Insane Systems RTS Starter Kit Guide

Content

Introduction

RTS Starter Kit asset is created to simplify development of RTS game. This asset is primary designed to create military RTS, which have vehicles, bullets, etc., but you can use it to other game styles. It contains a lot of components, which you need to know to work with asset and this guide help you to do it. But do not worry, these components are simple to understand.

Guide contains three main partitions:

- **Information** - information about asset objects, resources and components.
- **Settings and customization** - information about asset settings, which help you to customize examples for your RTS.
- **Programming** - this partition contains information, which help you to implement your own functionality.

You also can open this guide in google docs by using this link: https://docs.google.com/document/d/1jM-qJoNewQ2HnpzaUbwVG6VSX94sXPAGT5YRK15mUDA/edit?usp=sharing

In Google Docs you can quickly navigate between partitions by using Document Structure on left side of window.

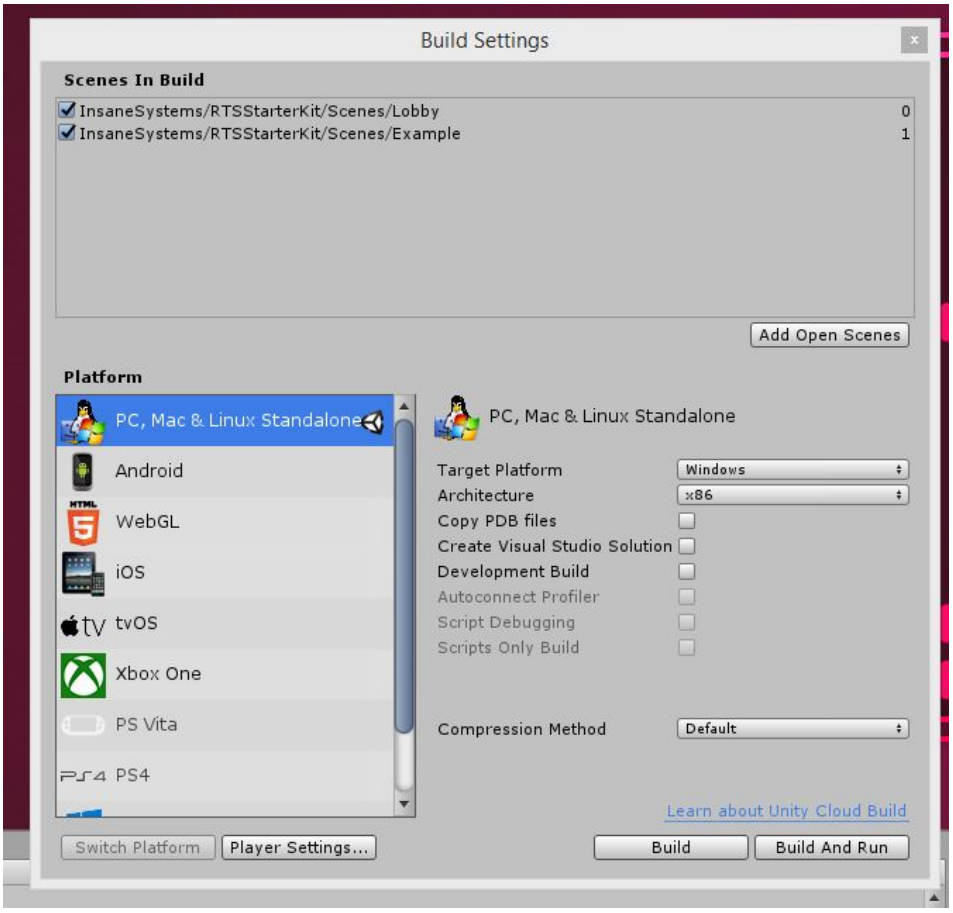
Information

Quick Start

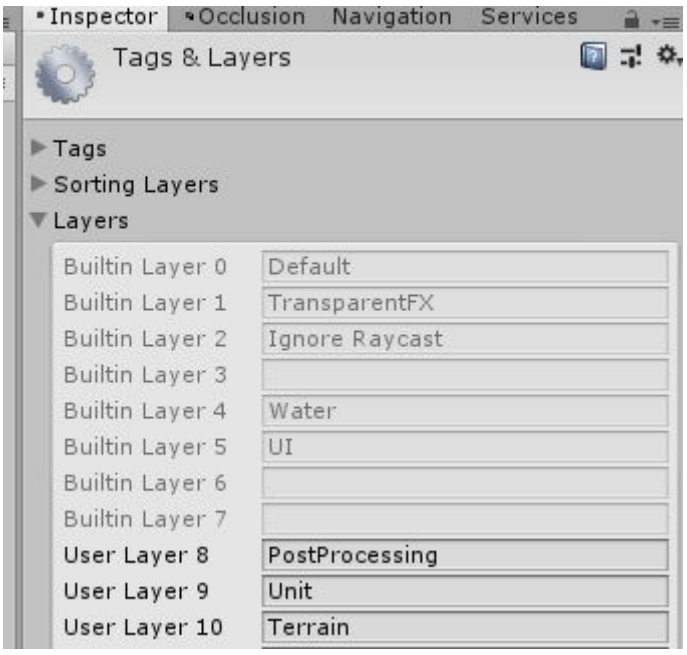
Best way to understand, how it all works - check the **Lobby** and **Example** scenes. It contains all used components which shown in simple RTS Game prototype. Have a detailed look to it and if you have questions - go ahead and read next sections of this guide.

Project preparation

Firstly, to run prototype properly, check that **Lobby** and **Example** scenes added to **Build Settings**. It should be done automatically by code, but if not, add it manually.



Next, you can check that asset added two new layers: **Unit** and **Terrain**. First is used for indicate ground object on maps, second – for all game buildings and units.



It should be done automatically, if it not done, please, add this layers manually.

Unit modules description

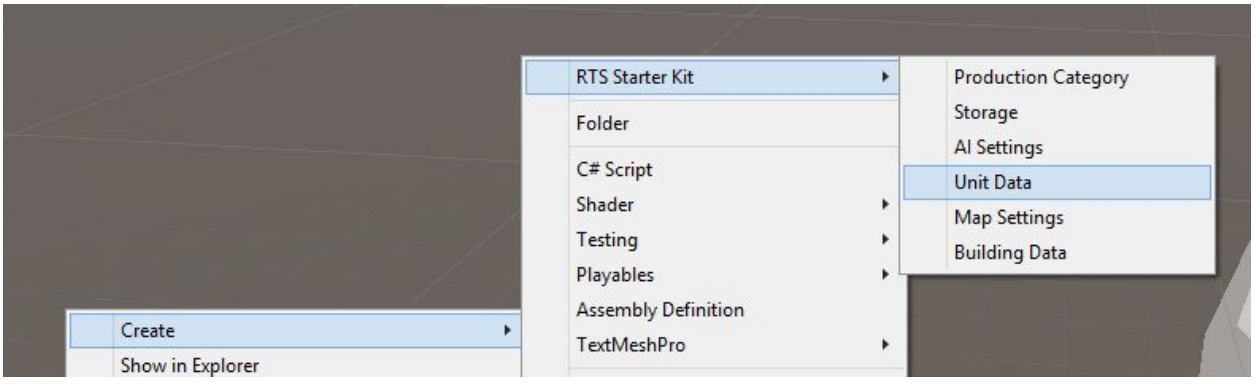
RTS Starter Kit is uses module system to customize game units and buildings, and there a list of primary modules:

- **Unit** – core module of any unit. Contains primary unit info and links other modules together.
- **Damageable** – makes an object possible to take damage. This component contains unit health info etc.
- **Movable** – this module represents unit moving system. If you want allow unit to move, add this module.
- **Attackable** – attack module. Unit, which have this module, gets the ability to attack other units.
- **Tower** – this module is needed for vehicles like tanks to rotate its tower to the targets.
- **Production** – allows this object to build units or buildings.
- **Harvester** – allows unit to gather resources.

Unit module is simple component, which can be drag n dropped to **Prefab** as any other **Unity** component or script. You can create your own modules and connect it to existing, if you needed. This described in **Programming** partition.

Unit data description

Unit data is an object in resources (inherited from **ScriptableObject** class), which contains all settings of one unit type. You can create new **Unit Data** by right-clicking **Project Window** and selecting in context menu option **RTS Starter Kit/Unit Data**.

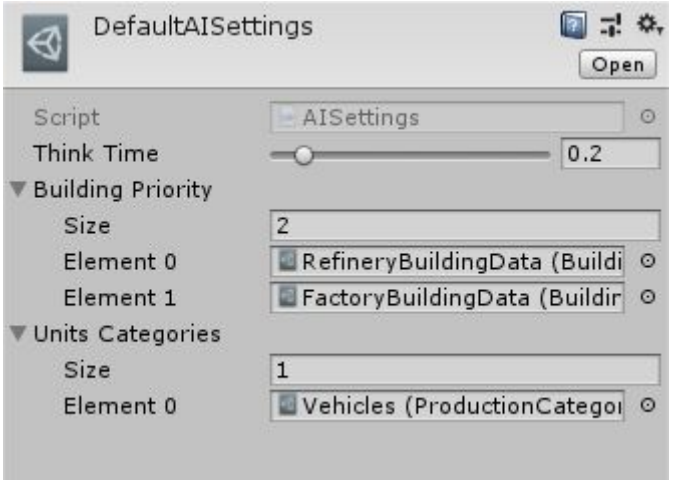


Production categories description

Production category is a list of units or buildings, which can be created by some unit or building with **Production** module. You can create your own production categories or change existing to set up custom lists of allowed buildings or units.

AI Description

This asset contains bots (computer enemies). Currently their AI is simple, but flexible for extend. Now AI Bots can create buildings, buy units and send it to attack enemies. Current **AI Settings** looks like this:



Think time is time between different AI Actions. Increase it to decrease bots actions per minute.

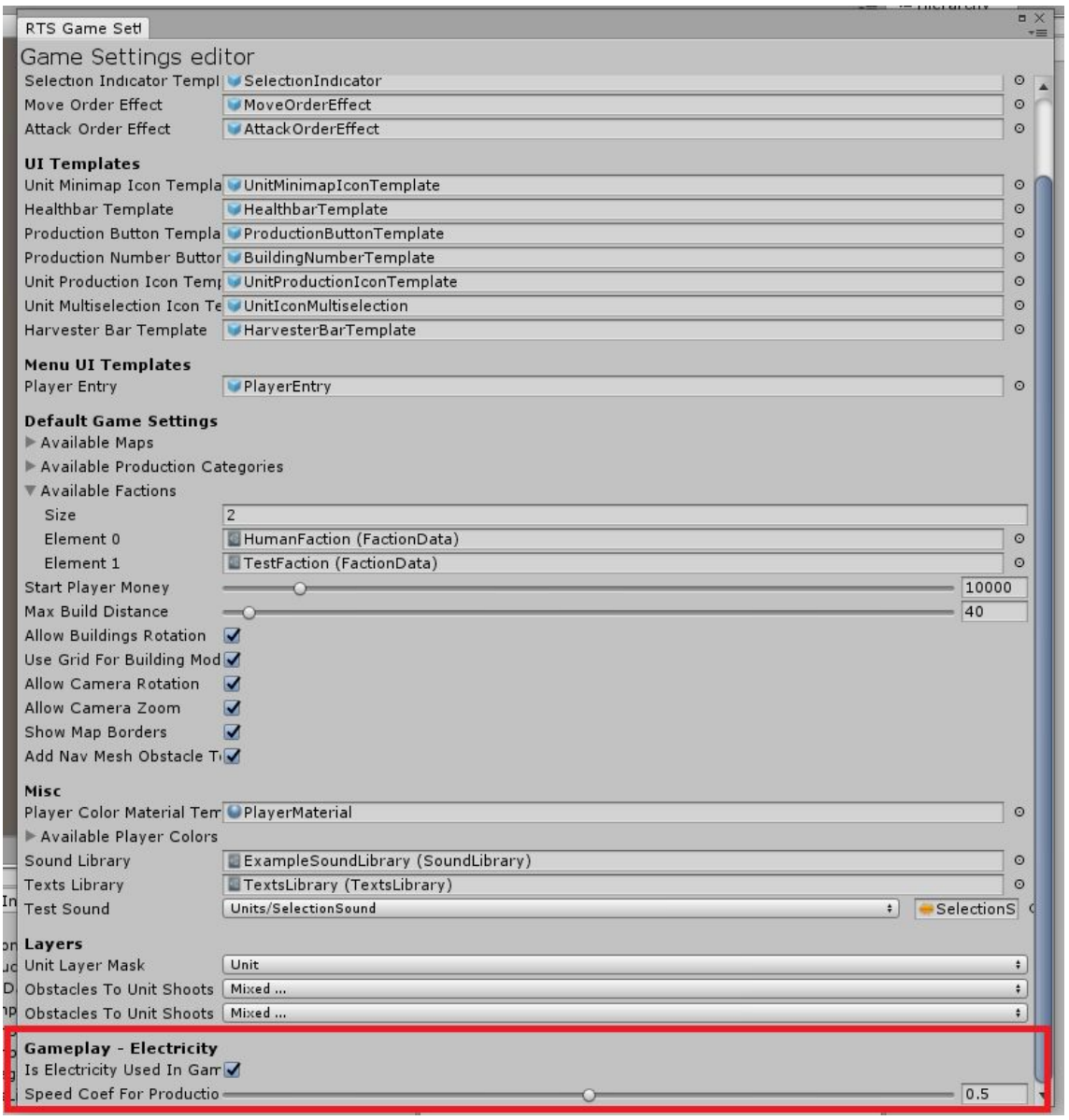
Building Priority is list of buildings, allowed to build for bots. Bots will create them in order of list.

Units Categories contains list of **ProductionCategory** objects, which can be built by bots. For example, if you add default Factory Building to **Building Priority** and want bots to build units from this factory, you should add to **Units Categories** its production category. In template this category is **Vehicles**.

