Voxeland is a next generation of a cubic-style terrain seen in Minecraft and others, but a bit more than that: it is a subdivided and adroitly smoothed cubical structure.

The tool comes with all the assets can be seen in demo: land and grass textures, tree objects, shaders, scripts and even a simple sky. Actually, there is enough assets to make a simple game basics. It will be a good help in a game creation or prototyping.

Special Grass feature automatically plants a shader-animated grass above a selected surface blocks. And a feature to assign any object as a terrain block gives an ability to bind objects to terrain, add or remove them in-game like any other block or even create terrains made of objects.

Quick Tutorial

To create Voxeland click GameObject -> Create Other -> Voxeland Terrain. A creation window will popup. Chunk size and initial generate parameters could be set there.

All the terrain is split into a separate meshes - chunks. Each chunk is 25 units in width and length by default. To change the chunk size enter a new int value. It is recommended to set the chunk size between 10 and 30.

Initial Terrain popup controls the type of initial terrain: “flat” will create a flat terrain with closed sides at a desired Initial Terrain Level, “generate” will generate terrain using default Generator settings, and “empty” will create an empty terrain. If you need to generate terrain with non-default settings, create an empty terrain and then use Generator feature. Use Initial Terrain Size field to set the starting size of created terrain.

Select Voxeland in a Hierarchy list, and you are ready to edit your terrain:

- To add block click left mouse button above the selected block.
- To remove a block click left mouse button with shift button pressed
- To blur the area click left mouse button with control button pressed (terrain could be blurred when brush extend is more than 0)
- To replace a block click left mouse button with control and shift pressed

Block type could be selected in a Block Types foldout by clicking a block type card. A new terrain has only one block type, it is named "Ground" and this block has a white texture. To change the name click on "Ground" label. To change the texture - assign it into a texture slot and press "Rebuild" button.

To enable terrain editing during playmode enable "Playmode Edit" in "Settings" foldout.

To load v2 terrains

Select old data asset file – inspector will display that it's script is missing. Assign VoxelandData.cs in the missing slot and save project (File → Save Project). Operations with large data file can cause lags.

Select Voxeland object and assign VoxelandTerrain.cs as a missing script. Then press “Load v2 Data” in Import and Export section. Load old data asset file.

If terrain has no mapping assign triplanar terrain shader or turn off “Weld Verts” in settings. If highlight has no materail – assign it in Materials section.

General properties

- **Brush size** – the extend of the brush. If size is more than 0 a volume brush is turned on, which allows to set multiple blocks in one click.
- **Spherify** – smoothes volume brush and shapes it in form of pseudo-sphere. Please note that Voxeland data is a cubical structure which cannot give clear spheres – all round objects will become a bit aliased.
- **Rebuild** – forces all terrain chunks to rebuild. This button should be pressed to make settings changes take effect on a displayed terrain. Destroys all the terrain chunks, objects and children and creates a new once using the terrain data.

Block Type properties

This section displays the list of of block type cards. Each card corresponds to a block type on terrain. All block settings changes require Rebuild operation.

- Block name label: unnecessary text field, but the block name assignment will help to find block type quickly.
- **Filled:** check this the block is filled with stratum, and uncheck if it is the air or object block.
- **Texture** and **Bump**: Diffuse texture map (left) and normals map (right) used by this block.
- **Different Top:** Some blocks may have different top texture - if face normal pointing mostly to the top it uses assigned top textures. This could be useful for snow or grass. Textures are assigned equivalent to main textures: the left one is diffeuse, right – bump.
- Smooth: Geometrical smoothness of current block type. Default is 1 – the smoothed block, 0 will make block geometry look like a box. Please note that geometrical smoothness does not affect normals.
- Grass: turn this on if block has standing grass above. The type of grass is set using Grass section. You can place different grass types on the same blocks.
- Prefab: Objects could be placed or removed using terrain editor. Placing object with prefab will instantiate prefab in scene.

Grass Type properties

- Type name label: unnecessary text field, but the block name assignment will help to find block type quickly.
- Material: Grass type material. It requires a grass shader and a grass texture with a special layout. The grass texture consists of 4 squares, each of them contains 2 triangles of grass. Each staright angle of a triangle - is the top grass point, and two others - a the bottom (ground) corners. Please use a demo grass texture as a reference.

