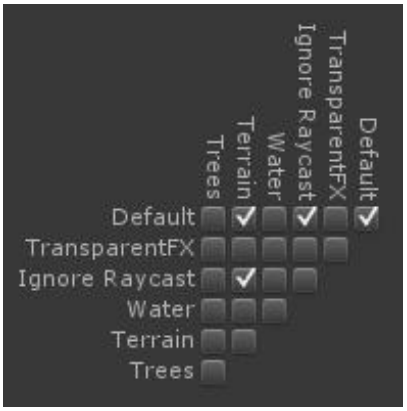
Features

- **Procedural Terrain Creation** – Infinite variations, fully controllable.
- **Dynamic Terrain Shader** – Make modifications, no need to repaint.
- **Custom Tree Placement** – And not limited to trees. Color tinting for extra variation.
- **Vehicles** – Tank to drive around, Helicopter to fly around. Fully physics-driven.
- **Turret** – aim with the mouse, left mouse button to fire.
- Optional **TNet**-based **Multiplayer**. Choose a vehicle then blow up your friends!

Installation

1. Set up the layers as follows:
    a. Edit -> Project Settings -> Tags
    b. Add “Terrain” as User Layer 8
    c. Add “Trees” as User Layer 9

2. Set up the Collision Matrix
    a. Edit -> Project Settings -> Physics
    b. Set up the matrix like so:



3. You will want to check “Run in Background” option. **Multiplayer won’t function properly without it!**
    a. File -> Build Settings -> Player Settings
    b. Resolution and Presentation section
    c. Check “Run in background”

Project Structure

Terrain Kit (#1)

- **Terrain Kit** contains the terrain generation logic and all the relevant scripts and resources. If you are only interested in the terrain generation aspect of this kit, then this is the only folder you need.
- **LibNoise** is a free open-source library that the kit uses in order to generate terrains.
- Tree models can also be found in this folder inside the Models subfolder.

Game (#2)

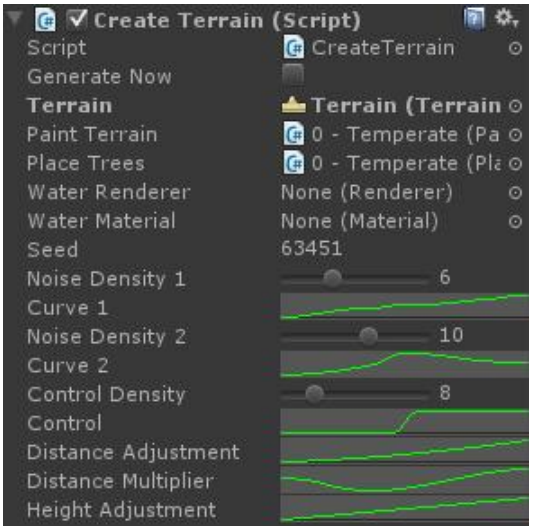
- The **Game** folder contains everything related to the actual game prototype. This is where you will find the **Tank** and **Helicopter** models, explosion particle emitters, sounds, and all the relevant scripts that tie them together.
- Don't worry if you don't see the "**Menu**" and "**Multiplayer**" scenes. They come with the **TNet** version of the package and can be imported by double-clicking on the optional unity package found in the MPGSK folder. You will need to have TNet imported in order to use multiplayer features.

Terrain Kit

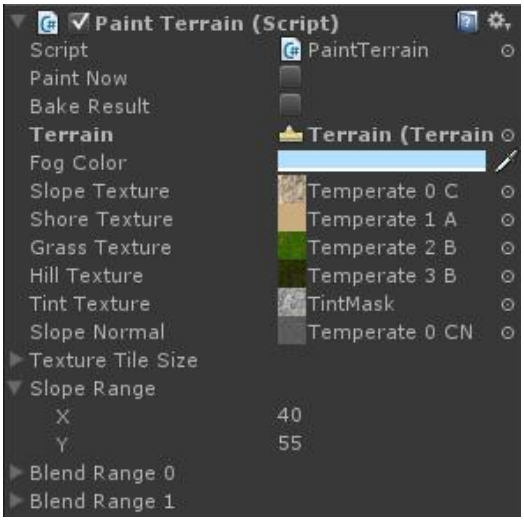
Terrain generator scene can be found in the Terrain Kit folder. It contains the Procedural Terrain Creator prefab as well as a basic movable camera to help you navigate around.

