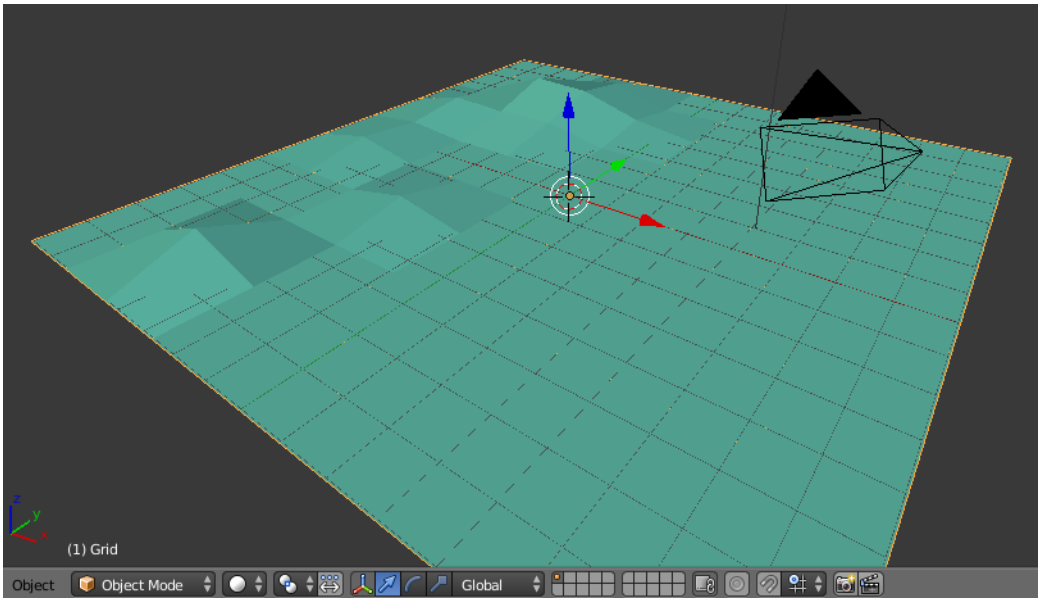
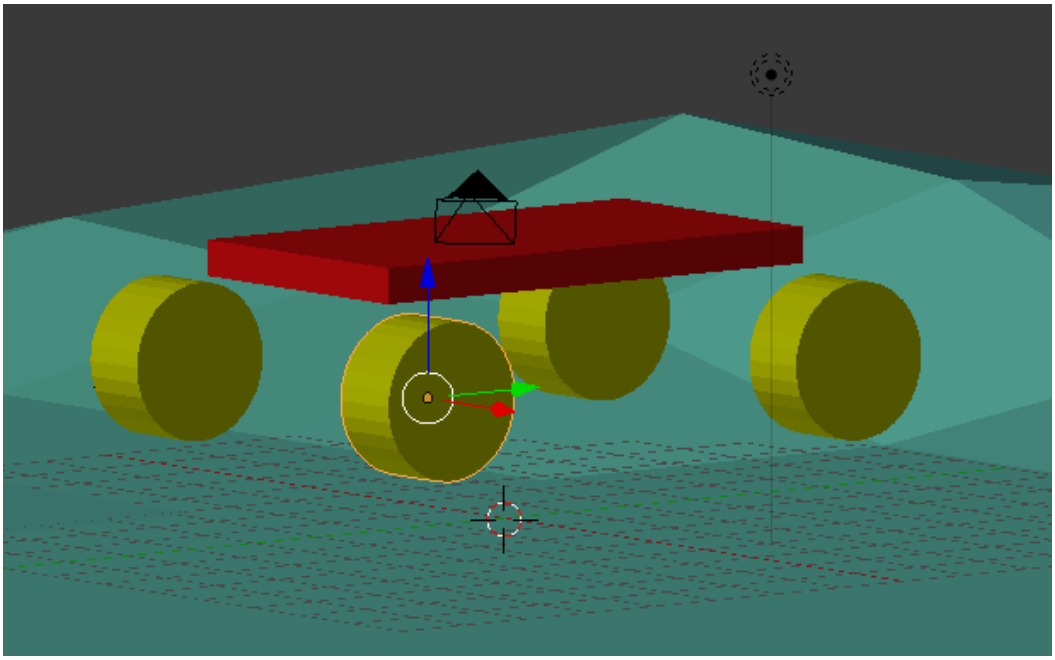
Делаем простейший автомобиль в Blender Game Engine

На этом уроке мы разберём принцип создания простого автомобиля. При этом наш автомобиль будет обладать некоторыми свойствами реальной машины. У него будут тормоза и амортизаторы, хоть и невидимые. Кроме того он будет иметь массу и, соответственно, инерцию.