Land Generator

Various algorithms for generating procedural terrain. You can turn any of them on or off with a checkbox near their names.

- Generate Center: the center of the area where terrain will be generated. **Use Camera Position** will set Generate Center to current camera position during generation.
- Generate Range: the extend of the generated area
- Seed: random seed used in noise generation.
- Overwrite: Enabling this toggle will clear area and generate new terrain. If this toggle off the terrain will remain, and new terrain will be created only where it is empty.
- Initial Level: the initail height of the terrain.
- Noise – creating terrain with fractal noise. This will add noise blocks to the planar filled blocks (if any).
    - Type – type number of noise blocks
    - Fractals – the number of noise fractals
    - Min Fractal Size – dimensions of the lowest fractal
    - Max Fractal Size – dimensions of the biggest fractal. The fractal sizes of fractals between min and max spread linear between min and max sizes.
    - Min Fractal Value – strength of a minimum fractal
    - Max Fractal Value – strength of a biggest fractal. Strength of the fractals between min and max spread linear between min and max values.

- Terrace – creates step-like landform
    - Min and Max Terrace: terrace size minimum and maximum
    - Incline: incline topper (most eroded) part of a terrace
    - Inlcline length: size of the inclined area
    - Opacity: total strength of the generated terrace map.
- Valley – big “main” terrace. Sets the height within Size to the Level.
    - Level: starting level of valley
    - Size: valley height
    - Opacity: total strength of the generated map
- Plateau – clamps the generated heightmap from the top
    - Level: clamp level
    - Opacity: total strength of the generated map
- Erosion – calculates rainfall erosion and sediment.
    - Type: the block type of sediment
    - Erosion Amount: amount of ground taken in each iteration. Bigger values will get more eroded but less realistic ground.
    - Sediment Amount: amount of ground that could be settled to map from mudflow.
    - Iterations: how many times script calculates erosion. Bigger values get more eroded soil, but will slow down calculations
    - Lower Sediment: subtracts this value from sediment map each iteration
    - Torrent Max: maximum amount of water that can carry soil
    - Torrent Blur: blurs torrent map with this amount. Lower values will make torrents more sharp, but will make erosion map more noisy.
    - Additional Noise: adds noise to prevent noticeable 1 unit height sediment steps.
- Grass – generates vertical grass map
    - Type – type of generated grass
    - Noise Size – min fractal of grass perlin noise
    - Density – zero density will not generate any grass, and density value 1 will plant grass everywhere
- Generate in Range – will generate terrain in Generate Range on Generate Center using other generator settings.
- Clear Terrain – will remove all terrain data and create zero terrain.

Materials

Rebuild does not need when any of these parameters changed.

- Land Shader - shader used in all chunks materials. Voxeland comes with 2 demo shaders - 4 blended textures with bump-maps and 4 blended textures with no bump. Assign the last one if you are not planning to use terrain with normal maps
- Land Specular – Land material specular color. Equivalent to material specular color
- Land Shininess – Land material shininess. Equivalent to material shininess
- Land Bake Ambient – Adds additional light to the areas laghted with additional ambient. Higher values will highlight the additional ambient effect but will make the picture over-bright
- Triplanar Tile – size of triplanar texture.
- Additional Ambient – calculates ambient occlusion using block structure. It does not darkens ambient that already exists, but adds new ambient of given color. The effect is strongest when scene ambient is dimmed.
    - Color – color of additional ambient
    - Spread – ambient occlusion blur. Higher values will bright the dark hollows
    - Margins – increase this to avoid seams in ambient between chunks. Should be equal or more than spread. Higher values will give less artifacts but will slow down calculations.
- Grass Material – Polygonal grass material. Requires a grass shader and a grass texture with a special layout
- Grass Animation Speed – How fast the grass is animated by the terrain. And the magnitude is set in a grass material
- Highlight Material – The material of a brush highlight. Material with another color or transparency could be assigned

