RTS Toolkit Free

v2022.1.1

email: chanfort48@gmail.com

https://chanfort.github.io/

29/12/2022

System requirements

Unity 2020, 2021 or 2022.

Compatibility and versioning

This table contains a compatibility list of versions between RTS Toolkit Free and Unity.

Unity version	RTS Toolkit Free version
2020.3.43f1, 2021.3.16f1, 2022.2.1f1	2022.1.1
2018.4.31f1, 2019.4.18f1, 2020.2.2f1	2020.2.1

1. Introduction

RTS Toolkit Free is a system, which allows to set up RTS battles and ease development of RTS games. This asset is designed for beginners, who are seeking how to setup simple RTS game in more optimised way to bring battles up to thousands of units. Developers, who are interested to dive deeper into RTS development are welcome to check RTS Toolkit with much more advanced features and continuous development, and improvements: https://www.assetstore.unity3d.com/en/#!/content/17660

RTS Toolkit Free allows game developers to simulate RTS battles with 6 phases: searching targets, approaching targets, attacking targets, damaged units selfhealing, dying phase and sinking phase. Different phases are used only for suitable objects and full script is attached just to one object in entire game (for example terrain), what allows to perform calculations and set parameters for all objects very quickly. Nearest neighbour search, used to find nearest targets coordinates is using kdtree.cs class (with additionally used sorting by heapsort algorithm for vectors), written by A Stark at 2009 and available here: http://forum.unity3d.com/threads/29923-Point-nearest-neighbour-search-class

WaitForSeconds used for long loops allows to increase performance even further. In general script is optimised to run with O(N log N) time assumptions. Performance analysis (which is performed in this script too) show that there are only 3 10 % of time spend for calculations inside loops compared with all time with WaitForSeconds.

Each unit has their individual movement, set by script, which is independent from other units and makes battles more realistic.

There are two examples with 4000 units and 12000 units used in scene respectively: https://dl.dropboxusercontent.com/u/248943005/1a.html and https://dl.dropboxusercontent.com/u/248943005/2a.html

2. Simple setup

This section will explain how to use prebuild example of RTS Tookit Free. To run it just follow these steps.

1. Open empty scene in Unity and import RTS Battle Simulator package (you can just simply copy RTS folder into your project assets folder). Before moving to next step make sure that you see the same folder structure as shown in the Fig. 1 bellow.

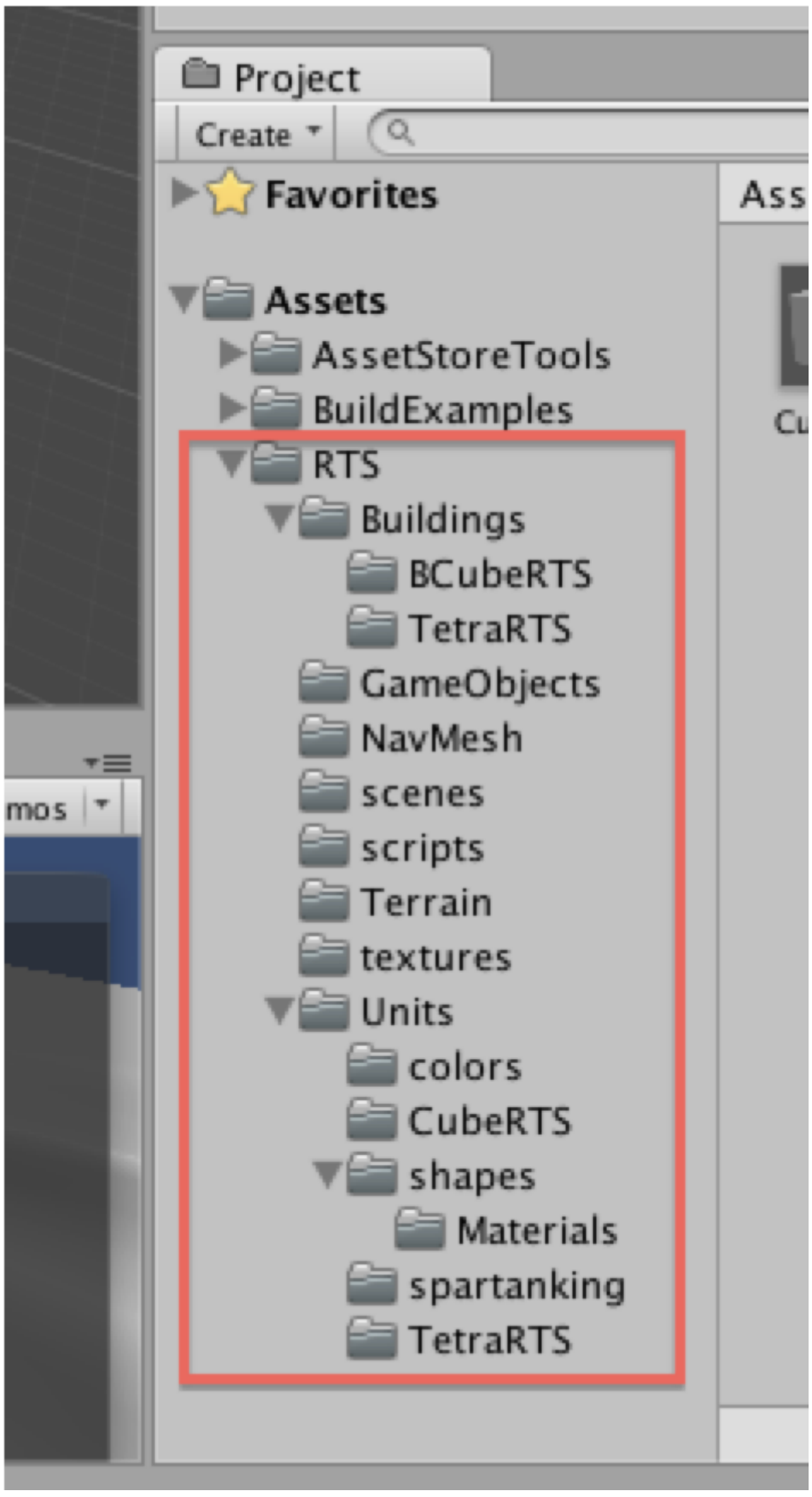

Fig. 1 - RTS Toolkit Free folder structure.

2. If you see all folders, marked with red rectangle, you can continue, if not, check if you downloaded this asset and if it is placed in your project assets folder.