Проведём некоторые подготовительные работы. Для начала в новом проекте создадим плоскость типа Grid и поднимем немного некоторые вершины, для придания ландшафту некоторой рельефности:



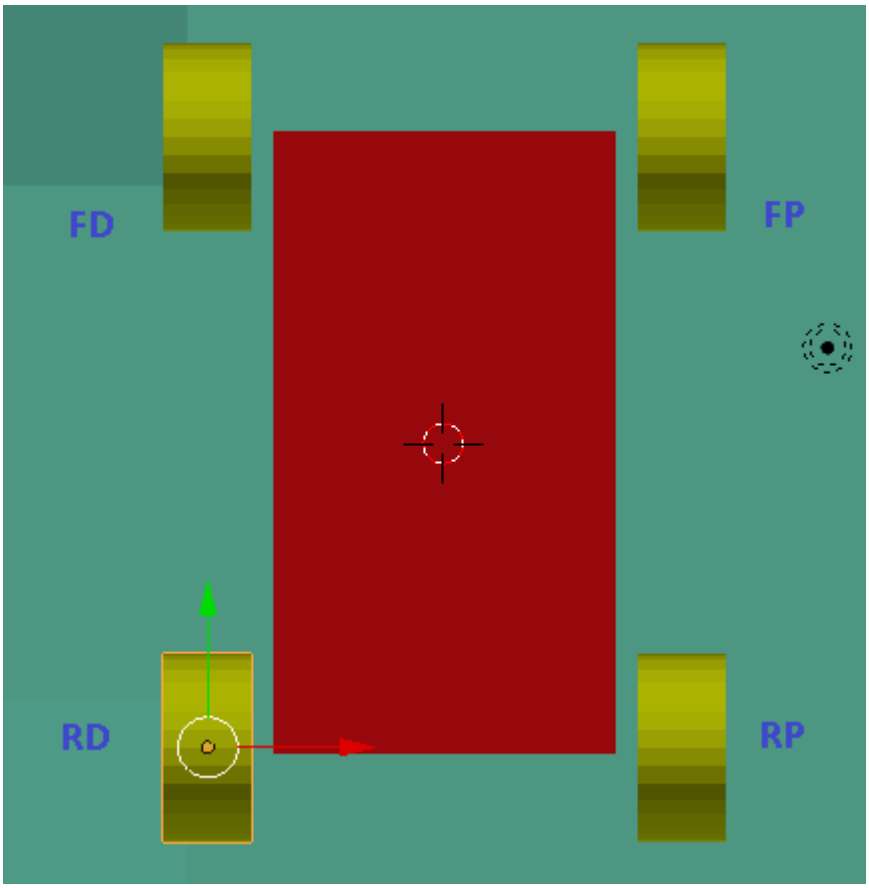
Теперь нам нужен корпус. Им будет обычный куб, растянутый до необходимых размеров(центр масс должен быть в центре). А колёсами послужат цилиндры, немного сжатые и повёрнутые на 90 градусов. Всё это необходимо немного приподнять над плоскостью. Колёса должны висеть над «землёй» на некотором расстоянии:



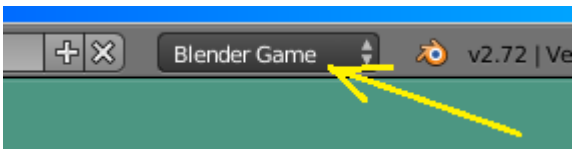
Важно! Выделить каждое колесо и применить Object->Apply->Rotation & Scale, иначе после применения скрипта колёса могут «разъехаться в разные стороны»!

Ещё одним важным моментом является переименование колёс. От этого будет зависеть правильность работы скриптов, которые необходимо написать. Напомню, что мы с вами уже не боимся скриптов. Кроме того, важные части скриптов мы разберём по возможности подробно.

И так, поворачиваем наш автомобиль (View -> Top), выделяем левое нижнее колесо и даём ему имя RD. Правое нижнее назовём RP. Левое верхнее – FD. Правое верхнее переименуем в FP.



Не забываем перейти в режим создания игры:



