Endless Road Generator 3D

The paradigm

First of all let me **thank you** for buying this asset.

This documentation describes the principles of the **Endless Road Generator 3D** asset.

Endless road generator 3D is a smart three-dimensional generator of road with traffic in the three-dimensional world. The main feature of the program is that the road is not pre-created initially and before you start scene, but directly in real-time depending on your purposes or any game events. This means that road is **constructing in real-time** to the **left**, **right**, **up** and **down**.

The ideology of this asset based on three-dimensional premade road segments(meshes) and use it in real-time generation depends on parameters in Inspector/script.

Moreover, another great feature of this asset is very **flexible control** of the **curvature** of the road and other settings in real-time.

Generated road is **optimized for mobile** devices. Road segments that a player has passed, are removed from the scene to clean up the memory.

How it works

Now I'll explain the files you will find in the asset and describe in detail how to configure the asset for your purposes.

First of all I want to split files into two types:
1) **main** and important files for road generation. I will tell in details about these files.
2) **secondary** and not so important files that I had created for the demo, so you can imagine the possibilities of asset. I will briefly tell about these files.

Let's start with the **main** files for road generation:

- **FBX models** of road segments, traffic and objects of the environment
    1. **Road_down.fbx** - road segment (downhill)
    2. **Road_left_easy.fbx** - road segment (very easy left turn)
    3. **Road_left_extreme.fbx** - road segment (very quite hard left turn)
    4. **Road_left_hard.fbx** - road segment (very hard left turn)
    5. **Road_left_light.fbx** - road segment (very very slight left turn)
    6. **Road_left_medium.fbx** - road segment (normal left turn)
    7. **Road_leftLean5.fbx** - road segment (spin, twirl, skew CCW of road by 5 degree)
    8. **Road_right_easy.fbx** - road segment (very easy right turn)
    9. **Road_right_extreme.fbx** - road segment (very quite hard right turn)
    10. **Road_right_hard.fbx** - road segment (very hard right turn)

11. **Road_right_light.fbx** - road segment (very very slight right turn)
12. **Road_right_medium.fbx** - road segment (normal right turn)
13. **Road_rightLean5.fbx** - road segment (spin, twirl, skew CW of road by 5 degree)
14. **Road_straight.fbx** - road segment(straight road)
15. **Road_up.fbx** - road segment (climb up)
16. **Side_ad_01.fbx** - environment (roadside advertising sign)
17. **Side_sign_01.fbx** - environment (roadside sign)
18. **Hill_medium.fbx** - environment (sloping hill)
19. **Mount_medium.fbx** - environment (sharp hill)
20. **Block_break.fbx** - on road obstacle (heavy crack hole)
21. **Block_repair_01.fbx** - on road obstacle (road reconstruction)
22. **Vehicle_car.fbx** - traffic (vehicle sedan type)
23. **Vehicle_SUV.fbx** - traffic (vehicle SUV type)

- ## **Prefabs** of these FBX models(with some additional GameObjects)
  1. **Road_down.prefab** - road segment prefab
  2. **Road_left_easy.prefab** - road segment prefab
  3. **Road_left_extreme.prefab** - road segment prefab
  4. **Road_left_hard.prefab** - road segment prefab
  5. **Road_left_light.prefab** - road segment prefab
  6. **Road_left_medium.prefab** - road segment prefab
  7. **Road_leftLean5.prefab** - road segment prefab
  8. **Road_right_easy.prefab** - road segment prefab
  9. **Road_right_extreme.prefab** - road segment prefab
  10. **Road_right_hard.prefab** - road segment prefab
  11. **Road_right_light.prefab** - road segment prefab
  12. **Road_right_medium.prefab** - road segment prefab
  13. **Road_rightLean5.prefab** - road segment prefab
  14. **Road_straight.prefab** - road segment prefab
  15. **Road_up.prefab** - road segment prefab
  16. **Hill_medium.prefab** - environment object prefab
  17. **Mount_medium.prefab** - environment object prefab
  18. **Side_ads01.prefab** - environment object prefab
  19. **Side_sign.prefab** - environment object prefab
  20. **Spoiler_hole.prefab** - road obstacle object prefab
  21. **Spoiler_repair.prefab** - road obstacle object prefab

- ## **Scripts** for road generation
  1. **roadDestruction.cs** - the script for the destruction of the far, no longer needed road segments
  2. **roadGenerate.cs** - the main script that generates the road.
  3. **vehicleScript.cs** - the script for traffic(you cannot call this script absolutely necessarily related to the road generation, but is that a road with no traffic ?)

- ## **Materials** and shaders

I don't think it is necessary to describe the materials and shaders asset, as they do not participate in the generation of the road and can be any brand of your choice. But you should know that the script of the vehicle determines the type of surface on which rides by the name of the material. So, if you want to use my vehicles your road must be named **‘road_asphalt_mat’**.

Detailed explanations about FBX models:

Road segments FBX - made in a 3d package 15 road segments. From straight one to very sharp left and right turns. Also, two climb up and downhill segments, and two swirling element for twist compensation.
(Twisting occurs when any turn road segment comes just after climb or downhill. After that the road begins to bends in three axes, not just horizontally. It becomes dangerous when this twist is accumulated more and more. In this case, the road can twist even 180 degrees(something like roller coaster). However, I guess that's not the desired effect. And as soon as the main script sees that the angle of the road starts to twisting(losing the horizon), it automatically inserts the compensation segments **Road_leftLean5.prefab** and **Road_rightLean5.prefab** to make it parallel to the horizon. If for some reason you seem to have fun when the road can have a profound twists, you can disable the insertion of compensation segments in the main script **roadGenerate.cs**. So, you'll get road like a roller coaster track.)
So, your road will be constructed out of these 15 road segments by main script - **roadGenerate.cs**.