3. Open prebuild scene by entering to “scenes” subfolder and double clicking “RandomSpread” scene file (Fig. 2).

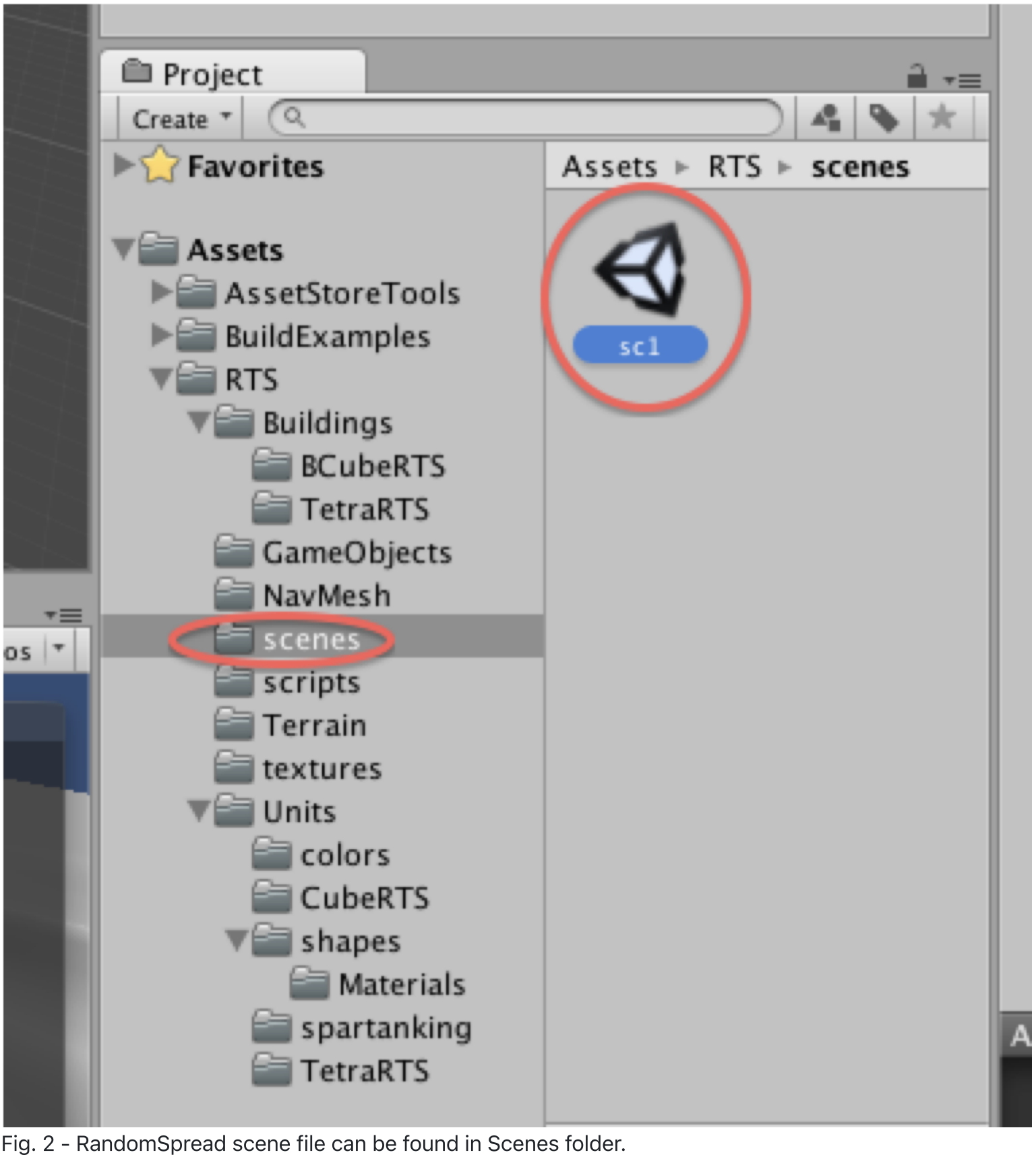

Fig. 2 - RandomSpread scene file can be found in Scenes folder.

4. You should see in Editor now something similar as shown in Fig. 3. Click “run” to see if prebuild example is running.

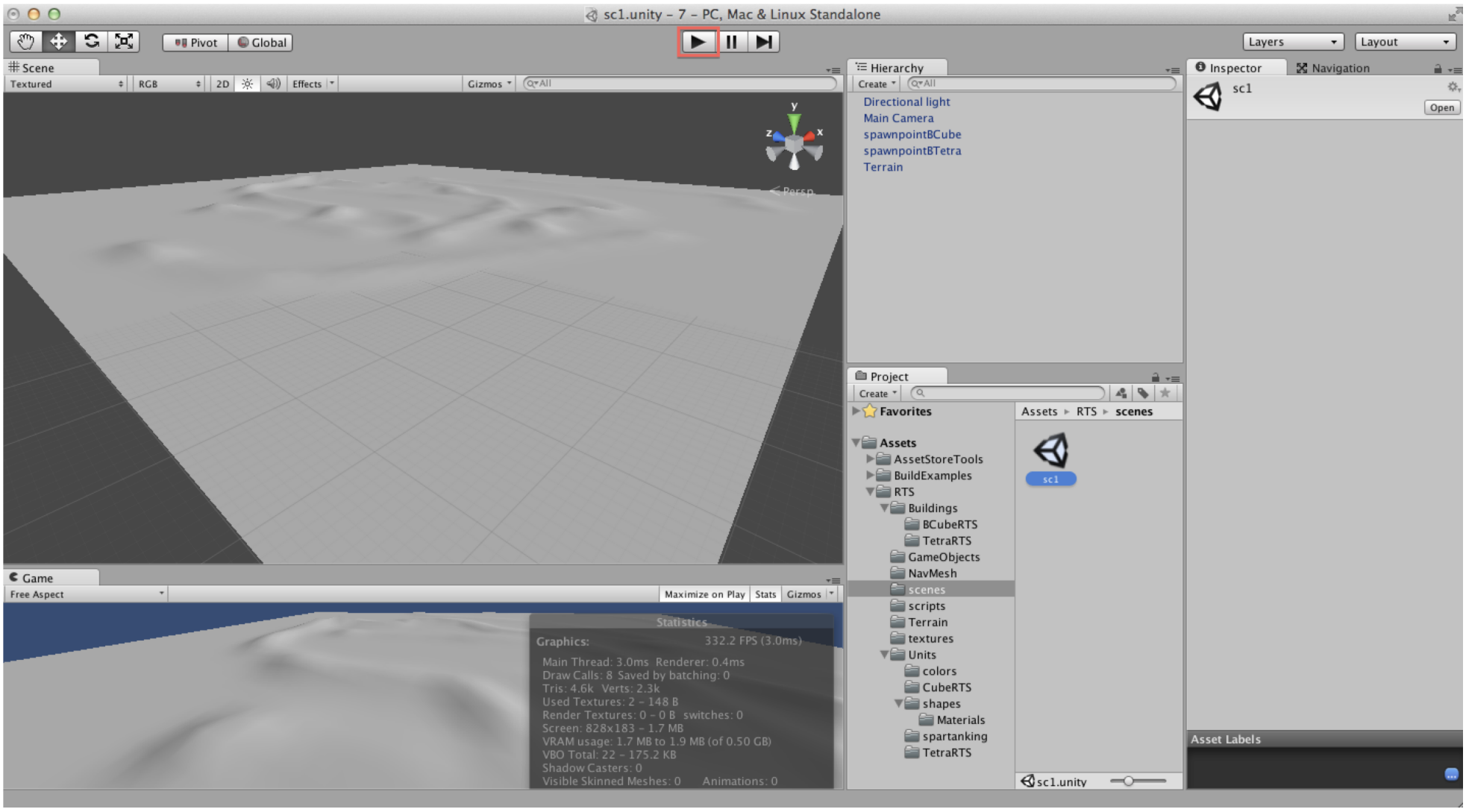

Fig. 3 - RandomSpread scene view.