Теперь переходим в игровую логику (Game Logic). Сразу выделим корпус автомобиля и переименуем его в Car. В правой части создаём новый скрипт (+New) и называем его CarSetup_Standart. Напомню, что в действительности все объекты и скрипты вы можете называть по своему. Но для начала сделайте так, как у меня, чтобы избежать путаницы. Теперь, при выделенном корпусе машины, создаём сенсор Always и контролёр Python (соединив их между собой). Контролёру назначаем скрипт CarSetup_Standart и пишем скрипт:

```
#import bge
import bge

# get current scene
scene = bge.logic.getCurrentScene()

# get object list
objList = scene.objects

# get vehicle named Car
car = objList["Car"]

# get obj1 physics ID
car_ID = car.getPhysicsId()

# there isn't any obj 2
obj2_ID = 0

# want to use a vehicle constraint
constraintType = 11

# create a vehicle constraint
vehicle_Constraint = bge.constraints.createConstraint( car_ID, obj2_ID, constraintType )

# get the constraint ID
constraint_ID = vehicle_Constraint.constraint_id

# create the vehicle
vehicle = bge.constraints.getVehicleConstraint(constraint_ID)

# save vehicle constraint as an object variable
car["Vehicle"] = vehicle

# use the object names to get the tires
# my tires are named TireFD, TireFP, TireRD, TireRP
tire_FD = objList["FD"]
tire_FP = objList["FP"]
tire_RD = objList["RD"]
tire_RP = objList["RP"]
```

```
# Front driver tire position from car object center
tirePos_FD = [ -1.26, 1.89, 0.0]

# Front passenger tire position from car object center
tirePos_FP = [ 1.26, 1.89, 0.0]

# Rear driver tire position from car object center
tirePos_RD = [ -1.26, -1.89, 0.0]

# Rear passenger tire position from car object center
tirePos_RP = [ 1.26, -1.89, 0.0]

# suspension angle from car object center
# using -z axis
suspension_Angle = [ 0.0, 0.0, -1.0]

# tire axis attached to car axle
# using -x axis of tire object
tireAxis = [ -1.0, 0.0, 0.0]

# set suspension height
suspensionHeight_FD = 0.2
suspensionHeight_FP = 0.2
suspensionHeight_RD = 0.2
suspensionHeight_RP = 0.2

# set tire radius
tireRadius_FD = 0.63
tireRadius_FP = 0.63
tireRadius_RD = 0.63
tireRadius_RP = 0.63

# tire has steering?
tireSteer_FD = True
tireSteer_FP = True
tireSteer_RD = False
tireSteer_RP = False

# Add front driver tire
vehicle.addWheel( tire_FD, tirePos_FD, suspension_Angle, tireAxis, suspensionHeight_FD,
tireRadius_FD, tireSteer_FD )
```

```
# Add front passenger tire
vehicle.addWheel( tire_FP, tirePos_FP, suspension_Angle, tireAxis, suspensionHeight_FP,
tireRadius_FP, tireSteer_FP )

# Add rear driver tire
vehicle.addWheel( tire_RD, tirePos_RD, suspension_Angle, tireAxis,
suspensionHeight_RD,  tireRadius_RD, tireSteer_RD )

# Add rear passenger tire
vehicle.addWheel( tire_RP, tirePos_RP, suspension_Angle, tireAxis, suspensionHeight_RP,
tireRadius_RP, tireSteer_RP )

################################################  Suspension
###  Tire friction

grip_0 = 30.0     # front driver's tire
grip_1 = 30.0     # front passenger's tire
grip_2 = 30.0     # rear driver's tire
grip_3 = 30.0     # rear passenger's tire

vehicle.setTyreFriction(grip_0, 0)  # front driver's tire
vehicle.setTyreFriction(grip_1, 1)  # front passenger's tire
vehicle.setTyreFriction(grip_2, 2)  # rear driver's tire
vehicle.setTyreFriction(grip_3, 3)  # rear passenger's tire

### Suspension compression

compression_0 = 6.0     # front driver's tire
compression_1 = 6.0     # front passenger's tire
compression_2 = 6.0     # rear driver's tire
compression_3 = 6.0     # rear passenger's tire

vehicle.setSuspensionCompression(compression_0, 0)  # front driver's tire
vehicle.setSuspensionCompression(compression_1, 1)  # front passenger's tire
vehicle.setSuspensionCompression(compression_2, 2)  # rear driver's tire
vehicle.setSuspensionCompression(compression_3, 3)  # rear passenger's tire

# set suspension damping

damp_0 = 5.0     # front driver's tire
```

```
damp_1 = 5.0     # front passenger's tire
damp_2 = 5.0     # rear driver's tire
damp_3 = 5.0     # rear passenger's tire

vehicle.setSuspensionDamping(damp_0, 0)  # front driver's tire
vehicle.setSuspensionDamping(damp_1, 1)  # front passenger's tire
vehicle.setSuspensionDamping(damp_2, 2)  # rear driver's tire
vehicle.setSuspensionDamping(damp_3, 3)  # rear passenger's tire

## set suspension stiffness

stiffness_0 = 12.5     # front driver's tire
stiffness_1 = 12.5     # front passenger's tire
stiffness_2 = 12.5     # rear driver's tire
stiffness_3 = 12.5     # rear passenger's tire

vehicle.setSuspensionStiffness(stiffness_0, 0)  # front driver's tire
vehicle.setSuspensionStiffness(stiffness_1, 1)  # front passenger's tire
vehicle.setSuspensionStiffness(stiffness_2, 2)  # rear driver's tire
vehicle.setSuspensionStiffness(stiffness_3, 3)  # rear passenger's tire

### set roll influence

roll_0 = 0.15     # front driver's tire
roll_1 = 0.15    # front passenger's tire
roll_2 = 0.15     # rear driver's tire
roll_3 = 0.15     # rear passenger's tire

vehicle.setRollInfluence( roll_0, 0)  # front driver's tire
vehicle.setRollInfluence( roll_1, 1)  # front passenger's tire
vehicle.setRollInfluence( roll_2, 2)  # rear driver's tire
vehicle.setRollInfluence( roll_3, 3)  # rear passenger's tire
```

