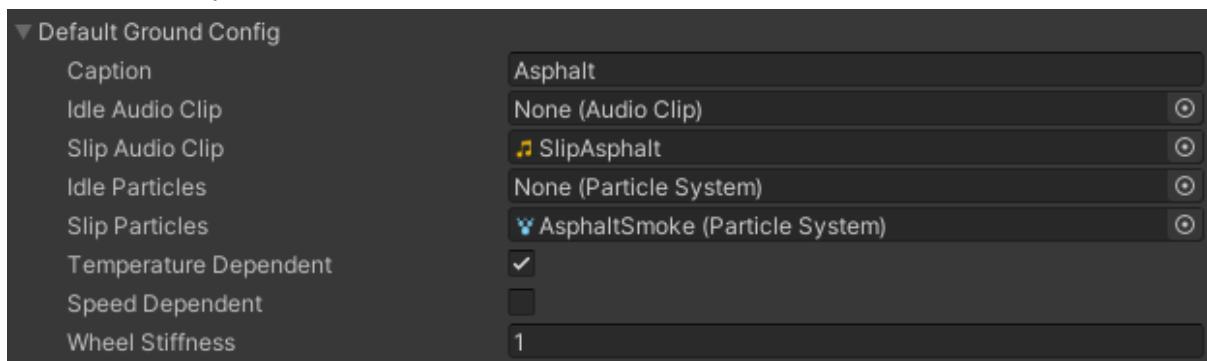
GroundDetection.

Класс запоминающий компоненты IGroundEntity находящиеся на GameObjects по которым проезжает колесо. Имеет только одно поле: DefaultGroundConfig типа GroundConfig. В данном проекте компонент GroundDetection находится в префабе GroundDetection в котором находятся все Системы частиц (Пыль/Дым из под колес), во всех GroundConfig ссылки именно на эти системы частиц (Просто для удобства настройки).

GroundConfig.

Содержит в себе настройки поверхности по которой едет колесо, для изменения силы трения колеса, звука и системы частиц.

Содержит следующие поля:



- Caption - Просто название настройки, для удобства редактирования конфигов в списках.
- IdleAudioClip - Звук который воспроизводится при простом движении по поверхности.
- SlipAudioClip - Звук который воспроизводится при скольжении колеса по поверхности.
- IdleParticles - Система частиц которая испускает частицы если колесо просто едет по поверхности (Без скольжения), если IdleParticles == null то частицы испускаться не будут.
- SlipParticles- Система частиц которая испускает частицы если колесо скользит по поверхности, если SlipParticles == null то частицы при скольжении испускаться не будут.
- TemperatureDependent - Зависимость от температуры колеса, чем выше температура, тем больше и дальше будут испускаться частицы (Например, на асфальте дым от покрышек увеличивается при росте температуры колеса).
- SpeedDependent- Зависимость от скорости машины, чем выше скорость, тем больше и дальше будут испускаться частицы.
- WheelStiffness - Множитель для значений трения колеса.

ObjectGroundEntity.

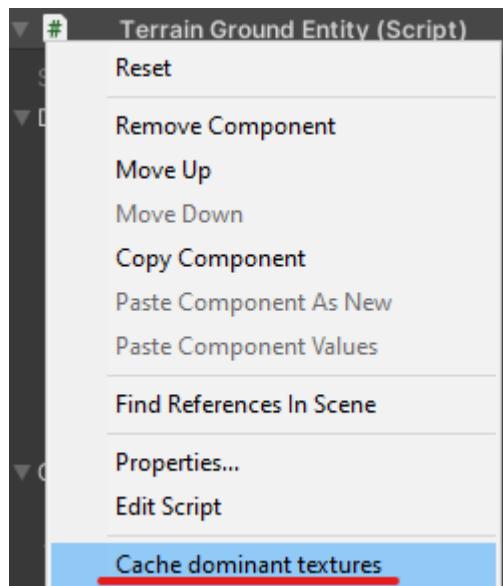
Этот компонент нужен для определения типа поверхности конкретному объекту (Например дороге).

Имеет только одно поле GroundConfig, когда колесо будет ехать по этому объекту, колесу будет передан этот GroundConfig.

TerrainGroundEntity.

!!!ВНИМАНИЕ!!! После редактирования террейна или TerrainGroundEntity

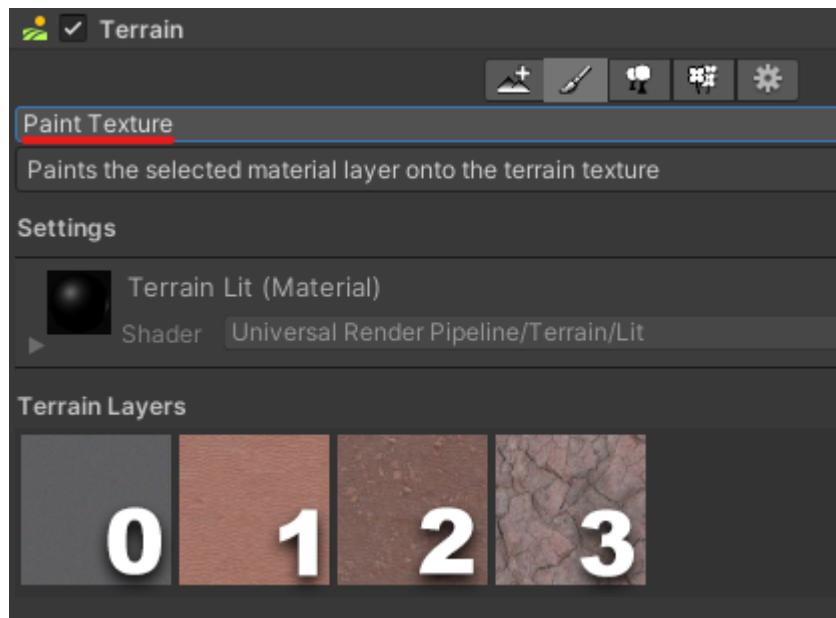
нужно выполнить команду из контекстного меню TerrainGroundEntity “Cache dominant textures” для расчета и сохранения всех текстур в скрытый массив DominateTextures.



TerrainGroundEntity - это компонент для настройки GroundConfigs террейна, для каждой текстуры террейна можно настроить уникальный конфиг. Имеет поля:

DefaultConfig - конфиг использующийся в случае если не найден конфиг для текстуры.
GroundConfigs - Список типа TerrainGroundConfig, конфиги для разных текстур.

TerrainGroundConfig обертка для GroundConfig имеет дополнительное поле TextureIndex в котором нужно указать индекс текстуры, индекс текстуры это индекс слоя террейна.

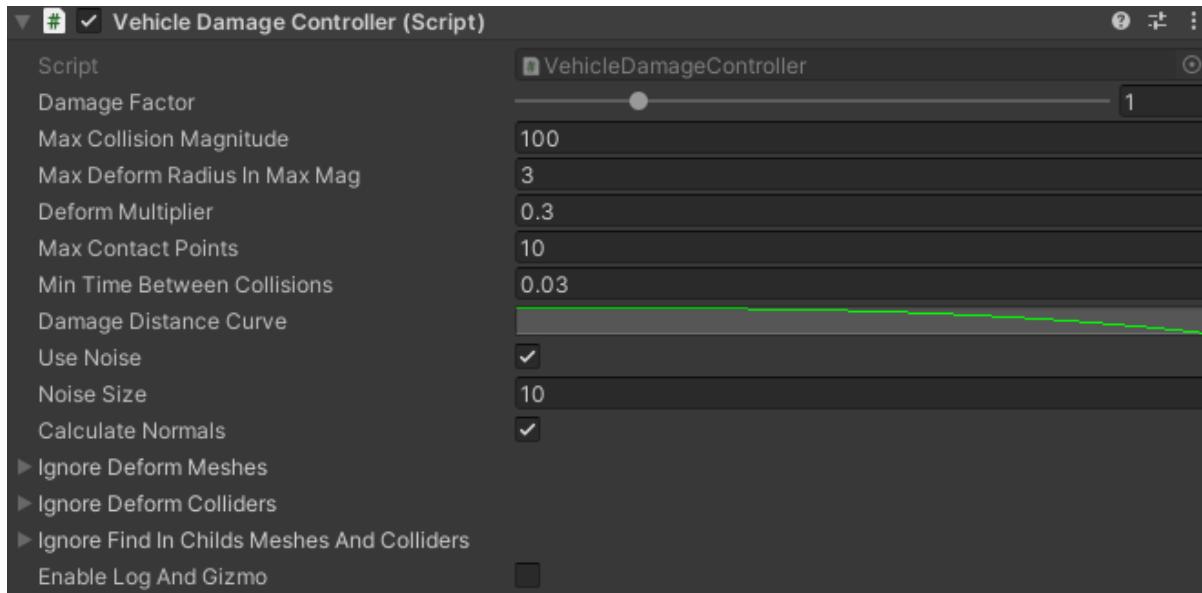


CarDamage.

VehicleDamageController.

Главный компонент обрабатывающий столкновения. При старте игры, этот контроллер ищет все дочерние Meshes, MeshColliders, DamageableObjects, DetachableObjects, MoveObjects (За исключением объектов из IgnoreDeformMeshes и IgnoreDeformColliders) добавляет их в соответствующие списки для дальнейшего повреждения.

Компонент имеет следующие поля:



- DamageFactor - множитель повреждения.
- MaxCollisionMagnitude - Максимальная сила удара.
- MaxDeformDistanceInMaxMag - Максимальная дистанция повреждаемых точек, при максимальной силе удара, например: Если сила удара равна 25(25% от максимальной) то удар будет распространяться на вершины и объекты в радиусе 0,75 юнита от точки столкновения (Чем дальше от точки столкновения, тем меньше повреждение).
- MaxContactPoints - Максимальное количество точек, при одном столкновении.
- MinTimeBetweenCollisions - Минимальное время между столкновениями.
- DamageDistanceCurve - кривая для расчета силы повреждения на расстоянии от точки столкновения. В скрипте везде используется квадрат длины (Для оптимизации), из за чего сила удара должна рассчитываться не линейно, в этом нам и помогает кривая для того чтобы избежать тяжелых вычислений.
- UseNoise - Использование шума при деформации видимых Meshes.
- NoiseSize - Размер шума (Чем больше тем ярче будет выражен шум).
- CalculateNormals - Пересчитать нормали после столкновения.
- IgnoreDeformMeshes - Список игнорируемых MeshFilters при инициализации.
- IgnoreDeformColliders - Список игнорируемых MeshColliders при инициализации.
- IgnoreFindInChildsMeshesAndColliders - Объекты в которых не будет происходить поиск повреждаемых мешей и коллайдеров.

DamageableObject.

Повреждаемый объект, он имеет показатель здоровья и ему наносятся повреждения при столкновениях (Наносимый урон зависит от силы столкновения и дальности LocalCenterPoint от точки удара). LocalCenterPoint - находится в центре масс вершин of Mesh если на этом объекте есть MeshFilter, иначе LocalCenterPoint == Vector3.zero. От этого объекта можно наследовать все повреждаемые объекты (Например двигатель, трансмиссия, рулевое управление), как сейчас сделаны некоторые элементы (колеса, стекла, фары).

Компонент имеет 2 поля:

Health - Здоровье на старте.

MaxDamage - Максимальное повреждение за одно столкновение.

DetachableObject.

Объект который может сначала частично отсоединиться от машины (Повиснуть на одном из нескольких возможных суставов), затем отсоединиться совсем, используется в основном для частей тела машины, при отсоединении всем дочерним элементам с DamageableObject наносится максимальный урон.

DetachableObject имеет следующие поля:

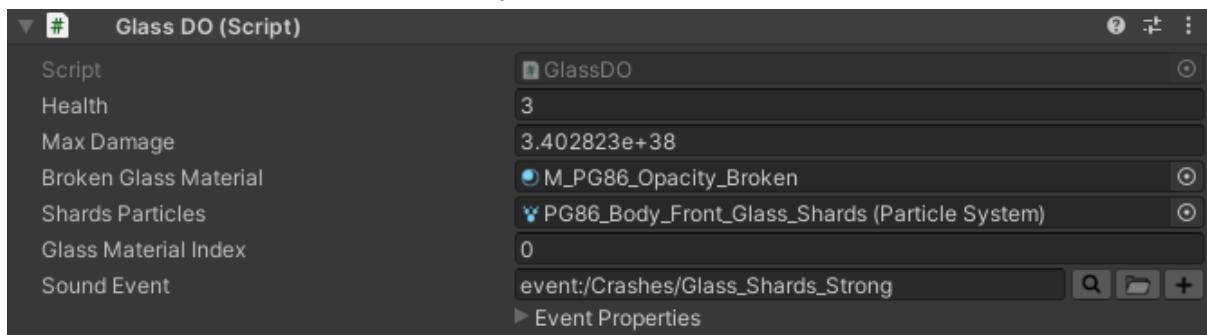


- Mass - Масса детали, эта масса присваивается Rigidbody созданному у данного объекта и отнимается от массы машины.
- Drag - Назначается в соответствующее поле созданного Rigidbody.

