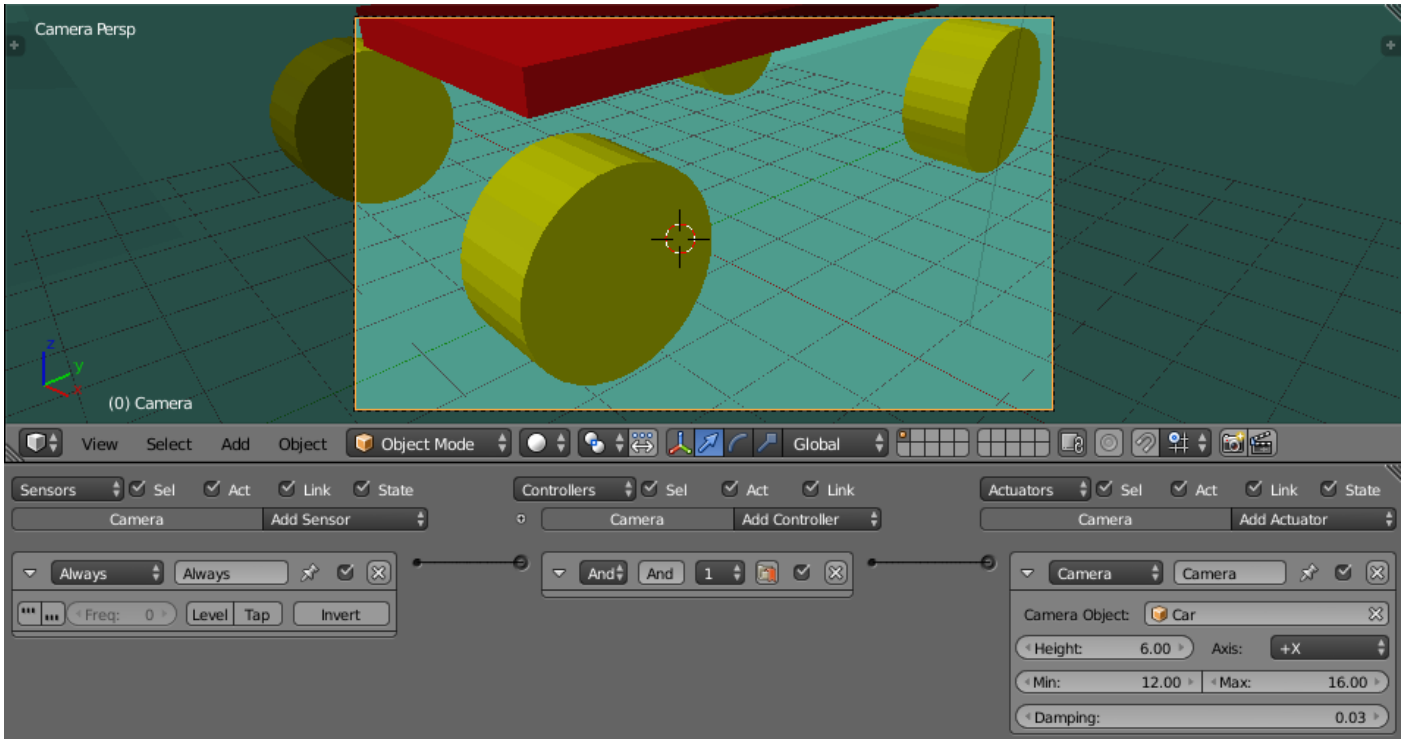
Делаем транспорт с разным количеством колёс.