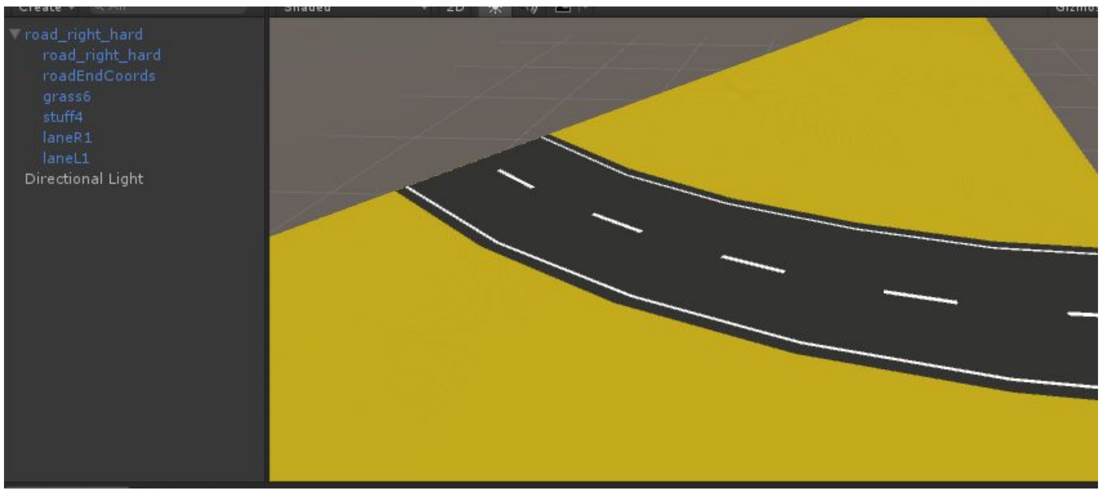
Environment FBX - made in a 3d package two models for hills and two models of side road signs. I can not call them really necessary, as the road can be generated without them, but in my opinion road, without the environment, traffic and so on is a not a sterling road.
So, these two kind of hills will be randomly scattered to the left and to the right of your road using **roadGenerate.cs** script. As well as two kinds of road signs.

On road obstacles FBX - made in a 3d package two models of road obstacles. As environment props it's not really necessary for road but what a road without crack holes and heavy garbage in a middle of lane ? These two kind of obstacles will be randomly scattered on left and right lanes of the road using **roadGenerate.cs** script.

Traffic FBX - made in a 3d package two models of vehicles represents traffic. These two kind of vehicles will be randomly scattered on left and right lanes of the road using **roadGenerate.cs** script and driven by AI using **vehicleScript.cs** script. Vehicles on right side moving same direction as you. The left side vehicles are is oncoming traffic moving to you.

Next part is prefabs explanation:

Prefabs are common FBX of road segments and other types of meshes with added some empty GameObjects, colliders and other stuff for the purpose of the road generator.



(example of road segment “road_right_hard”)

Each prefab of road segment contains of:

a) **model**(mesh) of road segment made in 3d package. I’ve separated model to 3 parts: **road_right_hard -** the roadbed**, grass6 -** off-road ground**, stuff4 -** white stripes on roadbed. You can have any number of objects in FBX **-** it is only a matter of your convenience.
b) **roadEndCoords** - is an empty GameObject positioned at the end of the road segment, deployed at an angle which bent segment. In this case, with turning hard it's 35 degrees. It’s very important object in prefab. It’s a pivot where this road segment ends and another one will be placed. So place this GameObject very precisely and make no mistake with the rotation angle to avoid any cracks between road segments.
c) **laneL1** and **laneR1** two empty GameObjects used to create waypoints for the AI navigation(honestly, it’s dublicated **roadEndCoords**, renamed and moved left and right to the lane as named).

So, to make new freshly crafted road segment compatible you need to **add three empty GameObjects** to it’s prefab: **roadEndCoords, laneL1** and **laneR1**. Also, don't forget to **assign** the mesh **collider** to your road meshes, so you can drive on this segment.

As for prefabs of **environment** objects - you need no additional actions to create prefabs of FBX. Don't forget to add mesh colliders to it.

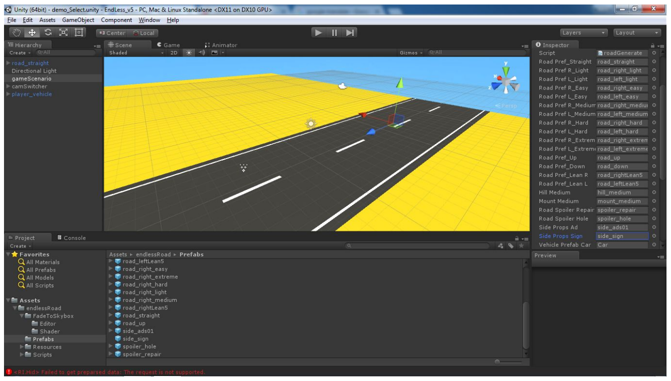
Road obstacles prefabs has their own collision objects. Also they are **rigidbody** for realistic physics collisions and crashes. It’s more about your gameplay and graphics/physics you wish in your project, so it’s up to you how to set up it well.

How generator of the road works.
And how you might control the process.

Hope you strongly understood that the construction of the road is an automatic process from pre-created road segments of the road. The road segments should be are created in a 3D package. Their number and variety is not limited, but as an example I'm using the 15 segments, which are listed above.

