Reyn's Voxel System

Table Of Contents

Credits

- Noise Library from https://github.com/JimmyCushnie/Noisy-Nodes
- Texture Samples from https://opengameart.org/content/big-pack-of-hand-painted-tiling-textures
- Free fly camera script from https://gist.github.com/ashleydavis/f025c03a9221bc840a2b

System Overview

Base World

Purpose

The World.cs base class is provided to serve as a basis for creating your own world systems, and in that regard, implements the majority of the functionality necessary to generate terrain.

Public Properties/Variables

- World Materials
    - These are the materials that are assigned to all generated chunks, there's generally two, one opaque material and one transparent.
    - Provided materials with appropriate voxel functions are found under
    - ReynsVoxelSystem/Materials
- Voxel Details
    - This is an array representing the different textures/colors you would like to use to represent your voxels. There can be up to 256
        - Texture
            - Optional
            - Desired texture to represent voxel
        - Color
            - Required
            - Fallback if no texture is used
        - Metallic
            - Metallic value used for shading this voxel
        - Smoothness
            - Smoothness value used for shading this voxel
- World Settings
    - Tick Time
    - Chunk Size
    - Max Height
    - Render Distance
        - This generally would be half of the total number of chunks to generate for a level system, given a number like 16, you'll end up with a total chunk size of 33x33 (1 chunk representing 0,0)
    - Smooth Normals
        - If true, normals will be calculated during the contouring process, resulting in a smoother terrain. If false, Unity's normal calculation will be used.
    - Use Textures
        - Controls whether textures assigned in Voxel Details are used
- ContouringShader

    - This should always be VoxelContour found in ReynsVoxelSystem/Scripts/ComputeShaders/VoxelContour.compute
    - Generates the mesh data from a voxel field
- DensityGenerationShader
    - This is your density generation shader
    - Examples includes are available in ReynsVoxelSystem/Scripts/ComputeShaders/ and have Density in the name
- Max Chunks To Process Per Frame
    - The amount of chunks that GenerationManager can start the generation process for per frame. Note, more than this number of chunks can be processed in the background, this only refers to starting.
    - Try to keep this number low

Overridable Functions

The World class implements the following abstract functions (functions marked as optional don't require a function body, but most be declared)

- OnStart()
    - Required
    - This call should be used to execute your generation function. Can also be utilized to initialize any other subsystems you may be using.
    - Example available at ReynsVoxelSystem/Examples/Scriptss/Worlds/WorldExample.cs
- InitializeDensityShader()
    - Optional
    - Used to set custom non-changing variables that would be used during Density Generation
- ExecuteDensityStage(GenerationBuffer genBuffer, int xThreads, int yThreads)
    - Required
    - Used to execute your Density shader
- DoTick()
    - Optional
    - Provided for calling voxel based ticks - calls from a background thread at an interval according to WorldSettings.TickTime
- DoUpdate()
    - Optional
    - Provided for calling any necessary calls during Update

Important Public Functions

The public functions within the World class are meant to be called via the World.Instance singleton and generally pertain to the chunks.

- SetVoxelAtCoord(Vector3 chunkPosition, Vector3 voxelPosition, Voxel value)
    - Sets the value of a voxel within a chunk at a position

Implementation Examples

- InfiniteTerrain
    - Complex example
    - Generates infinite terrain that tracks a transform
    - Uses ReynsVoxelSystem/Scripts/ComputeShaders/HeightMapNoise.compute
- IslandLevel
    - Medium example
    - Generates a small island surrounded by water
    - Uses ReynsVoxelSystem/Scripts/ComputeShaders/IslandDensity.compute
- WorldExample
    - Absolute bare bones World example
    - Uses ReynsVoxelSystem/Scripts/ComputeShaders/DensityTemplate.compute

World Modules

- World Modules are used to implement custom functionality in your World Generation process, and can be called automatically at certain times.
- Timings are as follows
    - Before Generation
        - Runs before the OnStart call
    - Before Meshing
        - Runs after Density Generation has run, but before meshing
        - Useful for things like structure spawning
    - On Tick
        - Runs during the World Voxel tick
        - Useful for voxel updates
    - On Generation Complete
        - Runs after the initial generation is finished
        - Useful for spawning
- Examples includes in ReynsVoxelSystem/Examples/Scripts/Modules
- Attached as a Game Object under your World System

Data Generation

This is the stage where you are most likely to do the customizations to produce the level you want, and most of it will be done in a compute file.

- The basis for your Density generation should be taken from ReynsVoxelSystem/Scripts/ComputeShaders/DensityTemplate.compute, as this will demonstrate how to properly handle the voxels internal density field, while also keeping generation fairly simple.
- Additional variables can be configured in the World override for ”InitializeDensityShader” and “ExecuteDensityStage” accordingly with the need to change the vars by chunkPosition.