- AngularDrag - Назначается в соответствующее поле созданного RigidBody.
- LoseForce - Сила удара при которой объект частично отделится (Повиснет на одном из суставах), при этом создается RigidBody.
- Joints - список Суставов, при частичной потери объекта, создается сустав между этим объектом и машиной с параметрами из списка (Выбирается случайный). К сожалению я не придумал адекватный способ для удобного и наглядного редактирования этих параметров, лучший способ это добавить HingeJoint настроить его как надо и перенести результаты настроек в список Joints, в дальнейшем я постараюсь придумать способ настройки.
- DamageCheckPoints - Точки с которыми происходит проверка на отсоединение при столкновении, в этом списке должна быть хотябы одна точка.
- UseLoseHealth - Флаг отвечающий за накопительный эффект повреждения, если он выключен то для отсоединения детали нужно получить повреждение больше или равно LoseForce за одно столкновение.
- SetNewLayerAfterDetach - Нужно ли назначить новый слой отсоединяемой части. Нужно например для фар, после отсоединения им назначается слой который не сталкивается с машиной (Для того чтобы фара не застряла в машине).

GlassDO.

Об этом компоненте уже было описание в разделе LightObject. Этот объект нужен для логики разбития стекол. Имеет следующие поля:



- BrokenGlassMaterial - Сломанный материал, если нужно отобразить осколки после разрушения, если это поле null то после разрушения объект будет невидим.
- ShardsParticles Система частич запускаемая при разрушении объекта
- GlassMaterialIndex - Индекс материала, на случай если Mesh имеет более одного SubMesh.
- SoundEvent - звук воспроизводимый при разрушении объекта.

MoveableDO.

Объект перемещаемый при столкновении в сторону направления удара (Зависит от силы и дальности удара). Используется в таких объектах как колеса и рулевое колесо.

GameController и PlayerController.

Вспомогательные компоненты для инициализации машин, камеры и UI.

GameController должен быть на каждой сцене, он ищет все автомобили на сцене и дает возможность переключаться между ними, если на сцене нет машин, то создается первая из списка B.GameSettings.AvailableVehicles. Этот класс нужен только для демонстрации ассета. В префабе GameController также находится меню, для переключения между сценами и инструкция по управлению.

PlayerController, в нем находится камера, UI показателей машины и UserInput.

Важно! Вы можете не использовать GameController, для игры достаточно разместить нужные объекты на сцене (Например: UserInput, CameraController, CarStateUI) и назначить им TargetCar, игра запустится без инициализации, вы так же можете создать свой GameController который будет инициализировать PlayerController.

UserInput.

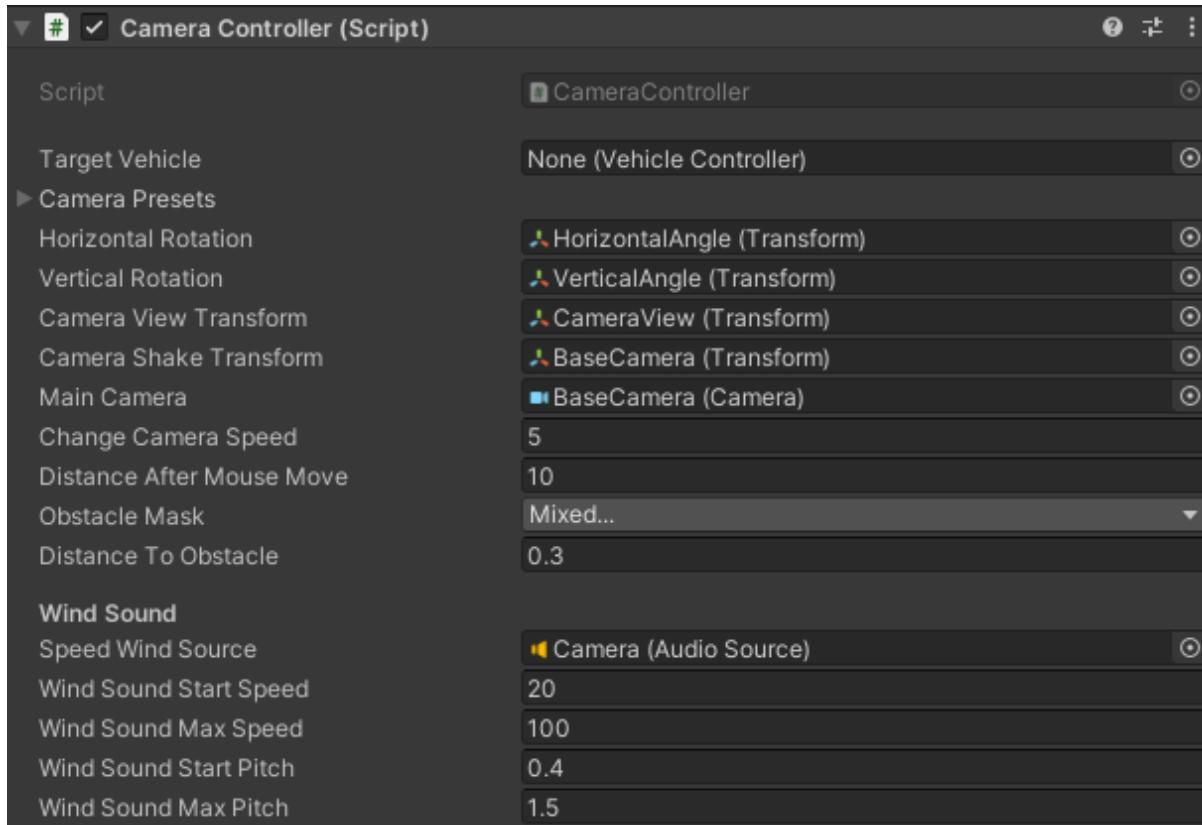
Для управления машиной используется InputManager.

UserInput имеет два поля:

- TargetCar - Управляемый автомобиль, это поле нужно для ручной настройки UserInput (Например: UserInput и машина находятся на сцене, и вам не нужна логика инициализации).
- HorizontalChangeSpeed - Для имитации стика при нажатии на клавиши клавиатуры. До внедрения этой логики машина немножко дергалась при резких поворотах при управлении с клавиатуры.

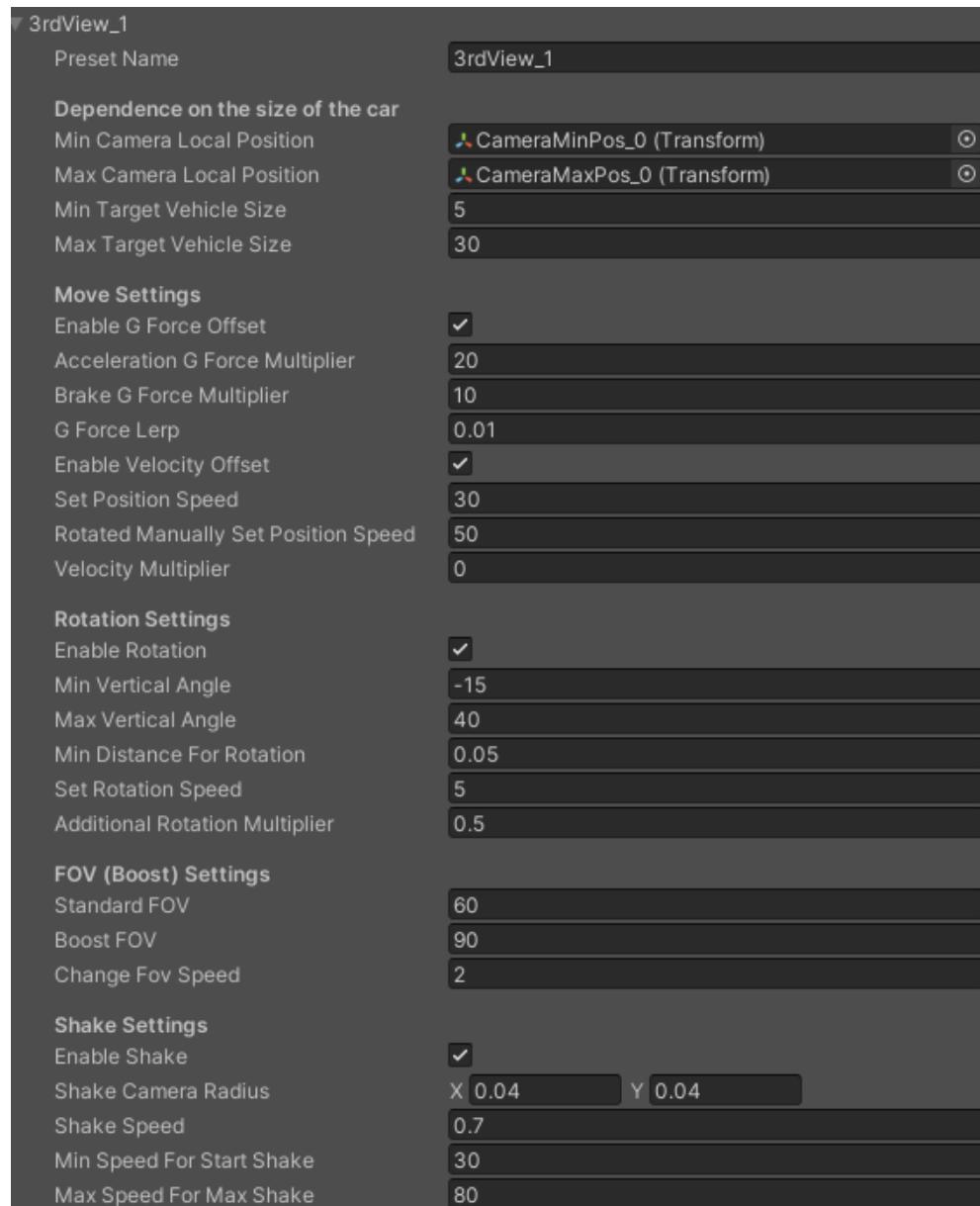
CameraController.

В этом компоненте находится логика управления камерой (слежение за точкой, вращение автоматическое и ручное), переключение между различными видами. Компонент имеет следующие поля:



- TargetVehicle - Автомобиль для слежения, это поле нужно для ручной настройки камеры (Например: Камера и машина находятся на сцене, и вам не нужна логика инициализации).
- CameraPresets - Список типа CameraPreset пресетов камеры, для настроек различных видов. HorizontalRotation - Объект который вращается по горизонтали.
- VerticalRotation - Объект который вращается по вертикали.
- CameraViewTransform - Объект камеры который занимает позицию в точке CameraLocalPosition в зависимости от выбранного вида, должен также быть дочерним объекту VerticalRotation.
- CameraShakeView - Объект которой трястется во время движения, должен быть дочерним объекту CameraViewTransform.
- ChangeCameraSpeed - Скорость смены видов.
- DistanceAfterMouseMove - Если камера была сдвинута вручную, то после преодоления машиной этого расстояния камера будет вращаться автоматически.
- SpeedWindSource - Source ветра во время движения.
- WindSoundStartSpeed - Скорость с которой ветер начинает воспроизводится.
- WindSoundMaxSpeed - Скорость при которой ветер воспроизводится максимально громко.
- WindSoundStartPitch - Pitch при минимальной скорости.
- WindSoundMaxPitch - Pitch при максимальной скорости.

CameraPreset имеет следующие поля:



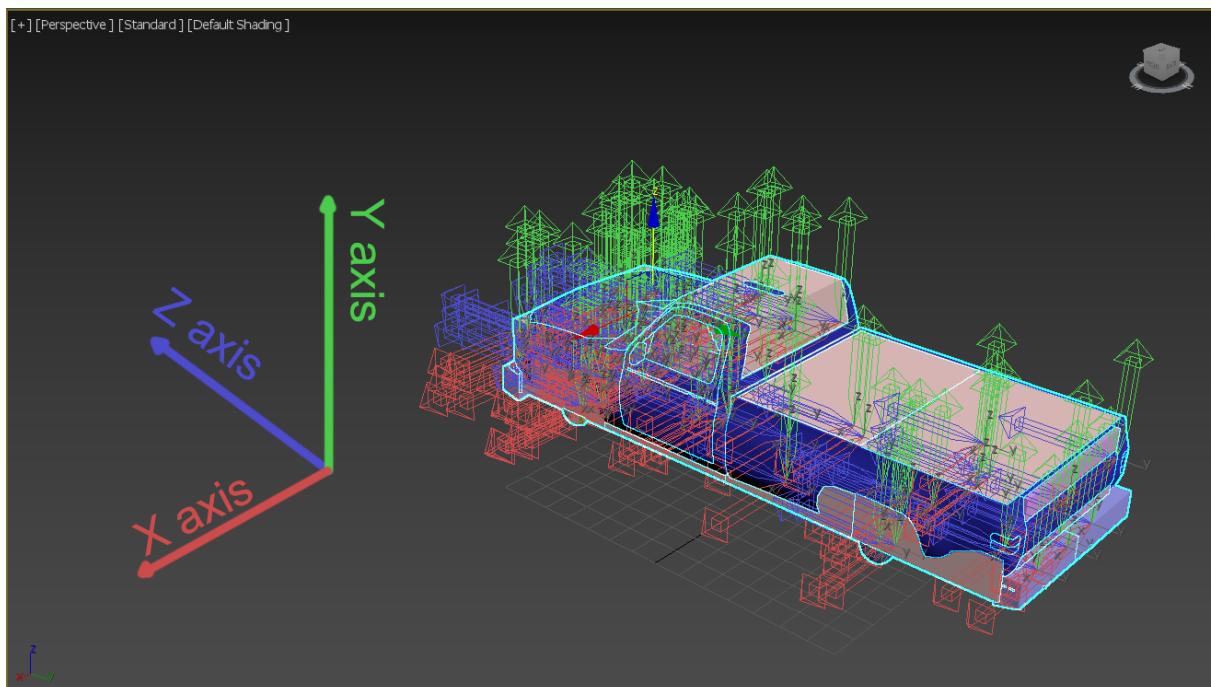
- PresetName - название пресета, для удобства редактирования.
- MinCameraLocalPosition и MaxCameraLocalPosition - в зависимости от размера машины (с прицлом), рассчитывается позиция и вращение камеры. MinCameraLocalPosition и MaxCameraLocalPosition объекты должны быть дочерними объекту VerticalRotation. После инициализации камеры объекты MinCameraLocalPosition и MaxCameraLocalPosition удаляются (Позиции и вращения запоминаются в переменные).
- MinTargetVehicleSize - Минимальный размер машины, если машина равна или меньше минимального размера то позиция и вращение для камеры будет MinCameraLocalPosition.
- MaxTargetVehicleSize - Максимальный размер машины, если машина равна или больше максимального размера то позиция и вращение для камеры будет MaxCameraLocalPosition.
- EnableGForceOffset - Включает смещение камеры во время ускорения/торможения.
- AccelerationGForceMultiplier - Сила смещения камеры (Назад) при ускорении.
- BrakeGForceMultiplier - Сила смещения камеры (Вперед) при торможении.
- GForceLerp - Интерполяция смещения для плавности.
- EnableVelocityOffset - Включает смещение в зависимости от текущей скорости.
- SetPositionSpeed - Скорость перемещения камеры.