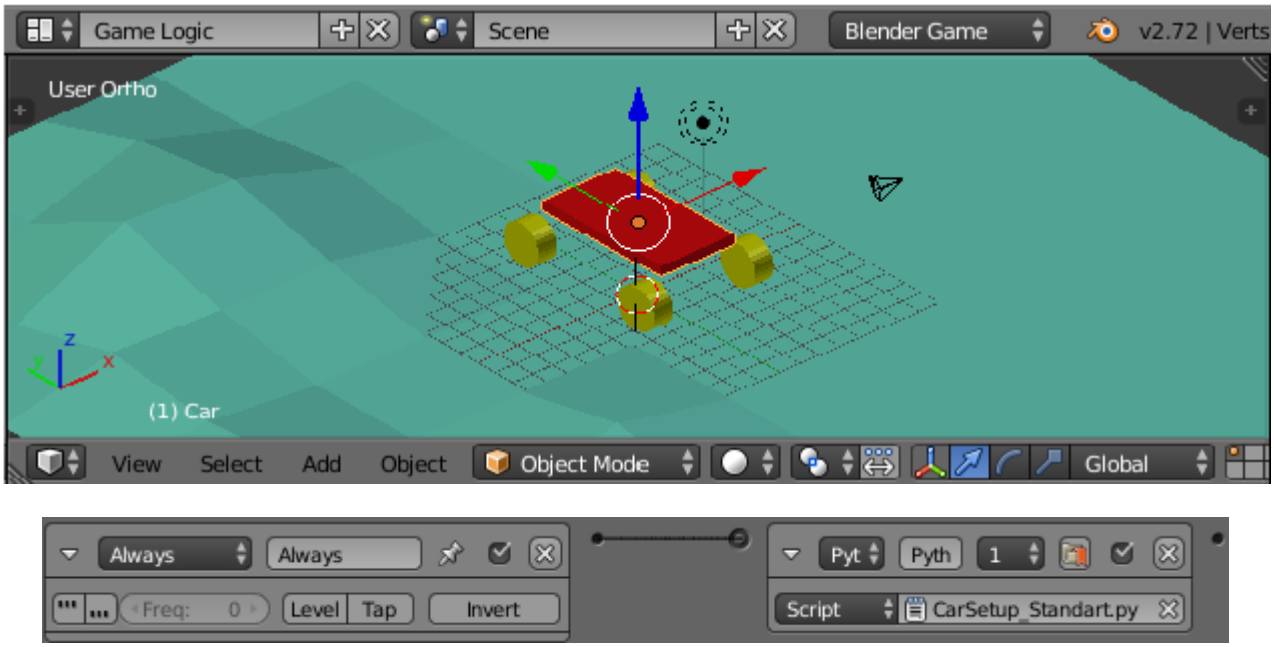
На первый взгляд скрипт кажется гигантским ☺ Поверьте, это не самый большой скрипт. Тем более, что буквально все элементы в нём важны. Это стандартный скрипт из официального примера Blender для автомобиля. Ниже мы разберём подробнее самые необходимые нам строки. Например строки

```
# get vehicle named Car
car = objList["Car"]
```

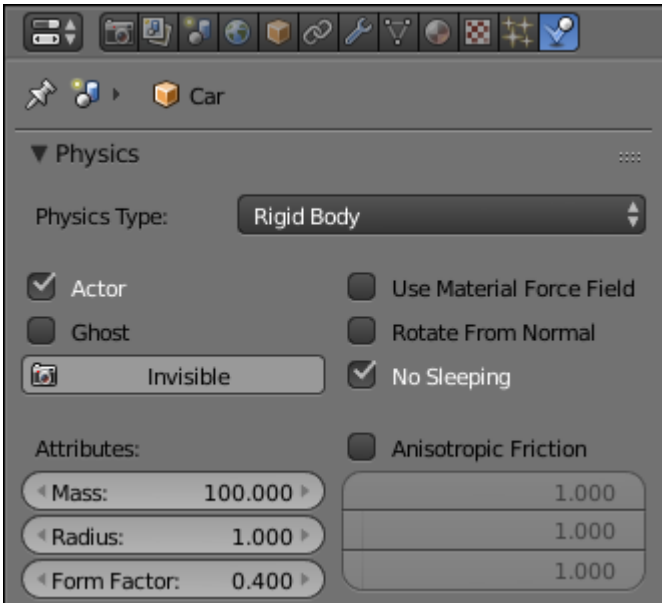
предлагают нам вставить имя корпуса автомобиля в кавычках.
Следующие строки, важные для нас