5. Successfully running build should show units placed on terrain, like in Fig. 4:

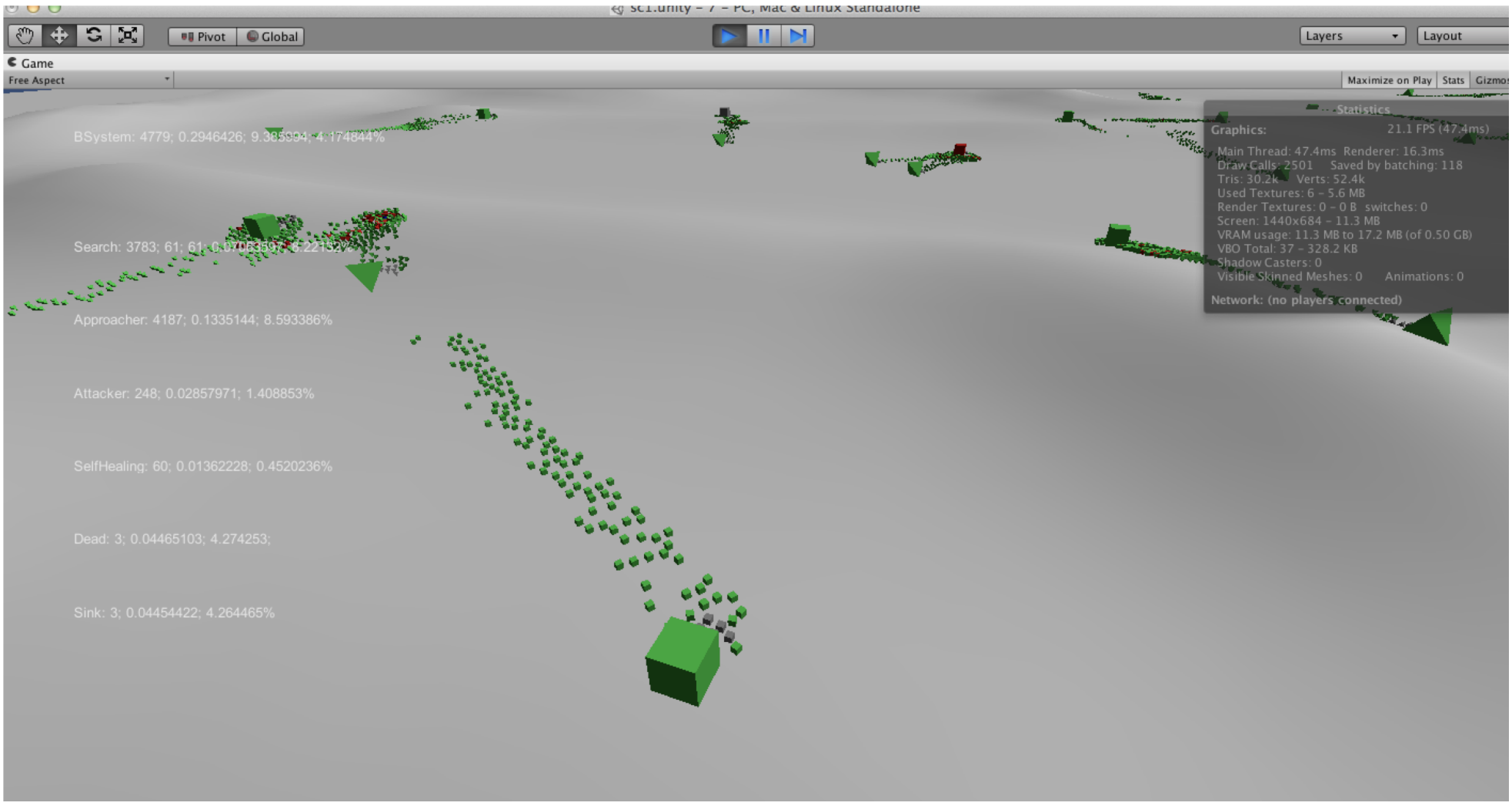

Fig. 4 - Gameplay mode of RandomSpread scene.

6. Well done – you managed to import and run package with prebuild example. You can now start looking how to set up everything correctly in your empty project from scratch or if you wish to skip manual set up, start playing with parameters.

3. Manual setup

1. Open empty project and import RTS package, like it was done in simple setup first step (Fig. 5).

Fig. 5 - View of empty scene after importing the package.

2. Create a terrain (GameObject > Create Other > Terrain) (Fig. 6).

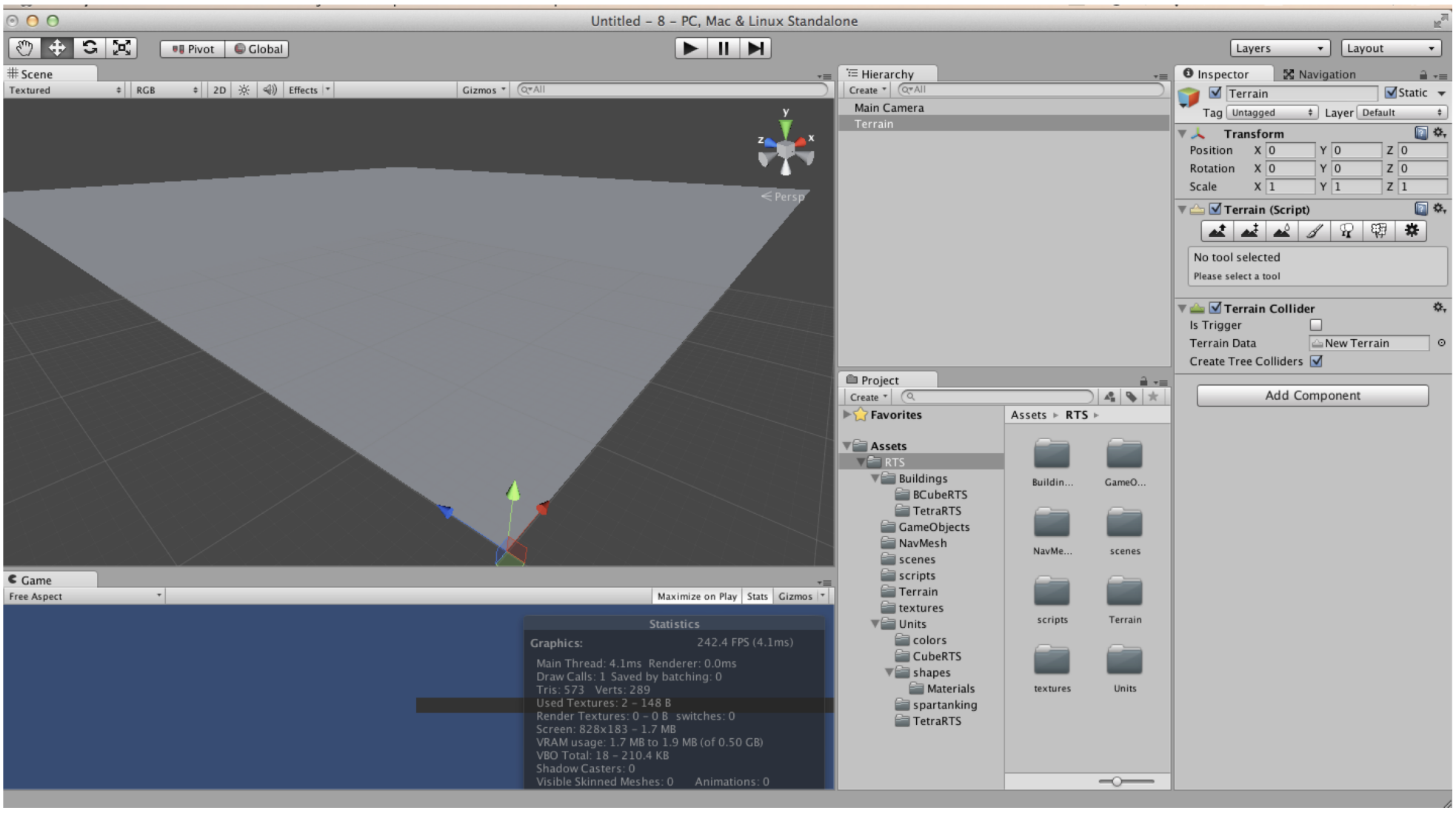

Fig. 6 - New terrain is added into empty scene.