- VelocityMultiplier - Сила опережения точки слежения (Например: Если этот параметр будет больше 0, то камера будет следить за точкой перед машиной в зависимости от скорости машины).
- EnableRotation - Включает/выключает вращение камеры (Ручное и автоматическое).
- MinVerticalAngle - Минимальный угол наклона камеры по вертикали.
- MaxVerticalAngle - Максимальный угол наклона камеры по вертикали.
- MinDistanceForRotation - Минимальная дистанция пройденная автомобилем за один кадр для обнаружения автоматического вращения (Нужно для того чтобы камера не вращалась хаотично пока машина стоит на месте).
- SetRotationSpeed - Скорость вращения камеры.
- AdditionalRotationMultiplier - Добавочное вращение камеры в сторону дрифта.
- StandardFOV - FOV использующийся по умолчанию.
- BoostFOV - FOV применяющийся во время использования буста.
- ChangeFovSpeed - Скорость изменения FOV при включении/выключении буста.
- EnableShake - Включает тряску камеры во время движения машины.
- ShakeCameraRadius - Радиус в котором будет трястись камера (Точки выбираются случайно).
- ShakeSpeed - Скорость тряски.
- MinSpeedForStartShake - Скорость на которой начинается тряска камеры.
- MaxSpeedForStartShake - Скорость на которой тряска камеры максимальна.

Создание автомобиля (CreateCarWindow)

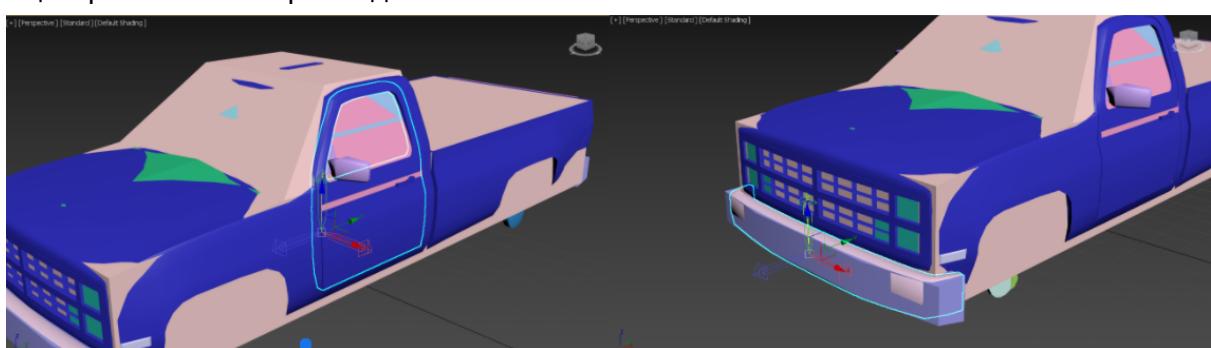
Подготовка модели автомобиля.

- 1) Автомобиль и все его части должны иметь правильный масштаб (примерно как в реальной жизни) transform.localScale всех частей должен быть равен Vector3.one (1, 1, 1), иначе VehicleDamageController (Повреждения) будет работать не корректно, его придется перенастраивать.
- 2) Автомобиль и все его части должны иметь правильное направление опорных точек, ОсьZ - ($z>0$)перед/($z<0$)зад, ОсьX - ($x>0$)право/($x<0$)лево, ОсьY - ($y>0$)верх/($y<0$)низ.

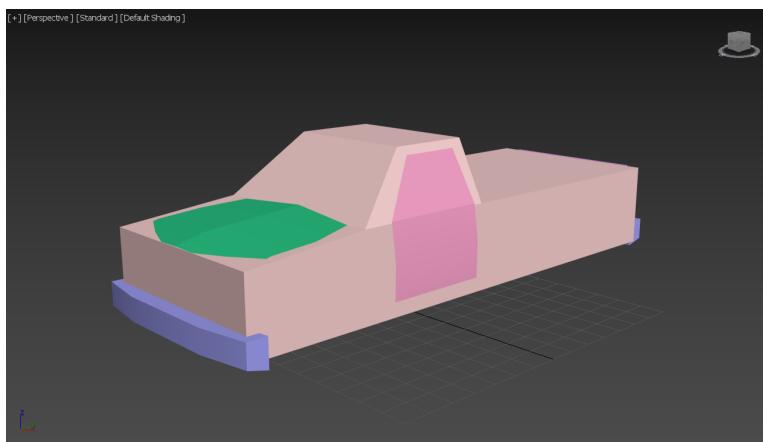


Если направления опорных точек некорректные могут возникнуть проблемы с определением колес, отсоединяемыми частями и т.д. так же могут появиться неизвестные мне проблемы.

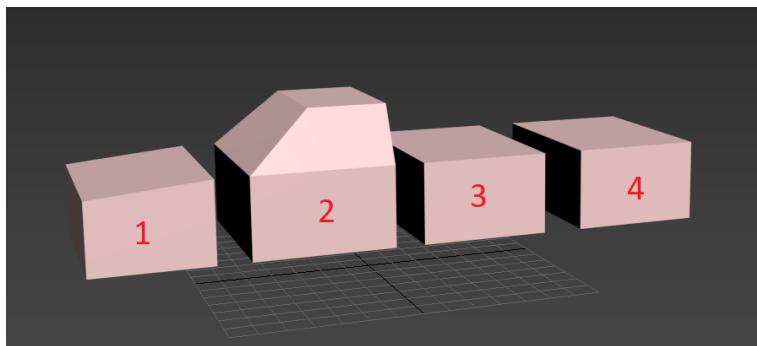
- 3) Отсоединяемые части желательно иметь правильное расположение опорных точек, например дверь должна иметь опорную точку на оси открытия двери (Для удобства настройки), бампер должен иметь опорную точку примерно в центре масс бампера и т.д.



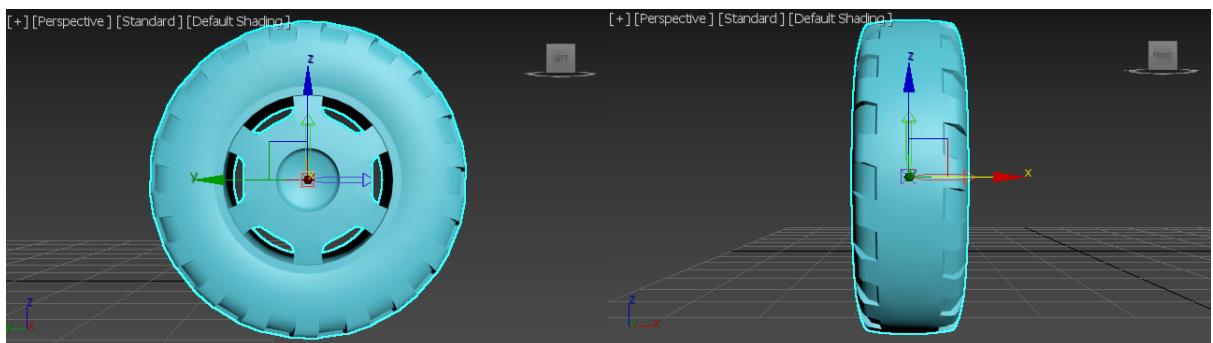
- 4) Коллайдеры для всех частей автомобиля по возможности должны быть простые (Должны иметь как можно меньше вершин).



- 5) Все коллайдеры должны быть выпуклые, поэтому вы можете разделить коллайдеры там где вам не нужны выпуклые, например коллайдер Body:

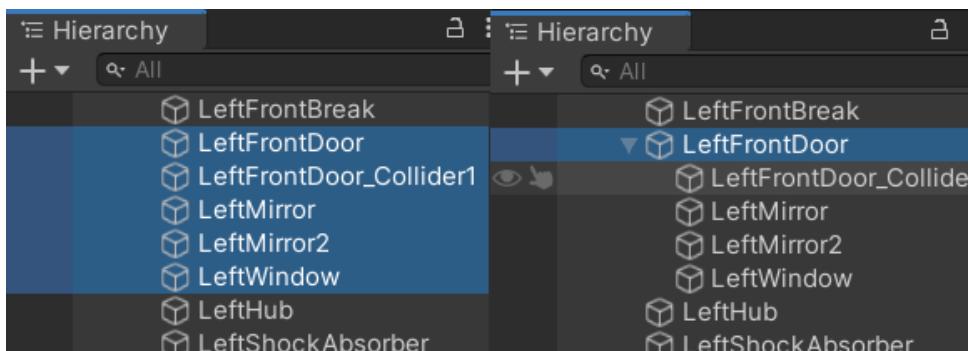


- 6) Точка опоры колеса должна быть по центру, в этой точке будет находиться WheelCollider.

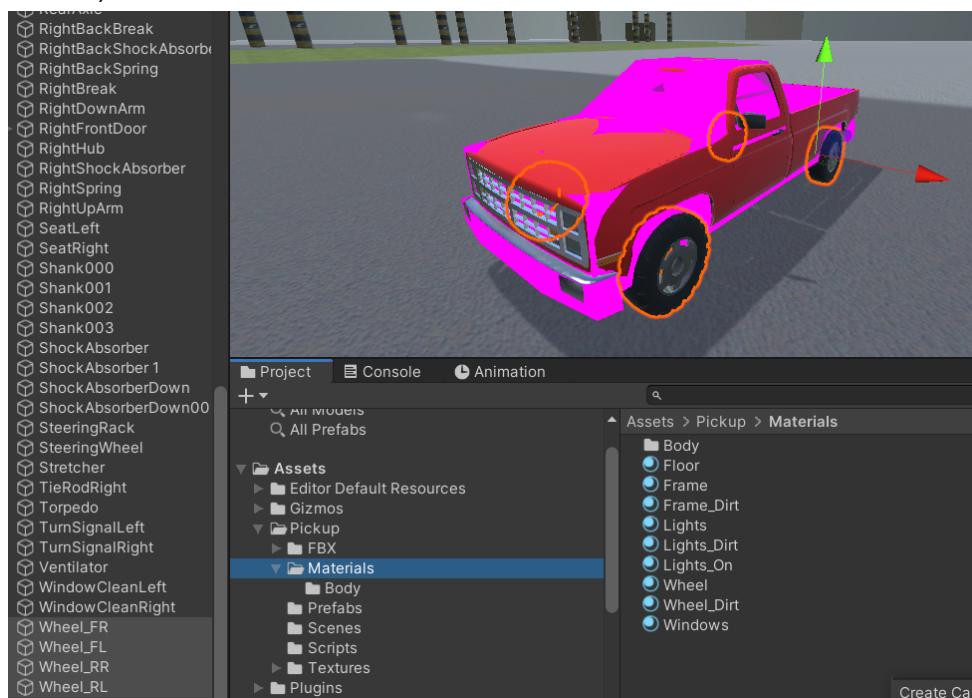


Подготовка префаба (Объекта) для создания автомобиля.

