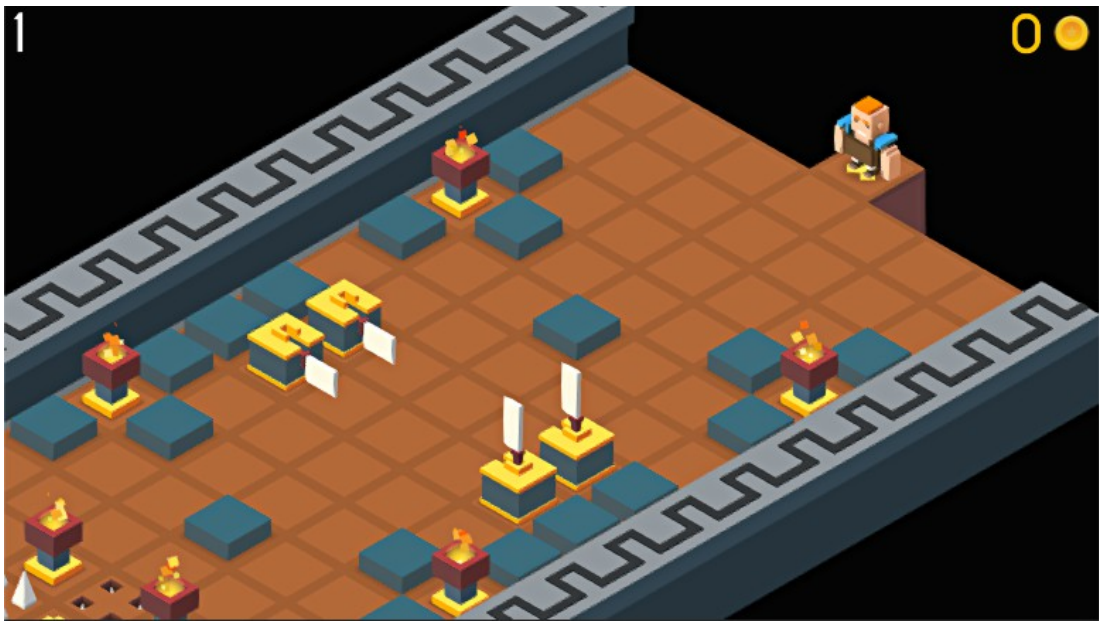


The Verby Noun Toolset

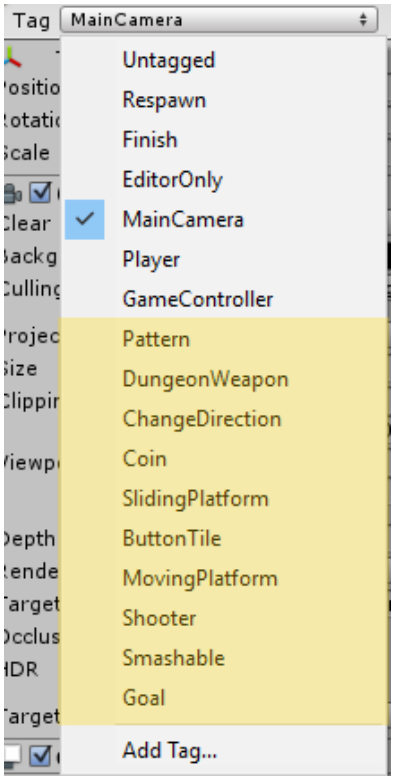
User Guide

1) Introduction

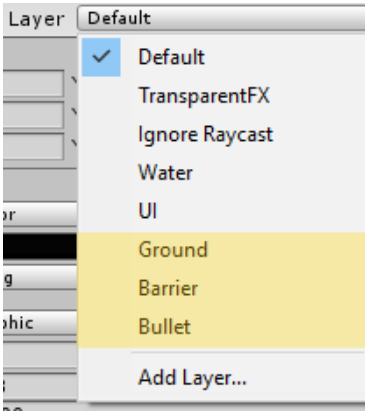
The Verby Noun Toolset seeks to help in the creation of pattern based endless 3D games such as crossy road, looty dungeon, etc. It allows you to control the pacing of the game by stitching hand authored level segments together relatively easily. Most other assets out there that seek to help out in the creation of similar games, randomly generate the levels on the fly, and while the verby noun toolset does use random combination of patterns, since the patterns are hand authored, the game designer can exert a lot more control in how the level eventually play.