(Currently to use more than 15 road segments you need to manually change the number of segments and the formula generating tight in the script **roadGenerate.cs**. In the later versions, which will be released this feature will be implemented through the Inspector, without need to change the lines of code script)

So, with these 15 premade segments of the road, we should connect the prefabs of these segments to proper fields of **roadGenerate.cs** script in Inspector(you may drop this script to any empty GameObject in scene). Do same thing with environment, obstacles and traffic prefabs as well.



Script	roadGenerate
Road Pref_Straight	road_straight
Road Pref R_Light	road_right_light
Road Pref L_Light	road_left_light
Road Pref R_Easy	road_right_easy
Road Pref L_Easy	road_left_easy
Road Pref R_Medium	road_right_mediu
Road Pref L_Medium	road_left_medium
Road Pref R_Hard	road_right_hard
Road Pref L_Hard	road_left_hard
Road Pref R_Extrem	road_right_extren
Road Pref L_Extreme	road_left_extreme
Road Pref_Up	road_up
Road Pref_Down	road_down
Road Pref_Lean R	road_rightLean5
Road Pref_Lean L	road_leftLean5
Hill Medium	hill_medium
Mount Medium	mount_medium
Road Spoiler Repair	spoiler_repair
Road Spoiler Hole	spoiler_hole
Side Props Ad	side_ads01
Side Props Sign	side_sign
Vehicle Prefab Car	Car

Now, when all prefabs properly linked, the main script will construct endless road just after you'll press "Start", based on the parameters that you set in the Inspector.

The main parameter is **Road Hardness**(curvature) - it describes how sharp are turns should be.

Hills percentage is the parameter that regulates how many climbs and downhills your road will contain.

Traffic - percentage of traffic vehicles on your road(both lanes).

Spoilers - percentage of obstacles on your road(both lanes).

Side props - percentage of off-road environment objects like road signs and so.

Max road height(it's abstract and not in meters) - how high climbs and how deep downhills should be. When road reach this height or depth(it's not really metters) the script will change direction of hill/valley to inverse.

You might to adjust any parameter like the hardness, traffic, obstacles any time the game(scene) is started. Moreover, these parameters can be associated with game events and triggers.

For example, if a player is too good and fast & furious driver, you can advisedly complicate the road - increasing sharpness of turns and adding more traffic/obstacles.

Knowledge is a treasure, but practice is the key to it

So, let's go all the way, from beginning to end.
Let's create a race scene from scratch and finally make it work.
At the end you'll get scene where you can drive your vehicle through endless road.
The plan for lesson is:

a) First, need to create assets for road, environment, and traffic in a 3D package.
b) Second, configure prefabs - need to add some important empty GameObjects as pivots to these assets.
c) The third and last thing is connect all this stuff with asset's scripts in Inspector.

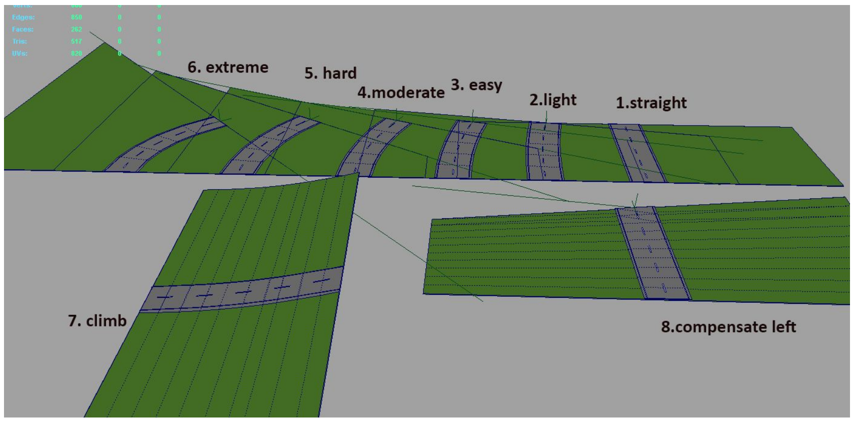
- **Part A - creating assets in 3D editor**

1. Initially create 15 road segments

So, in 3D modelling package we need to create 15 meshes of the road segments. Suppose we need 1 straight, 5 type of turns: very light turn, then little more serious turn(easy), then more curved moderate turn, a hard turn and finally a very sharp extreme turn. Also, we want road with some hills, like any road in life, except of course, the route of the salt lake Bonneville.

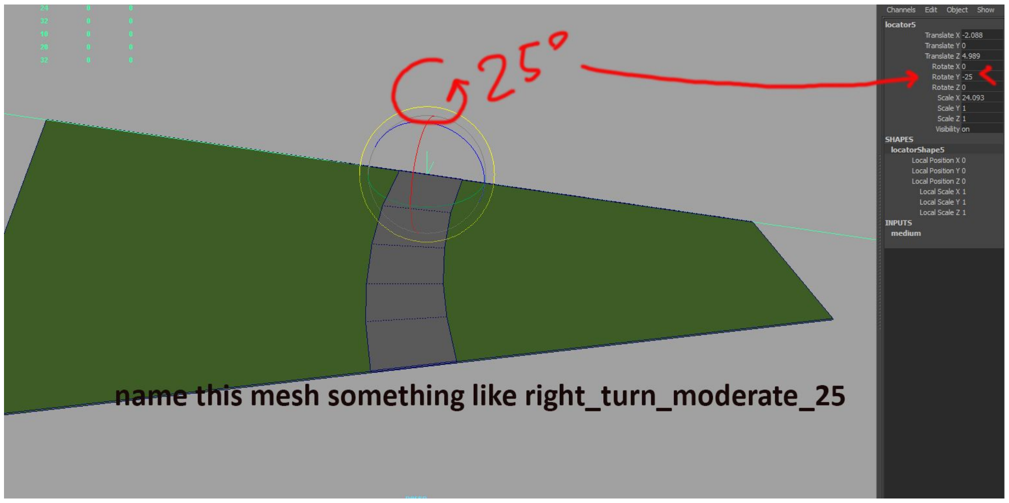
To do this, in addition to 1 straight and 5 turns(5 left and 5 right - 11 segments at sum), one ascent and one descent(13), we also will need two twisted straights segments to compensate for the twisting. The nature of the phenomenon of "twisting" I referred to earlier(15).