Electricity mechanic

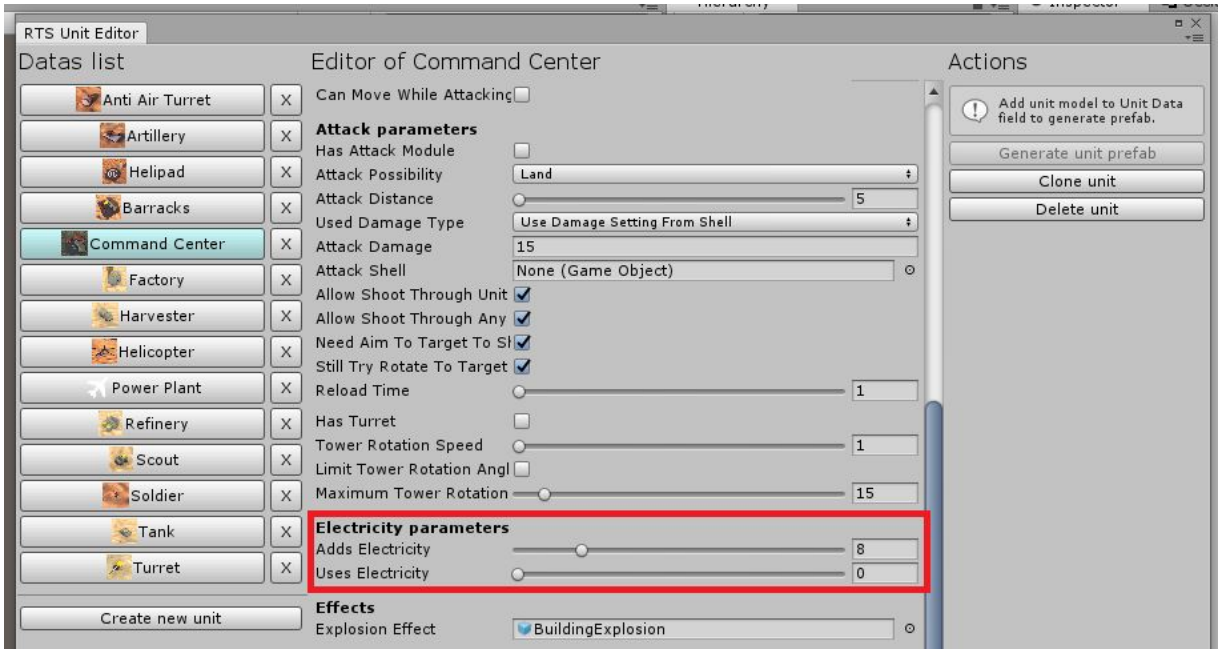
Electricity used in a lot of classic military RTS, so we decided that it will be good, if thing like this will be implemented in our asset.

This means that some buildings will produce energy (power plants, etc), which will be used by another buildings (factories, etc). To enable/disable this mechanic, you just need to edit this toggle in **Game Settings** (you can find it in top menu - RTS Starter Kit or just search Storage file, will be described below):



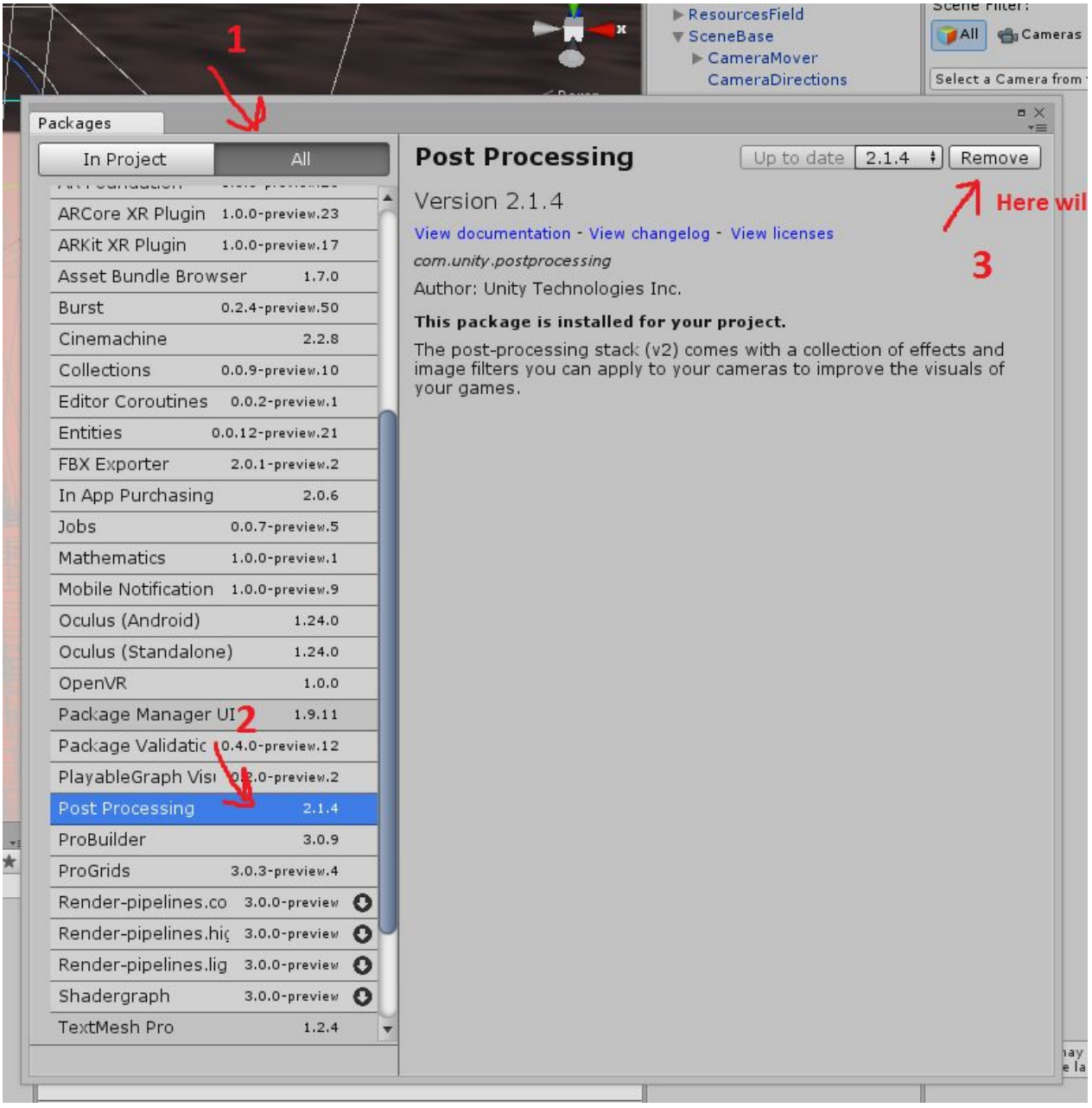
So, if player will be out of electricity, his units productions will be disabled or slowed down until electricity will not be restored.

Change adding or using by units electricity you can in Unit Data of needed units:



Dependencies

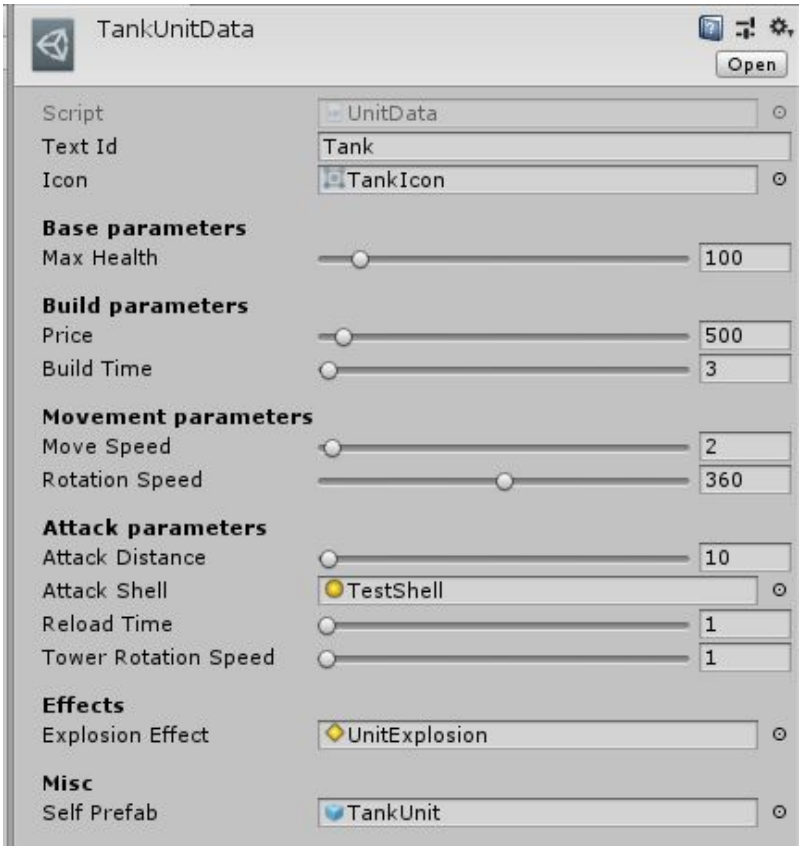
Asset uses **Post Processing Stack v2** for better graphics, so if you don't add it in **Package Manager** or by other way, it will not work. To add it from package manager, select in Unity top menu **Window** > **Package Manager**, in opened window do next steps from screenshot:


1
Packages
In Project
All
ARCore XR Plugin 1.0.0-preview.23
ARKit XR Plugin 1.0.0-preview.17
Asset Bundle Browser 1.7.0
Burst 0.2.4-preview.50
Cinemachine 2.2.8
Collections 0.0.9-preview.10
Editor Coroutines 0.0.2-preview.1
Entities 0.0.12-preview.21
FBX Exporter 2.0.1-preview.2
In App Purchasing 2.0.6
Jobs 0.0.7-preview.5
Mathematics 1.0.0-preview.1
Mobile Notification 1.0.0-preview.9
Oculus (Android) 1.24.0
Oculus (Standalone) 1.24.0
OpenVR 1.0.0
Package Manager UI 1.9.11
2
Post Processing 2.1.4
ProBuilder 3.0.9
ProGrids 3.0.3-preview.4
Render-pipelines.co 3.0.0-preview
Render-pipelines.hig 3.0.0-preview
Render-pipelines.lig 3.0.0-preview
Shadergraph 3.0.0-preview
TextMesh Pro 1.2.4
Post Processing
Up to date
2.1.4
Remove
Here wil
3
Version 2.1.4
View documentation - View changelog - View licenses
com.unity.postprocessing
Author: Unity Technologies Inc.
This package is installed for your project.
The post-processing stack (v2) comes with a collection of effects and image filters you can apply to your cameras to improve the visuals of your games.
ResourcesField
SceneBase
CameraMover
CameraDirections
All
Cameras
Select a Camera from

Settings and customization

How to change unit settings?

Most of unit settings are available in **UnitData** object in **Resources/Data** folder (but you can place it in any other location if you want). You just need to find data of needed unit and edit it settings. Or you can create new **Unit Data** (for new unit), if you need it. To create new **Unit Data** object, just **right-click** in **Project Window**, and in context menu select **Create -> RTS Starter Kit-> Unit Data**.



All settings are easy to understand, some of them have help hints (you just need move cursor to setting name and wait for hint appear) with additional info.

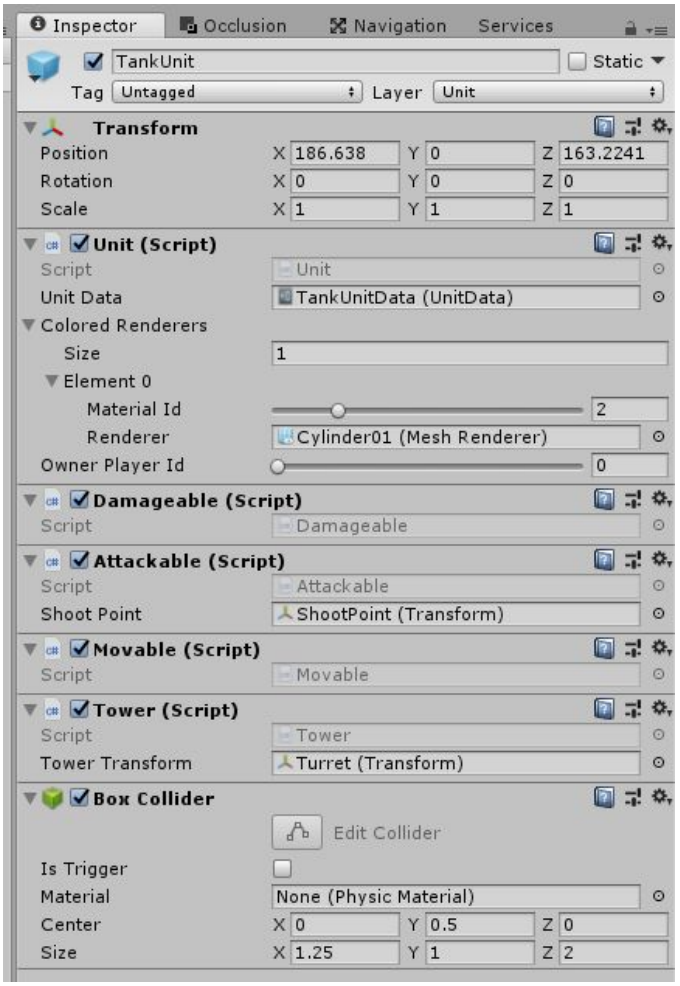
How to setup new unit?

You need complete few steps to do it:

1. Create (or clone existing in example) prefab for unit, add **Unit** component to it. **Unit** object should be in layer **Unit**.

2. Create **Unit Data** (how to do it you can read in previous section) with settings for this unit, set up your settings.