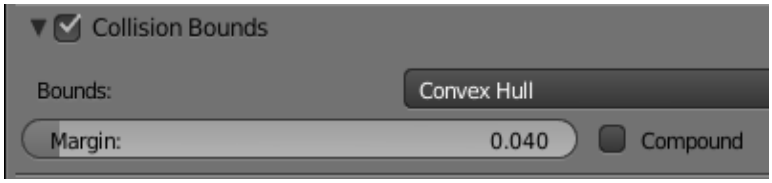
```
# use the object names to get the tires
# my tires are named TireFD, TireFP, TireRD, TireRP
tire_FD = objList["FD"]
tire_FP = objList["FP"]
tire_RD = objList["RD"]
tire_RP = objList["RP"]
```

Здесь необходимо в кавычках вставить имена наших колёс и т.д.



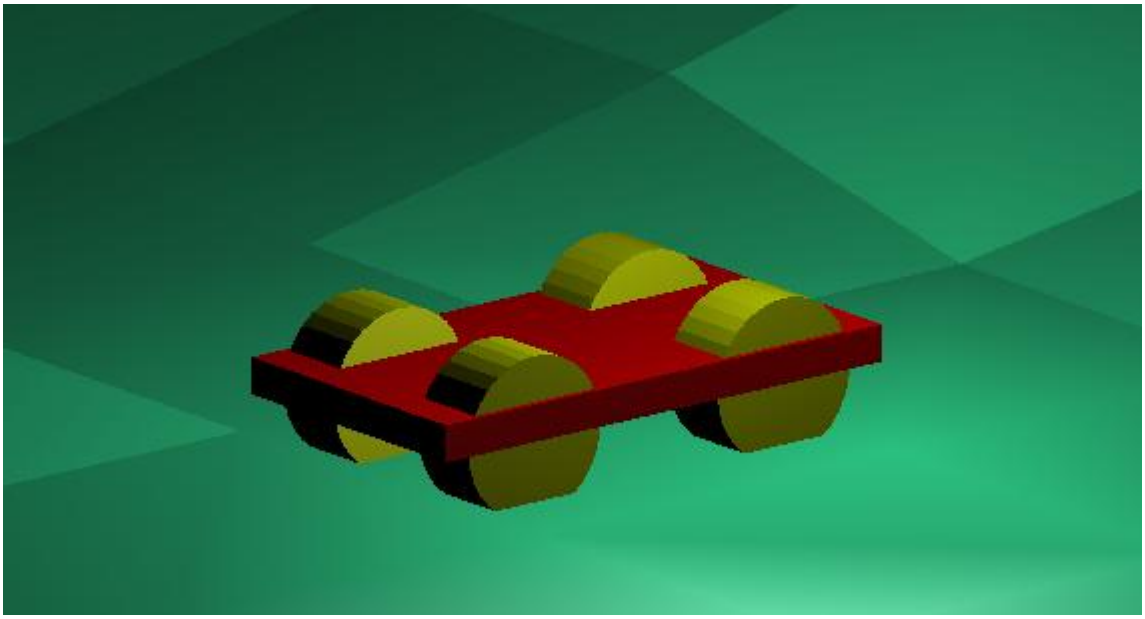
Если теперь (установив курсор в 3D окне) нажать лат.клавишу «P», то машина должна приземлиться на колёса. Но… увы, ничего не происходит. И всё лишь потому, что мы нашим физическим объектам не выставили свойства физики. Достаточно сделать корпус машины Rigid Body и придать ему массу, как всё станет на свои места и машина приземлится на землю:





Колёсам же можно поставить тип физики No Collision. Ибо они являются лишь оболочкой «физических» колёс. Именно по этой причине их можно сделать любой формы: хоть квадратными, хоть овальными.

Что же у нас получилось? Колёса встали совсем не так, как мы ожидали. Всё дело в настройках скрипта.

