Unity: procedure-generated levels

gamedev-wunder9l.blogspot.com

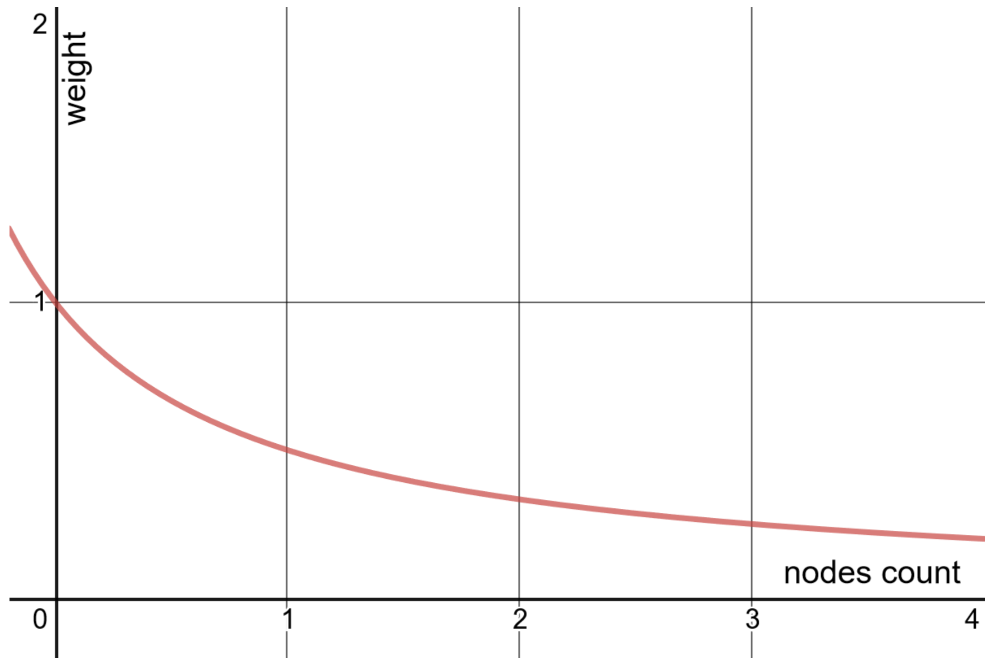
Unity: procedure-generated levels

To begin with, let me give some explanation what was the original goal. As a weekend project I was writing a 2d game. And there was a need for a dungeon map: number of rooms somehow connected with roads. I think that it should be auto-generator, that will create various levels by input parameters (like number of rooms).
At the very beginning I decided to divide the functionality into separate classes: one for generating level and another for rendering.

Generator

I have decided that generator has take maze X-length and number of rooms to generate as input parameters. Additionally it has to try keep the Y-limit, but it is not so strictly as X-limit.

Well, generator will take these parameters in constructor:
What about internals? I decided to use a container that can hold existing nodes and generate new one. I named it MapGrid. Inside it has a grid of nodes (List of List). First List - list of levels. It contains list of Nodes that have the same X-coordinate. On adding new Node to grid it calculates probability to be added to every level. Probability calculates on level weight. If level can be approached it takes weight from formula 1/(1+x):



IFRAME: [1]https://www.desmos.com/calculator/ebmg3wfepy?embed

What does "level can be approached" mean? It means that any potential Node in this level can be approached from existing one. Actually it means either this level has Nodes already or there are any neighbor-level from which new Node can be linked. Example: at the very beginning we have only one Node on the first level. In this case approachable levels are first and second (because from first level's Node could be a road to Node on second level).

When we calculate weights for every level we can choose a level containing future Node according to weights. After that we create Node in random possible position inside selected level. And the last step in Node generating is to make random roads to created Node. Pay attention: at least one road must exist.

Renderer

Tilemap renderer

Firstly I had a need only in rendering dungeon map as a tilemap. For this purpose I implemented a class that takes tilemap to fill, road tile and room tile, and has a method named RenderDungeonMap to render dungeon on this tilemap. Rendering is processed level by level, starting from first one. Each level is rendering from its lowest Y-position (room with lowest Y-coordinate). With this order of rendering we have only two possible roads: from left to right (to right neighbor) and from down to up (up neighbor):

As additional hint I reorder original MapGrid such way, that the lowest Y for all Nodes is equal to zero (shift Nodes by Y-axis).

Side note, as this renderer works with TileBase it is possible to integrate it with other extensions. For example in my project I prepared random tile with [2]this package. It takes multiple images and randomly choose one of them. As a result we have random level generator with random tiles.

How to use generator with tilemap renderer

You can find an example in git repository (scene TilemapRenderer). Link to git is below.