3. Drag And Drop this **Unit Data** to **Unit** component on your prefab in "**Unit Data**" field.



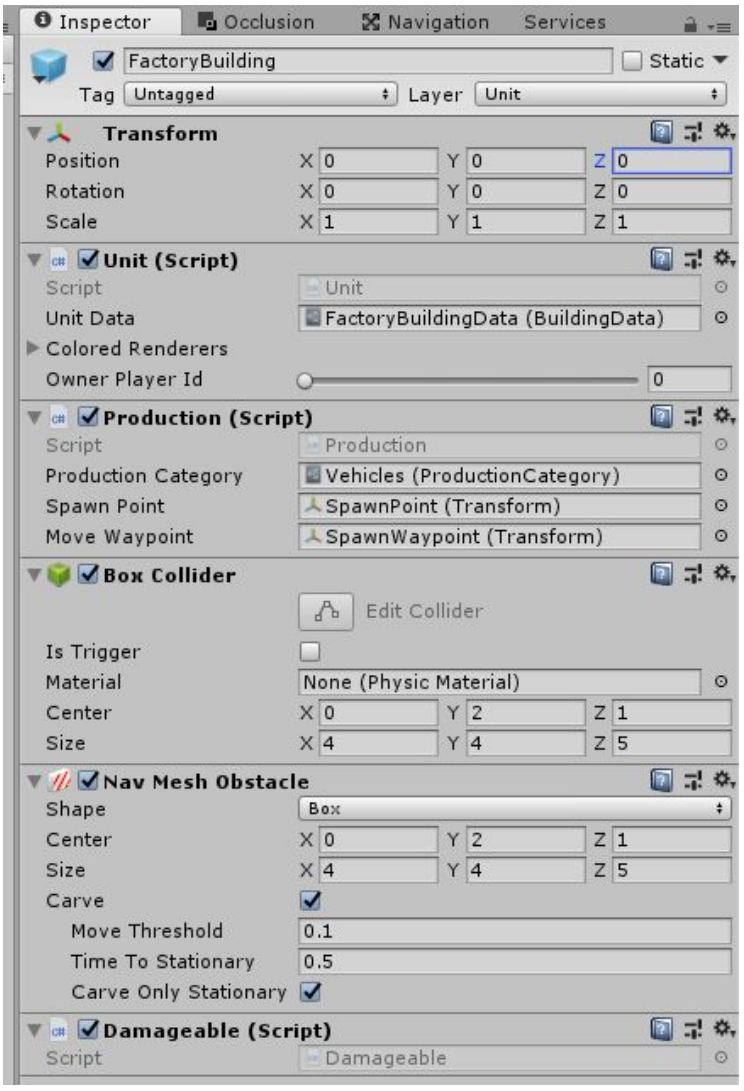
Now you can add other modules to unit to customize it as you want.

You also can **generate prefab automatically**, this is best way to setup base of unit from its data. After doing it you can customize its prefab as you want. See **Automatic unit/building setup** partition below for more details.

How to setup new building?

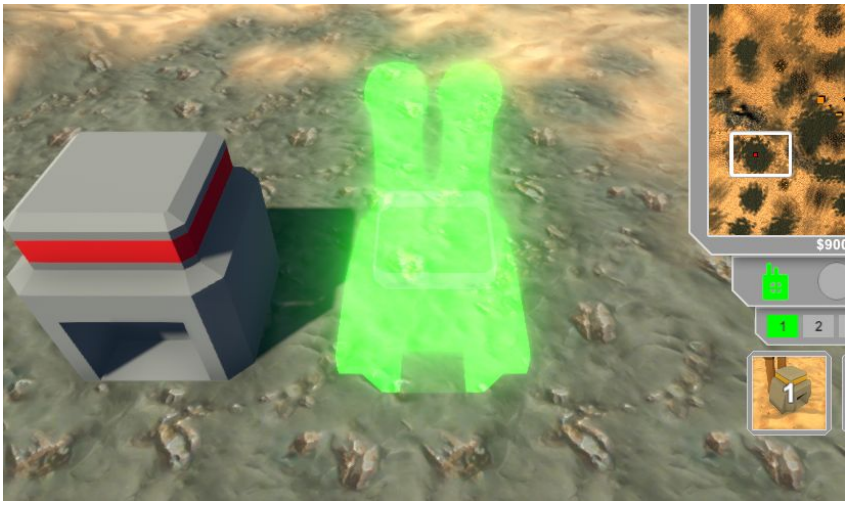
Same as usual unit, but set toggle **Is Building** to **true** in **UnitData**. You also can add **Nav Mesh Obstacle** with **Carve** parameter on to better units pathfinding. If you don’t do it, **Nav Mesh Obstacle** component will be added automatically when

building spawns, and its parameters will be same to building **Box Collider** parameters.

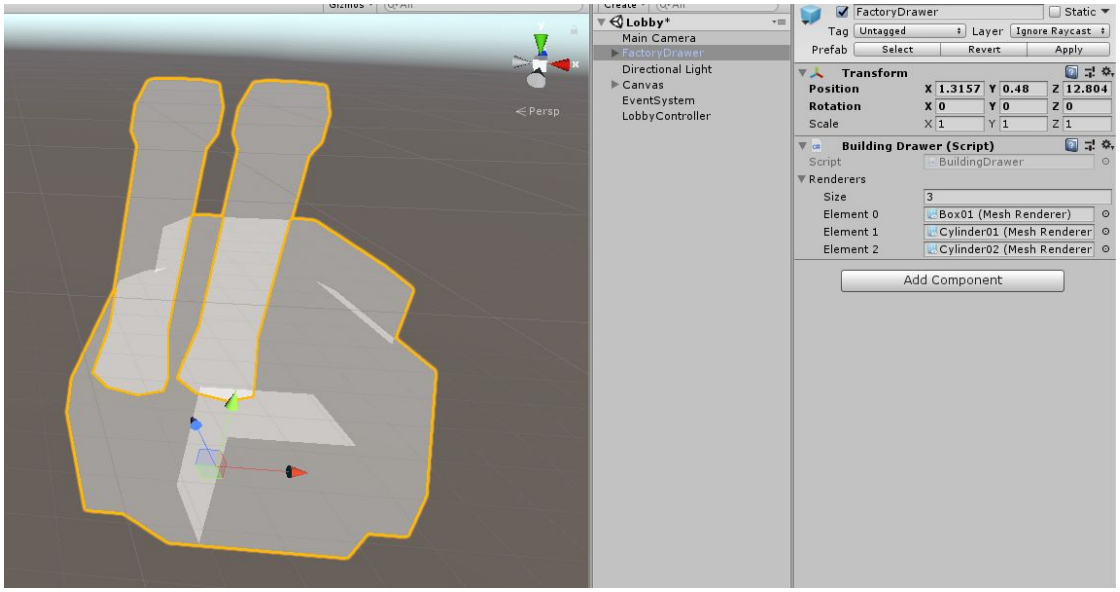


Also for buildings, which should produce units, you need to add **Production** component and setup it with needed **Production Category** object and other settings.

Buildings have one feature – **building drawers**. This is object, which shown when player is in Build Mode:



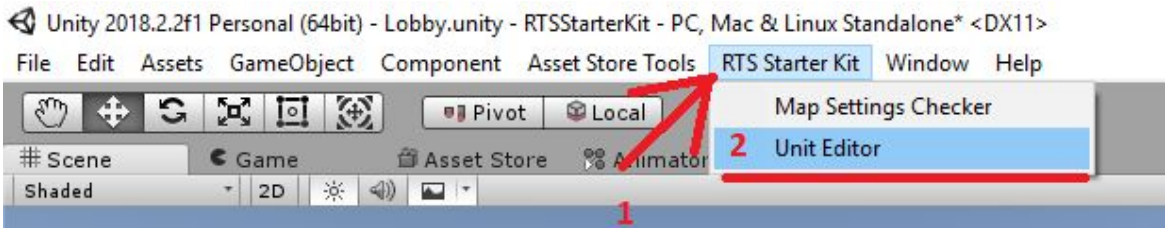
You should set up it for every building, which will be available for build by player. To do this, you need to create empty object, add model of needed building to it, and add the **Building Drawer** component to it. Next, drag'n'drop all meshes of building model to the **Renderers** field of **Building Drawer**. Finally, it should look like this:



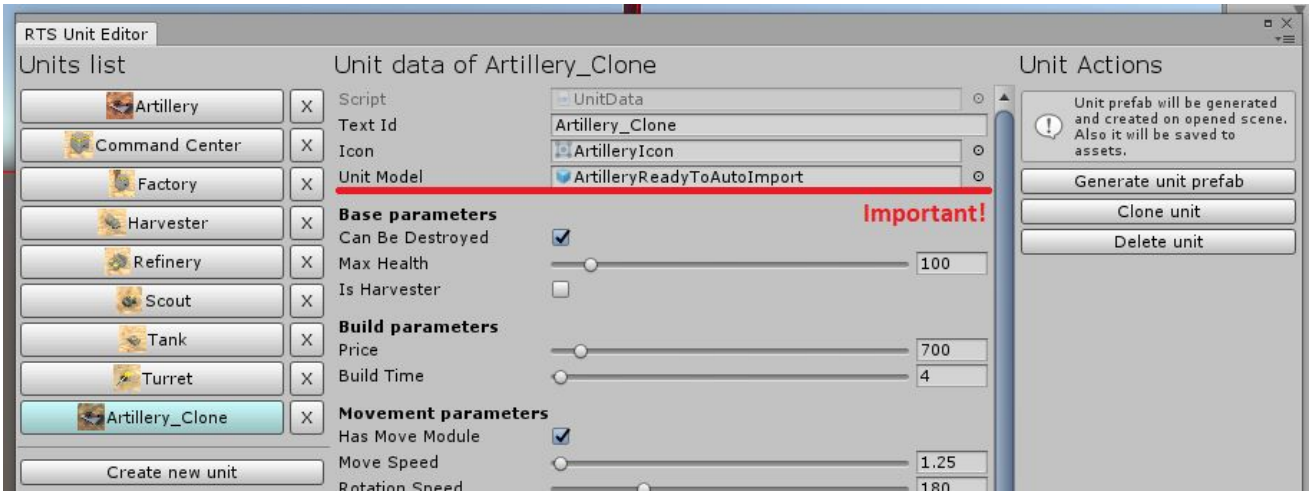
Transparent material named as **BuildingDrawer**, you can find it in **ExtraResources** folder and add to every mesh of drawer. You can also use your own material type with another effect.

Automatic unit/building setup

Since version **1.4.0** you're able to setup units semi-automatically. It can be done using **Units Editor** window. Open this window by clicking **top menu -> RTS Starter Kit -> Units Editor**:



In opened window select needed unit (if you need to create new unit, click **Create new unit** button. You're also can clone another unit by selecting it and clicking **Clone** button). Setup all parameters as you need, and don't forget to fill **Unit Model** field by model of your unit from assets:



Finally, you need to click **Generate unit prefab** button in right part of window. After click, created unit prefab will be spawned on scene and saved in **Prefabs/Units** folder. A lot of it's settings will be automatically configured, but there still some params, which you should setup manually:

1. Unit colored renderers - parts of unit, which shows its house color (read next partitions)
2. Collider size - size of your unit.

Other settings can be adjusted manually to, if you want.

Important thing is auto setup of model and bones.

First, your model will be rotated properly only if it's pivot is right for Unity (Z-axis is forward axis of your unit model).

Second, you can add several bones (empty objects or model parts) to your model, which will speedup setup of unit. It's namings listed below:

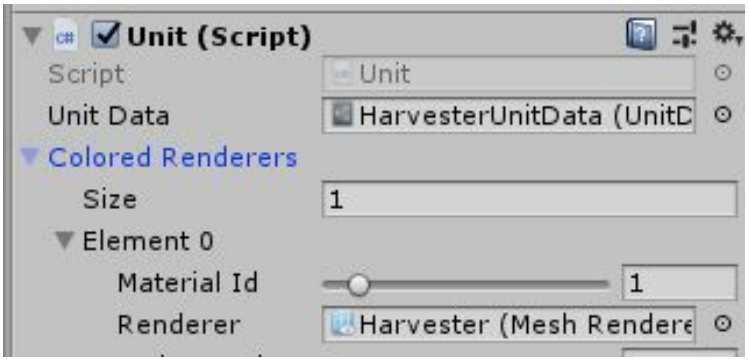
1. **WheelBone** - if unit has wheels, name it's bones like this, it will be automatically used by scripts.
2. **ShootPoint** - points, from which will shooted bullets by unit. In most cases you need only one shootpoint.
3. **TurretBone** - if your unit has a turret, you need name its bone like this to allow scripts automatically find object to rotate turret to enemies.

You can find example model of unit in project, just input in search **ArtilleryReadyToAutoImport** phrase.

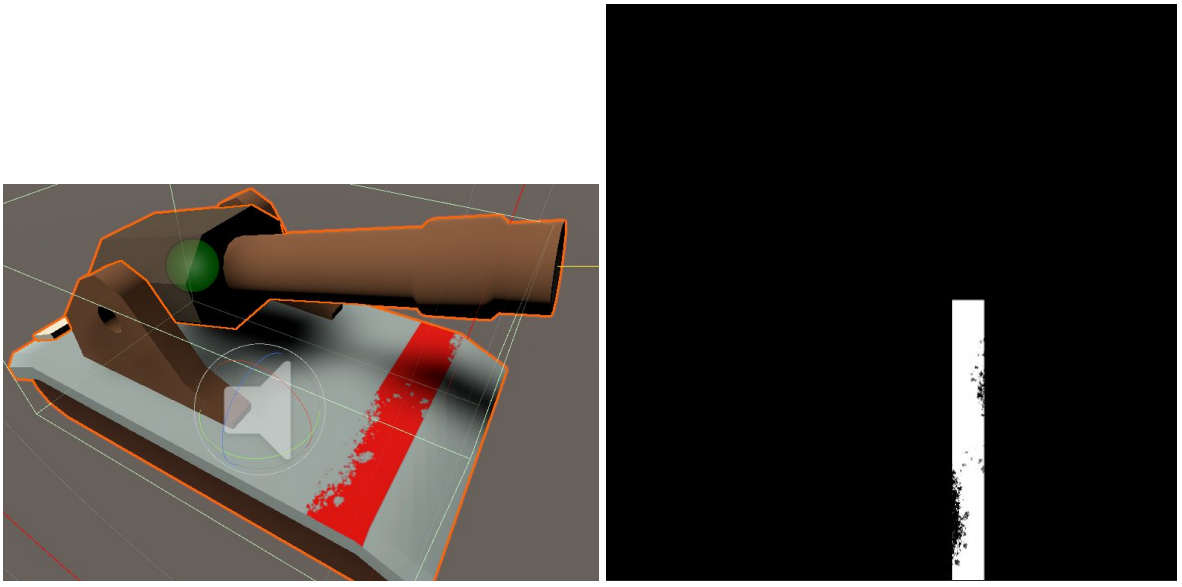
You can don't use these namings, but in this case you'll need to setup all these parameters manually (read next partitions).
In future will be added more bones variations for automatic setup.