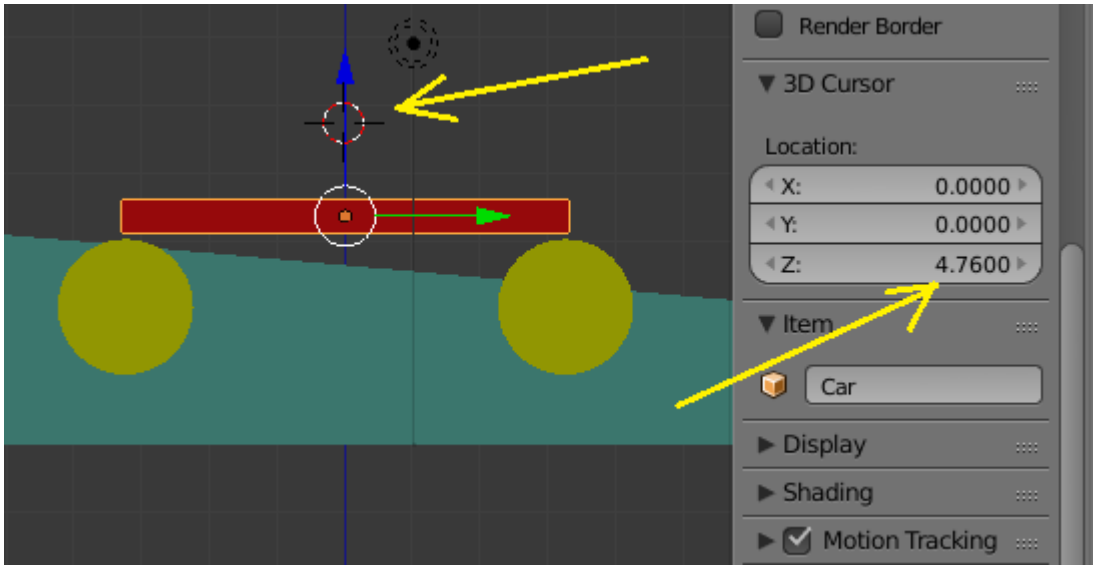
Берём с вами заготовку из предыдущего урока. Там у нас уже есть машина со скриптами и она ездит! ☺ Давайте прикрепим к ней камеру, чтобы было более удобно и реалистично за ней наблюдать. Для этого переходим в редактор логики, выделяем камеру, переходим на вид с камеры View->Camera и добавляем сенсор, контролёр и актуатор камеры:



И пусть вас не смущает, что машина не «влазит» в границы камеры. Ведь в настройках актуатора минимальная дистанция до объекта 12 единиц. А потому камера отодвинется на необходимое расстояние сама. Целью для камеры выставим объект Car. Готово. Можно убедиться в этом, нажав лат.клавишу «P». Если площади ландшафта маловато, можно его выделить и, нажав кл. «S», увеличить её до необходимых вам размеров. Скорее всего, это понадобится сделать, чтобы автомобиль после разгона не падал в «пропасть» ☺.

Теперь поговорим немного о характеристике массы. От данной величины зависит очень многое. Можно даже перечислить основное: инерция, длина тормозного пути, склонность к опрокидыванию, скорость разгона, проседание амортизаторов (их жесткость) и соответственно дорожный просвет. Поэтому массу нужно выбирать опытным путём для каждого созданного автомобиля. Кроме того, на способность опрокидывания влияет расположение центра масс. Если центр масс автомобиля выше оси колёс, то вероятность опрокидывания на поворотах велика. Если ниже оси – то мала. И конечно, способность скольжения т.е. «занос» тоже имеет значение. В нашем случае центр масс расположен точно по оси. Перевернуть такой автомобиль довольно трудно. Но давайте ради эксперимента поднимем центр масс выше колёс и попробуем проехать?! Переходим на вид справа, выделим корпус и поместим курсор в центре

масс Object -> Snap -> Cursor to Selected. Теперь, на вкладке справа, изменим координаты курсора, увеличив значение Z:



И, наконец, сместим центр масс в позицию курсора:

Object -> Transform -> Origin to 3D Cursor

Правда, теперь необходимо заглянуть в скрипт настроек авто CarSetup_Standart и изменить центр масс на столько же единиц (только с минусом) в строках:

```
53 # Front driver tire position from car object center
54 tirePos_FD = [ -2.51, 3.24, 0.0]
55
56 # Front passenger tire position from car object center
57 tirePos_FP = [ 2.51, 3.24, 0.0]
58
59 # Rear driver tire position from car object center
60 tirePos_RD = [ -2.51, -3.24, 0.0]
61
62 # Rear passenger tire position from car object center
63 tirePos_RP = [ 2.51, -3.24, 0.0]
64
```

Я выставил значение в -1.5

Теперь запустите игровой движок и попробуйте на скорости резко повернуть ?! Отчётливо заметно, как на повороте автомобиль кренится в обратную сторону.

Вывод – центр масс для машины нужно делать чуть ниже оси. Тогда она гарантированно не перевернётся.