3. Select your terrain, click Navigation tab near Inspector, then Bake tab and finally click Bake button to Bake NavMesh data. This will create NavMesh for your terrain, which will be used by script (Fig. 7).

Fig. 7 - Navigation tab view.

4. Move back to Inspector tab, create some Directional Light and move Main Camera to position, that it would be possible to see terrain from it (Fig. 8).

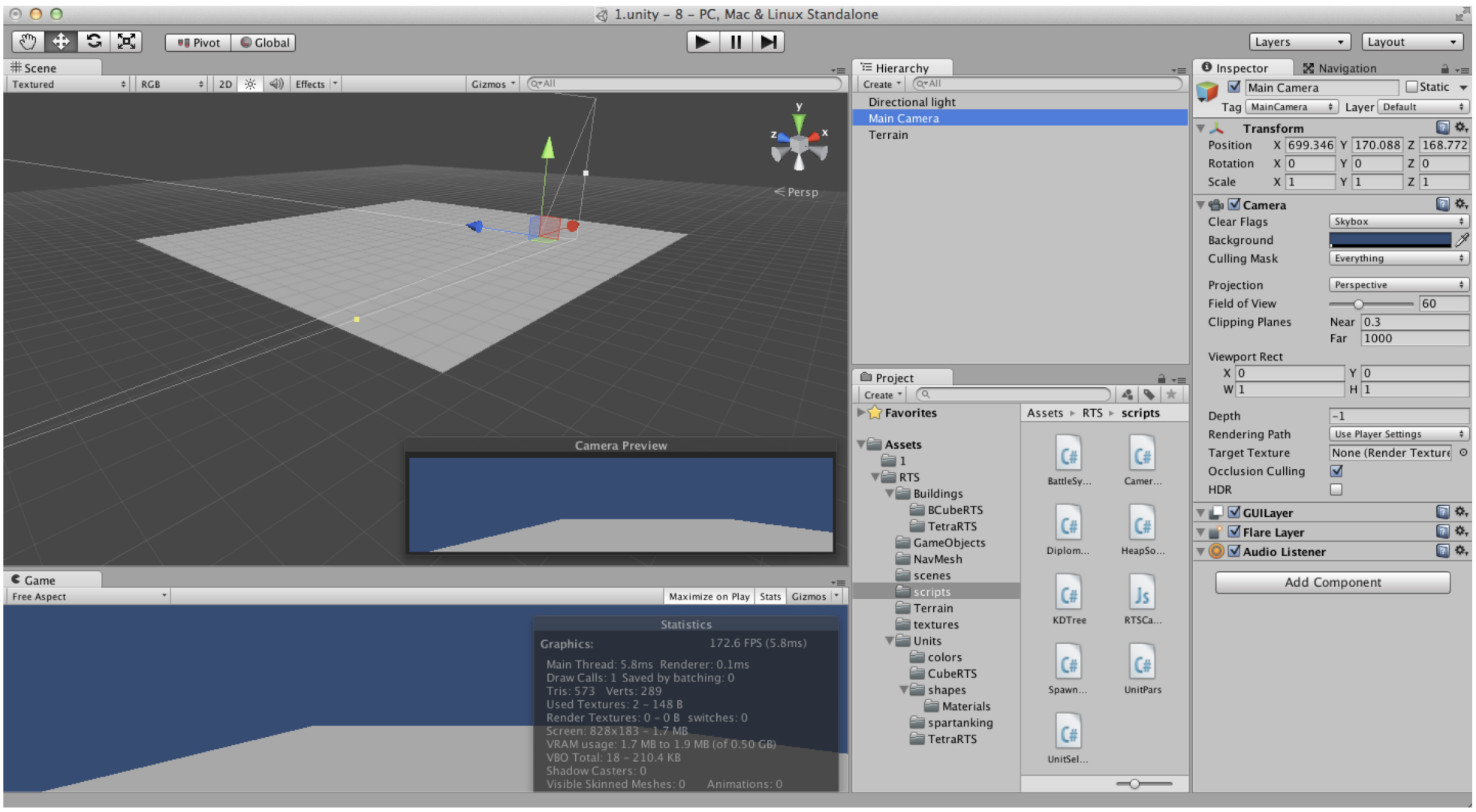

Fig. 8 - Creating Directional Light and moving camera into better position to be able to see terrain in game.

5. Select your Main Camera and from scripts subfolder attach “RTSCameraFree.cs” script to it (just drag script onto Main Camera to some empty space in Inspector, where are other Camera options nearby) (Fig. 9).

Fig. 9 - Adding “RTSCameraFree component to the main camera.

6. Click run to check if you can move RTS Camera with “WASD” buttons and rotate with right mouse button around in game scene.

7. Drag “spawnPointBCube” and “spawnPointBTetra” prefabs from “GameObjects” subfolder somewhere at the middle of terrain – these will be generator points, which will spawn randomly buildings onto terrain. They should appear in the list of other game objects (Fig. 10).