2) Setup Guide

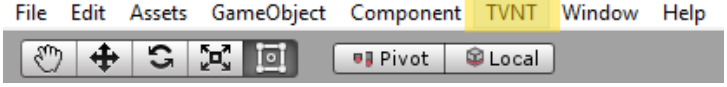
- Create a new unity project and import The Verby Noun Toolset Package into the project.
- These are the tags you need for your project to function optimally.



- Create three layers in your project named Ground, Barrier, and Bullet. These will be used to mark the level tiles that the player can stand on, and the level tiles that will block the player, and the bullets in the layer. The creation of these layers in essential for the sample projects to work.



- Delete the main camera in the your scene and drag the main camera prafab from TVNT/Prefabs folder into your scene. For now do not change any of the values that the camera come populated with (once you get a hold of how the asset works you can go your merry way altering any and every thing).
- To start creating the patterns that will eventually make up your game click on the TVNT menu item that should be showing up along with the other unity menu items. Click on the New Pattern menu option. This should create a new pattern object in you project heirarchy window.



- The pattern object comes with a custom inspector, so lets go over what this

inspector does and how to use it:

The Save Pattern Button

- The Save Pattern button should be the button you use to save the pattern you have created. **Do not use the regular methods of creating a prefab** and applying changes to it, since the pattern creator cleans the pattern of all its prefab connections before saving it as a pattern. This is done as a workaround for unity's nested prefab issue, thus letting you update your prefabs even after patterns that use them are created.
- Patterns are saved to the pattern folder specified in the pattern settings file located at TVNT/Scripts/PatternSettings.cs. Open this file and change the static variable patternPath to the folder you want your patterns saved in. Make sure this folder exists before you try using the save button to save patterns to it. The script will not create the folder structure for you.

Pattern Type

- Starting off, the verby noun toolset comes with 7 patterns types (Start, Connector, Easy, Medium, Hard, Enemy, and End). Again, once a user is familiar with the project they should be able to define customs patterns of their liking. The pattern types are defined in the TVNT/Scripts/Pattern.cs file. Adding to the Type enum will add to the pattern types that show up the pattern type field.
- That Start pattern type is used to tag patterns that serve as the starting blocks for endless runner type games. In the sample games that come along with this project the start pattern holds the start block that is used to know where to place the game character. So if the user wants to follow a similar setup, i would suggest that there is only one active start pattern in every game level.
- The Connector pattern type is just as the name implies. It is normally just a blank pattern made up of floors and walls, without obstacles of any sort to impede the players path. This area is used as a breather for the player to give them some rest after a hectic set of patterns, or before.

- The Easy pattern type is used to mark patterns with obstacles that fall into the difficulty category of easy. This is upto the users preference and starts to highlight the control a level designer can exert on level creation using the verby noun toolset. For instance, by programming a difficulty curve that uses maybe 3 easy patterns, 2 medium patterns, sandwiching an enemy pattern, and ending with a hard pattern the level designer can get easier control of the game flow than would be achievable through the use of a random level generator.
- The Medium pattern type is used to tag patterns that the user feels fall into the medium difficulty category.
- The Hard pattern type is used to tag patterns that the user feels falls in the hard difficulty category.
- The Enemy pattern type is used to designate patterns that contain enemy spawn points. Using this pattern type modifies that layout of the pattern inspector as two other fields appear upon its selection. The enemy type field and the enemy count field.
  - The enemy type field is used to tag the enemy pattern according to difficulty. It is populated with 6 initial types, but as with the pattern type can be extending by adding to the enemytype enum in the TVNT/Scripts/Pattern.cs file.
  - The enemy count field is used to keep track of the number of enemies spawned from this pattern.
- The End pattern type is similar to the start pattern type and can be used to mark patterns that need to show up at the end of levels. In Game2 of the sample games this pattern is used to hold the end level tile, that helps that game script keep track of when the player has reached the end of that given level.