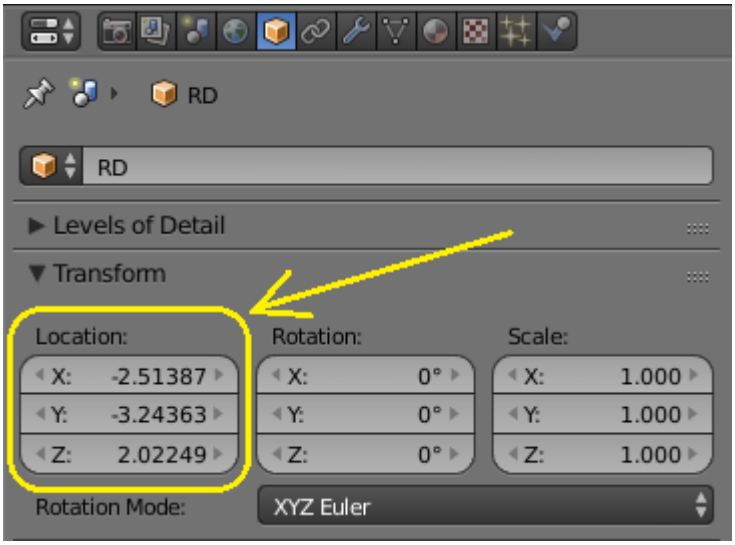


За позицию колёс в пространстве отвечает вот эта часть скрипта:

```
# Front driver tire position from car object center
tirePos_FD = [ -1.26, 1.89, 0.0]
```

```
# Front passenger tire position from car object center
tirePos_FP = [ 1.26, 1.89, 0.0]

# Rear driver tire position from car object center
tirePos_RD = [ -1.26, -1.89, 0.0]

# Rear passenger tire position from car object center
tirePos_RP = [ 1.26, -1.89, 0.0]
```

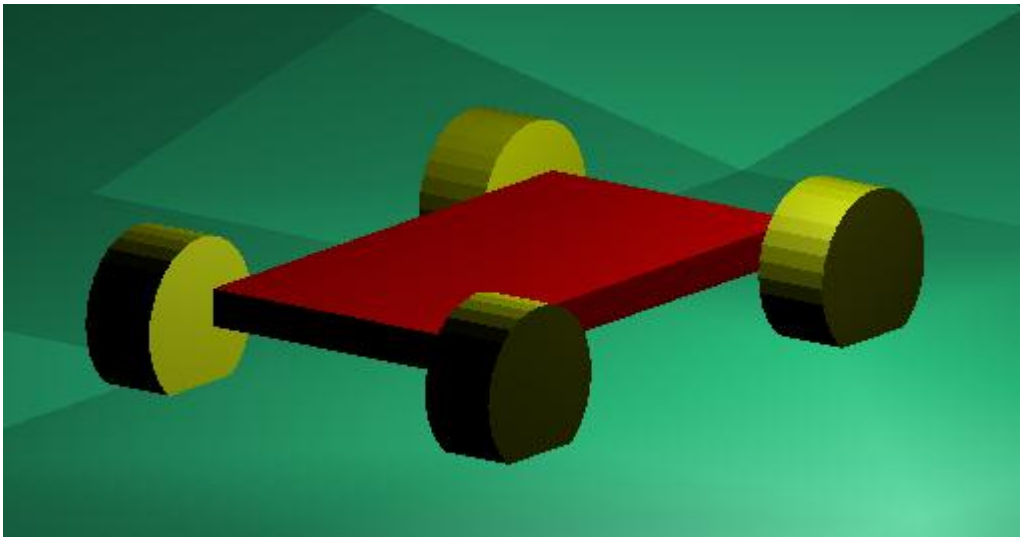
Это позиции колёс относительно центра корпуса автомобиля. Конечно, вы можете затратить массу времени и подставлять значения «методом тыка». Но гораздо легче посмотреть на позицию колеса вот здесь и скопировать значения первых трёх цифр каждой координаты:



Теперь наши координаты выглядят так:

```
# Front driver tire position from car object center
tirePos_FD = [ -2.51, 3.24, 0.0]

# Front passenger tire position from car object center
tirePos_FP = [ 2.51, 3.24, 0.0]

# Rear driver tire position from car object center
tirePos_RD = [ -2.51, -3.24, 0.0]

# Rear passenger tire position from car object center
tirePos_RP = [ 2.51, -3.24, 0.0]
```