- **Procedural Terrain Creator** object contains the Example Terrain GUI script on it. This is the script that's responsible for drawing the GUI you see when you hit Play.

- **Climates** can be found right underneath the procedural terrain creator. They are **Temperate**, **Barren**, **Tropics**, and **Tundra**. Each one contains 3 scripts on it – one to create the terrain, one to paint it, and the last one to plant trees on top of it.



Create Terrain script is the main one. It's what generates a procedural terrain using the parameters of your choice. The terrain is generated using 3 Perlin noises. One noise generates lowlands, another – highlands, and the third noise determines how the first two are mixed together. The script contains a variety of options and curves for you to play with. Feel free to play with them at Edit time. When you want to re-generate the terrain, click on the "**Generate Now**" checkbox.



Paint Terrain is what paints the terrains based on height of each vertex and the properties of your choice. If you modify the terrain in real time, you will notice the terrain get repainted right away. This is because the Paint Terrain script works with a dynamic shader rather than with splat maps, allowing the terrain remain a fraction of its usual size. If you change the properties, you will want to use the "**Generate Now**" checkbox.

Place Trees script is responsible for planting trees based on the terrain elevation and the properties of your choice.

Note that the trees are regular meshes, not Unity trees! This has its benefits and downsides, so whether you will want to use this script is up to you. But it does let you use all regular models here, not just those that use the "nature" shaders. Rocks, crystals, debris… whatever you like.

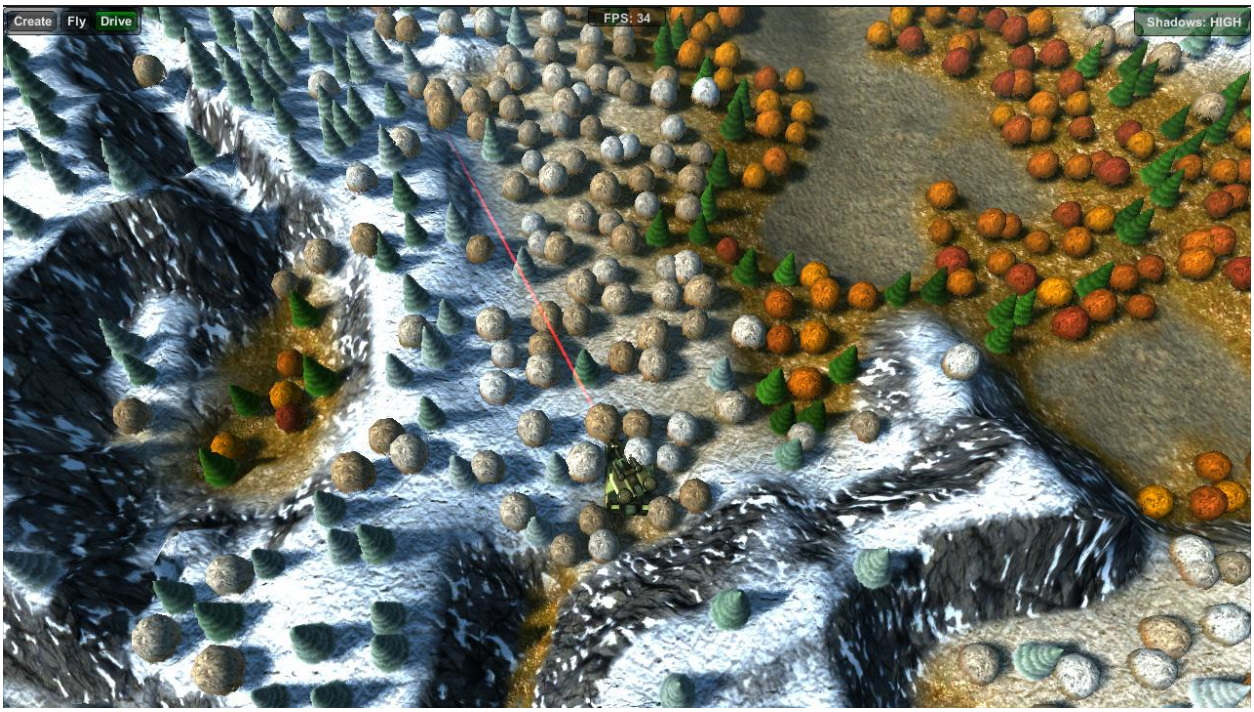
You can specify an array of "trees", the ideal height of where they will show up, as well as the height variance. For example if the palm trees are meant to grow from height of 0.25 to 2.25, you would specify the ideal height of 1.25, with a variance of 1.0.