How to set up house color parts of units?

Unit component have field **Colored Renderers**. It is array. Here you should add fields for all of unit mesh renderers, which have house colors. After, add one mesh renderer per field, and if mesh renderer has several materials, select needed material id number (number of material, which should be colored). Remember, that material numbers start from 0, not from 1.

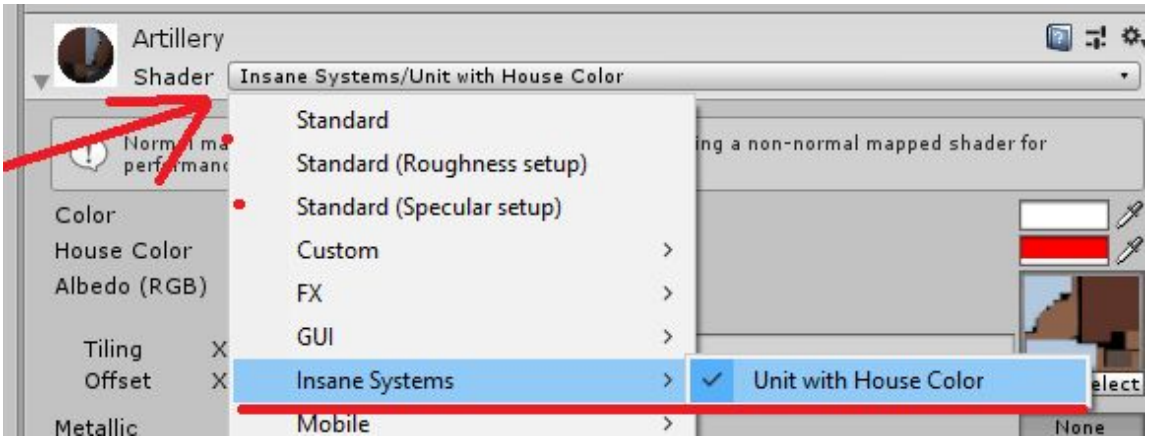


You also can use **house color texture** instead of coloring whole mesh. House color texture is same to emission texture, but it marks unit texture parts, where should be drawn house color, for example, check Artillery unit:

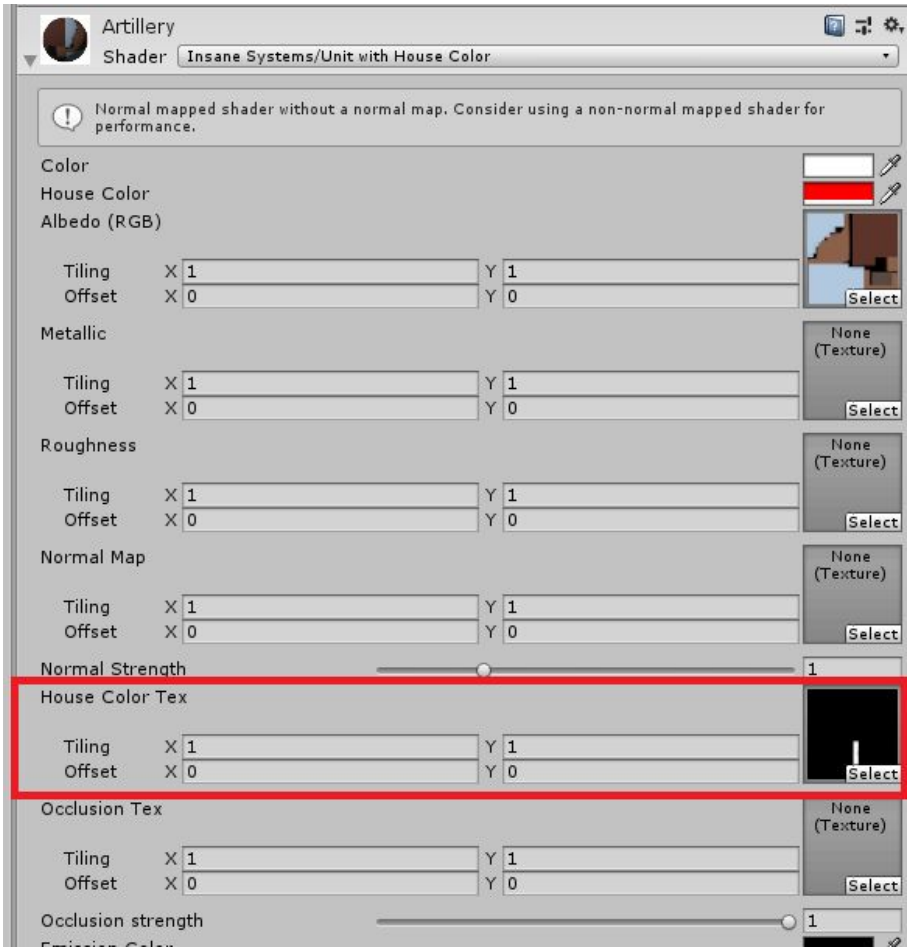


Right image - house color texture, left - how it drawn on unit.

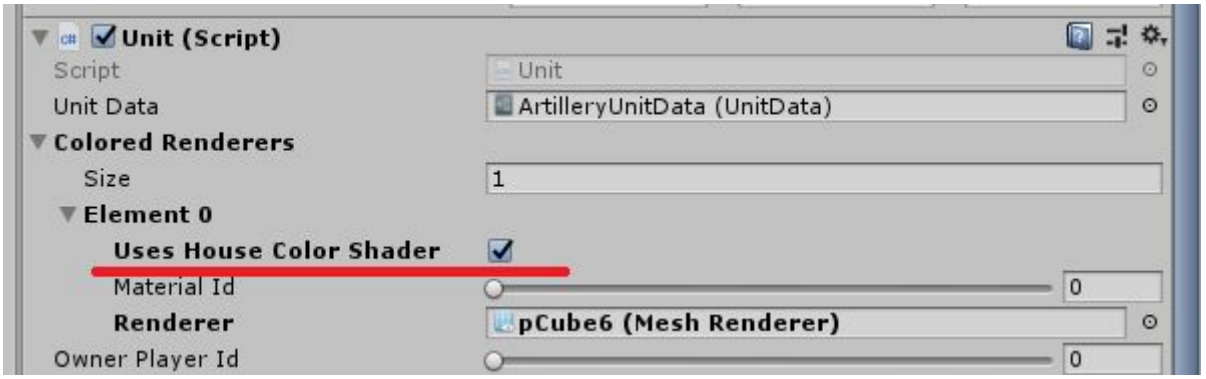
To use this type of house-coloring, you need assign unit model special shader named **Unit with House Color**:



It works in metallic + roughness flow, supports normal map, occlusion and emission. Assign your house color texture in **House Color Tex** field:



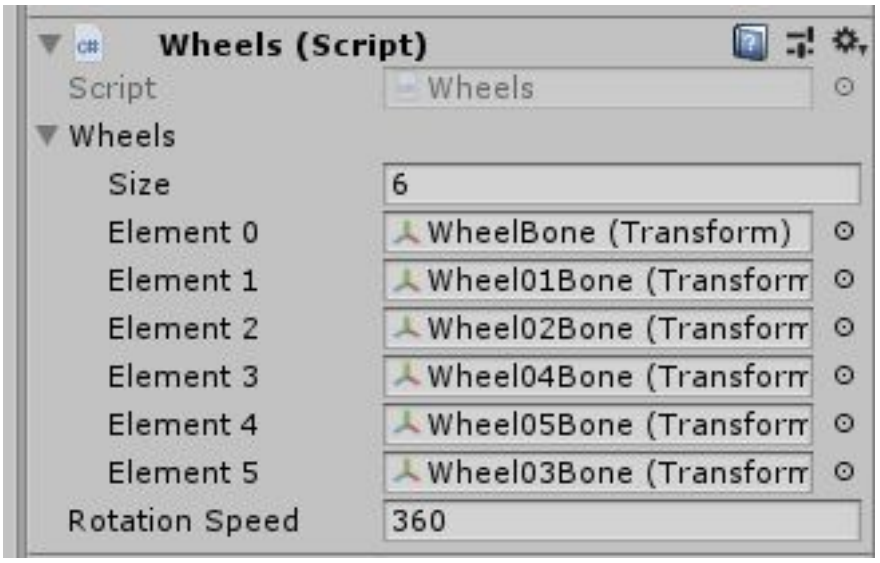
Finally, don't forget to enable toggle **Uses House Color Shader** in **Unit** script parameters:



This approach can boost your performance and more flexible at all.

How to setup wheels for vehicles?

Better way to explain it - ask you to look on default Harvester unit. It has fully tuned **Wheels** component.

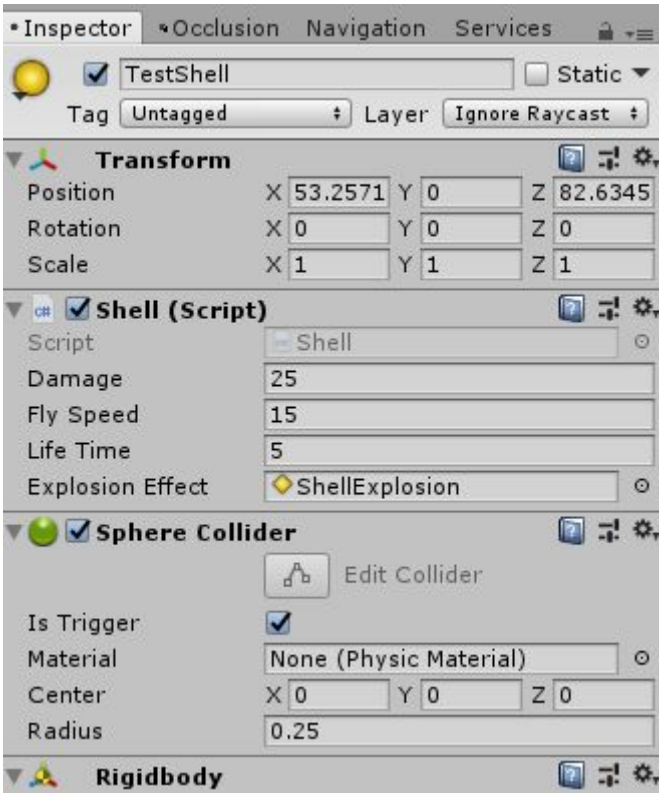


You should add all wheels to the array. If wheels have different pivot bone settings, you should add new identical bones and make each wheel child for its bone, and next add these bones to array.

Rotation speed is in degrees in second. It means that in current settings (360) wheels will fully rotate once per second.

How to setup custom rocket or shell?

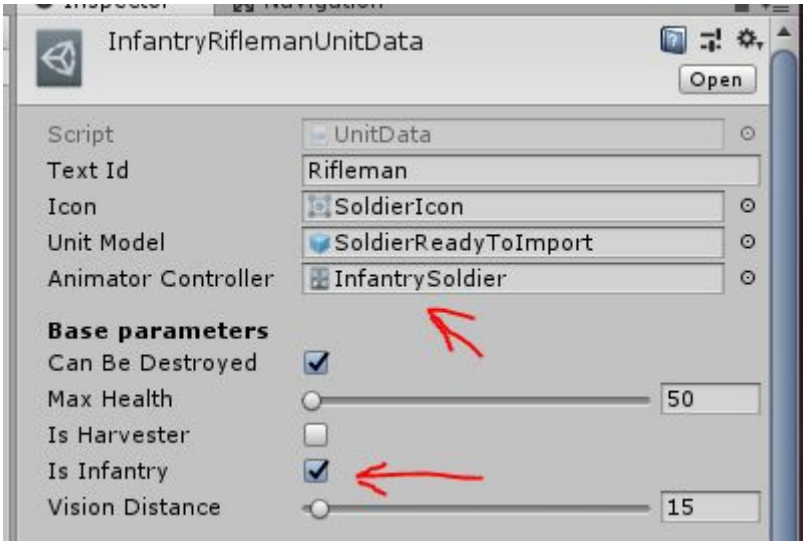
You can clone existing from asset Prefabs/Shells folder and customize settings as you want. Settings are easy to understand - damage value, fly speed, bullet lifetime and explosion effect:



To edit bullet physical size, change **Radius** value of **Sphere Collider**. You can use other collider types, if you need, just change the component.

