УДК 004.514

AUTOMATED TILE TEXTURING IN UNITY TILESET MAP

Utkin Y.E.
Scientific supervisor – senior lecturer Novikov Y.S.
Kharkiv National University of Radio Electronics
14 Nauky ave., Kharkiv, Ukraine, 61166, Department of Software Engineering
tel.: +38 (095) 381-02-70, e-mail: yurii.utkin@nure.ua

Tilemap system is one the most common instruments you would use when making a game using Unity. Especially when it is a 2D side-view or top-view game. Though, it could be also perfectly fit for isometric games. A regular tilemap level is drawn using different sprite variations of the same tile. For example, when you are drawing water and ground tiles together, you want the border between them to be smooth, where the closest to water ground tiles have a bit water on self, and the deeper ones don't have any. This creates a problem where your set of tiles might be too large to conveniently operate with. Even the simplest scenario of varying depending on the 8 neighboring tiles may result in up to 256 different sprites in your tile set. And if you want to vary sprites on the same position or have more than 1 types of tiles it could require even more sprites. Having so many sprites to choose from when drawing a level is way too hard and ineffective, so an instrument that would be able to do this instead of a you is a must.

Referring to the problem mentioned above, we can describe several requirements to the instrument:

1. The instrument should choose a sprite for the target tile by itself depending on the positions of the adjacent tiles.
2. The instrument should have an option of varying sprites with the same adject tiles.
3. The instrument should detect different type of tiles and choose a sprite depending on that.
4. The instrument should have a convenient interface to interact with.

The best way to solve a problem is to find a ready-made solution. And there is one great option for our problem. It is called "Rule Tiles" from Unity 2D Tilemap Extras package [1]. Though the package is still pre-release, it is quite stable and provides a wide range of features. The best part is as it was made by Unity it has a convenient interface and 100% compatibility with other Unity features from the box. The bad moment is we still need to modify it to meet our requirements.

First, it is necessary to define what functionality the Rule Tiles instrument already has and what must be added. If we look at RuleTile object we'll see that it has a list of rules each of which consists of target sprite, game object, collider, and condition under which it should appear. The condition is set using a grid representing the positions of the adjacent tiles, and signs on it which define whether a tile on this position must or must not be present, or it doesn't matter.

Also, there is an output type field, which has single, animation and random option. Single and animation options work perfectly, while random has one great problem. All sprites in the selection list when setting the random tile selection rule have an equal probability of appearing.

The next step is modifying this instrument to make randomizing tiles better and have an option to detect different tiles. For that purpose, we'll define an AdvancedRuleTile class inherited from base RuleTile class. Inside we'll also provide our Neighbor class inherited from base Neighbor class. Inside it we'll add all possible types of adjacent tiles including self, not self, any, specific tile N, where N – is number of specific tile type, for example, 1 is ground tile, 2 is water tile, 3 is sand tile. Also, we'll add a field with a chance for tile to appear. Using this we can provide a specific appearing chance for each sprite. The only thing that's left is override RuleMatch() method which contains logic of whether current sprite meets the tile rules with our added neighbor tile types and random chance to appear if set. Now we can create an instance of our AdvancedRuleTile and check it out (see Figure 1).

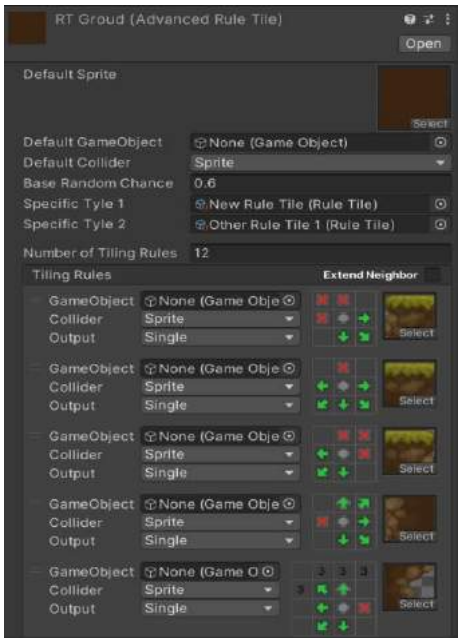

Figure 1 – AdvancedRuleTile object inspector

Other convenient feature we got from implemented tile randomization algorithm is that we can set different game objects and colliders to match different variations of the same tile. Also, we can now randomize variations for animated tiles, which is impossible with basic RuleTile class.

References:

1. Unity 2D Tilemap Extras (Package manual): https://docs.unity3d.com/Packages/com.unity.2d.tilemap.extras@2.2/manual/index.html (date of access: 13.04.2023).