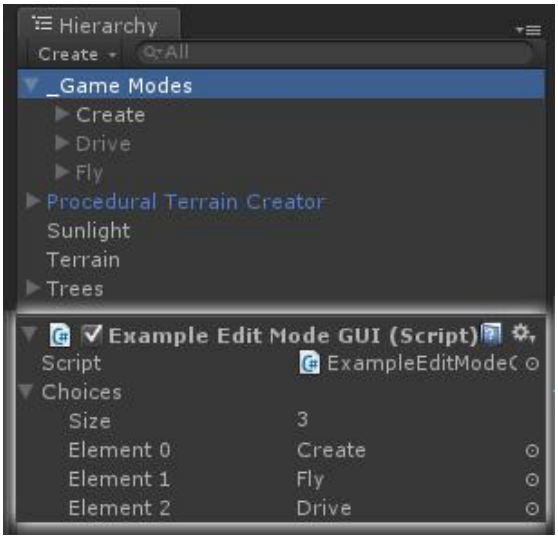
The "**Layer Mask**" is what controls which layers are considered to be blockers for tree placement. By default it's marked to be "default", meaning that everything on the default layer will be considered to block tree placement in the vicinity.

Once you hit Play, all trees get merged into few draw calls based on a certain "cell size" – usually into batches of around 5,000 to 10,000 vertices per draw call. If you're wondering what takes so long when you hit Play – this is it. Disabling trees will make the level load up instantly.

Game

Game folder contains everything related to the game prototype itself. It relies on the Terrain Kit for the terrain. **Singleplayer** scene contains the actual game logic inside, letting you not only create terrains, but also giving you the freedom to explore them by driving or flying around.





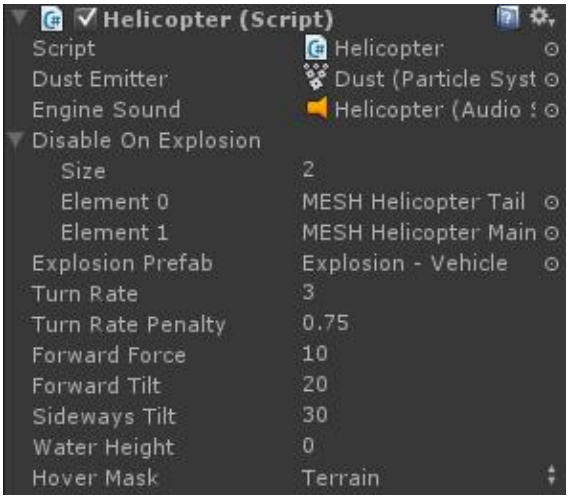
Game Modes object contains the GUI script that lets you change game modes shown in the top left corner.

You will still find the familiar **Example Terrain GUI** script attached to the **Create** object, but other objects don't have anything except children underneath.

You may notice that each of the 3 game modes contains its own camera pivot, and each camera pivot has the **Input Manager** script attached. Input Manager is what records the axes from Unity's Input, storing them in a static, easily-accessible location. This approach lets you easily change which axes do what without having to modify the code.

Input Manager's values are used to control the Vehicles (**Tank** and **Helicopter**).

