Enchanted Forest documentation

Project folder structure:

- Good Enough Development
    - Animations – bone based animations for some of the assets
    - Cutscenes – showcase of the demo level
    - Materials – Couple of materials for the background
    - Palettes – tile palette for painting the ground, walls and platforms
    - Prefabs – Configured prefabs for all placeable objects
    - Scenes – demo level and placeable object showcase
    - Scripts – parallax script for the background layers
    - Textures – packed spritesheet and also background images
    - Tiles – tiles for environment tile palette

Background parallax effect

Parallax effect for the background has already been setup, it uses the included InfiniteParallax.cs script. Please check the demo scene, there you will find 5 objects: Sky, MFar, MNear, TFar, TNear – those are the 5 layers of the background, each consisting of different images. Parallax script is attached to all of them and the value of parallax effect is based on how far the layer is from the camera.

Each of those gameobjects references the main camera, so the background images can be moved around based on the current camera position. You may simply use the Cinemachine virtual camera in order to follow a player, and the parallax background will synchronize with the player position.

Please note, parallax effect works only in Play mode, the background does not move while in Scene mode.

Using the tileset

Tile Palette has already been setup with the included tiles. In demo scene you will notice Grid object, which holds two layers – "Ground" is used for painting the actual level using the tile palette. It is using a composite collider to benefit from performance improvements. This is the layer where player will walk upon. The second layer is called Props, it contains prefabs which can be found under Prefabs -> Props folder.

Tile palette has been arranged in a logical way to see which tiles can be placed together. Tiles can be arranged to create different types of platforms, ground, water and also a cave system. Please refer to the demo scene in order to understand how everything fits together.

Animations

Couple of assets have been animated using "Creature 2D" software which is a bone-based system. Similarly how 3D models are rigged with a skeletal bone structure, also 2D images can be rigged in a similar fashion. Once that is done, we used IK (inverse kinematics) to create the desired animation effects.

In order to use these animations, you do NOT need the Creature software, but you need to use the engine runtime, which is a Unity plugin, please refer to the official Creature documentation for setting up the plugin:
https://www.kestrelmoon.com/creaturedocs/Game_Engine_Runtimes_And_Integration/Unity_Runtimes.html

Once the plugin has been setup, all of the animations will work without any issues. All of them follow the same structure – parent gameobject, for example, WillowTree and two child objects – CreatureRenderer, which contains the mesh renderer along with the object’s image, and CreatureAsset, which contains the JSON file for controlling the animation.

You may place the animated gameobjects directly in the scene and they will start working on a loop as soon as the game is launched.