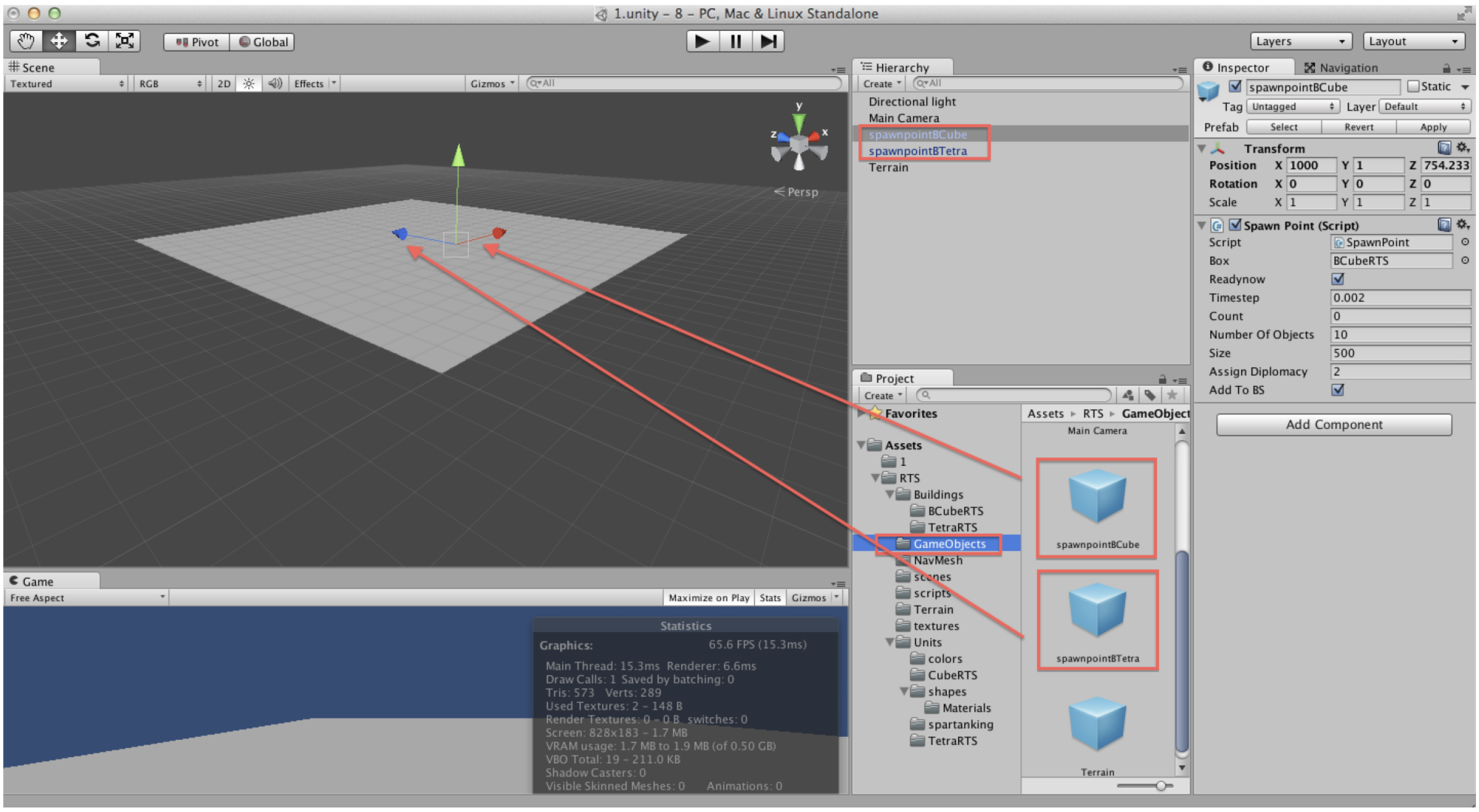

Fig. 10 - Adding prefabs into scene in order to spawn units.

8. Don't run the game just yet but attach main script “BattleSystem.cs” to Terrain (Fig. 11).

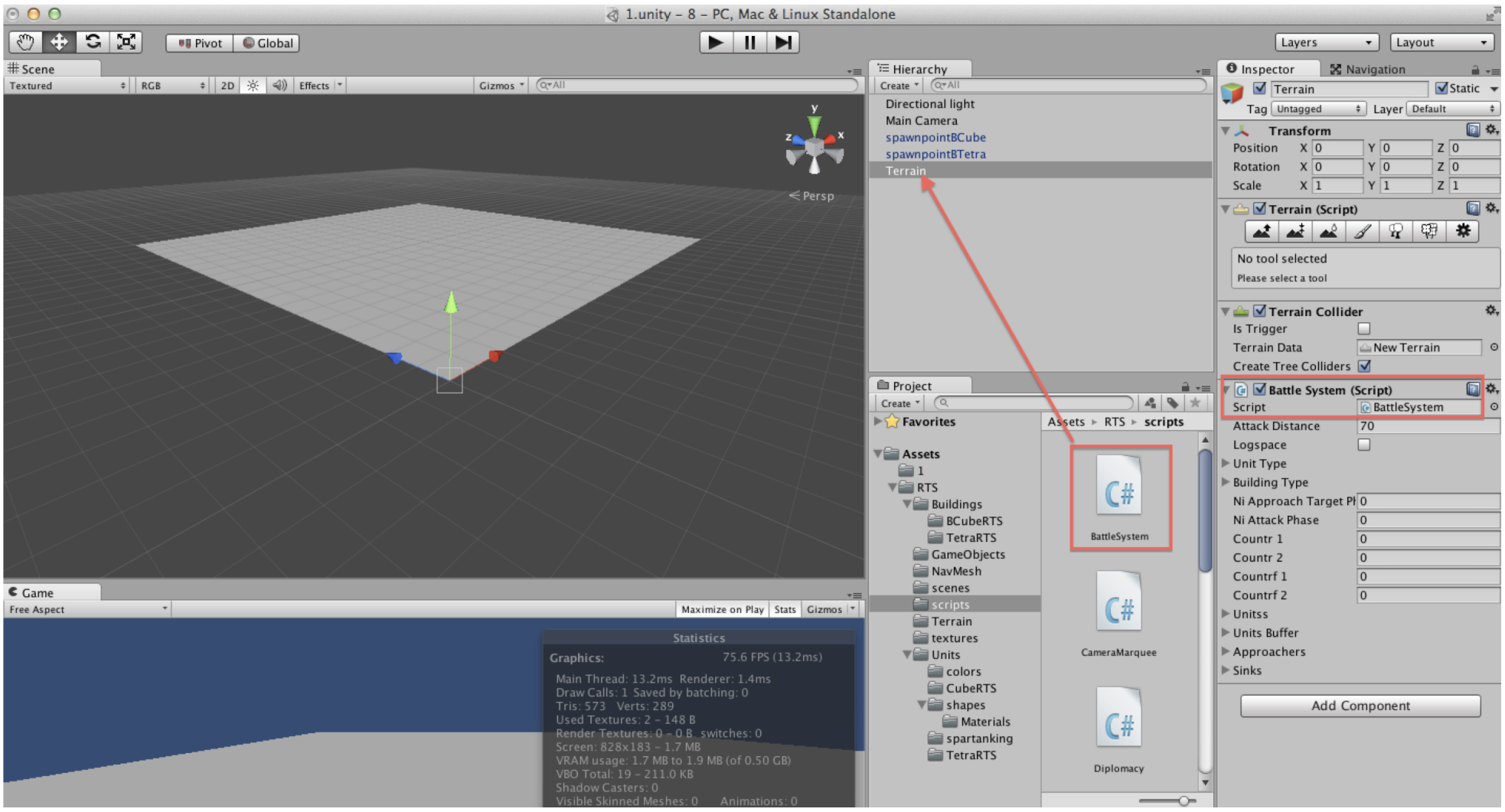

Fig. 11 - Adding BattleSystem script to the scene.