How to setup infantry units

First of all, you need to setup specific fields in Unit data, it shown on screenshot below

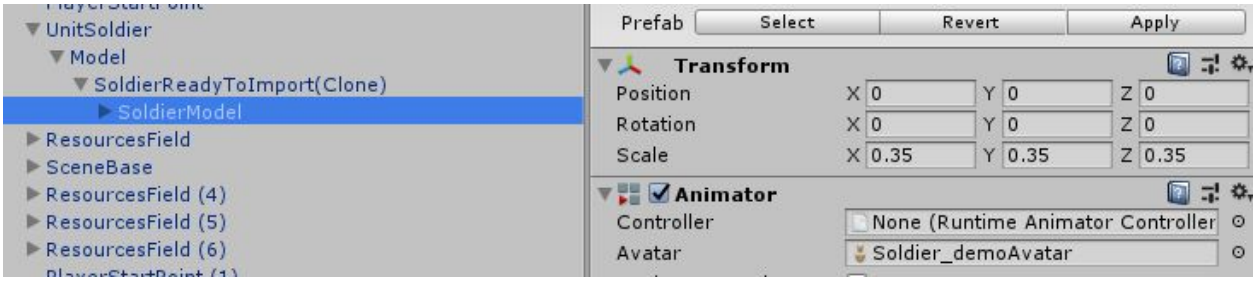


Animator Controller needed to allow animated character play its animations (our asset will setup it to Animator in prefab automatically)

Is Infantry simple marks this unit as infantry.

Now you need to make changes for infantry-specific needs in your unit prefab. On automatic prefab generation it will do it, you know, automatically. :) But if something goes wrong or you want to do it manually, you need to add **Infantry** component to the unit prefab, and setup it's field **Animator** - it will allow unit to play animations:

This animator can be found in one of childs of prefab. If there no one in your model, add it manually:



Setting up of animated character model is same to default unity pipeline.

After this things done, it should work fine and play animations, if you added it in your Animation Controller.

For more info you can check our asset example infantry soldier prefab, animator, etc.

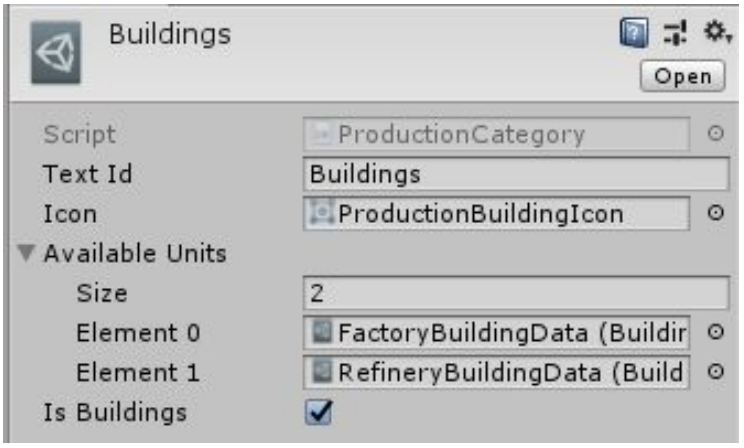
How to add or edit Production Category?

To create new production category, you need right-click in **Project Window** menu and select **Create -> RTS Starter Kit -> Production Category.** But you can clone existing in example project **Production Category**. You can find it in **Resources/Data/ProductionCategories**.

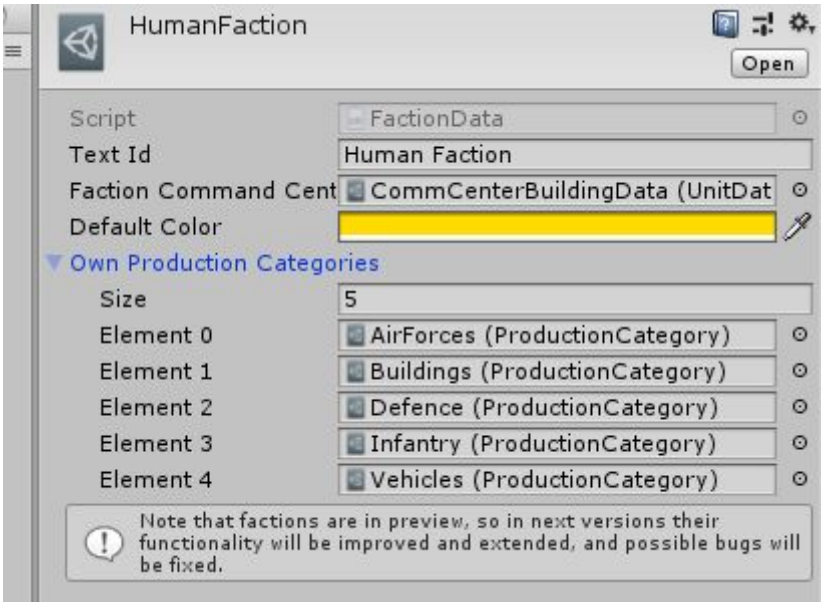
Production category contains next settings:

1. **Text id**.

2. **Icon** of category. It will be drawn on UI.
3. **Available Units** - list of units (unit datas) for this category. It is all units, which will appear on UI when this category selected.
4. **Is Buildings** parameter. Check it, if this category contains buildings (not units) objects. Otherwise remove check mark from this toggle.



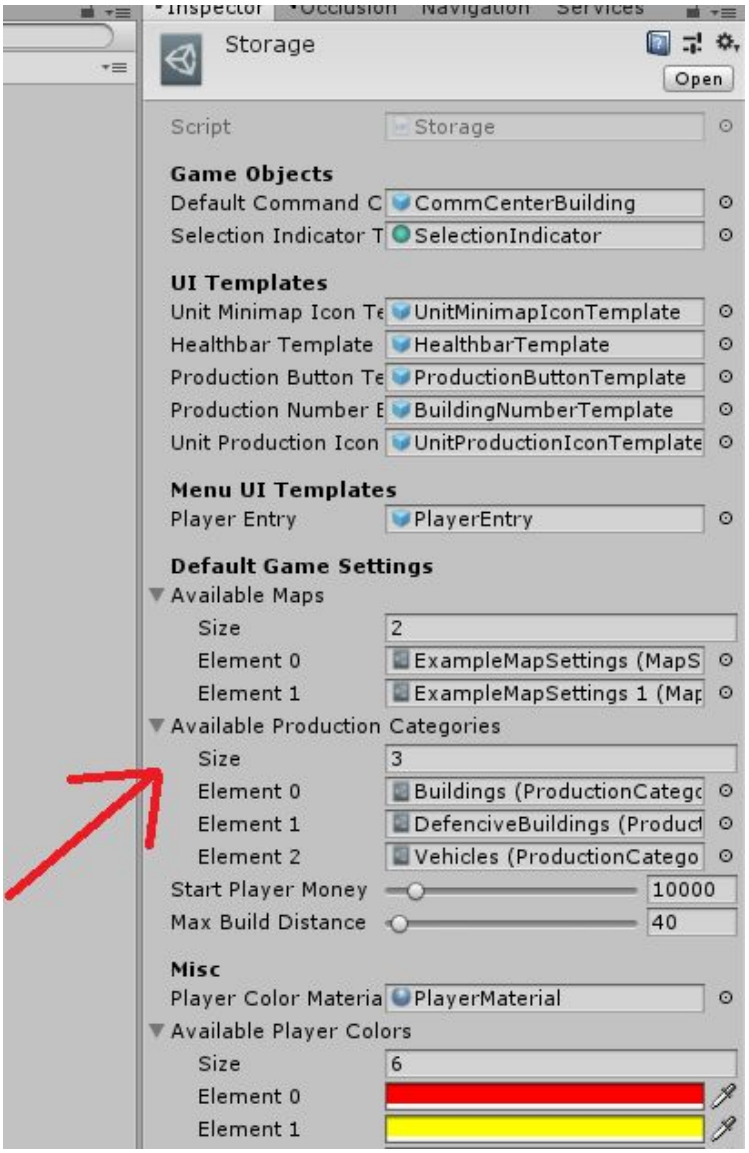
Don’t forget to add new production categories to specific faction, which should have it:



Note that different factions can have same production category, if you need it for tests, prototyping or smth.

Also note, that in current version it needed just to show production icons on panel ingame, but if any building of this fraction will have non-added to faction production category, on selection it still be able to buy units of this category.

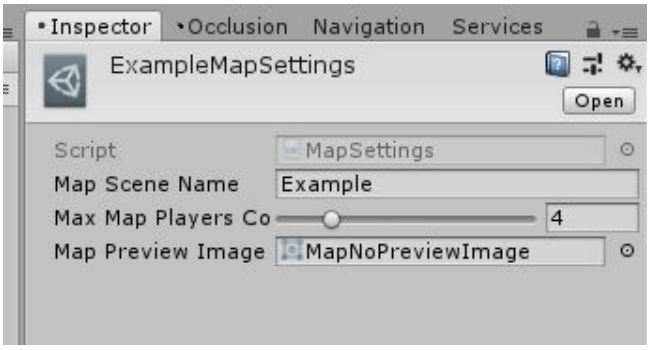
After adding new **Production Category**, add it to **Storage** object in **Available Production Categories** field.



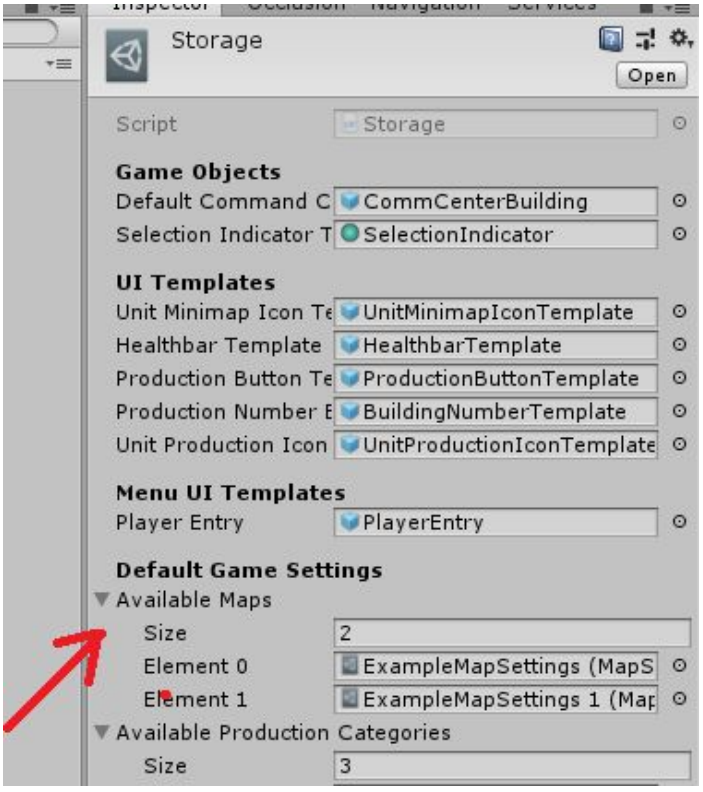
How to add and setup new map?

First of all, you should create **Scene** with your map. This scene should have terrain (or any object with collider and marked with **Terrain** layer), baked **NavMesh** (to allow units move) and **SceneBase** prefab from **RTS Starter Kit**. Don't forget to place **Player Start Points** on map (you can find this prefab in RTS Starter Kit Prefabs folder).

Next, you should add this scene to the **Build Settings**. After it done, create **Map Settings** (right click on **Project Window**, in context window select **RTS Starter Kit/Map Settings**), and fill its fields (map scene name and map players count).



Finally, drag and drop this **Map Settings** object to the field **Available Maps** of **Storage** object.

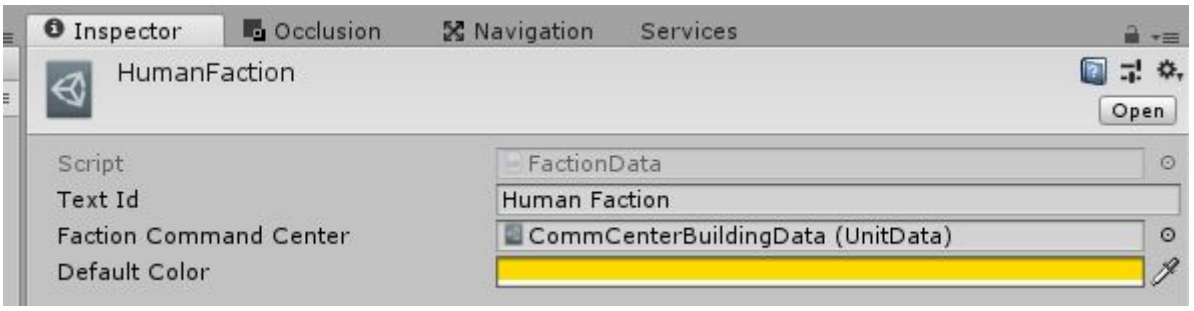


How to add new faction (race)?

To add new faction/race you just need to create **Faction Data** and fill it with your data.