In total we should have 15 road segments:



In this picture is still missed: 5 same left turns(light, easy, moderate, hard, extreme), downhill, compensate right. So, with 8 showed above and missed 7 counted in this string, we have 15 road segments in total.

So, there is no restrictions for 3d modeling, but I want you to pay attention to the final degree of curvature of the turn. You must accurately know the angle/degrees, and keeping it to make your mode. Don't make an uneven surface in the beginning/end of the roadbed, otherwise they will connects each other with ugly gaps. Therefore, I propose to enter the rotation angle in the model name, to avoid confusion at prefabs configuration stage.


25°
name this mesh something like right_turn_moderate_25
Channels Edit Object Show

```
locator5
      Translate X -2.088
      Translate Y 0
      Translate Z 4.989
         Rotate X 0
         Rotate Y -25
         Rotate Z 0
          Scale X 24.093
          Scale Y 1
          Scale Z 1
       Visibility on
SHAPES
  locatorShape5
   Local Position X 0
   Local Position Y 0
   Local Position Z 0
      Local Scale X 1
      Local Scale Y 1
      Local Scale Z 1
INPUTS
  medium
```

Here is an example of how every segment should be done. Note the locator having a turn at the Y by 25 degrees. It exactly cut a road segment when it will be attached to another segment, they would be perfectly 'welded' without any gaps. Also, please note that I suggest keep angle of rotation as part of .FBX's name.

2. Create related elements of the environment.

Also, in addition to the road segments we will make two hills/mountains, two road sign and two piles of garbage that will also stay as the surroundings on our way and next to the road. Of course, we could create a few vehicles, but we'll Import them from standard Unity assets.

two road signs
two hills
two obstacles

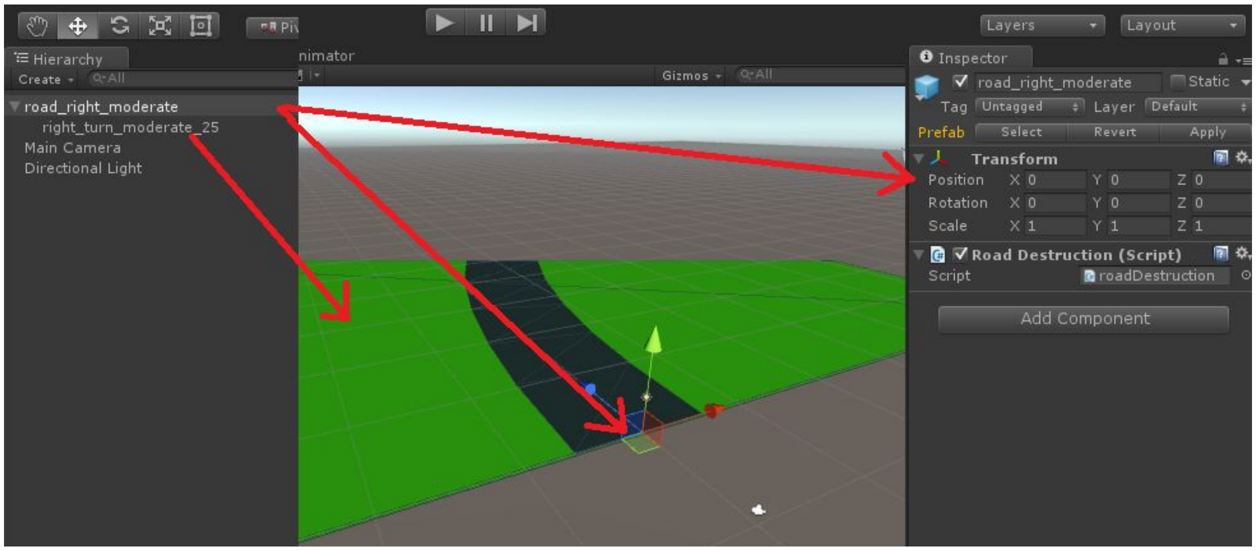
Also we need to make: 2 hills, two road signs, two obstacles(Just for example ! You might don't use any of this objects for your road)

The last thing to do is to **export our models to 21 FBX files**: 15 road segments, 2 hills, two road signs, two obstacles.

- **Part B - create and configurate prefabs from 3D models**(FBX we just made)

1. Now it's time to configure our prefabs to use it. This way all road segments need to be complemented with three empty objects. **roadEndCoords**, **laneL1** and **laneR1**.

To do this in an empty scene, create an empty GameObject. Let's name it **road_right_moderate** and put in 0, 0, 0. Then drag the mesh **right_turn_moderate_25.FBX** to road_right_moderate as child. The parent GameObject road_right_moderate is the point where our mesh road segment right_turn_moderate_25.FBX will be attached to the previous road segment. So, position the middle and the beginning of the mesh right_turn_moderate_25.FBX to GameObject road_right_moderate at is shown on picture:



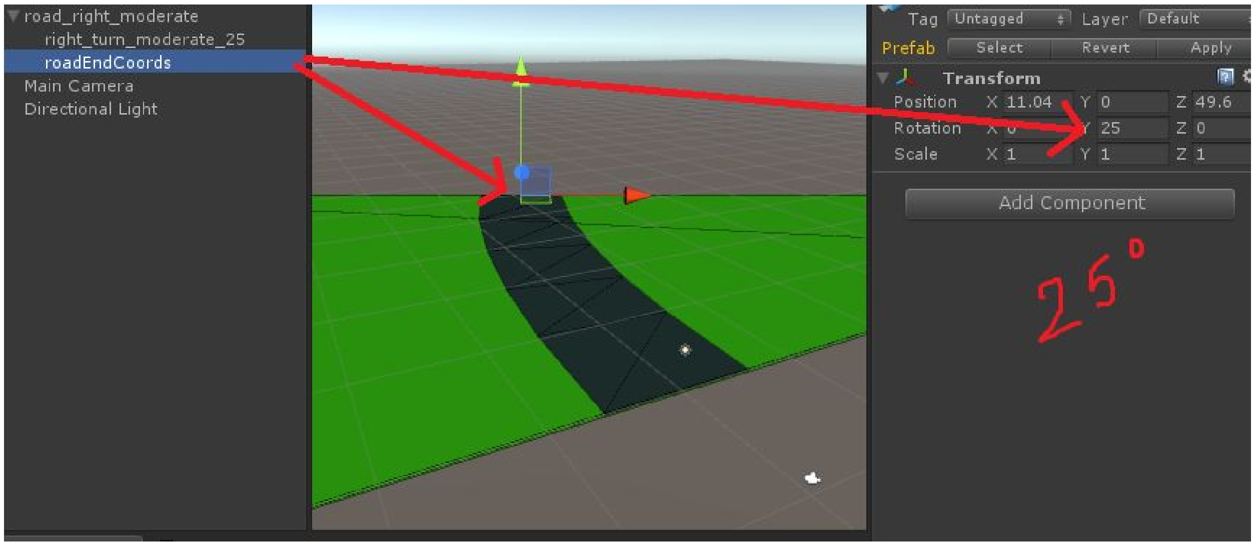
Drag your mesh right_turn_moderate_25.FBX as child to new created GameObject road_right_moderate as parent.

2. Create anchors for attaching of road segments to each other.

It's time to create the first of three necessary GameObjects that will be used for technical purposes: attaching road segments, AI navigation and scattering of environment props.

roadEndCoords - is the place where will be attached the next road segment. It needs to be placed at the end of the road segment and is turned to the angle which the roadbed is angled. In this example, the angle is 25, which we know because we've named our FBX turn_right_moderate_25. So, find the end of the roadbed, and create an empty GameObject. Name it **roadEndCoords**, then rotate it along the Y-axis to 25 degrees to the right and place it right on the edge of the canvas.

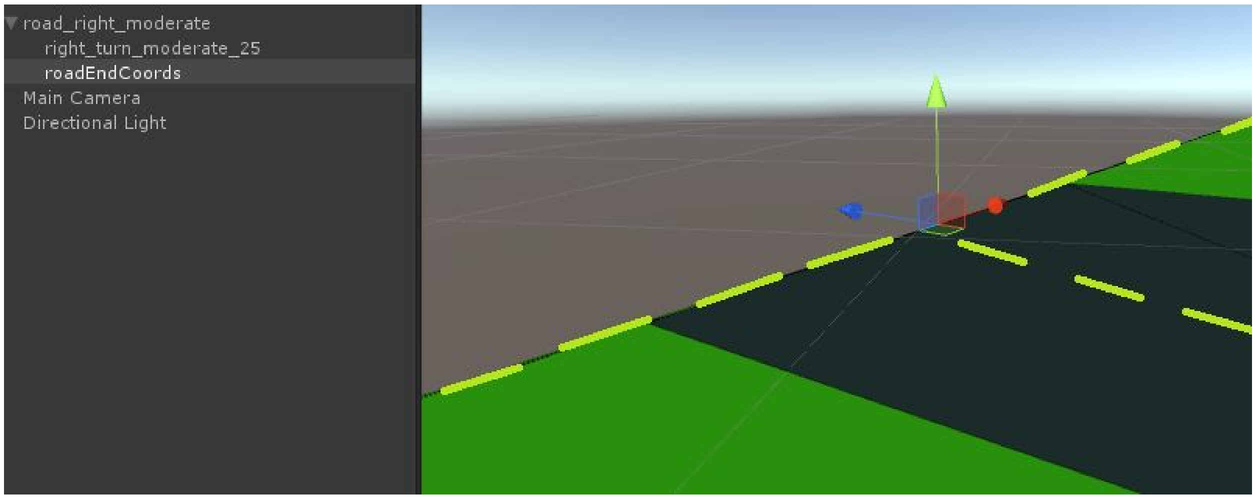
Everything like on those two pictures:



road_right_moderate
right_turn_moderate_25
roadEndCoords
Main Camera
Directional Light

Tag Untagged Layer Default
Prefab Select Revert Apply
Transform
Position X 11.04 Y 0 Z 49.6
Rotation X 0 Y 25 Z 0
Scale X 1 Y 1 Z 1
Add Component

Create empty GameObject, name it roadEndCoords**,** and put it into container road_right_moderate as child.
Turn this GameObject by actual angle of your mesh turn(25).