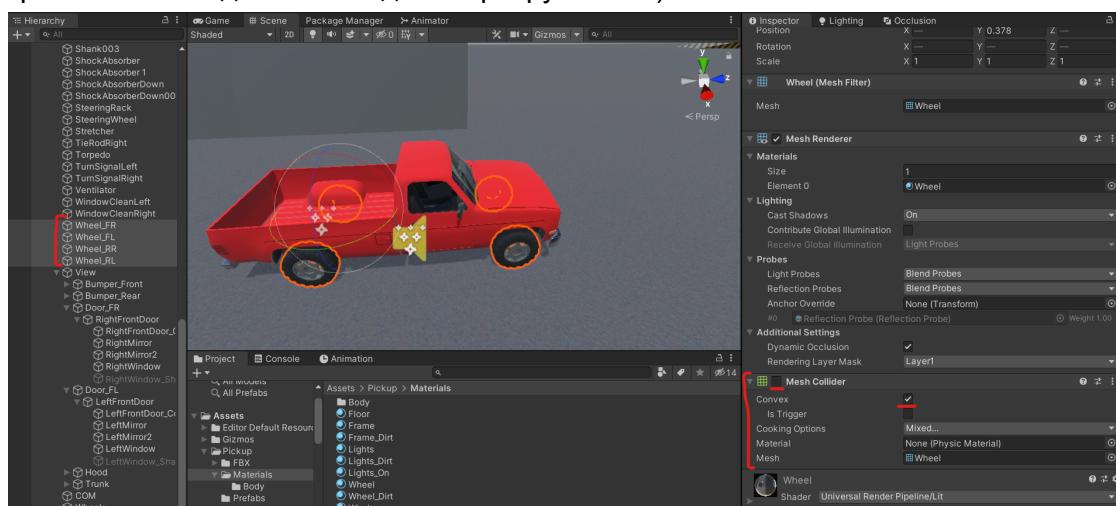
- 1) Для удобства создания вы можете сделать все объекты относящиеся к одной части дочерними этой части (например зеркало двери, стекло двери сделать дочерними этой двери), чтобы не искать объекты одной части по всей иерархии.



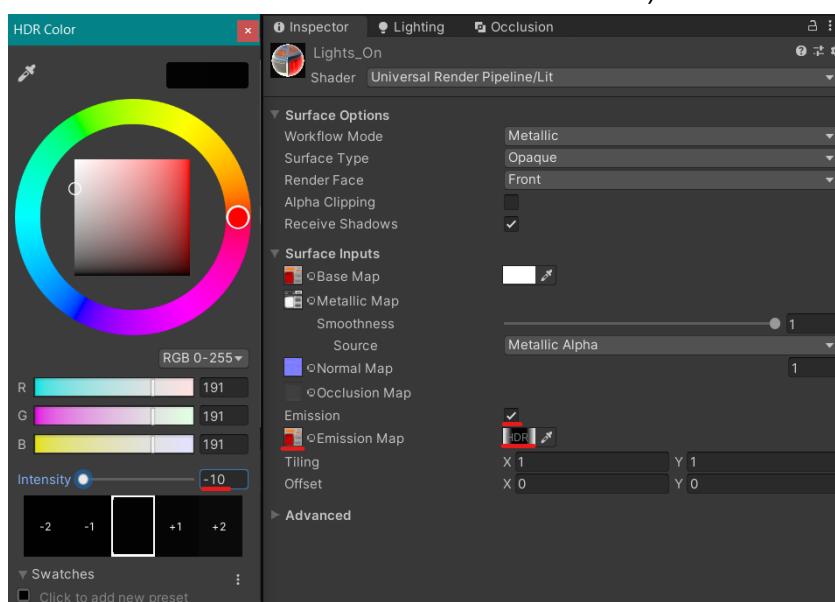
- 2) Установить колеса в нужные места (В случае если у модели только одно колесо).



- 3) Добавить колесам MeshCollider и выключить его объект (Чтобы колесо не провалилось под землю когда оно разрушается).



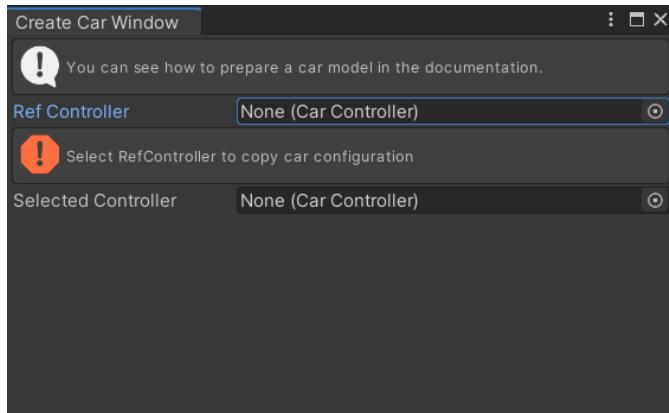
- 4) Создать дубликат материала Lights и сделать его светящимся (Если будут использоваться световые сигналы автомобиля).



Окно создания автомобиля.

Есть видео с примером создания автомобиля [How to create a car in UVC](#)

Для открытия окна выполните команду: "Window/Perfect Games/Create Car Window".

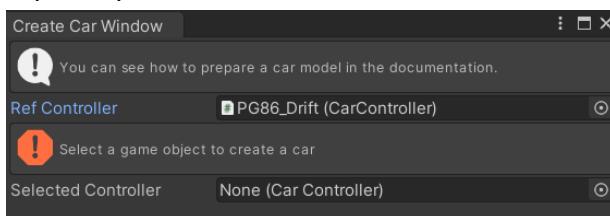


RefController - Ссылка на существующую машину с которой будут копироваться параметры двигателя, управления, трансмиссии, колес. При создании нового автомобиля желательно не оставлять это поле пустым.

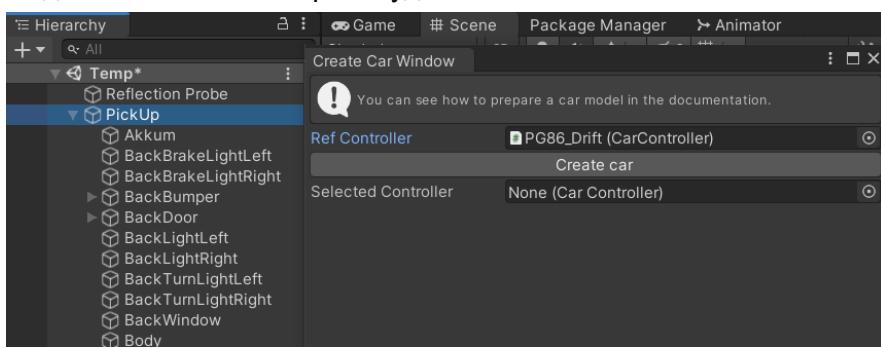
SelectedController - Редактируемый автомобиль, автоматически устанавливается при нажатии на кнопку "Create car", также можно указать уже созданный автомобиль для редактирования.

Создание автомобиля:

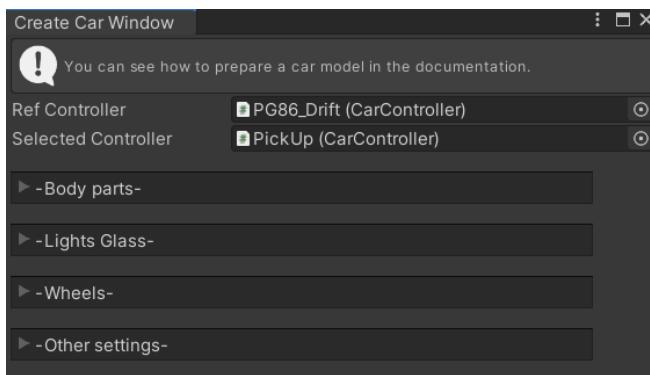
- 1) Указываем существующий автомобиль с которого будут копироваться параметры.



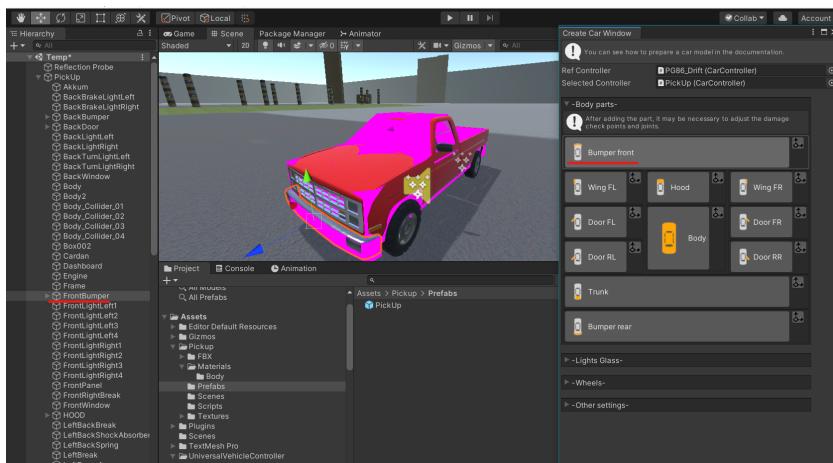
- 2) Выделяем объект который будет автомобилем.



- 3) Нажимаем кнопку “Create car”. После нажатия на кнопку автомобилю добавятся необходимые компоненты, объекты и префабы.



- 4) Настраиваем все части автомобиля (Вкладка “Body parts”), для этого выделяем объект (Объекты) части и нажимаем соответствующую кнопку.



Если объектов несколько, то опорная точка может быть не в том месте, для исправления этого выберите объект с правильной опорной точки и нажмите на

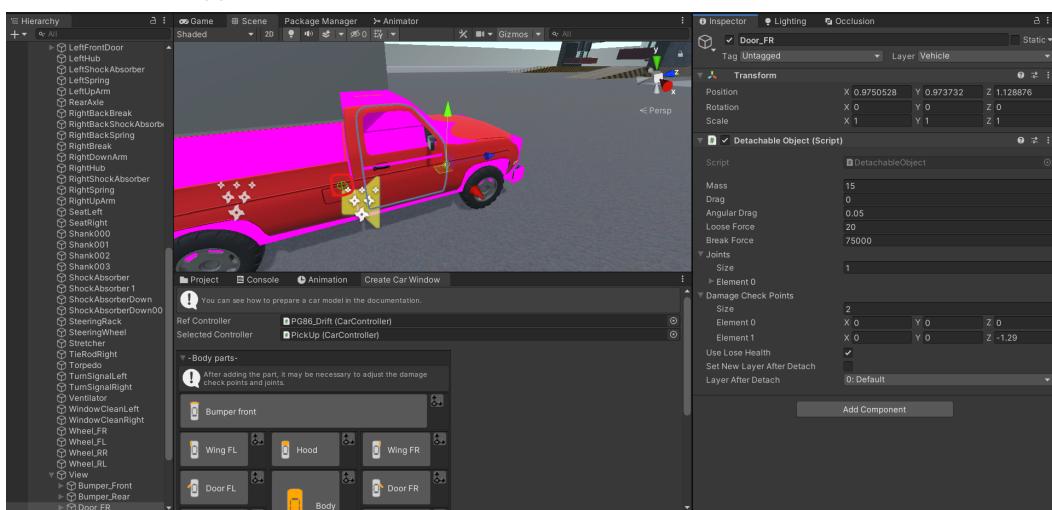


кнопку , после этого опорная точка части будет в позиции опорной точки выделенного объекта.

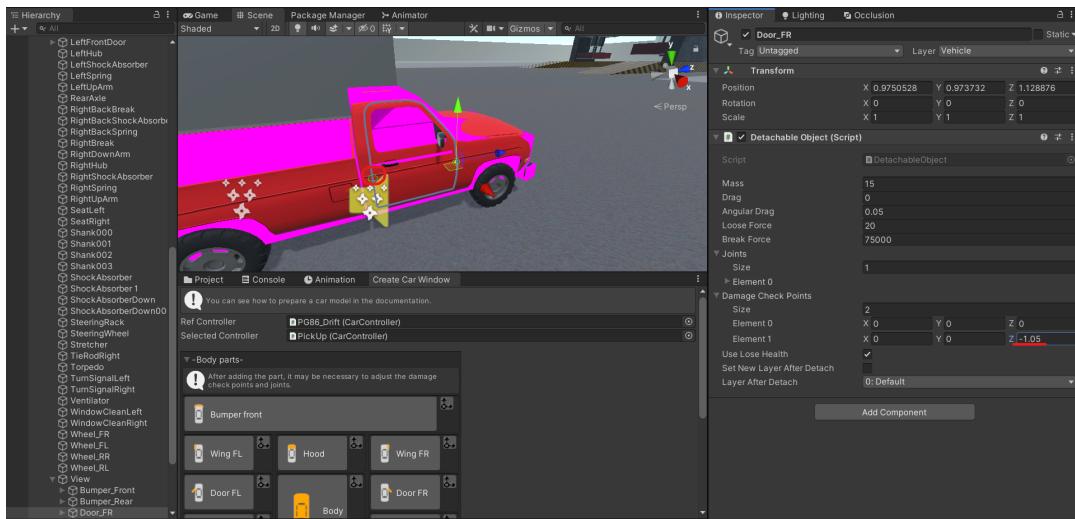
После создания части возможно потребуется настройка **DetachableObject** **DamageCheckPoints** и **Joints**.