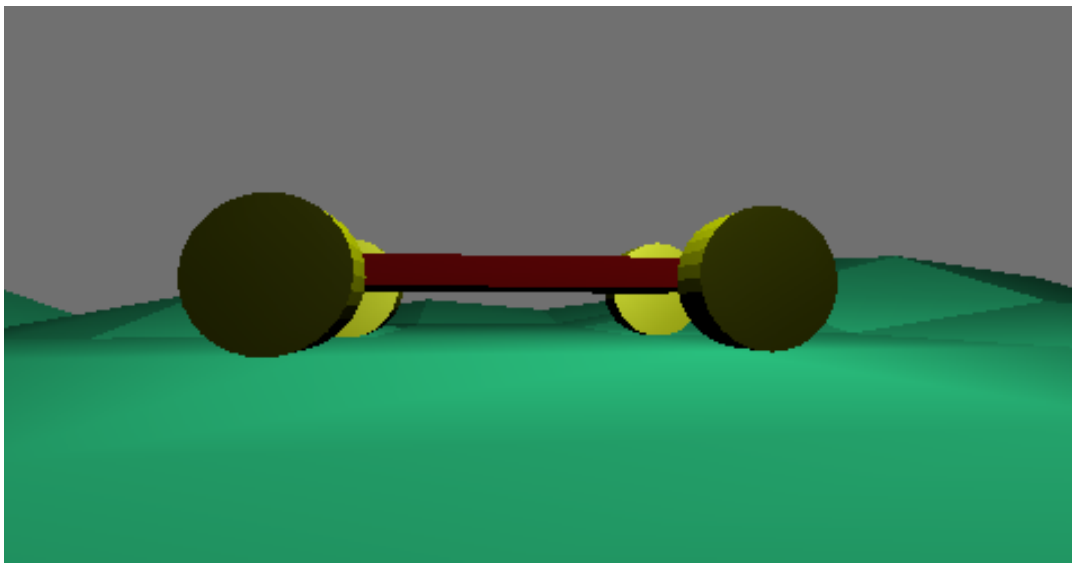
Координату Z мы пока оставим без изменений. Вот теперь должно получится правильно:



И нам осталось лишь выровнять «физические» колёса с их оболочкой. Тогда они перестанут проваливаться в «грунт». Вот эти заветные строчки:

```
# set tire radius
tireRadius_FD = 0.63
tireRadius_FP = 0.63
tireRadius_RD = 0.63
tireRadius_RP = 0.63
```

Диаметр наших колёс-цилиндров равен 1(единице). Даваёте подставим значение 1 вместо 0.63. Вот теперь машина приземляется как положено ☺