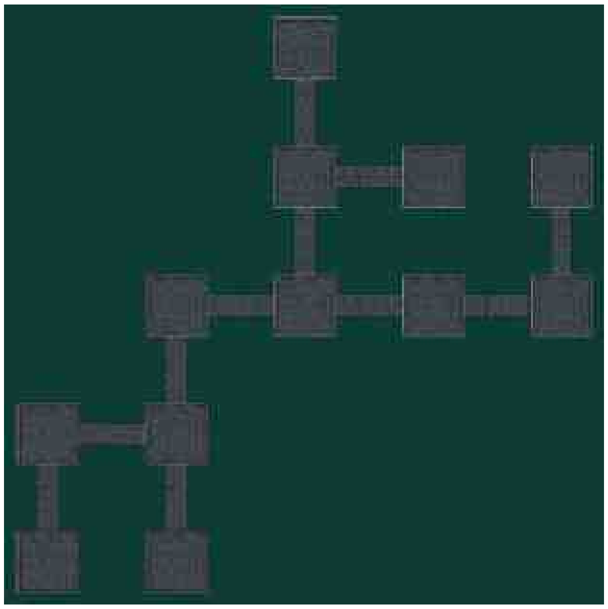
Firstly you should create new tilemap and tile palette ([3]more info you can read here). In the following example I used tiles sheet that is free and taken [4]from here. I assume that you have a palette and tiles for road and room.

Create a tilemap (GameObject->2D->Tilemap) and rename it to something sensible, for example DMTilemap. Move it somewhere that it interfere our ui in future (for example I shifted it by 1000 along X-axis). Add two scripts to this game object: DungeonMapCreator and DungeonMapTilemapRenderer. First one is high level script that communicate with interface and controls Generator and Renderer. Set fields of DungeonMapTilemapRenderer: road tile, room tile and tilemap (itself, you can find it in Inspector). Additionally you can set fields of DungeonMapCreator.

Create new Camera (GameObject->Camera). It will be used to render a texture (from our tilemap to image) so I put it as a child of DMTilemap. Name it (for example DMCamera). Set Camera's projection to Orthographic. Add DMCamera script to it as a component. Fill its TilemapRenderer field with previously created DungeonMapTilemapRenderer. Now we need to create [5]RenderTexture. It will take the picture from camera. ISet it as TargetTexture for DMCamera.

Lets make base interface: Button to generate new DungeonMap and RawImage that will shows us the result. Set our RenderTexture as RawImage's Texture. Set Button's click action to call DungeonMapCreator->TestCreateMap (it creates demo dungeon map). That's it. You can test it by clicking on Button.

One of results:

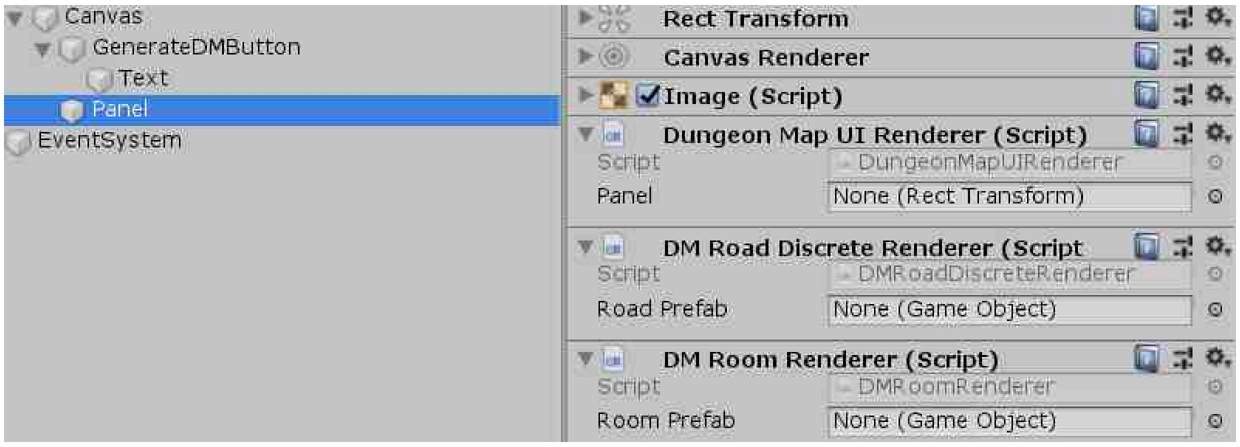


How to use generator with UI-renderer

You can find an example in the same git repository (scene UIRenderer). Link to git is provided at the end of page.

Let's create another scene to demonstrate work of rendering DungeonMap as UI-elements. What is benefits? You can interact with it because rooms are buttons-implemented objects. So you can apply your actions on Button click and provide with your own logic.