Примеры:

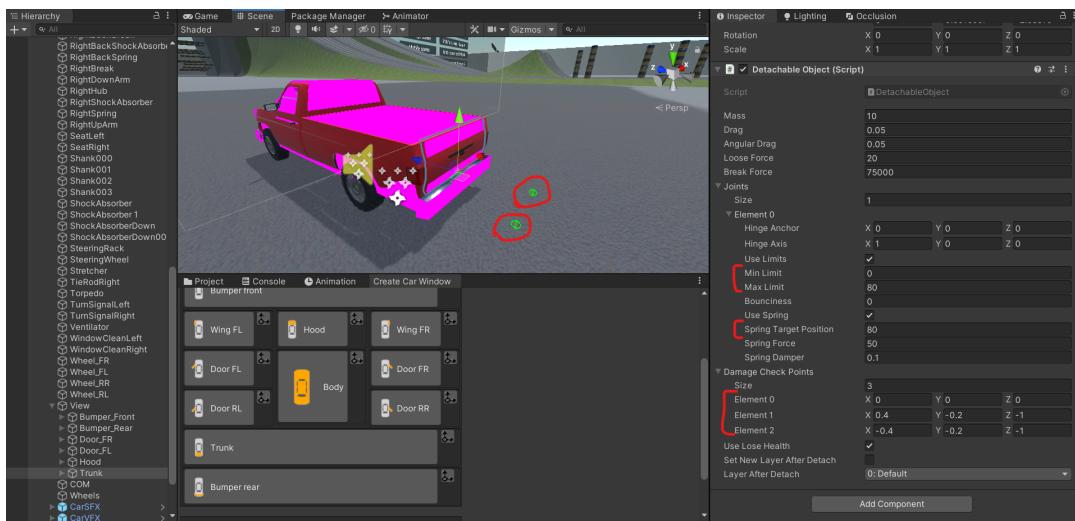
Door после создания



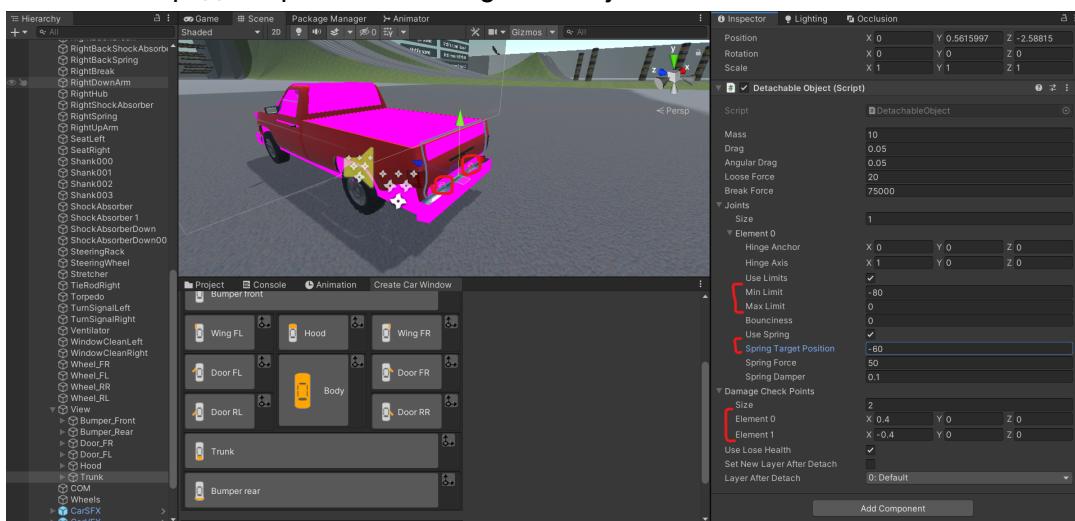
Door после редактирования DamageableObject



Trunk после создания



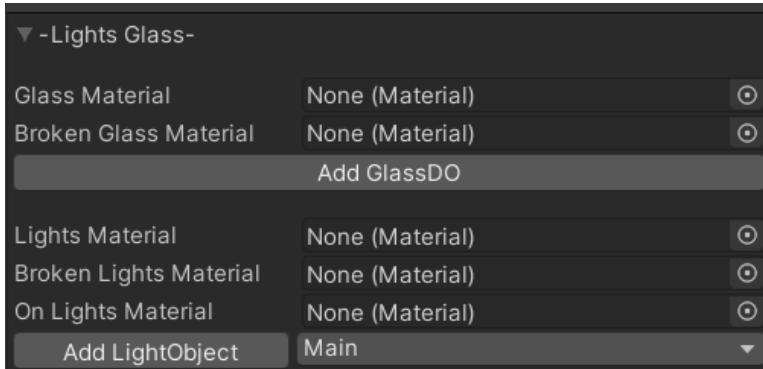
Trunk после редактирования DamageableObject



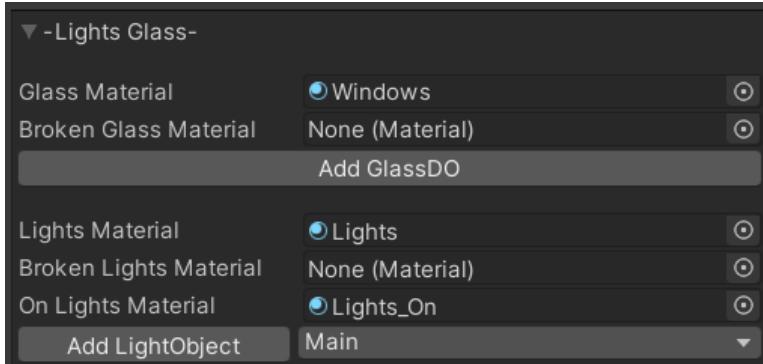
По аналогии нужно создать и изменить все имеющиеся части автомобиля (Если каких то частей машины в вашей модели нет, то просто пропустите их).

- 5) Все оставшиеся объекты можно выделить и нажать кнопку "Body".
- 6) После создания всех частей нужно создать коллайдеры, для этого нажмите на кнопку "Convert all MeshColliders", если у вас префикс коллайдеров отличается то вам нужно изменить "ColliderPrefix".

7) Стекла и световые приборы.

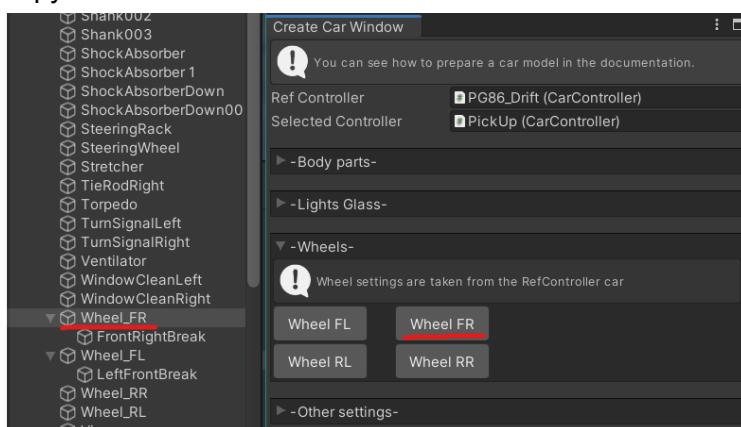


Для начала нужно указать материалы стекол и световых приборов.

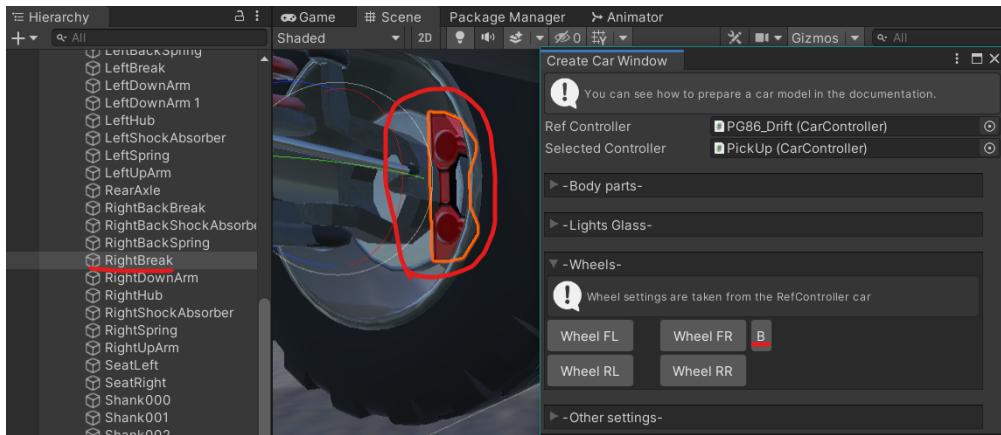


Broken материал нужен для обображения объекта после уничтожения, например торчащие осколки стекл. Если не указать **Broken** материалы то при уничтожении этот объект будет исчезать.

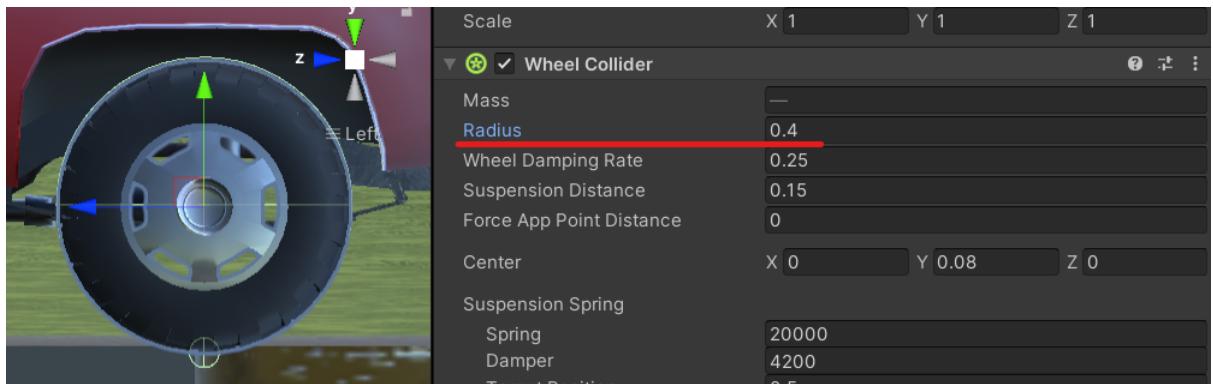
- 8) Выделяем стела и нажимаем кнопку “Add GlassDO”, при этом создадутся частицы осколков стекла, которые будут появляться при уничтожении объекта, при необходимости вы можете настроить направление испускаемых частиц, по умолчанию направление рассчитывается от центра выбранного автомобиля.
- 9) По аналогии со стеклами выбираем все световые сигналы одного типа, выбираем тип и нажимаем кнопку “Add LightObject”, нужно повторить это действие для каждого типа светового сигнала (Main, Turn left, Turn right, brake, reverse).
- 10) После создания световых сигналов можно добавить источник света нужным объектам (Например для головного света или для добавления источника света невидимому объекту без MeshRenderer).
- 11) Выбираем колесо и все его части (Диск, покрышку, тормозной диск), важно чтобы точка опоры всех объектов была в одной позиции, если это не так то лучше все объекты сделать дочерними колесу и выделить только колесо. Нажимаем соответствующую кнопку во вкладке Wheels, все настройки колеса берутся из RefController.



После создания колеса появится возможность добавить ему тормозной суппорт, для этого выделите нужный объект и нажмите на кнопку “B” возле соответствующего колеса.

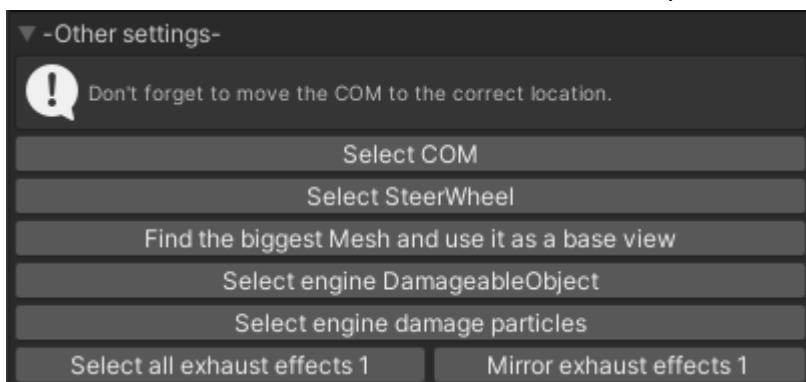


Также нужно изменить радиус WheelCollider если он отличается



Остальные колеса настраиваются аналогично.

12) Вкладка “Other” в ней находятся остальные настройки.