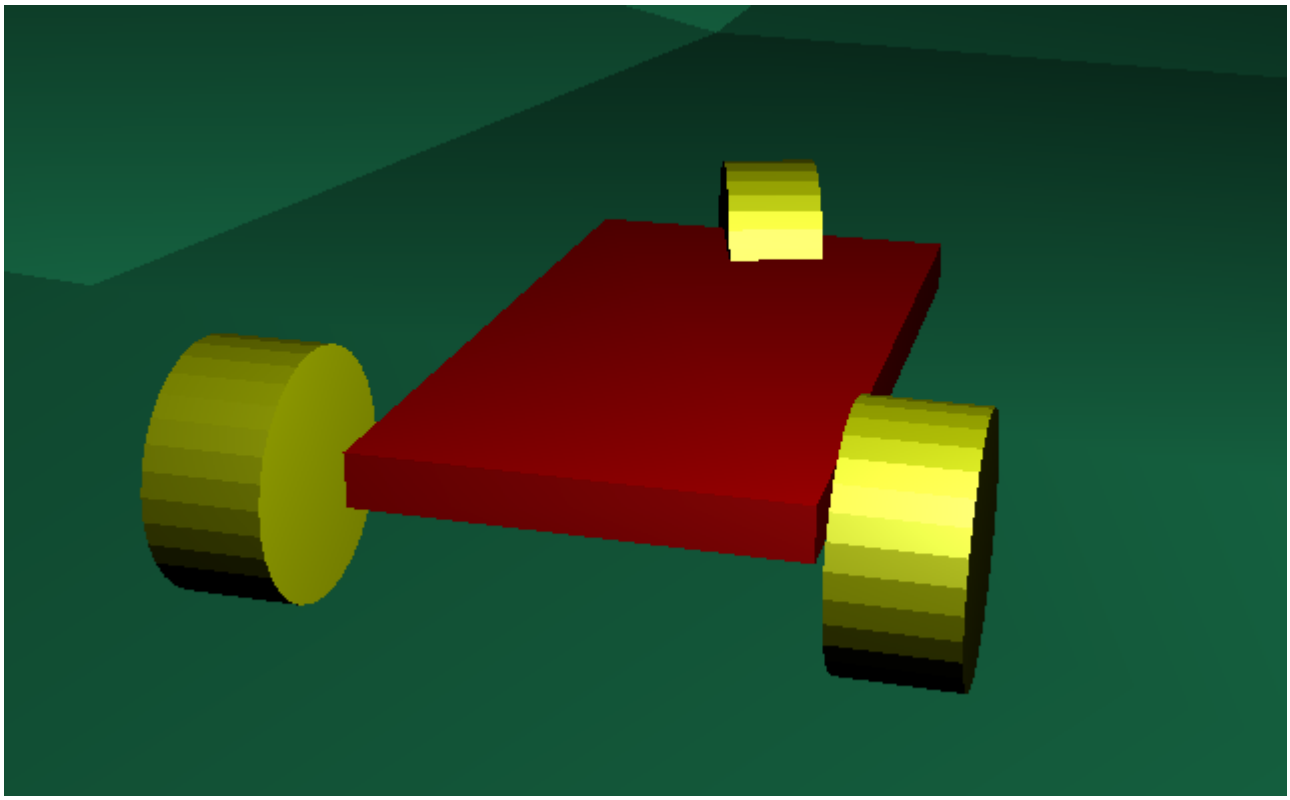
Но, давайте вернём центр масс на место

Object -> Transform -> Origin to Center of Mass

и продолжим эксперименты ☺.

Как сделать трёх колёсный транспорт? Неужели нужно переписывать скрипт, в котором и так-то трудно разобраться? Вовсе нет! Мы поступим гораздо проще и просто совместим два передних колеса в одних координатах:

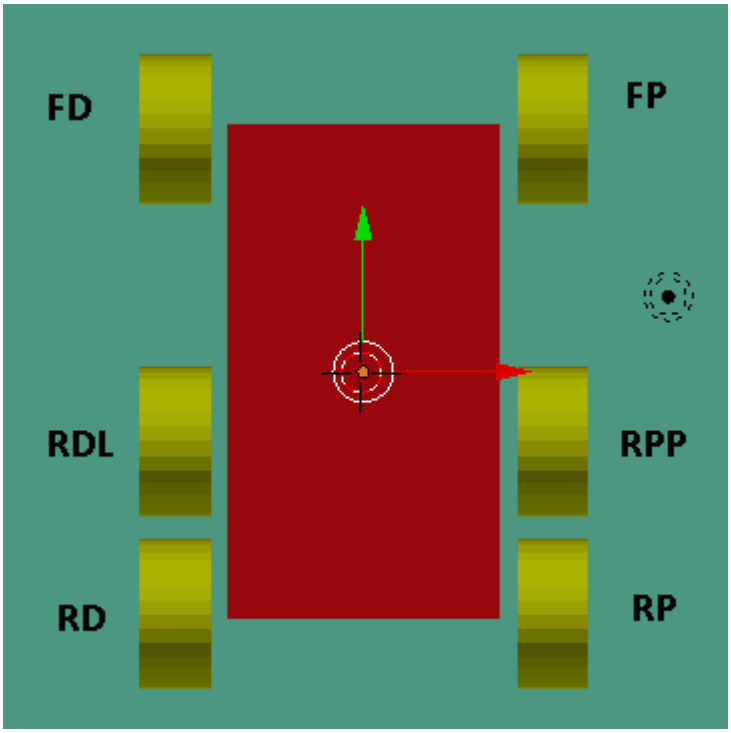
```
53 # Front driver tire position from car object center
54 tirePos_FD = [ 0.0, 3.24, 0.0]
55
56 # Front passenger tire position from car object center
57 tirePos_FP = [ 0.0, 3.24, 0.0]
58
59 # Rear driver tire position from car object center
60 tirePos_RD = [ -2.51, -3.24, 0.0]
61
62 # Rear passenger tire position from car object center
63 tirePos_RP = [ 2.51, -3.24, 0.0]
```



По тому же принципу делается и двух колёсный транспорт – мотоцикл. Только у мотоцикла нужно продумать расположение центра масс, чтобы при повороте он наклонялся в правильном направлении.

А как, спросите вы, сделать шесть колёс или восемь? А вот здесь без «танцев с бубном» вокруг скрипта не обойтись. По сути, нужно переписывать весь скрипт настроек. Но я вам серьёзно помогу. Я заранее переписал стандартный скрипт, немного его модернизировав. И вот что получилось – добавлять колёс в моём скрипте можно сколько угодно, но управляющих может быть только четыре. Я так пока и не понял, как сделать управляемыми все колёса. Но всё же, для шести и восьми колёсного транспортного средства этого вполне достаточно.

И так, добавляем ещё парочку колёс посередине, путём дублирования задних. И называем их RDL и RPP:



Далее пишем скрипт настроек, который я назвал MyCarSetup_Proba:

```
##########################################################################
#                                                                        #
#  составил Niburiec                                                     #
#  все мои примеры http://blender-3d.ru/forum/index.php/topic,796.0.html#
#                                                                        #
##########################################################################


#import bge
import bge

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

# get object list
objList = scene.objects
```

```
# get vehicle named Car_Red
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"]

tire_RDL= objList["RDL"]

tire_RPP= objList["RPP"]


# Front driver tire position from car object center

tirePos_FD = [ -2.5, 3.2, -0.2]


# Front passenger tire position from car object center

tirePos_FP = [ 2.5, 3.2, -0.2]


# Rear driver tire position from car object center

tirePos_RD = [ -2.5, -3.2, -0.2]


# Rear passenger tire position from car object center

tirePos_RP = [ 2.5, -3.2, -0.2]


tirePos_RDL= [ -2.5, -0.9, -0.2]


tirePos_RPP= [ 2.5, -0.9, -0.2]


# 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 = 0.3
suspensionHeight_FP = 0.3
suspensionHeight_RD = 0.3
suspensionHeight_RP = 0.3
suspensionHeight_RDL= 0.3
suspensionHeight_RPP= 0.3

# set tire radius
tireRadius_FD = 1.0
tireRadius_FP = 1.0
tireRadius_RD = 1.0
tireRadius_RP = 1.0
tireRadius_RDL= 1.0
tireRadius_RPP= 1.0

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

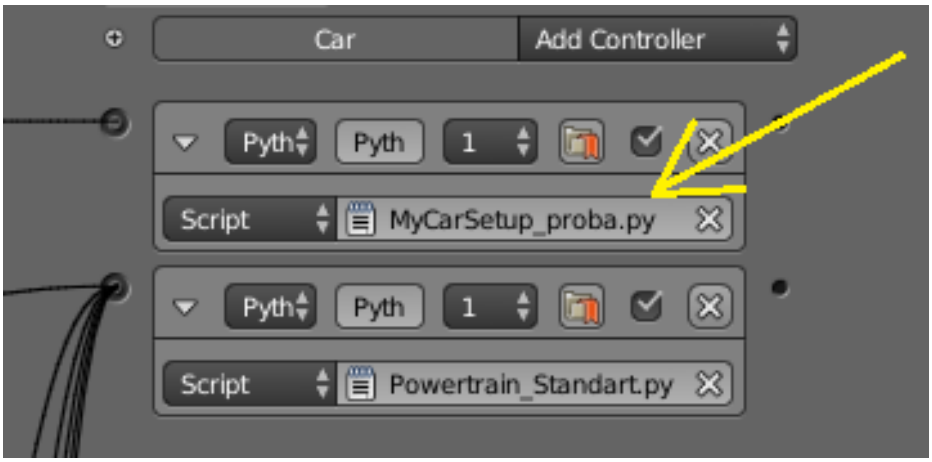
```
# Add front driver tire