9. Run the game and enjoy your manual setup results (Fig. 12).

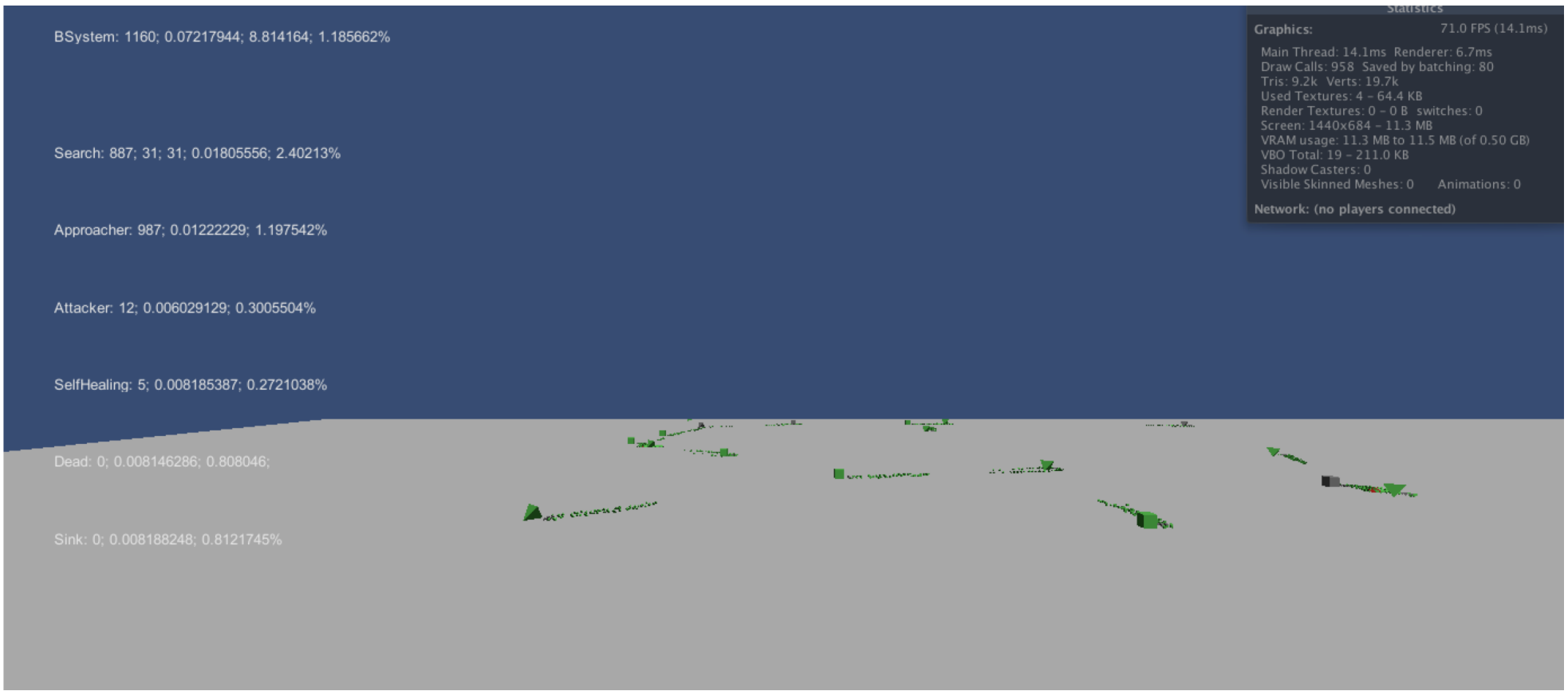

Fig. 12 - Scene gameplay view after manual setup.

4. Parameters

If you want to use your own buildings, you should create prefabs (or just make by duplicating existing ones) exactly the same as “BCubeRTS” and “BTetraRTS” in buildings folder (Fig. 13).

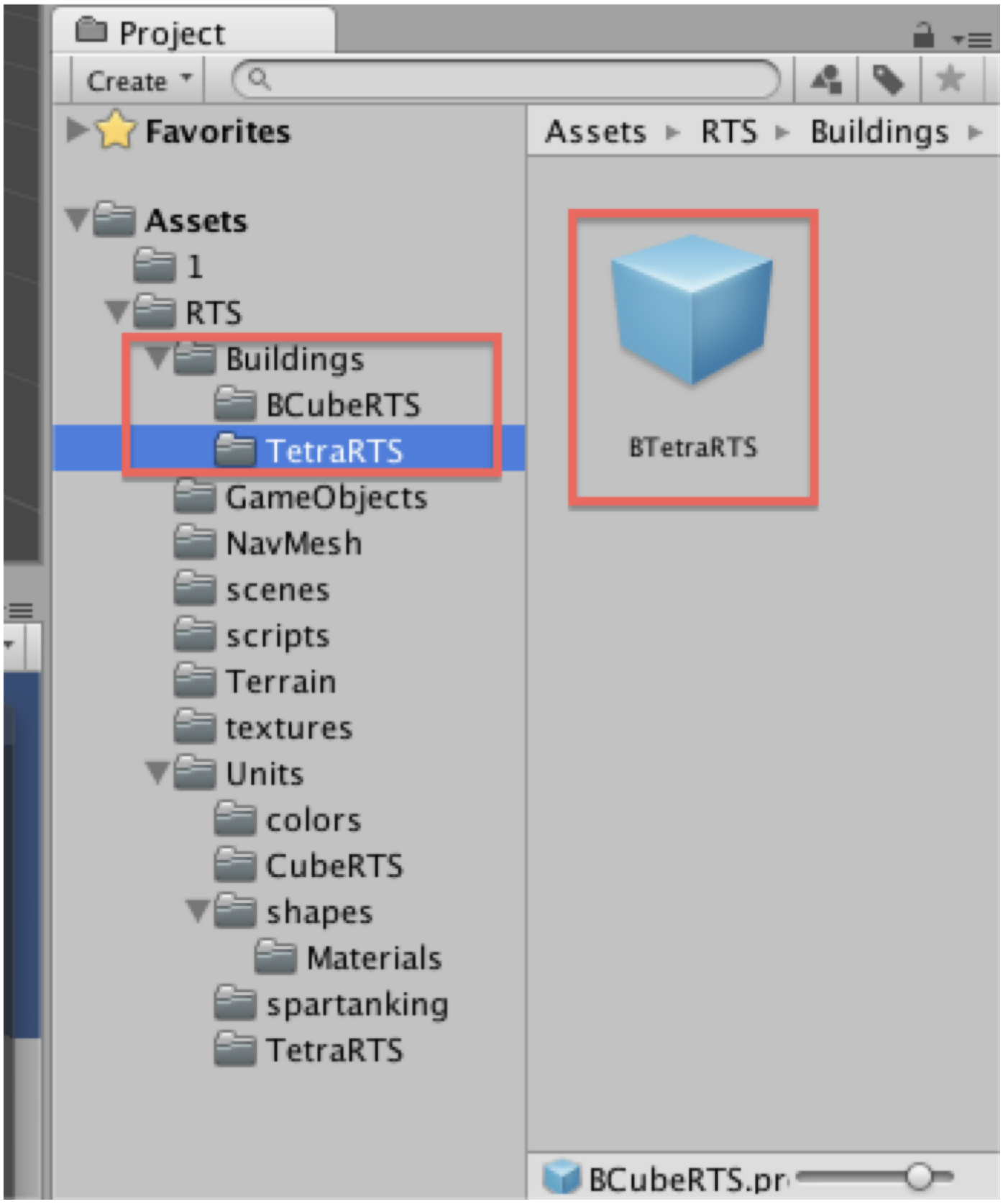

Fig. 13 - Prefabs in Buildings folder.

These prefabs has attached spawn point script themselves, so you can also repeat the same replacement procedure for unit prefabs in Units/CubeRTS and Units/TetraRTS subfolders respectively.

“SpawnPoint.cs” script, which randomly generates objects has parameters as shown in Fig. 14.