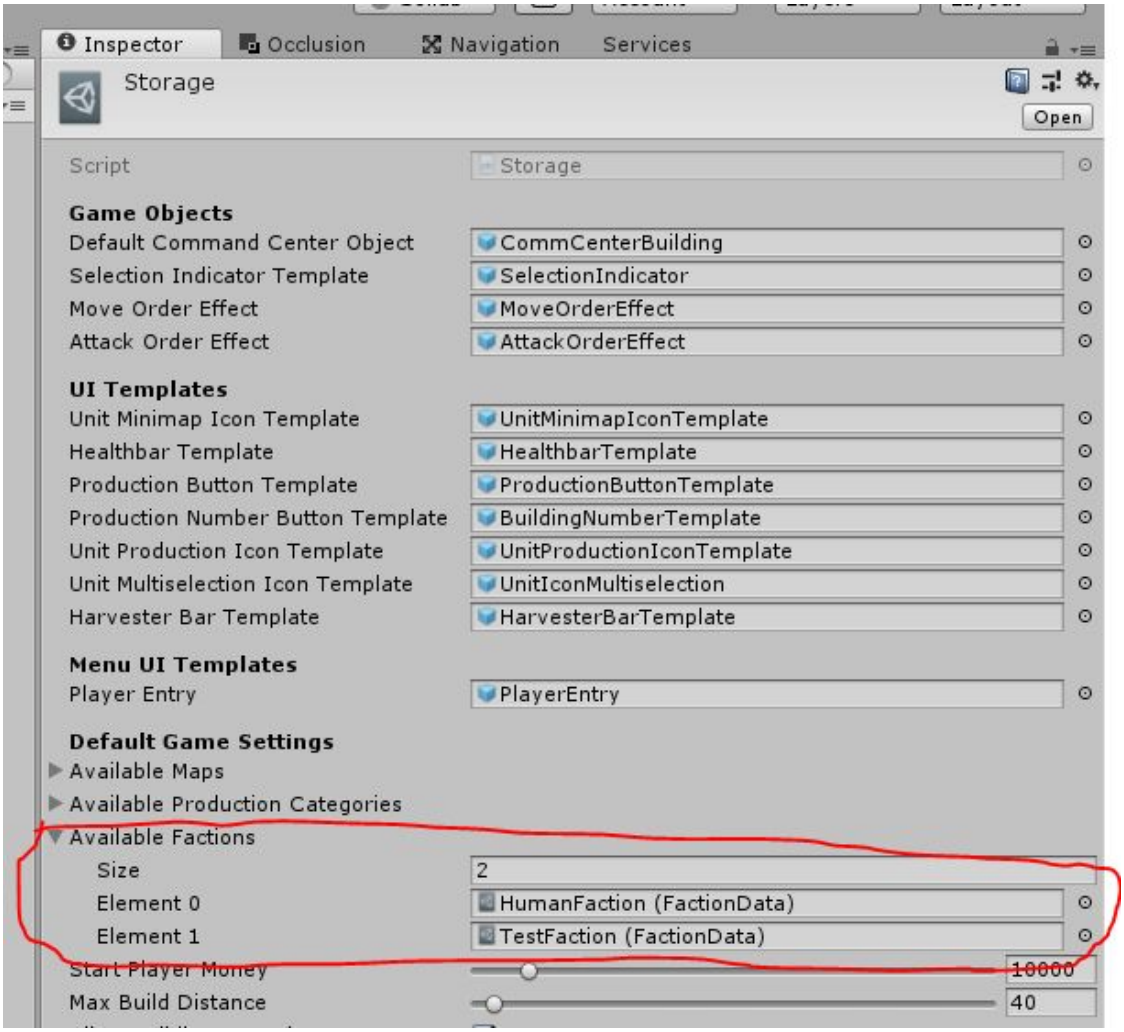
To create new **Faction Data**, you just need to **click RMB** in **Project Window**, and select **Create -> RTS Starter Kit -> Faction Data**.

In **Faction Data** you can setup parameters of your faction:



It is very simple. You write its name and add **Unit Data** of this faction **Command Center** building. All data about available units and buildings already setted up in **Unit Datas** of this faction.

Finally, you need add it to the **Storage** in **Available Factions** field:



Fog of War

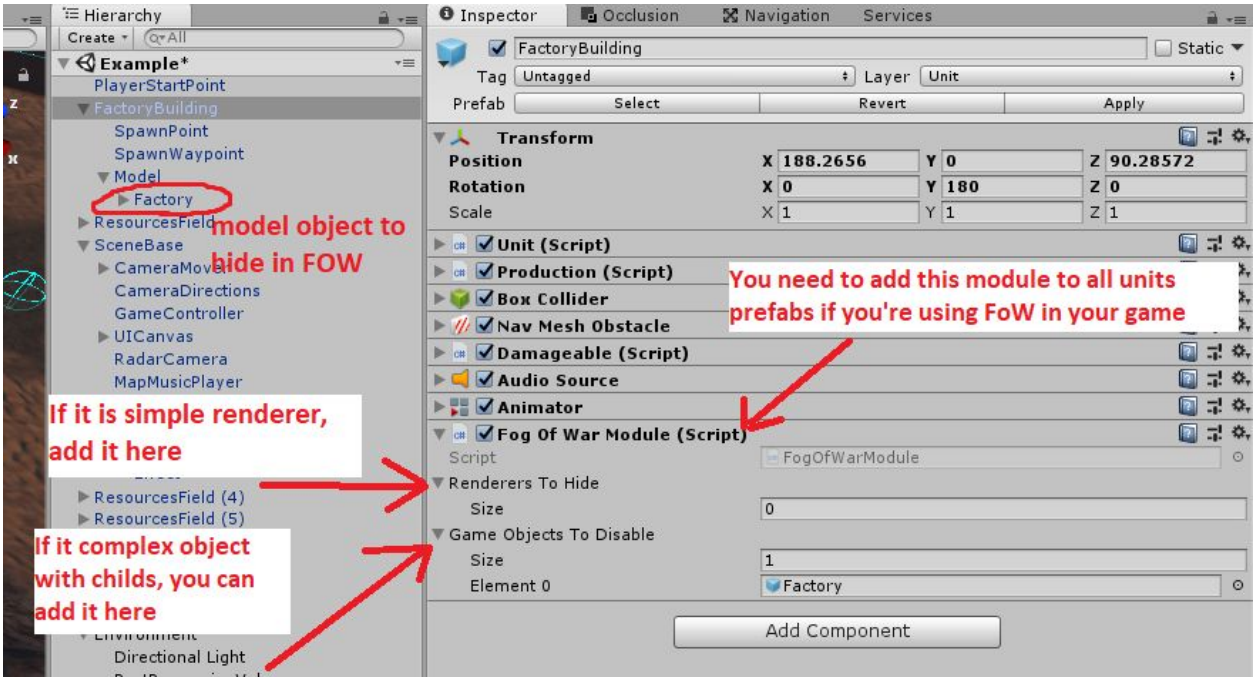
You can enable **fog of war** in **Game Settings/Storage** asset file. Result shown on screenshot:

This is classic mechanic for most of RTS games. Hides enemies units in fog, your units have ranged visibility.

To change visibility range of specific unit, select it in **Units Editor** or select his UnitData file and edit field named **Vision Distance**.

Note that fog of war max units count (which have vision in FOW - player and his teammates units) is 1000. It can be increased in shader files manually, but not recommended.

You also need to setup all units prefabs to work in FoW. Screenshot below shows how to do it. You need add the **Fog of War Module** to the unit, next you need to select model in unit hierarchy and add it to the list of hideable objects (Do not add there unit prefab itself, it will break unit in playmode):

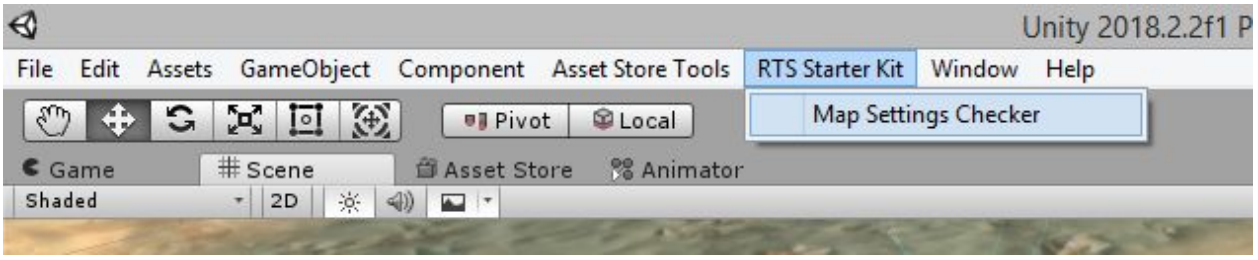


If you don’t do it or do it wrong, enemy unit will be visible in fog of war. If you see something like this, re-check your unit Fog of War Module settings.

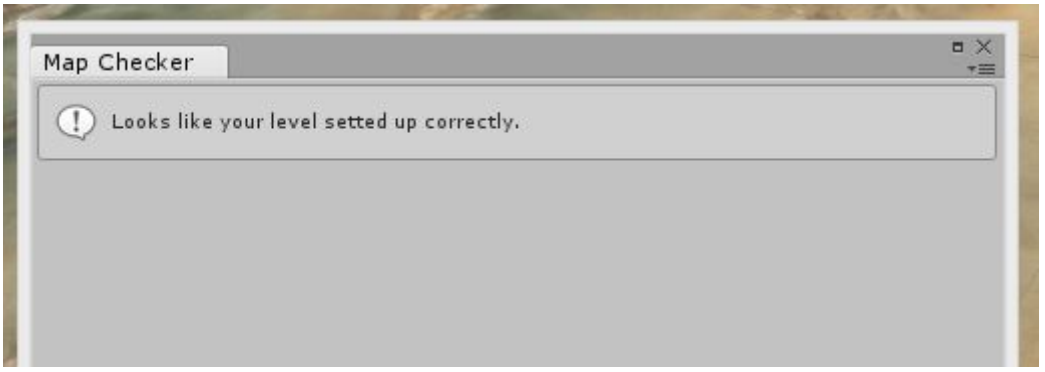
Map settings Checker

Map Settings Checker is a special window, which will check all scene settings and show warnings, if you forget to setup something. If scene setted up correctly, it will notice you that all fine.

You can access this window from top Unity menu by clicking **RTS Starter Kit/Map Settings Checker**:

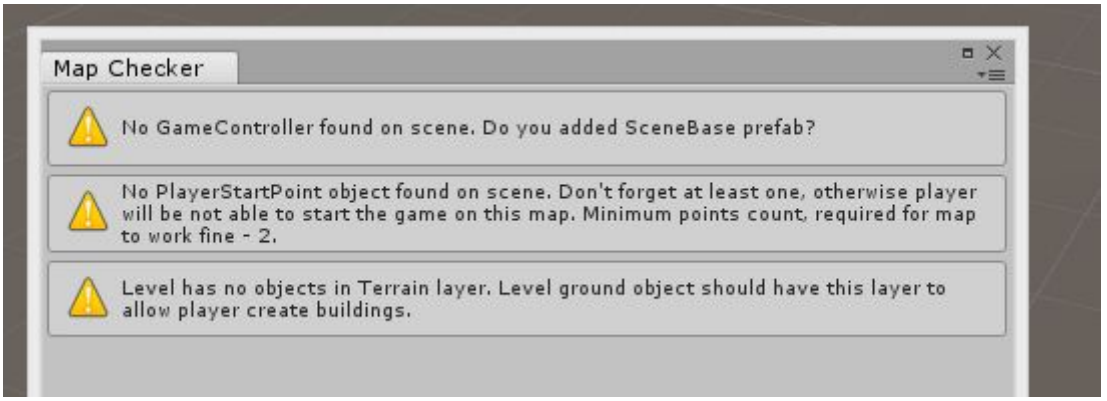


This window looks like this when scene settings is correct:



Map settings checker

Or like this, if you forget to setup something:

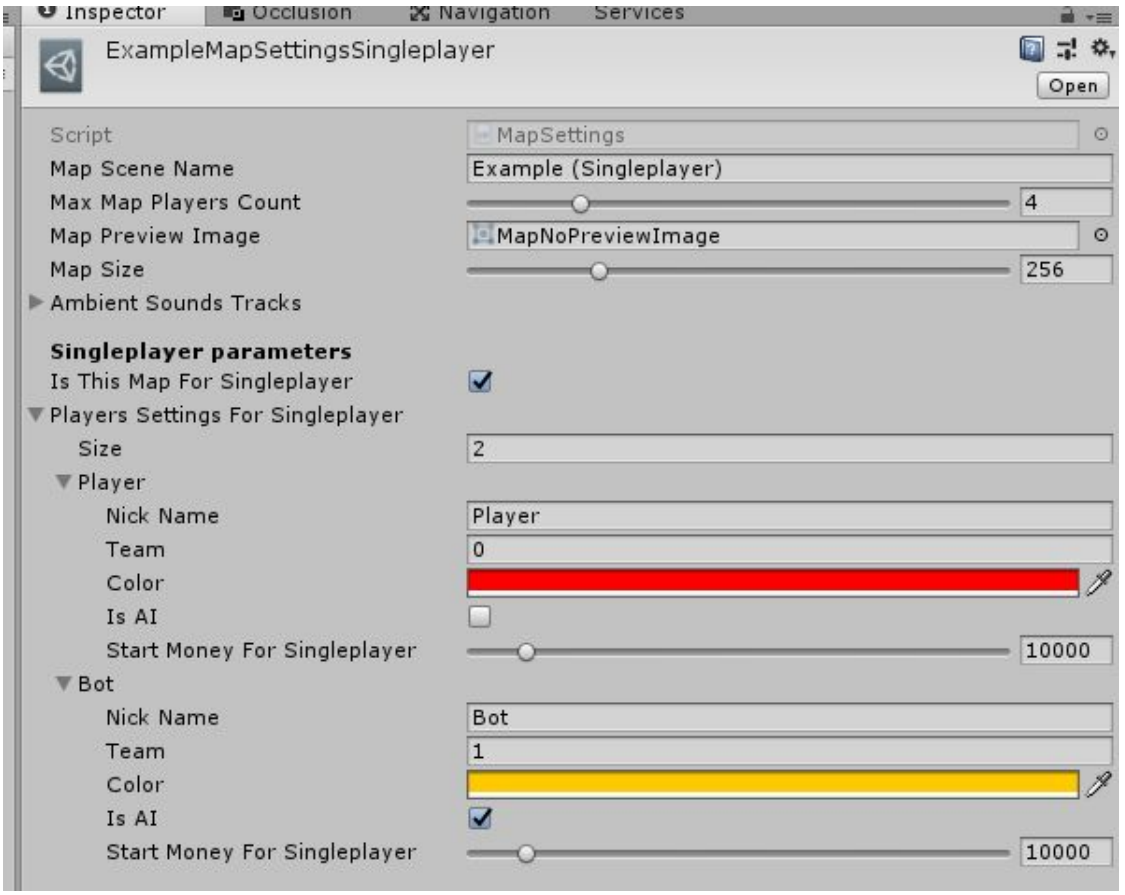


Map settings checker

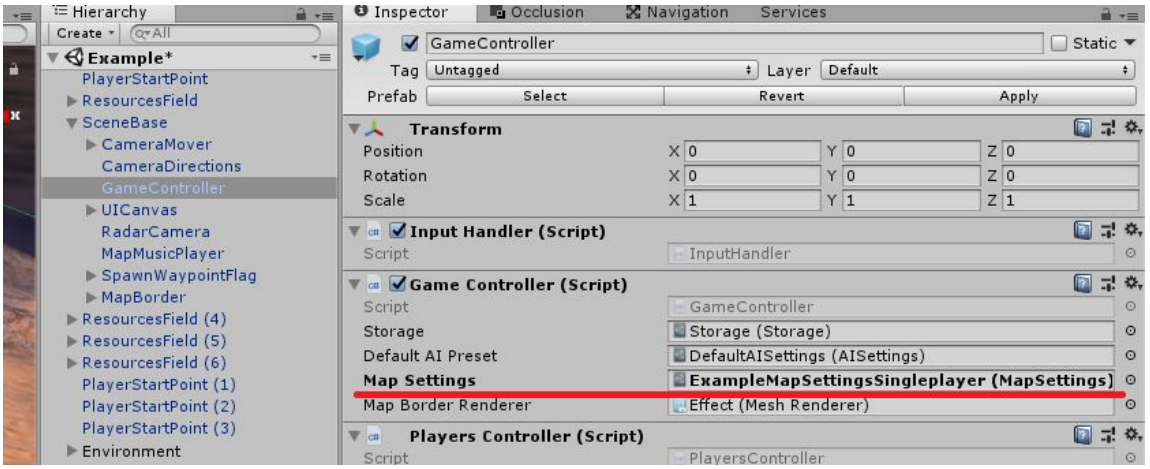
Setting up single player (campaign) map

By default, all maps settings designed for lobby-style matches. But if you're making single player game with campaign, of course you don't need any lobbies.