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 )


vehicle.addWheel( tire_RDL, tirePos_RDL, suspension_Angle, tireAxis,
suspensionHeight_RDL,  tireRadius_RDL, tireSteer_RDL )


vehicle.addWheel( tire_RPP, tirePos_RPP, suspension_Angle, tireAxis,
suspensionHeight_RPP,  tireRadius_RPP, tireSteer_RPP )


################################################  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
grip_4 = 30.0
grip_5 = 30.0

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
vehicle.setTyreFriction(grip_4, 4)
vehicle.setTyreFriction(grip_5, 5)

### Suspension compression

compression_0 = 3.0     # front driver's tire
compression_1 = 3.0     # front passenger's tire
compression_2 = 3.0     # rear driver's tire
compression_3 = 3.0     # rear passenger's tire
compression_4 = 3.0
compression_5 = 3.0

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
vehicle.setSuspensionCompression(compression_4, 4)
vehicle.setSuspensionCompression(compression_5, 5)
```

```
# set suspension damping

damp_0 = 9.0     # front driver's tire
damp_1 = 9.0     # front passenger's tire
damp_2 = 9.0     # rear driver's tire
damp_3 = 9.0     # rear passenger's tire
damp_4 = 9.0
damp_5 = 9.0

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
vehicle.setSuspensionDamping(damp_4, 4)
vehicle.setSuspensionDamping(damp_5, 5)

## 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
stiffness_4 = 12.5
stiffness_5 = 12.5

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
vehicle.setSuspensionStiffness(stiffness_4, 4)
vehicle.setSuspensionStiffness(stiffness_5, 5)


### 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
roll_4 = 0.15
roll_5 = 0.15


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
vehicle.setRollInfluence( roll_4, 4)
vehicle.setRollInfluence( roll_5, 5)
```

Теперь главное не забыть заменить скрипт в контролёре настроек:



И всё! Можно запускать и кататься. Вы заметите, что при езде по неровностям колёса пружинят совершенно независимо друг от друга. Но, если говорить честно, то эти дополнительные колёса лишь катятся по поверхности. Чтобы они участвовали в торможении или наборе скорости необходимо переписать и скрипт управления. Ниже я даю вам мой вариант. Если вы творчески мыслите, то вы сможете его усовершенствовать сами ☺ :

```
##########################################
#
#  Vehicle Constraint
#
#  Blender 3D 2.6x - 2.7 using Bullet Physics
#
#  составил Niburiec
#  все мои примеры http://blender-3d.ru/forum/index.php/topic,796.0.html
#
#
#
##########################################

#import bge
import bge

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

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

################ get sensors
reverse = controller.sensors["Reverse"]             # sensor named "Reverse"
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_Red
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 = 200.0
gasPower = 200.0

# 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)
vehicle.applyEngineForce( power, 4)
vehicle.applyEngineForce( power, 5)

############## 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 заключении скажу, что управляемыми могут быть колёса с номерами 0, 1, 2, 3 .
Удачи вам в творчестве!