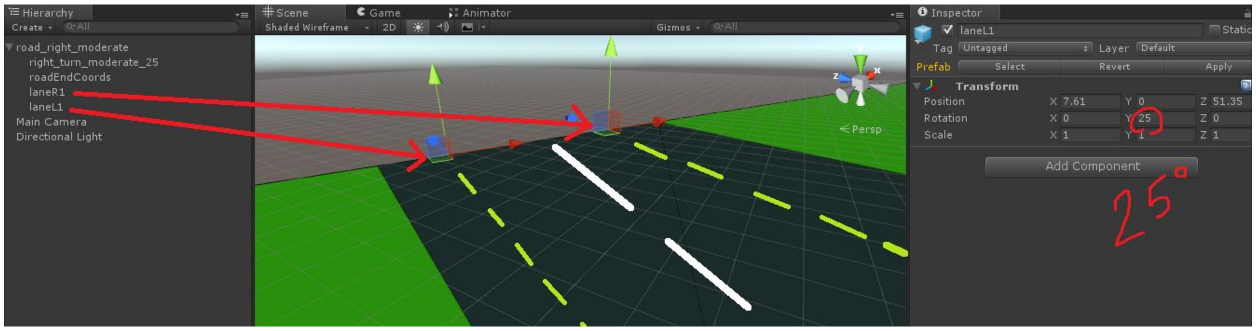


road_right_moderate
right_turn_moderate_25
roadEndCoords
Main Camera
Directional Light

Place roadEndCoords at the end of road. And place it very accurately in the middle and on the edge of the roadbed.

3. Create navigation points for the AI.

Next you need to create two empty GameObjects: **laneL1** and **laneR1**. These two objects that will be used as waypoints for vehicles driven by AI. Do it by simple way. Just duplicate roadEndCoords, change it's name to **laneL1**, switch the Unity's transition system to 'local' and move it left to the middle of the left lane. Do the same things for laneR1, and drag it to the right lane same way.



laneL1
Transform
Position X 7.61 Y 0 Z 51.35
Rotation X 0 Y 25 Z 0
Scale X 1 Y 1 Z 1

Place LaneL1 & LaneR1 at the end of road. Keep turn 25 same as roadEndCoords. And place it at middle of lanes(left and right)

4. Add component Physics/Mesh Collider

To be able to drive on this road, you need to assign a mesh collider to your road segment. Or, if you have a very complex road segment with lots of geometry and compute physics doesn't make sense, then connect a simplified mesh collider to this segment.

5. Assign script **roadDestruction.cs** on each prefab segments.

And the last thing we need to do is assign script **roadDestruction.cs** to GameObject **road_right_moderate**. If the player's vehicle is too far away from segment this script will destroy the entire road segment and all that it contains: obstacles, mountains, etc.

By default it's 300 meters. You might to change to any distance you need.

Save that as a prefab.

Seems quite easy, right ? Repeat all steps with the other 14 road segments .FBXs.

6. Other stuff like environment objects, traffic and so on.

You need just two actions with environment objects - assign collision meshes and to save them as a prefabs.

For the obstacles on the road I also suggest to add component - rigidbody. These objects could be involved in accidents computed by the physics In contrast to immobilized characters.

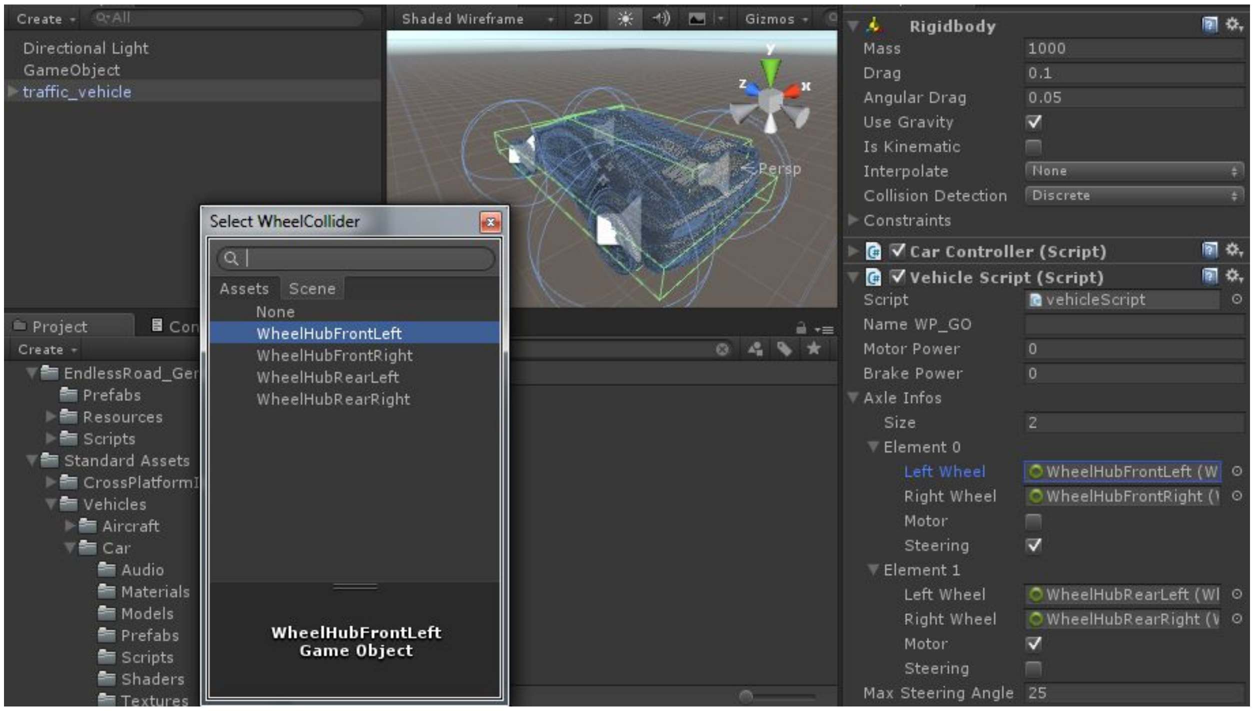
7.The most essential thing that we need is a vehicle.

We're not going to model it in 3D package, but just import the standard Unity Car - this is the best show how asset can be used for any vehicles you have.

That's how we'll get vehicles for traffic too.

First, we need to import standard Unity's car Assets/Import Packages/Vehicles.