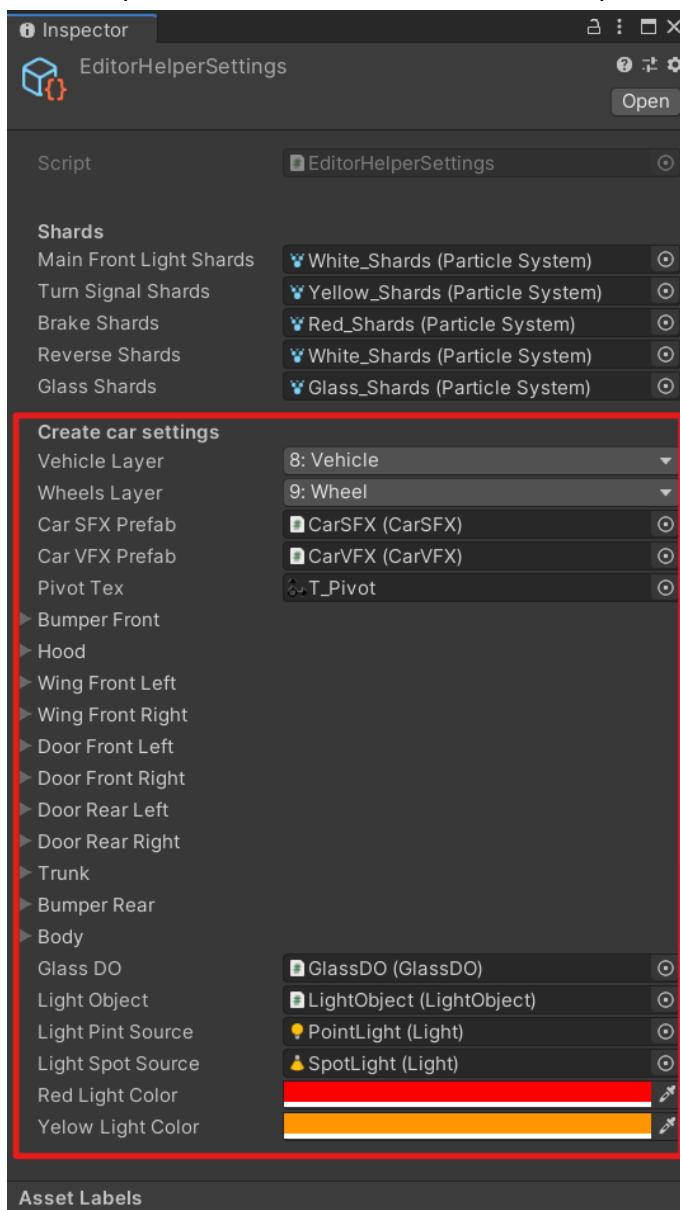
Кнопка “Select COM” делает активным объект COM, переместите его как вам нужно, от этого зависит поведение автомобиля, желательно чтобы COM находился примерно в центре автомобиля и немного ниже осей колес (вы можете посмотреть положение COM у других автомобилей в ассете).

- 13) Выберите рулевое колесо (Если оно есть у вашей модели) и нажмите кнопку “Select SteerWheel”, после этого рулевое колесо будет вращаться по оси Z в сторону поворота.
- 14) Нажмите кнопку “Find the biggest Mesh and use it as a base view”, будет найден самый большой MeshRenderer и добавится в список CarController.BaseViews, при необходимости вы сами можете определить базовые объекты, это необходимо для определения видимости машины, когда машина не видна не работают эффекты и вращение колес.
- 15) Кнопка “Select engine DamageableObject” выделяет или создает в случае отсутствия объекта отвечающего за повреждения двигателя. От расположения этого объекта зависит сила повреждения двигателя при ударах.

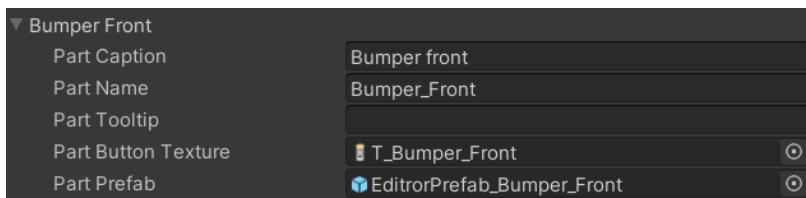
- 16) Кнопка “Select engine damage particles” выделяет все системы частиц повреждения двигателя, если эти эффекты есть в префабе CarVFX.
- 17) Кнопка “Select all exhaust effects n” выбирает все эффекты трубы для удобства изменения их положения.
- 18) Кнопка “Mirror exhaust effects n” делает дубликат эффектов в зеркальной позиции по X.

Настройки создания автомобиля.

Все настройки находятся в ассете EditorHelperSettings



- Shards - ParticleSystems для создания эффекта осколков для стекол и фар. Создаются автоматически при создании стекол и фар.
- VehicleLayer - Слой который будет назначаться создаваемому автомобилю.
- WheelsLayer - Слой который будет назначаться колесам создаваемого автомобиля.
- CarVFXPrefab - Перфаб визуальных эффектов который будет добавляться при создании автомобиля.
- CarSFXPrefab - Перфаб звуковых эффектов который будет добавляться при создании автомобиля.
- PivotTex - Текстура обозначения опорной точки (Нужна только для редактора).
- BumperFront (И остальные части) - Поле типа CarPart, в нем хранится информация о создаваемой части. CarPart имеет следующие поля:



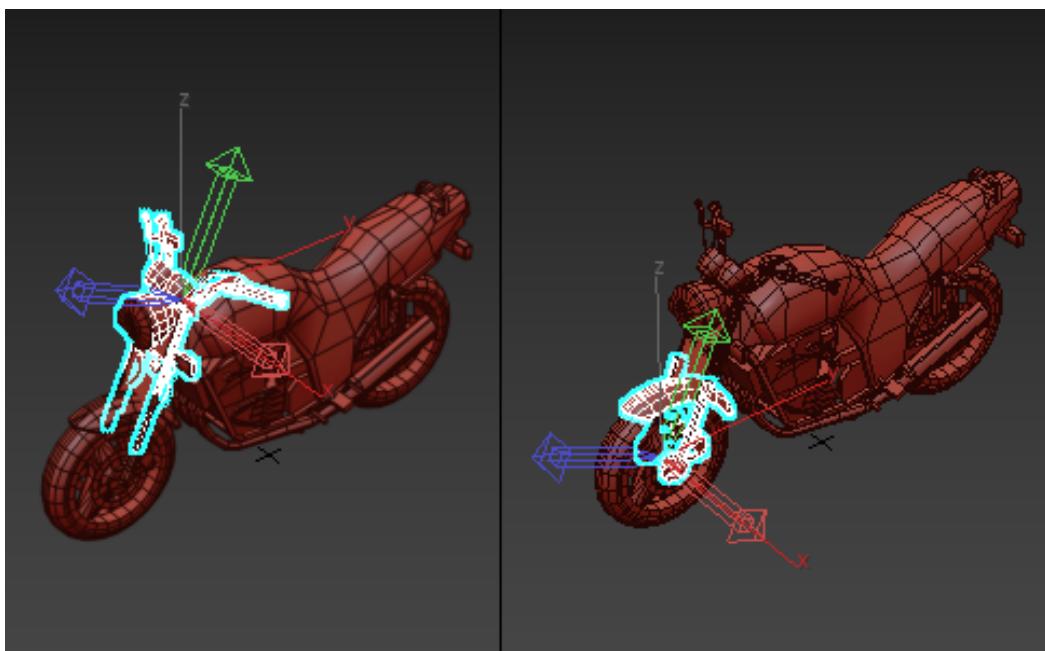
- PartCaption - Название части которое будет отображаться на кнопке.
- PartName - Имя части, так будет называться создаваемый объект в иерархии.
- PartTooltip - Tooltip для кнопки части.
- PartButtonTexture - Текстура для кнопки части.
- PartPrefab - Префаб части который будет дублироваться для выбранной части, все подобные префабы хранятся в папке "Assets\UniversalVehicleController\Editor\EditorPrefabs", вы можете изменять их по своему желанию и создавать машины используя собственные префабы.
- GlassDO - Префаб с настройками объекта стекла (Здоровье, звук и тп.).
- LightObject - Префаб с настройками объекта света (Здоровье, звук и тп.).
- LightPointSource - Префаб с источником света типа Point;
- LightSpotSource - Префаб с источником света света типа Spot (Для головного света);
- RedLightColor - Цвет для источника света, выбирается в зависимости от типа светового сигнала (Тормоз, задний главный свет).
- YellowLightColor - Цвет для источника света, выбирается в зависимости от типа светового сигнала (Левый поворот, Правый поворот).

Создание байка (CreateBikeWindow)

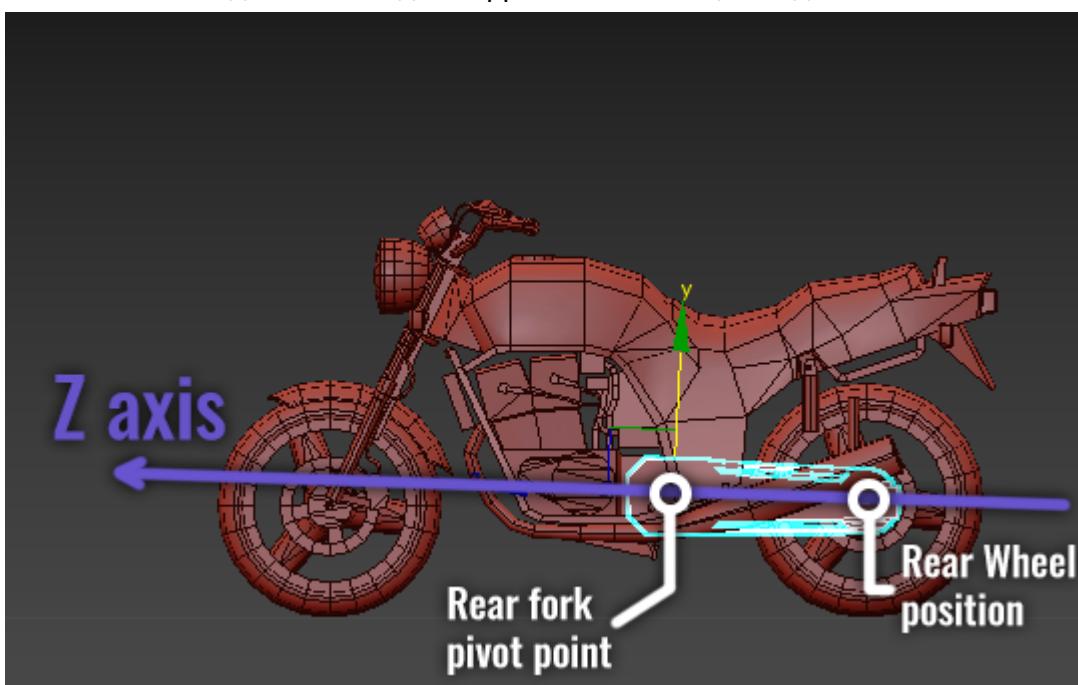
Подготовка модели байка.

Модель байка настраивается по аналогии с моделью автомобиля, вы можете посмотреть 1 - 6 в разделе “Подготовка модели автомобиля”.

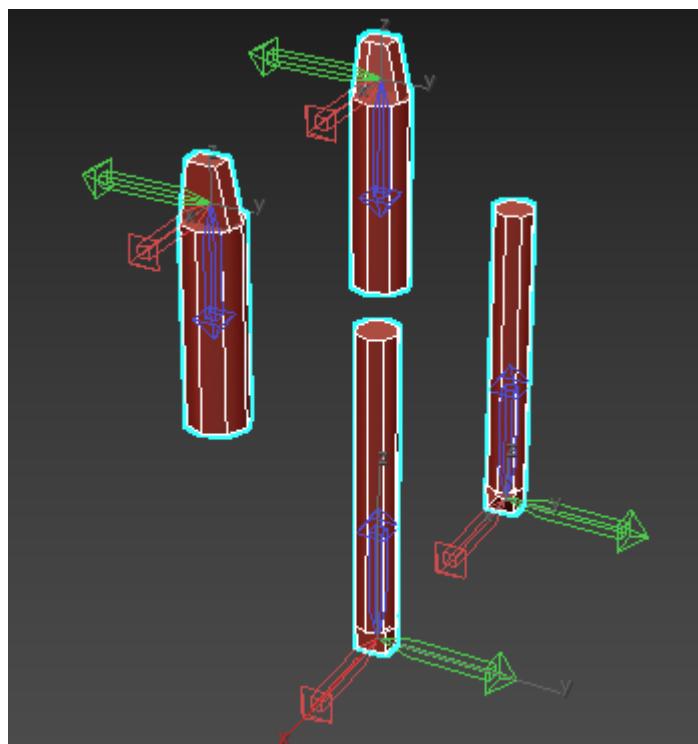
- 7) Вращение опорной точки руля байка и передней вилки байка должны быть одинаковы для корректной анимации подвески. Желательно чтобы ОсьY совпадала с осью руля байка, если это не так то для оси руля байка можно задать угол вручную. Желательно чтобы позиция опорной точки руля должна совпадать с осью поворота руля байка. Позиция опорной точки должна совпадать с позицией колеса.



- 8) Позиция опорной точки задней вилки должна быть в оси крепления задней вилки, направление опорной точки должно быть таким чтобы заднее колесо было на ОсьZ задней вилки, для корректной анимации подвески.

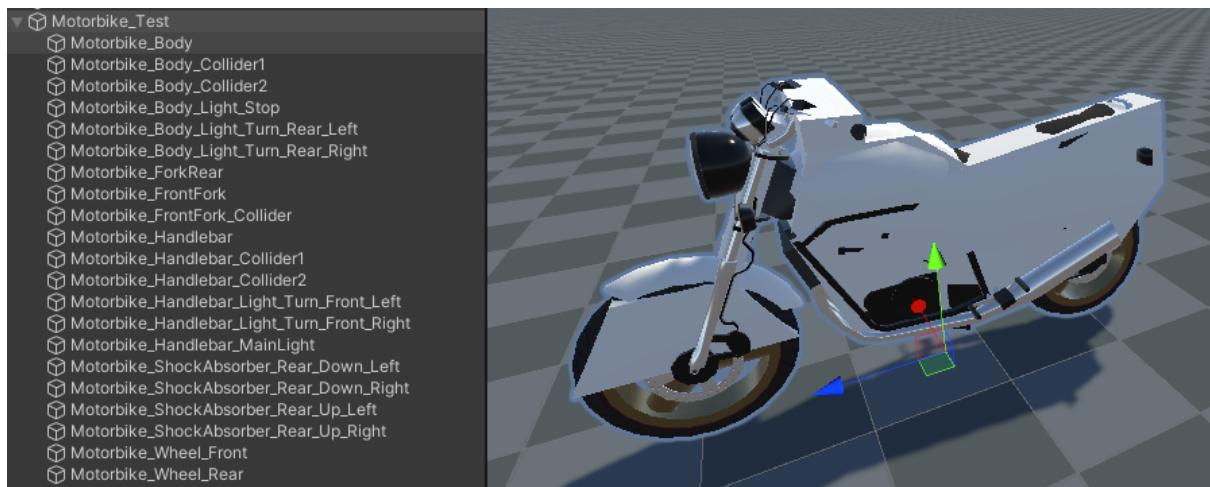


- 9) Если у вашей модели есть амортизаторы то у них ОсьZ точки опоры должна совпадать с осью амортизаторов, ОсьZ должна быть направлена в сторону объекта слежения.