First of all create two elements: a button to generate DungeonMap (as in previous example) and a panel (UI->Panel) that will contain further DungeonMap. Add a component to panel named DungeonMapUIRenderer. It has a field named panel - you can drag and drop to it a panel which will be used to draw DungeonMap. By default DungeonMapUIRenderer uses itself gameObject as a panel. You also have to add two more components to the object: room renderer (now only one implementation - DMRoomRenderer) and road renderer (DMRoadDiscreteRenderer).



```
▼ Canvas
  ▼ GenerateDMButton
      Text
    Panel
EventSystem
```

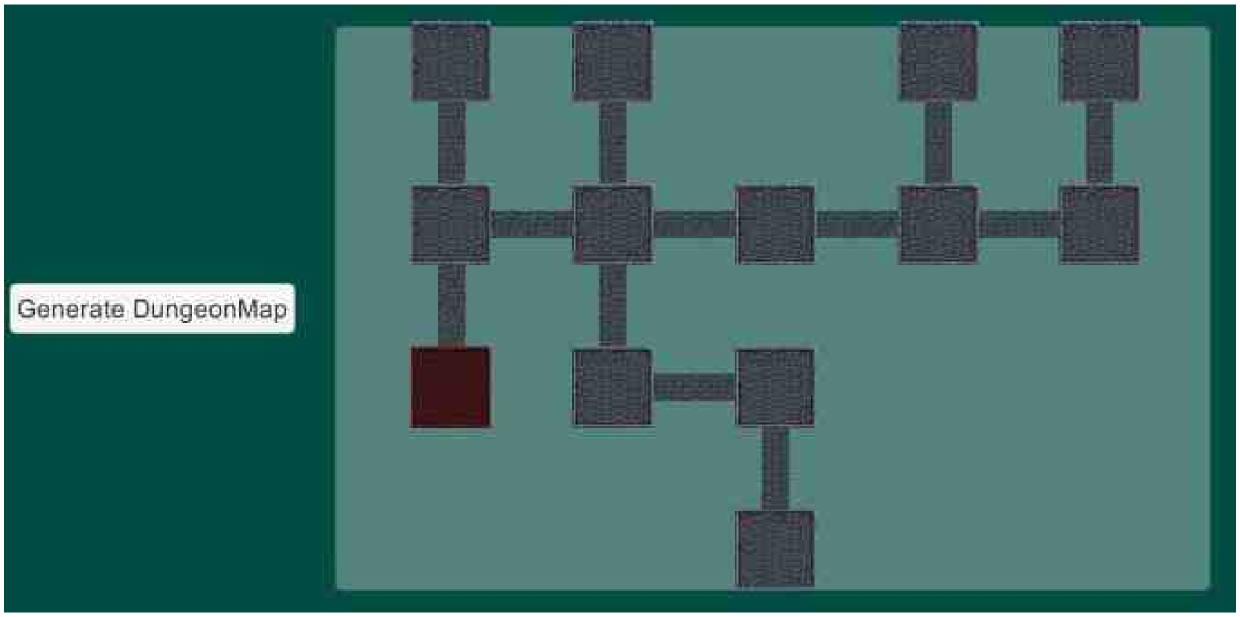
```
▶ Rect Transform
▶ Canvas Renderer
▶ ✓ Image (Script)
▼ Dungeon Map UI Renderer (Script)
  Script          DungeonMapUIRenderer
  Panel           None (Rect Transform)
▼ DM Road Discrete Renderer (Script
  Script          DMRoadDiscreteRenderer
  Road Prefab     None (Game Object)
▼ DM Room Renderer (Script)
  Script          DMRoomRenderer
  Room Prefab     None (Game Object)
```

Both renderers (room and road) takes a prefab as a parameter. What is that? It is prefab of game object that would be rendered as DungeonMap component. Let's create them.

Road prefab would be a simple image. Create it (UI->Image) and name somehow like RoadPrefab. You can use any image as its background. I will use same sprites as in Tilemap. After finishing add it to prefabs (drag and drop to Prefabs folder)

For room I will use Button as prefab. You should take into account that it is just example of use, you can create any prefab what you want and use it. My prefab is a button that on click highlighting red. So creating Button, rename it to RoomPrefab and set your image as button background. After that add a component to it: script that controls button clicks (SimpleRoomButton). Also add this button to prefabs.

Now we can set prefabs to appropriate fields in Panel->DMRoomRenderer/DMRoadDiscreteRenderer. The last step is to add DungeonMapCreator and set calling its method TestCreateMap as button (Generate-DungeonMap) listener. There is no difference where to add DungeonMapCreator. So let's create it as a component of button. You just need to provide a reference to Panel to creator's field "Object With Renderer" (because we put our DungeonMapUIRenderer to panel). Do not forget to set button's OnClick to call creator's method. Final result example:



That's all. You can find sources and example of use [6]here

1. `https://www.desmos.com/calculator/ebmg3wfepy?embed`
2. `https://github.com/Unity-Technologies/2d-extras#master`
3. `https://docs.unity3d.com/Manual/Tilemap-Palette.html`
4. `https://opengameart.org/content/sewer-tileset`
5. `https://docs.unity3d.com/ru/current/Manual/class-RenderTexture.html`
6. `https://github.com/Wunder9l/unity_dm_generator`