So, you can check **Is This Map For Singleplayer** toggle in **Map Settings** and your map will be work as single player-only map.
Also you need to pre-setup this map players parameters in **Map Settings**. My example looks like this:

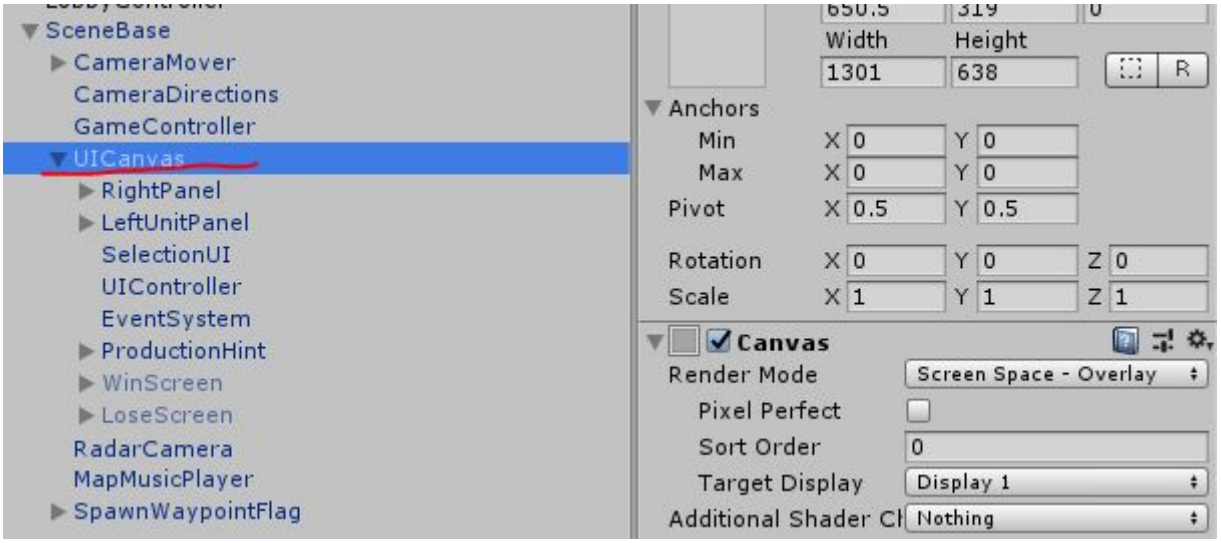


Finally, you need add this **MapSettings** asset to **GameController Map Settings** field on scene of your map:

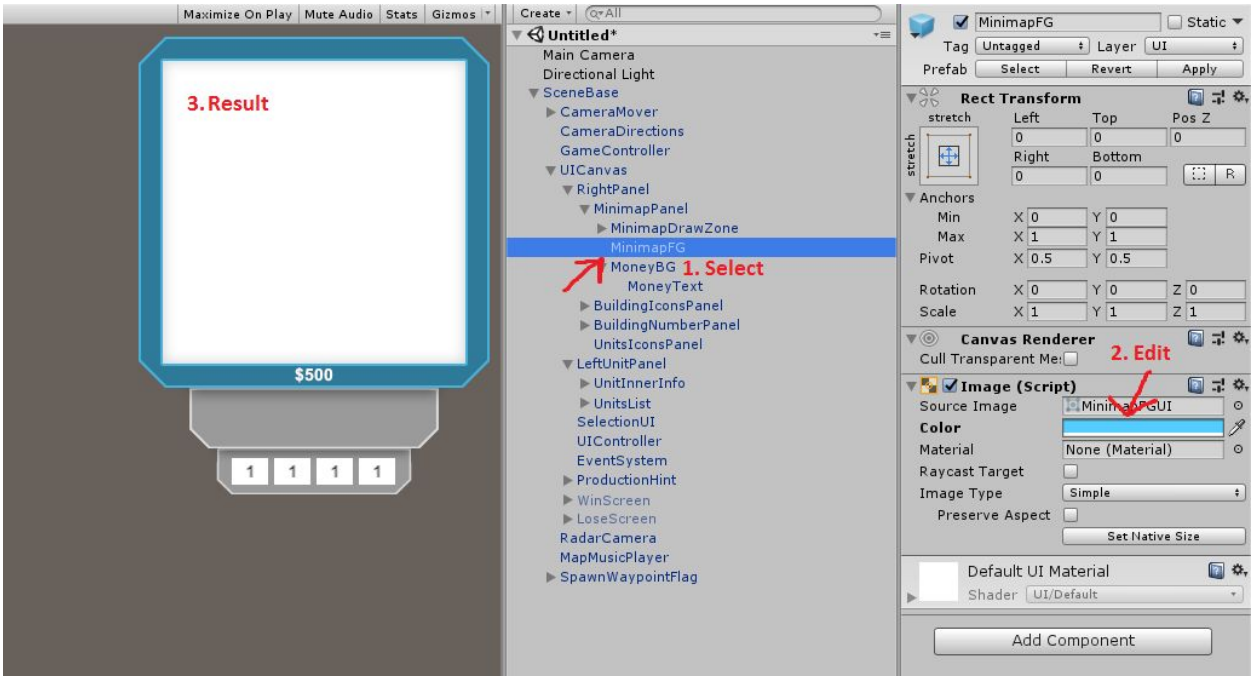


How to customize game UI?

You can easily customize game UI with your images and colors, just drag'n'drop **SceneBase** prefab to **Scene** (or open in Prefab Editor, if you're use 2018.3 or newer version), and edit **UI Canvas** child UI Elements:



For example, I'd changed color of one UI windows:

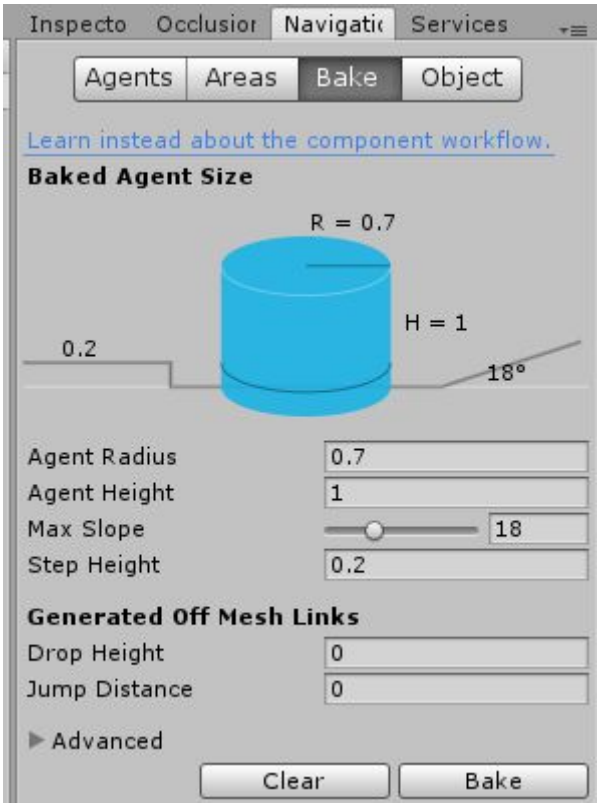


Don't forget to save prefab after your changes.

You can also change position of elements, images, size etc. But a lot of this elements works with code, so don't remove any of them from prefab, it can cause errors.

Best settings for NavMesh

In current version asset uses these NavMesh settings:



It good, because disallow units to move at very high angle of terrain. But it works fine only for example units scale. If scale of your units is bigger or smaller, you need to find your own best settings of NavMesh.

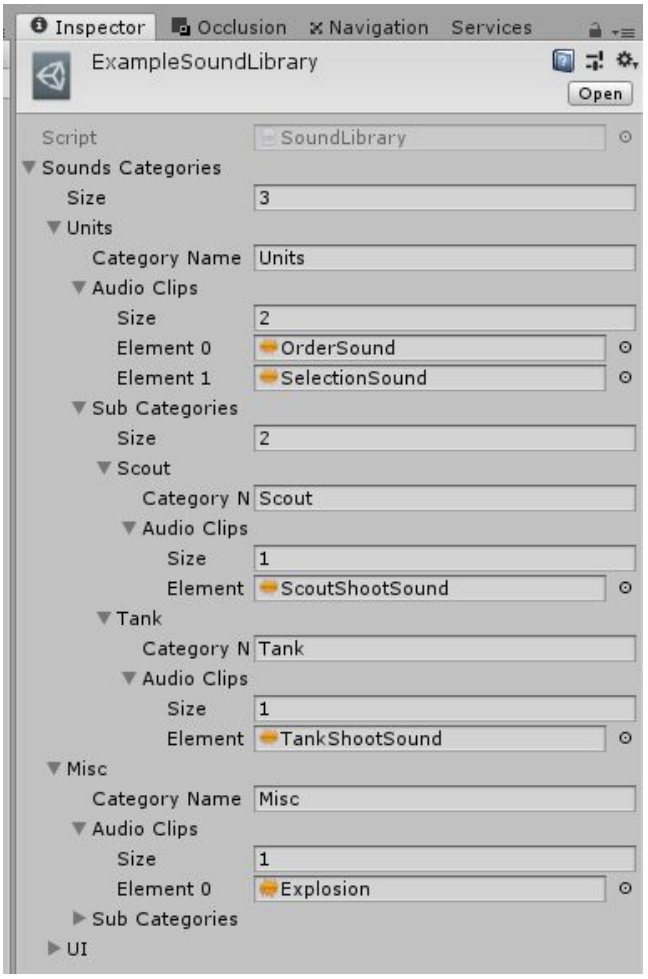
But you always can look at our settings and make something like this, but with proportion to your scale.

Sound Library and Sound Editor

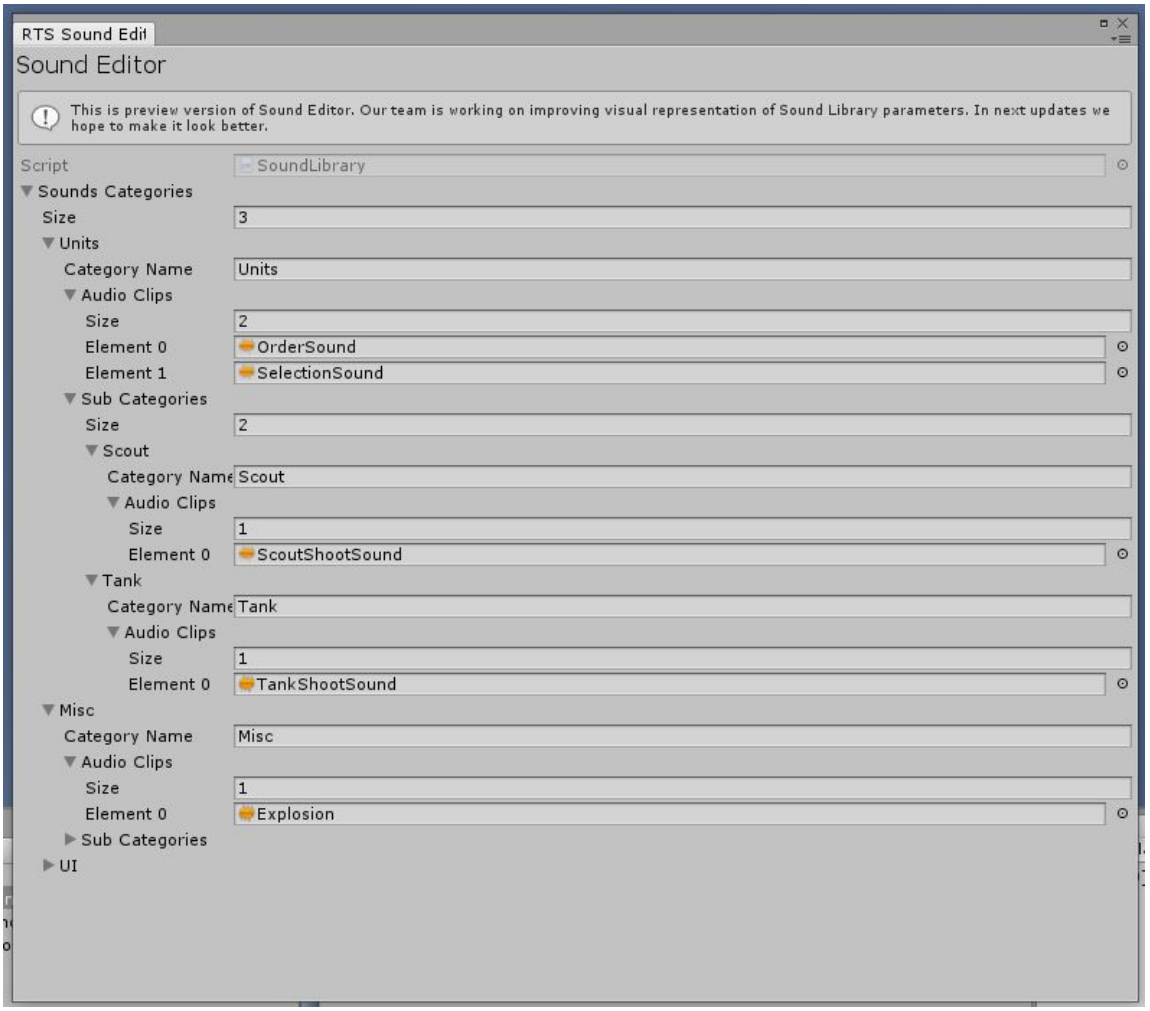
Sound Library is an resources asset file like **Storage** (described before). It used to manage all project sounds in more conveniently way than default. It allows to create sounds categories (you can divide units sounds and UI sounds for example) and subcategories (for example, subcategories for different units). Next, in **Unit Data** settings you can simple select needed sound from dropdown. It is simpler than searching them through all project.