Подготовка префаба (Объекта) для создания байка.

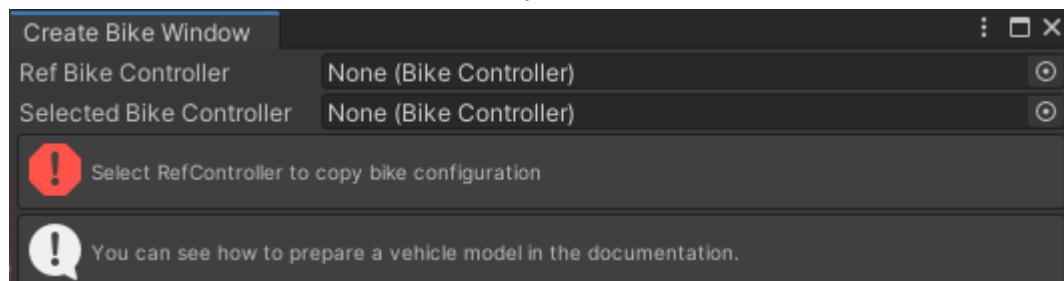
Так как у байка не много частей, то подготовка не требуется, можно не изменять иерархию объектов.



Окно создания байка.

Есть видео с примером создания байка [How to create a bike in UVC](#)

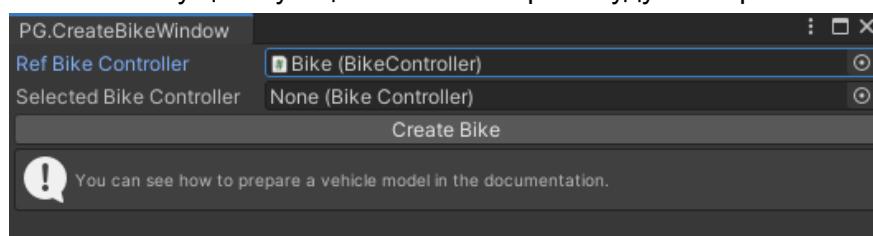
Для открытия окна выполните команду: "Window/Perfect Games/Create Bike Window".



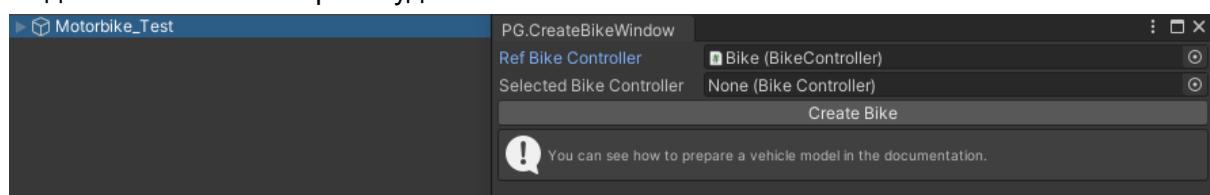
RefBikeController - Ссылка на существующий байк с которого будут копироваться параметры двигателя, управления, трансмиссии, колес, настройки байка. При создании нового байка желательно не оставлять это поле пустым.
SelectedBikeController - Редактируемый байк, автоматически устанавливается при нажатии на кнопку “Create bike”, также можно указать уже созданный байк для редактирования.

Создание байка:

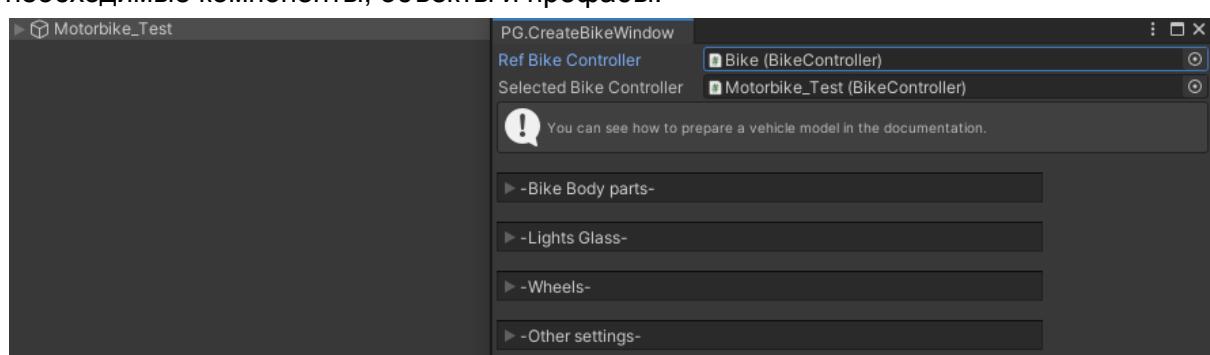
Указываем существующий байк с которого будут копироваться параметры.



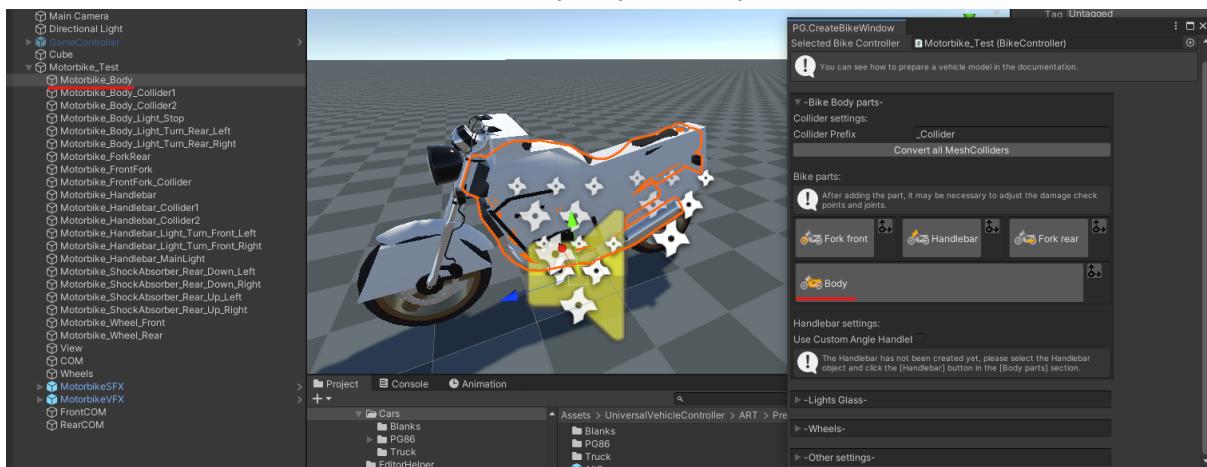
Выделяем объект который будет новым байком.



Нажимаем кнопку “Create bike”. После нажатия на кнопку байку добавятся необходимые компоненты, объекты и префабы.



Настраиваем все части байка (Вкладка “Body parts”), для этого выделяем объект (Объекты) части и нажимаем соответствующую кнопку.



Если объектов несколько, то опорная точка может быть не в том месте, для исправления этого выберите объект с правильной опорной точки и нажмите на кнопку , после этого опорная точка части будет в позиции опорной точки выделенного объекта.

Если у вашей модели есть отдельные сетки для коллайдеров то нажмите на кнопку “Convert all MeshColliders”, если у вас префикс коллайдеров отличается то вам нужно изменить “ColliderPrefix”.

!!!ВАЖНО!!! Коллайдеры обязательны для всех транспортных средств. Если у вашей модели нет сеток коллайдеров то можете:

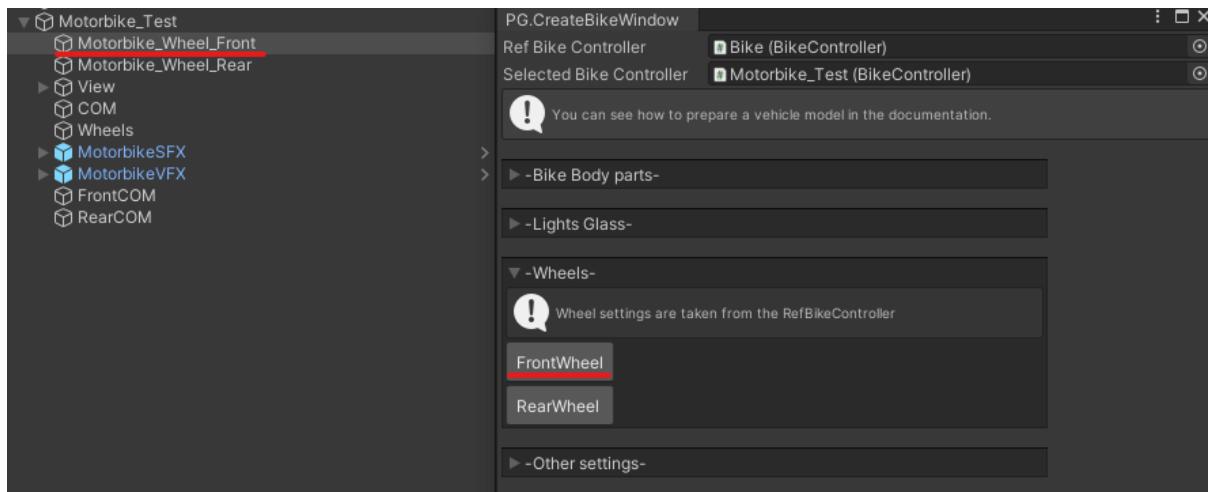
- Попробовать использовать существующую сетку части (Если сетка имеет много полигонов то может возникнуть проблема с производительностью).
- Использовать примитивные коллайдеры (Например box collider), в таком случае коллайдер не будет изменять форму при повреждениях.

Рекомендую использовать сетки специально подготовленные для коллайдеров (Можете посмотреть как это сделано в стандартных моделях UVC).

(Вкладка “Lights Glass”) Стекла и световые приборы настраиваются по аналогии с автомобилем.

(Вкладка “Колеса”)

Выбираем колесо и все его части (Диск, покрышку), важно чтобы точка опоры всех объектов была в одной позиции, если это не так то лучше все объекты сделать дочерними колесу и выделить только колесо. Нажимаем соответствующую кнопку во вкладке Wheels, все настройки колеса берутся из RefBikeController.



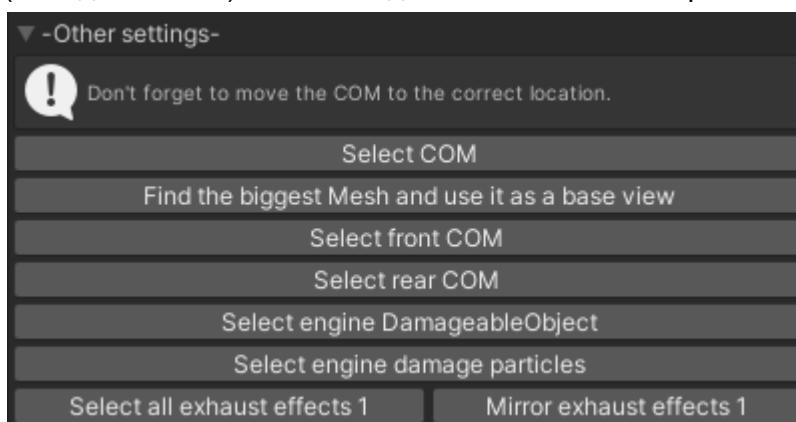
Также нужно изменить радиус WheelCollider если он отличается



!!!ВАЖНО!!! WheelColliders и WheelViews находятся в разных объектах (Для имитации подвески байка).