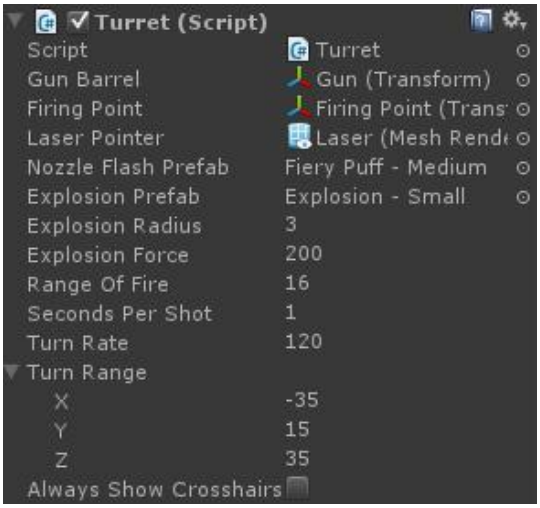
Follow Camera Target script attached to each Camera Pivot determines how quickly the camera follows the **Camera Target** object – a script found on both the **Helicopter** and the **Tank**.



Speaking of vehicles themselves, each one has an assortment of properties that control how they behave. **Helicopter** lets you adjust the turn rate, forward movement force (what makes it fly forward), as well as tilt angles that determine how much the helicopter is able to tilt when flying around.

The **Tank** has even more options that control the properties of its engine and wheels. The tank also has a "**Water Explosion**" property. If you have **Tasharen Water** and drop it into the scene, you can make it possible for the tank to drown by specifying **Explosion – Water** prefab for it.

If a water explosion is specified, the tank will use the "**Drown Height**" property to determine when it should be considered to be drowned. Go ahead, try it! It will make more sense when you add water and drive into it.



In both vehicles aiming and firing is achieved via the **Turret** script found below the Renderers object within the vehicle's hierarchy. This is where you can specify the **nozzle flash** that appears when you fire, as well as the **explosion** that gets created in the distance.

You can also choose the **explosion radius** and the amount of **force** applied at the origin. **Range of Fire** and **Seconds per Shot** properties can also be found here. You may also notice that while the Tank's Turret can rotate the full 360 degrees (Turn Range's Z), the Helicopter's only gives 35 degrees of rotation, making it only possible to shoot at what's in front of it.

If you play multiplayer, you will notice that the game is set up to one-shot kill on a direct hit. If you wanted to add damage mechanics instead, you would do that by opening up the Vehicle class and finding the line marked as "TODO":

```
// TODO: This would be the place to deal damage and subtract from the vehicle's health
```

Of course for the damage to make sense you'd have to have more than one vehicle in the game, which brings us to the last topic… multiplayer with TNet.

TNet Integration

Have **TNet**? Fantastic. You get more content! Import **TNet** then double-click on the **TNet Integration** unitypackage. Choose to overwrite the files it prompts for. Add the Menu and Multi Player scenes to the build list and hit Build. That's it!

If you select the vehicles (**Helicopter** or **Tank**), you will notice that they now have a **TNObject** and a **TNAutoSync** script on them. Auto Sync is set to synchronize the position, rotation and velocity of the vehicle, and only 4 times per second. Yet if you hit Play, you will notice that everything is nice and smooth. How is it accomplished? Quite simple:

TNAutoSync is used to force-synchronize the values every so often. You can safely drop the **Updates Per Second** down to 1 and you won't notice any difference. Everything remains in sync because the vehicles themselves synchronize the input axes in the Update function once they change significantly enough. With the input in sync, vehicles will move the same on all clients, and AutoSync takes care of forcing them into the correct position from time to time ensuring that the positions don't deviate too much.

When something changes, such as an explosion occurs (**Vehicle.AddExplosionForce** function), **TNAutoSync** is asked to synchronize the values right away in order to ensure that everything is proper.

The amount of **jitter** is also minimized by separating the Physics from the Renderers. If you select the Renderers object under each vehicle, you will notice that it has the "**Vehicle Renderers**" script on it. This script adjusts the position and rotation smoothly, ensuring that there are no sudden "jumps". This lets the physics always be in the right spot, but the visual renderer remains nice and smooth trailing ever-so-slightly behind.

- **NOTE: Tasharen Water is not included, and is available as a separate package.**

Questions, Comments and Support

Although the package was designed to be as intuitive as possible, I know what's intuitive for some may not be for others, so I am always available to answer questions. The best way to do so is to go to the forum (http://www.tasharen.com/forum/index.php?board=6.0) and ask there. Due to the very high volume of correspondence I am no longer able to answer direct support-related emails as they can't help others – but the forum posts certainly can.

So TLDR version: have a question? Just ask on the forum. ☺