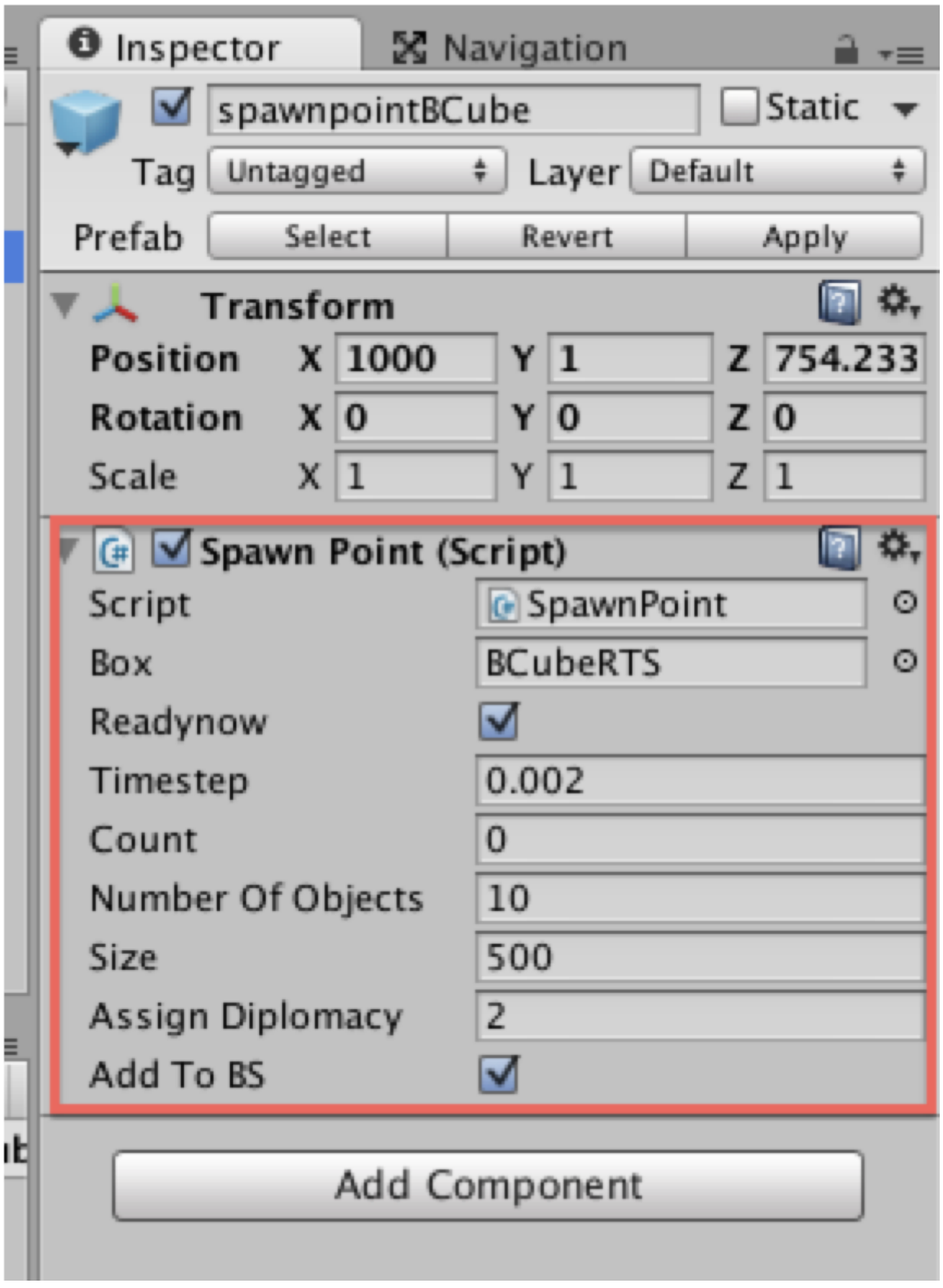

Fig. 14 - SpawnPoint script attached to the prefab.

At the moment “Timestep”, “Number of Objects”, “Size” and “Box” parameters are supported to modify. Other parameters are not yet fully checked (they are for debugging only) and user must be extremely carefully with them, as they can make entire simulator not working if wrong parameters would be set and saved.

- [Timestep] - spawns object every x seconds, where x is number, assigned for thi parameter.
- [Number of Objects] - the number of objects which “Spawn Point” will create.
- [Size] - the size of the box in space, where objects will be spawned.
- [Box] - game object to be spawned.

“BattleSystem.cs” uses parameters from “UnitPars.cs” script, which is attached to units and buildings prefabs (Fig. 15).

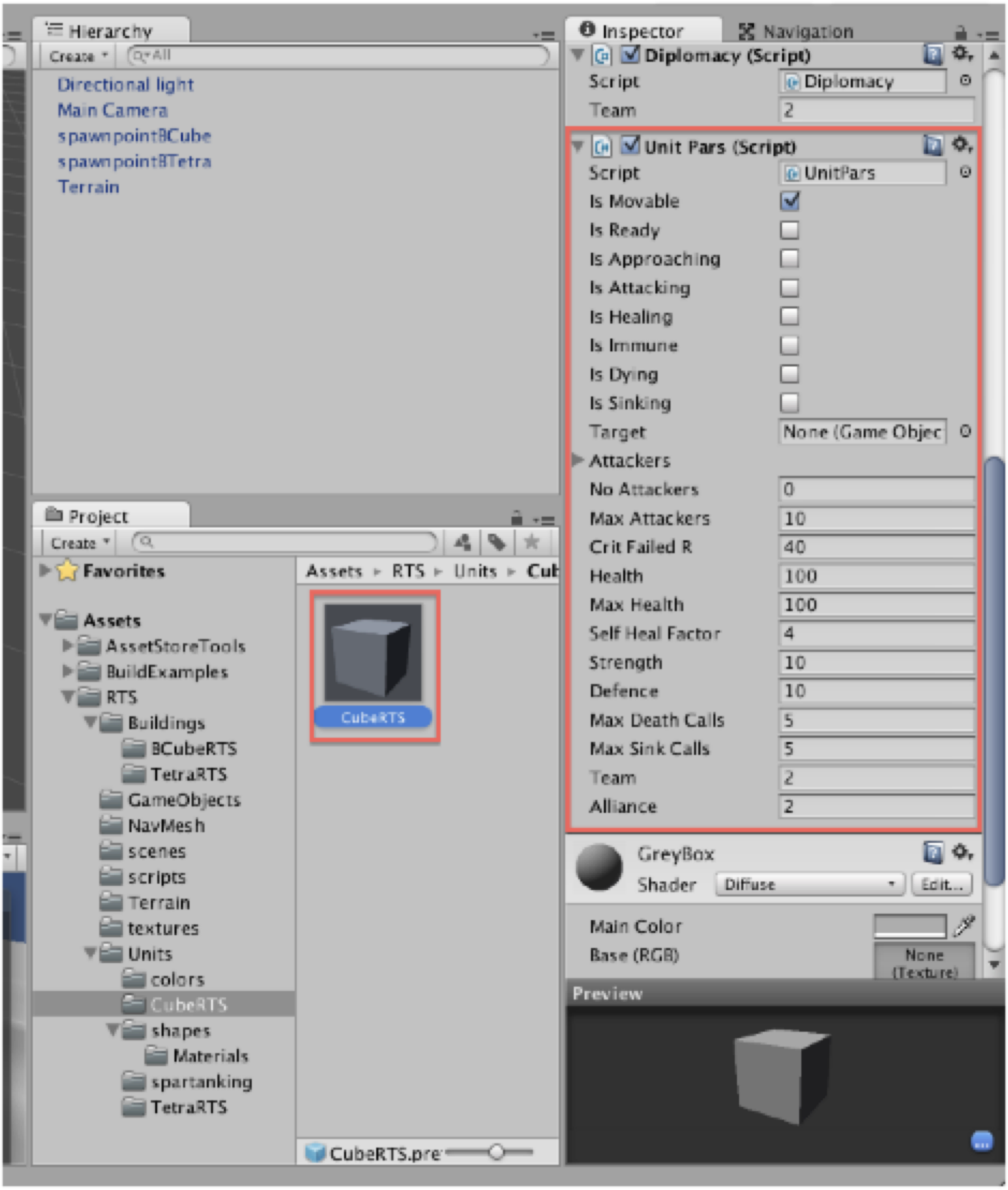

Fig. 15 - UnitPars script attached to the prefab.