Example project already contains example **SoundLibrary** asset, and it also setted up in **Storage** (you need drag and drop here **SoundLibrary** asset which you want to use, or left it as it is and edit example **SoundLibrary** when you want to add new sounds).

Currently it looks not the best, but can be edited as any array in Inspector: to add new categories or sounds - extend the array size. After it done, setup newly appeared field as you need.



Sound Editor is just quicker way to access currently used in project **Sound Library**, you can open it from **top menu -> RTS Starter Kit -> Sound Editor**.



If you want use Sound dropdown in your code, just write **[Sound]** before your **Audio Clip** variable.

Triggers

Triggers allows you to setup custom gameplay situations on your maps. It is very useful thing for singleplayer and campaign games (but can be used in multiplayer too, but of course requires improvements in code).

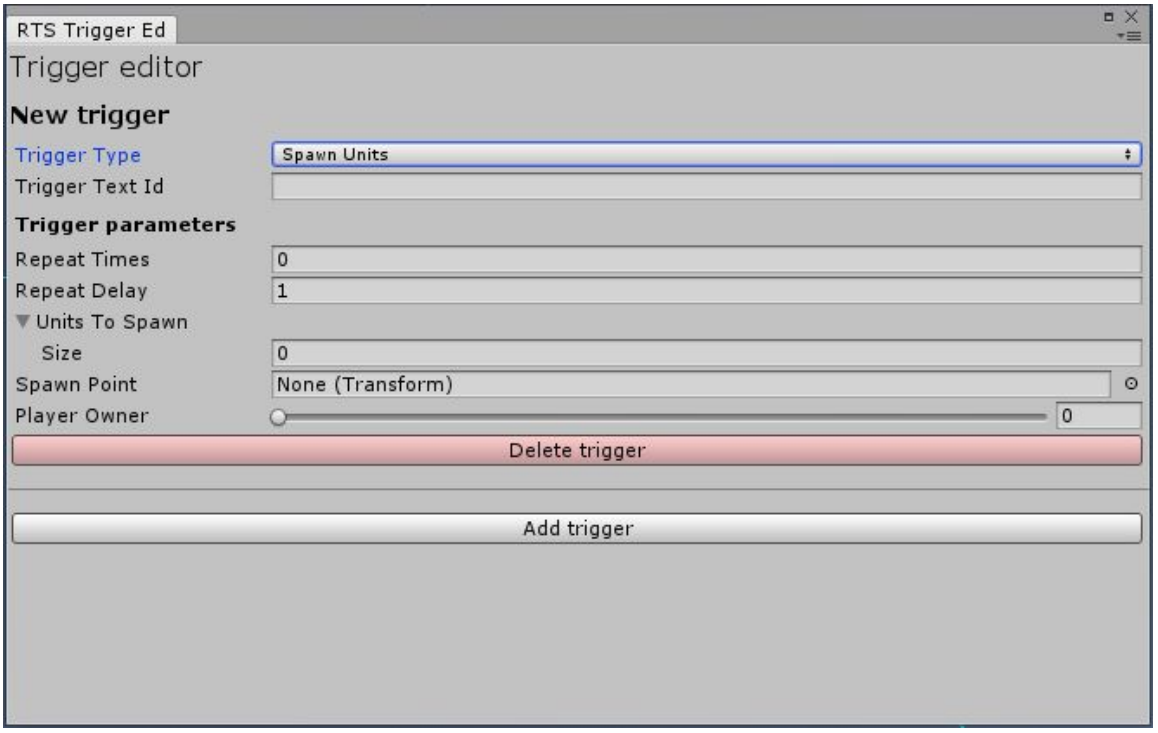
For example, imagine the situation, when computer AI should start attacking player units after them moving in some zone setted up before. With trigger you simple select which units should attack player and select, where it should move to meet player units.

Or, another example. Player units reaches some destination, where player should build the base. You can give player a money with trigger or you can spawn

Command Center (or another building which allows to create other buildings), so player continue playing, but now have possibility to build the base.

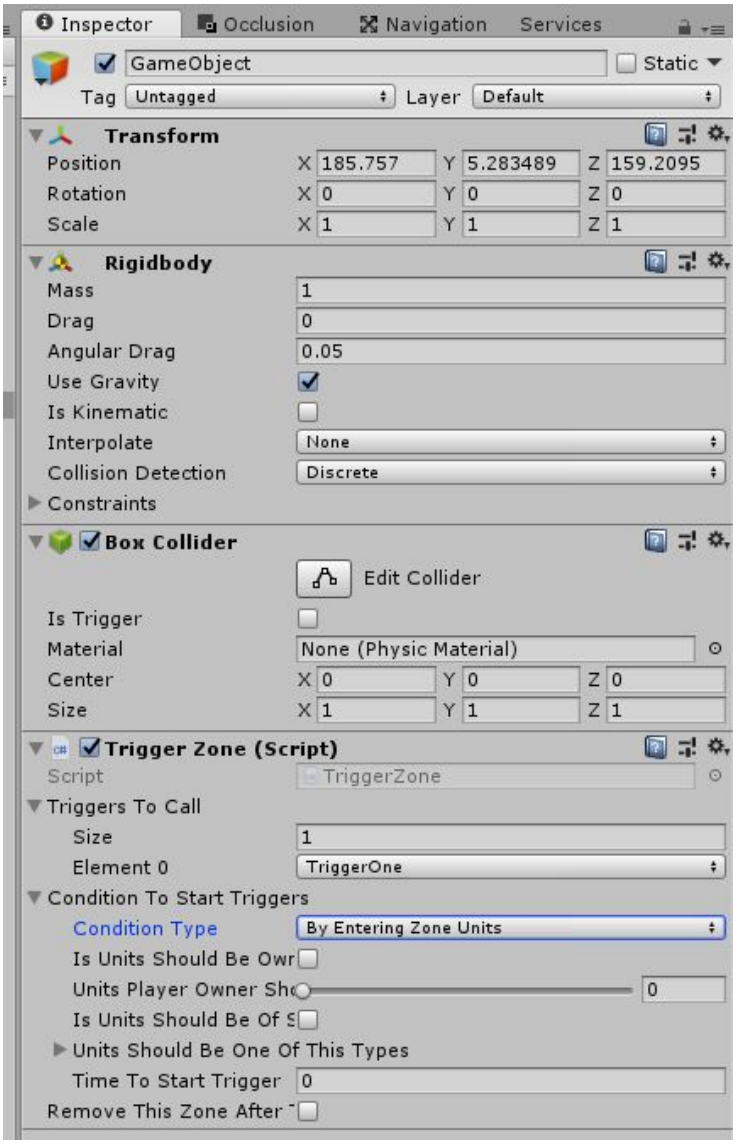
Our asset includes implementation of such triggers.

To start working on triggers, open **top menu -> RTS Starter Kit -> Trigger Editor**. You should do it on your map, not in Lobby scene etc.



Trigger Editor allows you to manage map triggers, add new and setup all their parameters. If you already worked in other editors like this, it should be very intuitive for you.

Trigger Zone component allows you to create trigger zones mentioned before. Just add it to empty **GameObject**, setup **Box Collider** of trigger zone (size etc). After this done, in **Trigger Zone** component you can select triggers which should be called when this zone being triggered. Do it in **Triggers To Call** field.

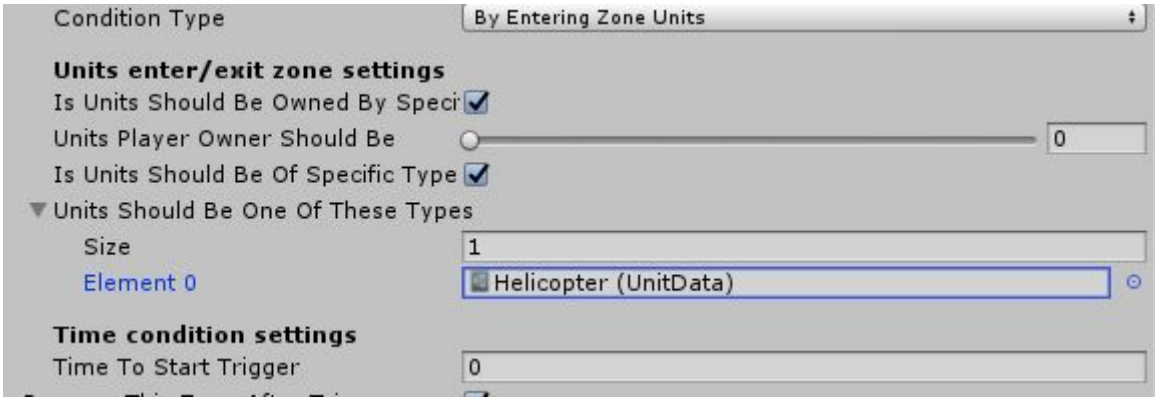


Note that if you will rename triggers, you need to update parameters in **Trigger Zones** too.

There also exists a **Trigger Condition**, which allows you to filter things which should call this triggers.

So **Trigger Zone** not only used for check entering units, but you also can use it to make trigger which waits some time since level start.

This is an example condition of triggering zone:



This zone **will be triggered only if** unit of type **Helicopter** owned by **player 1** (with index 0) **enters the zone**. It is very easy to use, isn't it?

Our asset already have some ready-made triggers, conditions and we will add new ones in updates. You also can code your own triggers (use existing as example), it is not too complicated. Don't forget to add your trigger name in **TriggerType** enum. Note that it also should end with **Trigger** word, but you don't need to include this **Trigger** word to a **TriggerType** name of trigger.
Example: trigger named **TestTrigger** should be added to a **TriggerType** as a **Test.**

Programming

How to create my own module?

All what you need is create new **C# Script**, and place your logics here. If you want, you can inherit it from **Module** component – this is basis component for all Unit modules in **RTS Kit**. It contains link to main **Unit** component, so you're can easily access to it from your own module, for example:

```
public class CustomModule : Module
{
    void Start()
    {
        Debug.Log(selfUnit.OwnerPlayerId);
    }
}
```

How to use features of RTS Starter Kit modules in my own?

If you inherited your custom module from core **Module** class as it shown in previous section, you can get access to the other modules by **Unit** component, it contains links to all other default modules, if it added to the unit:

```
public class CustomModule : Module
{
    void Start()
    {
        selfUnit.movable.MoveToPosition(Vector3.zero); // access to the Movable module

        Debug.Log(selfUnit.attackable.attackTarget); // access to the Attackable module

        selfUnit.damageable.TakeDamage(15); // access to the Damageable module

        Debug.Log(selfUnit.production.unitsQueue.Count); // access to the Production module
    }
}
```

You can check accessible module for null, if you not sure that it exist on unit object.

To send new order to unit, use **selfUnit.AddOrder** method. You can send move, follow and attack orders:

```
    void Start()
    {
        var moveOrder = new MovePositionOrder();
        moveOrder.movePosition = Vector3.zero;

        selfUnit.AddOrder(moveOrder, false);
    }
```

Using of RTS Starter Kit events

Rts Starter Kit code contains a lot of events, which you can connect your code to. It simplifies development of new features very well.

For example, you can do something when any unit spawned. You can receive info about spawn by connecting to **Unit.unitSpawnedEvent**.

```
using UnityEngine;
using InsaneSystems.RTSStarterKit;

public class ExampleDebug : MonoBehaviour
{
    void Start ()
    {
        Unit.unitSpawnedEvent += OnUnitSpawned; // now every time when any unit being spawned,
                                                // OnUnitSpawned method will be called and...
    }

    void OnUnitSpawned(Unit unit)
    {
        Debug.Log("Spawned a new unit" + unit.name); // ... this message will appear in Console.
    }
}

```

Full list of events can be found in online API Documentation.

Online API Documentation

You can get much more info about **RTS Starter Kit Asset** source code in our **online API Documentation**. Beta version of API Documentation available here:

http://dzhuraev.com/APIDocumentation

We often update it, and still working on improvement of documentation, remember that it still in beta.

To get access to the **RTS Starter Kit** partition, you'll need to enter this token: **67gfx17nj**.

Roadmap

Possible roadmap for the next versions. You can read it, if you looking for some feature, but RTS Starter Kit asset doesn't have it now.

Relevant roadmap can be found here: https://trello.com/b/50aM7pH1

Contacts

You can ask your questions or send your suggestions to us. ☺

To contact us email **godlikeaurora@gmail.com**

Credits

RTS Starter Kit by **Insane Systems**.

Used free content for preview:

Environment Terrain textures by **Yughues** - patreon.com/Yughues

Low Poly Soldiers Demo by **Polygon Blacksmith -** assetstore.unity.com/publishers/17894