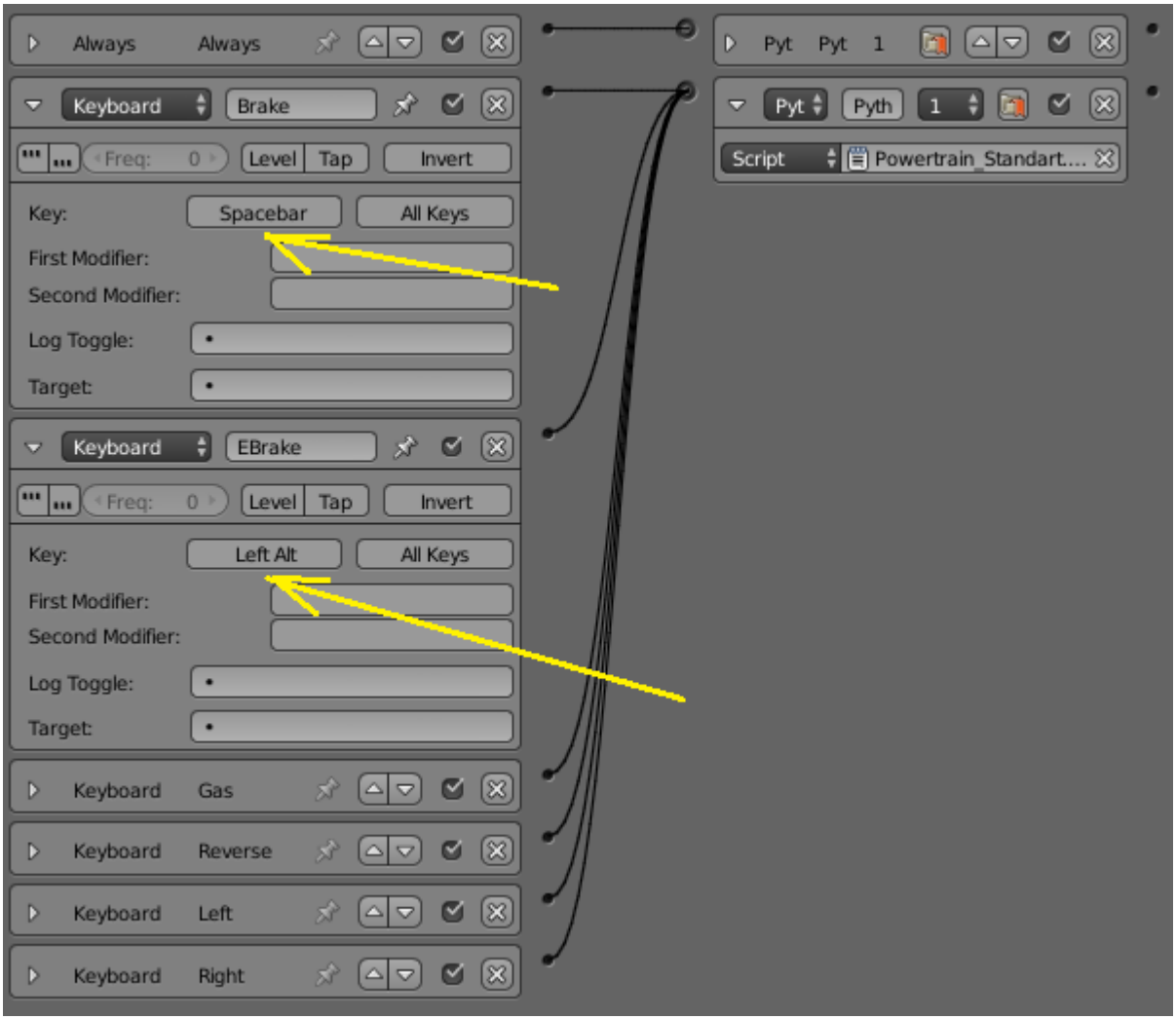
Теперь займёмся управлением. Нам необходимо управлять автомобилем с клавиатуры. Корпус авто должен быть выделен. Пишем новый скрипт с названием Powertrain_Standart:

```
#import bge
import bge

# get current scene
scene = bge.logic.getCurrentScene()

# get the current controller
controller = bge.logic.getCurrentController()

################ get sensors
brake = controller.sensors["Brake"]                 # sensor named "Brake”
emergency = controller.sensors["EBrake"]            # sensor named "EBrake"
gas = controller.sensors["Gas"]                     # sensor named "Gas"
reverse = controller.sensors["Reverse"]             # sensor named "Reverse"
steerLeft = controller.sensors["Left"]              # sensor named "Left"
steerRight = controller.sensors["Right"]            # sensor named "Right"

################ get the car

# get object list
objList = scene.objects

# get vehicle named Car
car = objList["Car"]

# get the saved vehicle ID
vehicle = car["Vehicle"]


###  Brakes
brakeAmount = 40.0      # front and back brakes
ebrakeAmount = 100.0    # back brakes only

# emergency brakes
if emergency.positive == True:

        front_Brake = 0.0
        back_Brake = ebrakeAmount
        brakes = True

# brake
```

```
elif brake.positive == True and reverse.positive == False:

        front_Brake = brakeAmount
        back_Brake = brakeAmount
        brakes = True

# no brakes
else:

        front_Brake = 0.0
        back_Brake = 0.0
        brakes = False

# brakes
vehicle.applyBraking( front_Brake, 0)
vehicle.applyBraking( front_Brake, 1)
vehicle.applyBraking( back_Brake, 2)
vehicle.applyBraking( back_Brake, 3)

################### gas and reverse

# set power amounts
reversePower = 100.0
gasPower = 400.0

# brakes
if brakes == True:

        power = 0.0

# reverse
elif reverse.positive == True:

        power = reversePower

# gas pedal
elif gas.positive == True:

        power = -gasPower

# no gas and no reverse
```

```
else:

        power = 0.0

# apply power
vehicle.applyEngineForce( power, 0)
vehicle.applyEngineForce( power, 1)
vehicle.applyEngineForce( power, 2)
vehicle.applyEngineForce( power, 3)

############# Steering

# set turn amount
turn = 0.3

# get steering sensors

# turn left
if steerLeft.positive == True:

        turn = turn

# turn right
elif steerRight.positive == True:

        turn = -turn

# go straight
else:
        turn = 0.0

# steer with front tires only
vehicle.setSteeringValue(turn,0)
vehicle.setSteeringValue(turn,1)
```

В логике создаём 6 (шесть) сенсоров клавиатуры и один контролёр Python. Имена сенсоров прописаны в начале скрипта. Поэтому очень важно их точно повторить:



Для тормозов (Brake и EBrake) назначим клавишу пробела и левый Alt. Для газа (движения вперёд) – стрелку вверх. Для движения назад – стрелку вниз. Для движения влево и вправо – стрелки влево и вправо.
Контролёру Python назначаем наш новый скрипт и соединяем все шесть сенсоров с контролёром.

Что важного в скрипте управления? Например, можно настроить силу тормозов:

```
###  Brakes
brakeAmount = 40.0      # front and back brakes
ebrakeAmount = 100.0    # back brakes only
```

Там же можно настроить, какие колёса будут тормозить (передние или задние) и при каких условиях. Ещё можно настроить мощность импульса двигателя при движении вперёд или назад:

```
# set power amounts
reversePower = 100.0
gasPower = 400.0
```

Угол поворота колёс тоже легко изменить:

```
# set turn amount
turn = 0.3
```

Короче говоря, все настройки в ваших руках! Достаточно приложить немного фантазии и смекалки и ваш автомобиль будет двигаться как вам угодно!

Запускаем лат.клавишей «P» и радуемся! ☺

Урок составил Niburiec для сайта http://blender-game.ucoz.ru/ 17.04.2015.