Take the prefab of the Unity Car, remove all scripts except the CarController.cs. And drag to it script vehicleScript.cs. Then set up as shown in the picture below: the power of the motor 50, the brake 100, axis 2. Connect the necessary connection of the wheels with the Unity Standard Car's wheel colliders. Max steering angle is 25.
(Of course, all values are approximate - just to work in these conditions. In your project, with your own vehicle all those values should be relevant for your purposes)



Rigidbody
Mass 1000
Drag 0.1
Angular Drag 0.05
Use Gravity ✓
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints

Car Controller (Script)

Vehicle Script (Script)
Script vehicleScript
Name WP_GO
Motor Power 0
Brake Power 0
Axle Infos
Size 2
Element 0
Left Wheel WheelHubFrontLeft (W
Right Wheel WheelHubFrontRight (
Motor
Steering ✓
Element 1
Left Wheel WheelHubRearLeft (Wl
Right Wheel WheelHubRearRight (
Motor ✓
Steering
Max Steering Angle 25

Standard prefab of Unity Car with removed default scripts: CarUserContro.cs and CarAudio.cs. Do not touch CarController script.cs. Also, you need to assign script vehicleScript.cs to this Car's prefab and configure number of axis, links to wheel colliders and max steering angle.

Save it as new prefab, and name it to **traffic_vehicle.**

- **Part C**(last one) - connect everything to the main script **roadGenerate.cs.**

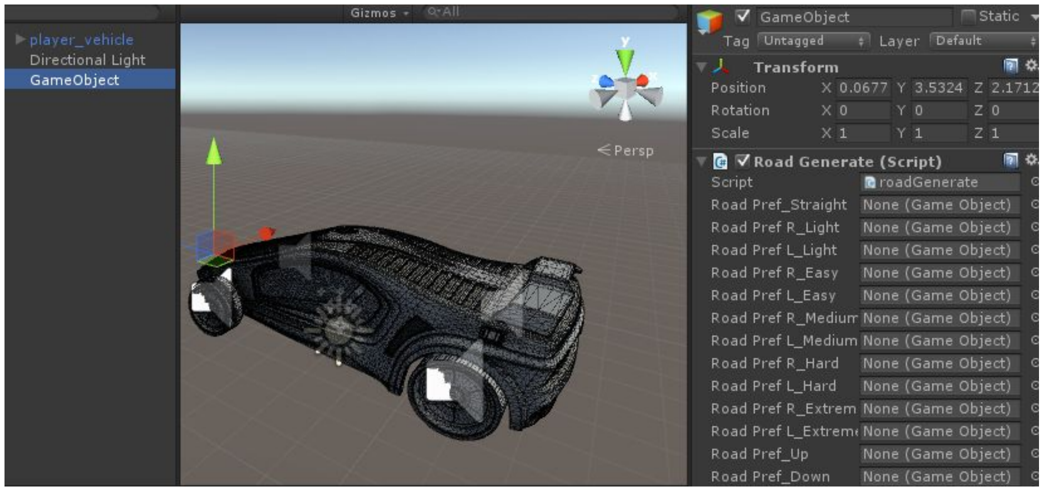
1. Set up new scene and create a player's car.

In a new clean scene, create an empty GameObject. Drag **roadGenerate.cs** script to this GameObject(it will be a container for this script only, just to make it work when scene is running).

Now drag prefab of standard Unity Car to scene and place it at any position, let it be: 0, 3, 0(3 meters above zero ground).

Also, you need to rename standard Unity Car to '**player_vehicle**'.

Attach the camera directly to the Unity Car.



player_vehicle
Directional Light
GameObject

GameObject Static
Tag Untagged Layer Default

Transform

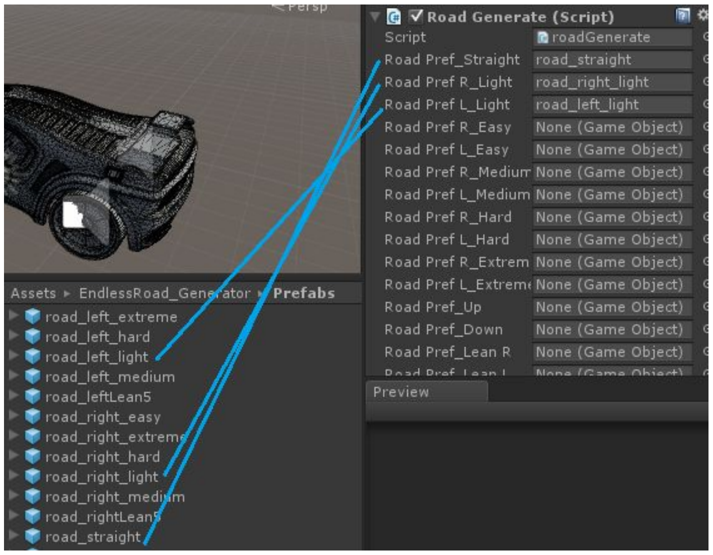
Position	X 0.0677	Y 3.5324	Z 2.1712
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Road Generate (Script)

Script	roadGenerate
Road Pref_Straight	None (Game Object)
Road Pref R_Light	None (Game Object)
Road Pref L_Light	None (Game Object)
Road Pref R_Easy	None (Game Object)
Road Pref L_Easy	None (Game Object)
Road Pref R_Mediur	None (Game Object)
Road Pref L_Medium	None (Game Object)
Road Pref R_Hard	None (Game Object)
Road Pref L_Hard	None (Game Object)
Road Pref R_Extrem	None (Game Object)
Road Pref L_Extrem	None (Game Object)
Road Pref_Up	None (Game Object)
Road Pref_Down	None (Game Object)