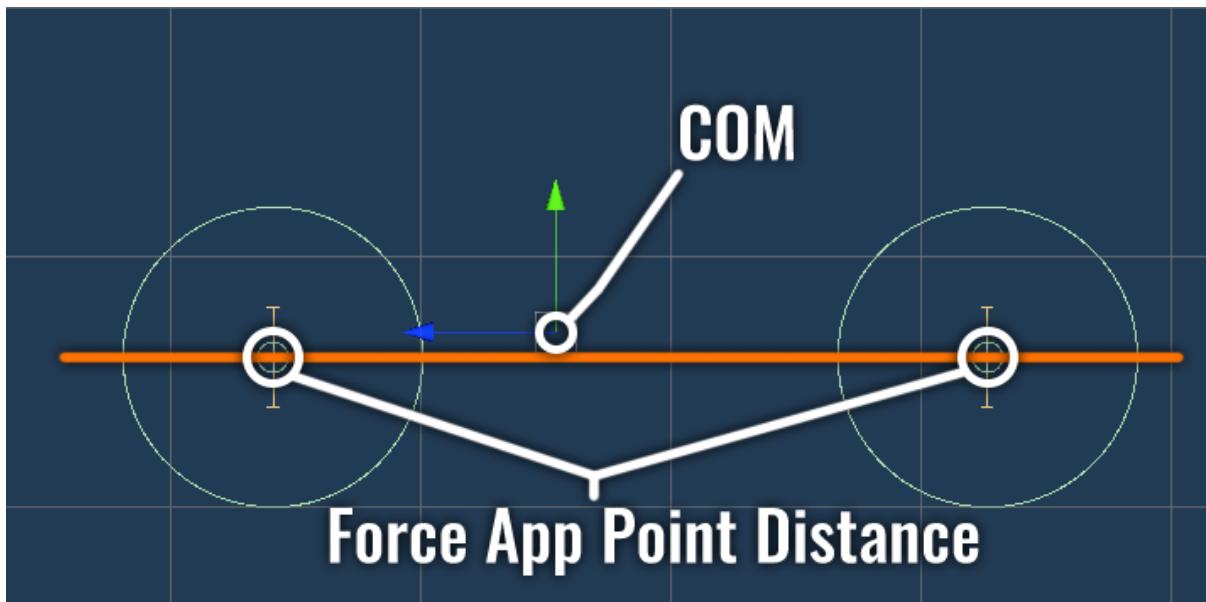
Остальные колеса настраиваются аналогично.

(Вкладка “Other”) в ней находятся остальные настройки.



Кнопка “Select COM” делает активным объект COM, переместите его как вам нужно, от этого зависит поведение автомобиля, желательно чтобы СОМ находился примерно в центре автомобиля и немного ниже осей колес (вы можете посмотреть положение СОМ у других автомобилей в ассете).

!!!Внимание!!! Позиция COM сильно влияет на поведение байка. Чем ближе позиция COM к линии между ForceAppPointDistance колес, тем стабильнее ведет себя байк при изменении RB.Velocity (Ускорение, торможение, повороты).



Нажмите кнопку “Find the biggest Mesh and use it as a base view”, будет найден самый большой MeshRenderer и добавится в список BikeController.BaseViews, при необходимости вы сами можете определить базовые объекты, это необходимо для определения видимости байка, когда байк не виден не работают эффекты и вращение колес.

Кнопка “Select engine DamageableObject” выделяет или создает в случае отсутствия объекта отвечающего за повреждения двигателя. От расположения этого объекта зависит сила повреждения двигателя при ударах.

Кнопка “Select engine damage particles” выделяет все системы частиц повреждения двигателя, если эти эффекты есть в префабе BikeVFX.

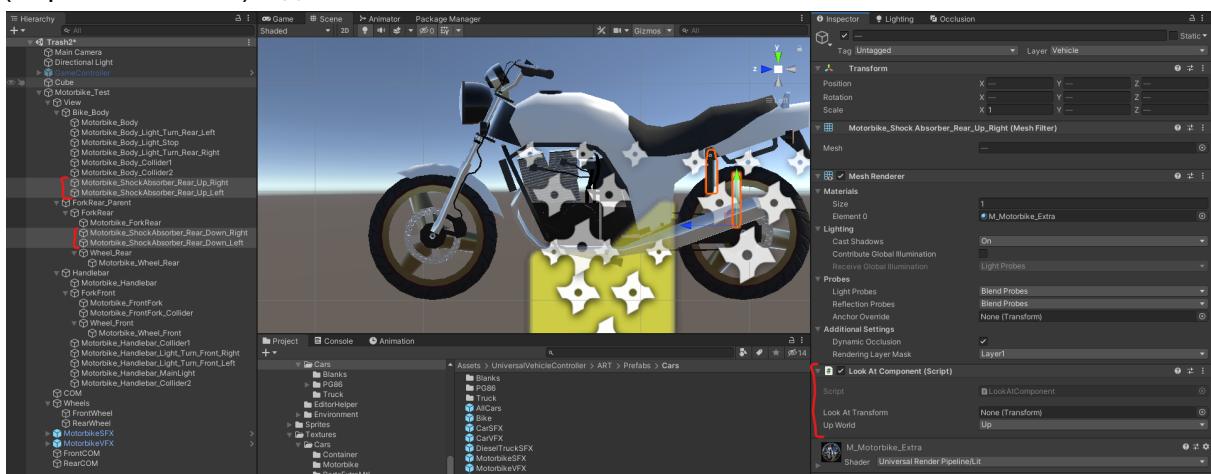
Кнопка “Select all exhaust effects n” выбирает все эффекты трубы для удобства изменения их положения.

Кнопка “Mirror exhaust effects n” делает дубликат эффектов в зеркальной позиции по X.

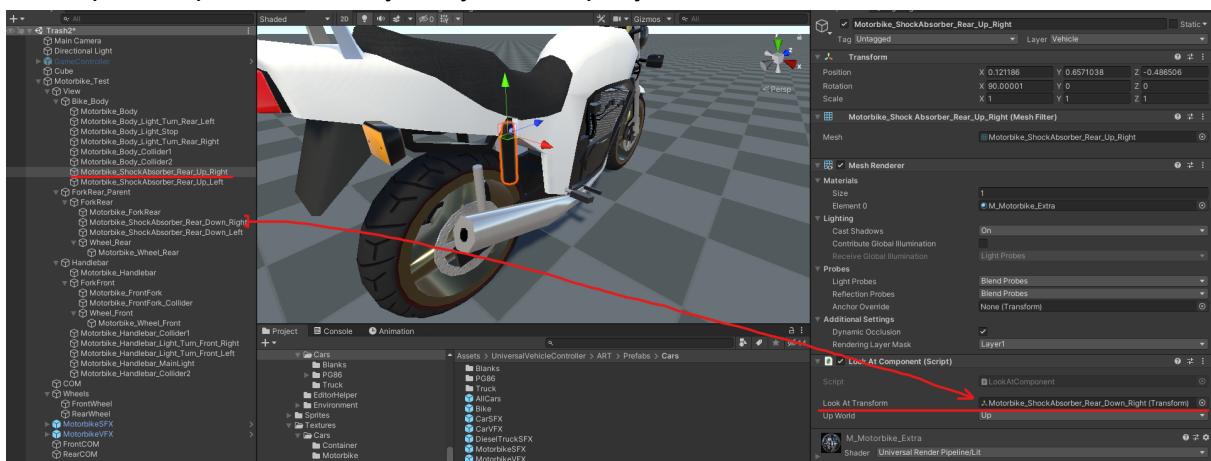
LookAtComponent

Для анимации подвески байка (Амортизаторы) добавлен LookAtComponent. Корректная работа амортизатора (Правильное расположение) зависит от правильной настройки Точек опор амортизаторов, как настроить правильно описано в разделе “Подготовка модели байка”.

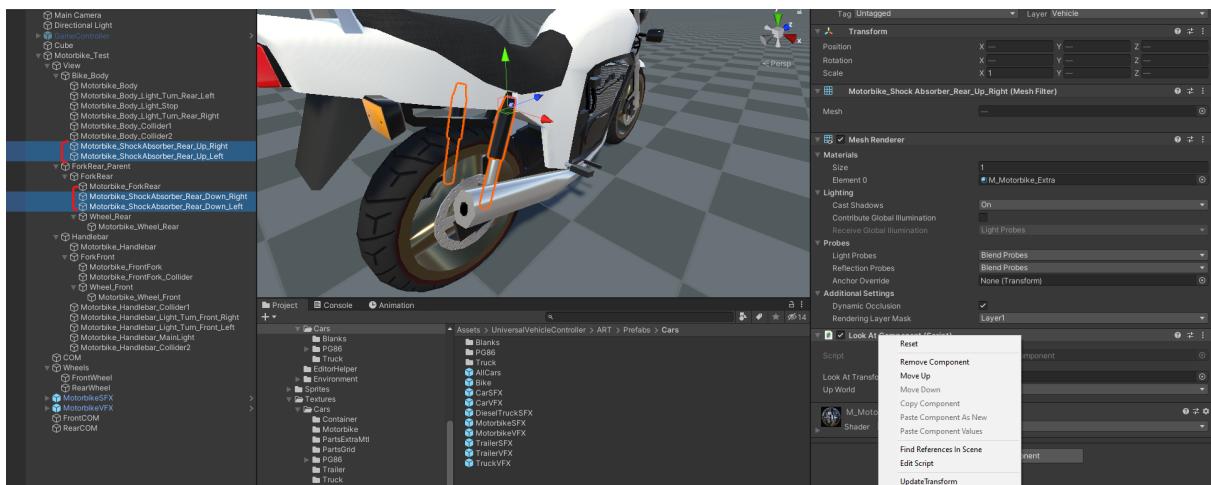
- 1) Если у вашего байка есть амортизаторы, выберите все части амортизаторов (Верхние/Нижние) и добавьте им этот компонент.



- 2) В LookAtTransform укажите противоположную часть амортизатора, например: для верхней правой части нужно указать правую нижнюю.



- 3) В некоторых случаях требуется изменить UpWorld, например для данного байка нужно изменить UpWorld на Down для верхних частей амортизаторов.
4) После настройки всех частей амортизаторов можете выполнить команду "Контекстное меню/UpdateTransform".



Если настройка сделана правильно, все части займут правильную позицию.

Мобильная версия.

Ассет можно использовать для разработки мобильных проектов, все скрипты хорошо оптимизированы и работают даже на очень слабых устройствах.

Основные ресурсы GPU и CPU уходят на качество графики и контент (Модели, текстуры, частицы и т.п.), несколько советов по оптимизации:

- По возможности старайтесь не использовать Terrain. Даже с низкими настройками качества Terrain, на слабых устройствах очень сильно влияет на производительность.
- В моделях используйте как можно меньше полигонов и отсоединяемых частей, это облегчит расчет повреждений автомобилей и снизит нагрузку на видеокарту.
- Также хочу посоветовать вам изучить статьи по оптимизации, [например эта статья](#), или подобные ей.

В ассете мобильное управление осуществляется с помощью кнопок “CustomButton”.

UI управления находится в префабе “MobileUI”. Префаб “MobileUI” является вложенным префабом в “GameController”, объект “MobileUI”, включается/выключается в методе “Start” если приложение запускается на мобильном устройстве или TargetPlatform == Android | iOS.

Так же на мобильных устройствах создается другой PlayerController, отличается от основного только в расположении CarState и в настройках камеры, в дальнейшем может быть и другие отличия.

“CustomButton” - Это наследник “Button”, с проверкой на нажатие.

Транспортные средства в проекте.

Вы можете легко создать свою машину, [есть видео как это сделать](#).

В проекте представлены 3 варианта машин и 1 мотоцикл:

1. Стоковая PG86_Stock. Медленная, тяжело управляется.
2. Для дрифта PG86_Drift. Средне быстрая с небольшой максимальной скоростью, легко управляется, легко начинает входить в дрифт.
3. Для гонок PG86_Race. Быстрая, с большой максимальной скоростью, хорошо управляется на высоких скоростях.
4. Для скоростных гонок по прямой PG86_Drag. Очень быстрая, с очень большой максимальной скоростью, подходит для заездов по прямой или с очень незначительными поворотами.
5. Motorbike - Мотоцикл с 2-мя колесами.
6. Truck - тяжелый тягач, имеет 4 ведущих колеса и 2 рулевых, трение колес высокое, трансмиссия и двигатель настроены на сильную мощность при этом максимальная скорость относительно низкая.
7. Trailer - Прицеп для тягача, очень тяжелый с 6 неуправляемыми колесами (Работает только торможение).

Разница в настройках машин PG86, сравнить которые вы можете в проекте:

- Все WheelColliders, Сила пружины подвески, высота подвески, длина подвески, жесткость подвески, трение колес.
- RigidBody машин, параметр Drag, для увеличения максимальной скорости.
- CarController:
 - CarController.COM - Центр тяжести.
 - Engine.MaxMotorTorque - Максимальная мощность двигателя.
 - Steer.MaxSteerAngle - Угол поворота колес.
 - Steer.SteerChangeSpeedToVelocity и SteerChangeSpeedFromVelocity - Скорость поворота рулевых колес.
 - Steer.HelpDriftIntensity - Помощь в подруливании в сторону дрифта.
 - Steer.PositiveChangeIntensity и OppositeChangeIntensity - Сила изменения AngularVelocity (Силы вращения).
 - GearBox.GearsRatio и MainRatio - Передаточные числа коробки передач.

Менять можно и другие параметры, такие как: ведущие колеса, вес автомобиля, кривая мощности двигателя, скорость переключения передач и многие другие, дальше в дело вступает баланс, от правильного баланса напрямую зависит правильное ощущение от езды.

Так же в проекте вы найдете заготовки для всех типов машин с названиями:

- Blank_DriftCar
- Blank_motorbike
- Blank_RaceCar
- Blank_StockCar
- Blank_DragCar
- Blank_Truck
- Blank_Trailer

Контакты.

По любым вопросам связанными с проектом вы можете обращаться любым удобным для вас способом.

e-mail:

PerfectGamesStudio@gmail.com

Bugrov1990@gmail.com

war.90@mail.ru

[Discord](#)