It should be safe to modify parameters [Max Attackers], [Crit Failed R], [Health], [Max Health], [Self Heal Factor], [Strength] and [Defence].

- [MaxAttackers] - number of attackers, allowed to attack this unit (currently this parameter not always working well, but it is safe to use it).
- [Crit Failed R] - when approachers are approaching their targets, distance between them must decrease. If not – it is counted as failure to approach. This parameter set maximum allowed failures before resetting new target.
- [Health] - unit current health – if it drops bellow 0, unit dies.
- [MaxHealth] - maximum health, which unit can hold.
- [Self Heal Factor] - amount of health, which unit automatically receives per time unit.
- [Strength] - strength of attacker – factor at which attacker removes opponent health.
- [Defence] - possibility to avoid attacker damage.

“BattleSystem.cs” itself has only one supported parameter at the moment: “Attack Distance”, which sets maximum allowed distance between units to be used by “BattleSystem.cs”.

5. Build-in colours of objects used by simulator

- [Yellow] - unit is ready obtain target.
- [Green] - unit is approaching its target.
- [Red] - unit is attacking its target.
- [Blue] - unit is dying.
- [Violet] - unit is sinking.

More complicated approaches may require modification of scripts. Please view comments inside scripts for more information. Currently more detailed comments are given for “BattleSystem.cs” script, which is the most important script for this simulator.

6. Performance

Battle simulator has build-in performance reporter for detailed analysis how much computing power is taken by each of 6 phases. This reporter is running in game and show performance in right side of the screen (Fig. 16).

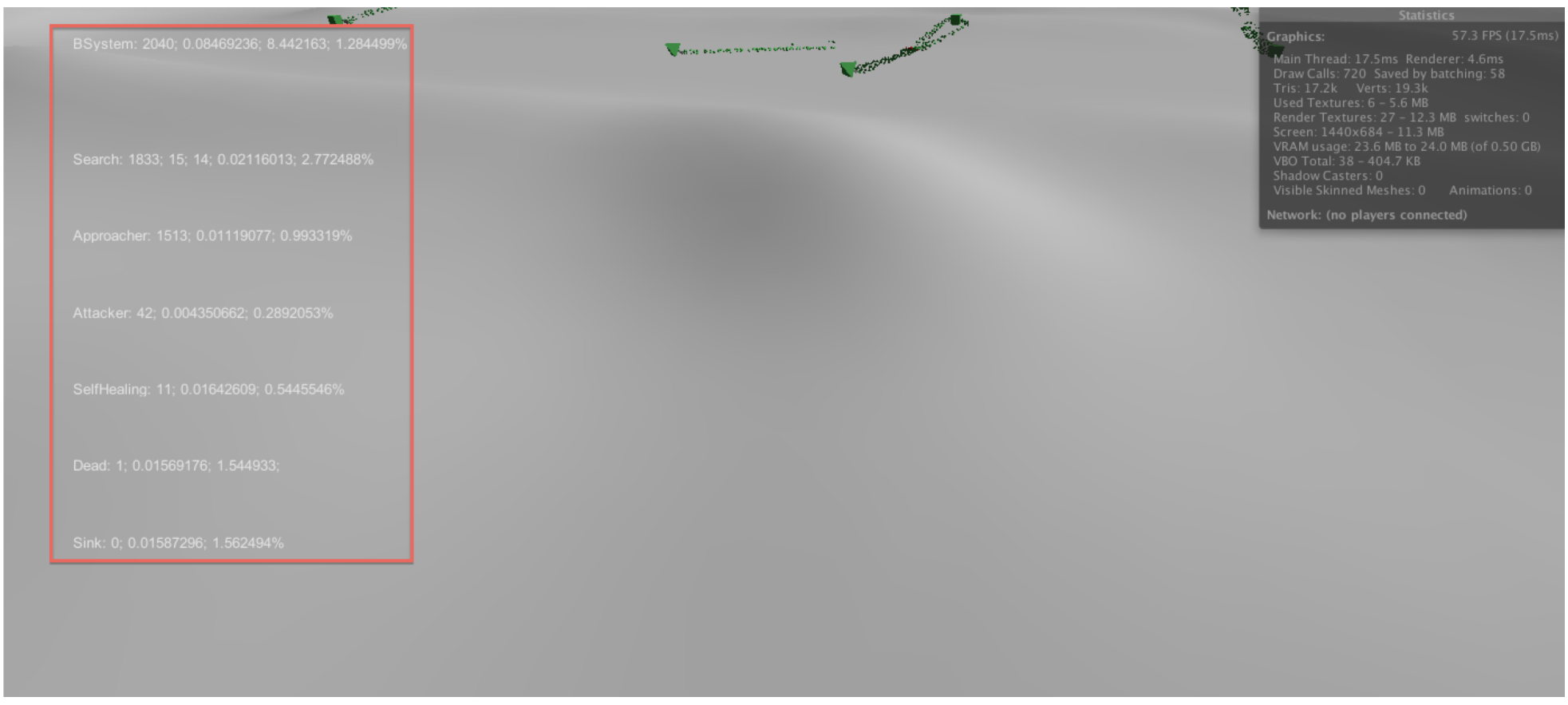

Fig. 16 - BattleSystem performance reporter.

First integer number show how many objects are used by each phase. Search phase has two more integers, showing how many attackers are searching their targets in each of two teams. First float number show time spend in calculations, second float number show total time spend in calculations and WaitForSeconds and third float shows fraction between these two times in percentage, which is phase performance.

First line represents total number of units used by entire simulator, sums of all 6 phases times respectively, and average performance for all 6 phases.

7. Trivia

Older and not well optimised version of this simulator is available for free as demo version on:

https://forum.unity.com/threads/3-9k-units-battle-system.223334/
If you experience an error in your code feel free to post on forums or email to chanfort48@gmail.com with the following details explained:

- Parameters, which you used and how they are different from original ones.
- Error message.
- Double click on Error message to get highlighted line in a script: include a few lines of code, where error appeared, that I could look for reason.
- Any other details, you mentioned and think they could be related with error.

Any notices about unnatural behaviour, “strange things” and other feedback should be posted in

https://forum.unity.com/threads/3-9k-units-battle-system.223334/
to make them available for discussion in community.

8. Credits

- KDTree class used for neighbour searches in order to assign nearest targets for attackers (The code has been adopted to RTS Toolkit needs):
  http://forum.unity3d.com/threads/29923-Point-nearest-neighbour-search-class
- Unit rectangle selection script was based on Camera Marquee tutorial and was adopted to RTS Toolkit Free needs:
  http://paulbutera.wordpress.com/2013/04/04/unityrtstutorialpart1marqueeselectionofunits/