New scene. Standard prefab of Unity's Car just imported from Assets/Import Package/Vehicles. Car was renamed to **player_vehicle**. Also drag on a GameObject created a script **roadGenerate.cs**

Now connect all the empty slots roadGenerate.cs with previously prepared prefabs:



Road Generate (Script)

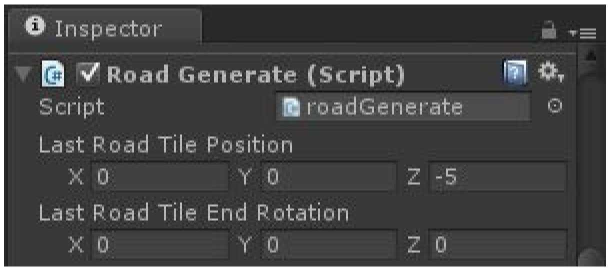
Script	roadGenerate
Road Pref_Straight	road_straight
Road Pref R_Light	road_right_light
Road Pref L_Light	road_left_light
Road Pref R_Easy	None (Game Object)
Road Pref L_Easy	None (Game Object)
Road Pref R_Medium	None (Game Object)
Road Pref L_Medium	None (Game Object)
Road Pref R_Hard	None (Game Object)
Road Pref L_Hard	None (Game Object)
Road Pref R_Extrem	None (Game Object)
Road Pref L_Extrem	None (Game Object)
Road Pref_Up	None (Game Object)
Road Pref_Down	None (Game Object)
Road Pref_Lean R	None (Game Object)

Assets ▸ EndlessRoad_Generator ▸ Prefabs

- road_left_extreme
- road_left_hard
- road_left_light
- road_left_medium
- road_leftLean5
- road_right_easy
- road_right_extreme
- road_right_hard
- road_right_light
- road_right_medium
- road_rightLean5
- road_straight

Now connect all the empty slots of the **roadGenerate.cs** with the early created prefabs.

Now indicate where the road should begin and where initially to be rotated.
We want the coordinates 0, 0, -5 - there will be built a straight segment that the player was comfortable to begin his movement.



Inspector

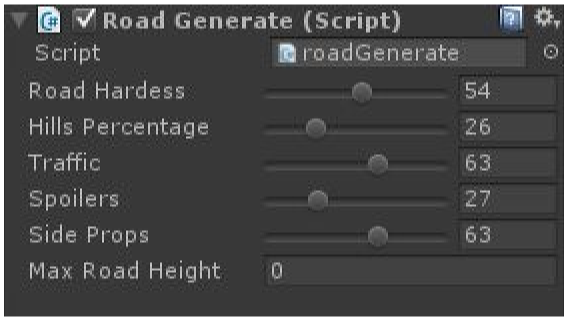
Road Generate (Script)

Script: roadGenerate

Last Road Tile Position
X 0 Y 0 Z -5

Last Road Tile End Rotation
X 0 Y 0 Z 0

If you want, you can pre-set in the Inspector in the script roadGenerate.cs approximate curvature of the road, its height, the presence of traffic and so on. If you do not, the default will be a straight road without traffic and obstacles.



Road Generate (Script)

Script	roadGenerate
Road Hardess	54
Hills Percentage	26
Traffic	63
Spoilers	27
Side Props	63
Max Road Height	0

It’s all ready !
Click ‘**Play**’ and you can drive to infinity on the infinity endless road.

Well, just for the sake of completeness, leaving no questions…

Not so important stuff

Next, I will describe other files of the project that are not directly related to the generation of the road and written mostly for the demo purposes. Also it can be used well as an example of your game template.

Here is a list of the files that are created for the demo and showing how you can use the asset to create your own game:

- **'FadeToSkybox' directory** full of scripts and shaders for the FX of the ground to sky 'soft' transition :

FadeToSkybox - this folder contains everything relates to FX a smooth transition ground into the sky. This FX written by Keijiro Takahashi, and is used with his kind permission.

- **Texture** files made especially for demo purposes:
  1. **Black_texture.png** - black texture for the FX of darkening the screen during the transition from scene to scene
  2. **Smoke_particles.tga** - dust/smoke texture for a situation when the vehicle leave the road and goes off-road
  3. **Skydome.tga** - sky texture.

- Additional **Prefabs**(with added collider-trigger) made for demo purposes
  1. **Gates.prefab** - prefab made of standard cubes(no fbx was made)

- **Scripts** written especially for demo purposes
  1. **camSwitcher.cs** - script switches two camera modes: chase and orbit by right mouse button
  2. **distanceCounter.cs** - script calculates the distance, which player has drove
  3. **fadeIn.cs** - script for darkening the screen when scene changes one to another
  4. **gameTypeSelect.cs** - the script for scene selection in main menu
  5. **gatesCreator.cs** - script that creates the gates for the finish line
  6. **gatesScript.cs** - script for the finish line gates, watching when the player will pass the gate and finish the race
  7. **onScreenConrols.cs** - script for onscreen GUI controls. To gain the ability to adjust the road generation in real time.
  8. **timerCountDown.cs** - script of countdown timer
  9. **toMainMenu.cs** - script for return to the main menu from any demo scenes
  10. **vehiclePlayerScript.cs** - script that allows you to drive the vehicle yourself(almost a classic wheelCollider script from the Unity Tutorial)

- **Scenes** made especially for demo purposes
    1. **demo_Select.scene** - demo main menu to select demo's type gameplay
    2. **demo_curvatureControls.scene** - demo scene with onscreen GUI controls
    3. **demo_countDown.scene** - demo scene with 'time eliminator' race

Version changes:

1.0 - first release

That's all.
Feel free to ask me anything about that asset by e-mail: **smokerr@mail.ru**
Boris Chuprin 2017