Grid Z

- The grid Z field the value for the length of a pattern. Adjust this value to see the sceneview grid that is drag when the pattern object is selected grow or shrink along the z axis.
- The **grid X** can be found in the PatternSetting files at TVNT/Scripts/PatternSettings.cs. This value is freely open for editing since in most cases a game will use a constant pattern width, varying only the length, and this is the system that this asset uses.

Clear Grid Button

- The clear grid button removes all the objects that have been laid out in a pattern. It is used as a quick way to start afresh if the pattern layout that you are currently pursuing does pan out.

Refresh Prefab Connections

- The prefabs that make up a pattern sometimes lose connection to the nested prefabs that they initialize thus preventing the modification of the initial state of those prefab. Use the Refresh Prefab Connections Button incase you find that the modifications that you make on the levelTile script do not make changes to the objects in the pattern.

Enable Editing/Disable Editing

- The enable editing button does exactly as its name implies. It enabled the user to edit the selected pattern, placing or removing level tiles. Once the user is done editing the pattern they are advised to click on the disable editing button to enable navigation in the scene view without making inadvertent changes to the pattern.

Place Solitary

- The place solitary check box is used to make sure that two level tiles do not occupy the same space unless this is the desired outcome. When this checkbox is off, objects like coins, pots, shooters, chopping blocks, or other level decorations can be placed over ground tiles.

Level Tiles

- Level tiles act as holders for the prefabs that are used to build the pattern. Instead of directly inserting the your level prefabs into a pattern, they are indirectly initialized through the levelTile script found at TVNT/Scripts/LevelTiles.cs. This script can be extended to provide additionally functionality if you level prefab demands it. To get an idea of what can be accomplised take a look at the scripts in TVNT_Samples folder. The pattern inspector populates it level tiles field with all the level tiles it finds at the location specified for them in the Pattern Settings file located at TVNT/Scripts/PatternSettings.cs. Changing the value for levelTilePath will load the level tiles from this new path.

Bottom Entrances

- The bottom entrances buttons are used when patterns are matched. In order to prevent the stitching together of two patterns that might lead to a dead end, the bottom entrances and are matched to the top entrances of the previous pattern laid down to ensure that there is atleast one path through the laid down levels. When these buttons are red it indicates that the block at that particular point in the pattern is not free to be occupied by the character.

Top Entrances

- The top entrances buttons are likewise used to mark of the tiles on the top of the pattern that are not occupied and free for the player to pass through. This is used when requesting the next pattern to make sure that there is always atleast one way through the laid down patterns.

3) The Pattern Settings File

- The pattern settings file is used to hold the global settings that should apply to all pattern you create.
- The pattern path variable hold the location to which the pattern prefabs will be saved.
- The level tile path holds the location from which level tiles will be loaded.
- The tiledSize variable holds the size of the tiles. In the case of the sample project, the tiles are 4x4x4.
- The playerYOffset variable holds the the y offset of the player. For example, the sample projects contrain tiles that are 4x4x4. So the player offset is set to 2 since this is the distance on the y axis that would place the player over the

ground tile, but not floating in the air.

3) TVNTObjectPool

- The TVNTObjectPool object must be included in every scene in which you intend on using the verby noun toolset. The object pool recycles the level prefab in order to keep the memory consumption manageable within your project. Just drag the TVNTObjectPool prefab from the TVNT/Prefabs folder into you scene and you should be good to go.

3) TVNTPatternLoader

- The TVNTPatternLoader is a simple script that loads patterns and provides an easy way to retrive these patterns. You can write your own pattern loader, but if you need an easy way to retrive patterns, using the pattern types that come with this project you can use the pattern loader that comes with this project. To do that you will have to drag the TVNTPatternLoader prefab from the TVNT/Prefabs folder into your scene and provide it with the location at which your patterns are saved.