Settings

- Infinite: turning this checkbox off will limit terrain to area with Size of Terrain Extend based in Terrain Center.
- LOD Distance – the distance of turning high-poly chunk mesh off and enabling it's low-poly version, which has 4 times less triangles. This feature does not use Unity's LODGroup component so it works with non-pro version.
- Build Distance – chunks in this range from camera are automatically build. When a camera comes to chunk it is build, and when camera goes away it is destroyed. This will save memory and draw calls on big terrains.
- Auto Generate – turn off automatic chunks build. This will freeze already built chunks and will not build new chunks when camera changes position. Press “Generate Now” button to refresh chunks.
- Chunk Size – dimensions of a terrain element mesh. It is recommended to set it somewhere between 10 and 30. Higher values will speed up the whole terrain building and reduce the number of draw calls but will slow down the terrain editing.
- Horizon Plane - planar (non-voxel) terrain to display on far distances as lod.
    - Extent – half-size of far plane in chunks
    - Density – grid density (verts per chunk)
- Playmode Edit – Enables in-game terrain editing.
- Weld Verts – welds all chunk verts. Turn this off to enable old-style mapping
- Multithread Edit – using all cpu threads in terrain build and editing.
- Hide Chunk Objects – hides chunk objects in inspector.
- Hide Chunk Wire – do not display terrain wireframe.

Import and Export

- Import 2.0 Data – loads old Voxeland data. Assign VoxelandData.cs script to data asset before loading.
- Import Heightmap – loads heightmap in PNG format. The average of three channels (r,g,b) is used as height value
- Load TXT – loads string data from text format
- Save TXT – saves data to text format
- Bake – bakes terrain to Unity meshes and turns Voxeland script off. Note that terrain should not be infinite.

Scipting FAQ:

How can I get or set blocks with scripting?
Use `Voxeland.GetBlock(int x,int y,int z)` (returns type:int) and `Voxeland.SetBlock(int x, int y, int z, byte type)` functions. X,y and z are coordinates of the block, and type is the number in block types array. Please note that types array is 1-based, type 0 is empty (air) block.

How can I set multiple blocks at once?

When you set many nearby blocks in one frame it is better use `Voxeland.SetBlocks(int x, int y, int z, byte type, int extend, bool spherify, bool replace)` command, where extend is radius of blocks area, spherify sets the sphere area form (when spherify is false form is cube), and if replace is true function will ignore non-filled blocks.

How can I edit the terrain in-game using playmode camera or ray?

You can use `Voxeland.Edit(aimRay:Ray, mouseDown:boolean, shift:boolean, control:boolean)` command. Aim ray is the ray pointing on Voxeland (it could be `Ray(Camera.main.transform.position, Camera.main.transform.forward)` for games with cursor locked to the center of screen or `camera.ScreenPointToRay(Input.mousePosition) for games with free cursor`). Other parameters – are the analogs of Voxeland editing in scene view, they are exposed to make custom assignments. For example if you wish to use space instead of mouse pressing you can use this: Edit(aimRay, Input.GetKey(KeyCode.Space), shift, control). And certainly you can use Input.GetButton instead.

The selected block type could be set with Voxeland.selected. All block types are stored in Voxeland.types array. Each type has these variables:

```
name : String
filled : boolean
texture : Texture
bumpTexture : Texture
hasGrassAbove : boolean
object : Transform
```

For example to get selected type's name use:

```
voxelandComponent.types[voxelandComponent.selected].name
```

How can I save and load terrain in-game?

All the terrain data stored in a IntervalTree object, that could be accessed as voxelandTerrainInstance.dataholder.data. IntervalTree can convert all the data to byte list or string, which could be saved in desired location. Use IntervalTree `SaveToString()` (returns string) and `LoadFromString(string)` functions.

This example saves terrain data to persistent data path folder:

```
using (System.IO.FileStream fs =
new System.IO.FileStream(Application.persistentDataPath + "/VoxelandData.txt",
System.IO.FileMode.Create))
      using (System.IO.StreamWriter writer = new System.IO.StreamWriter(fs))
            writer.Write(land.dataholder.data.SaveToString());
```

And this will load saved data:

```
using (System.IO.FileStream fs =
new System.IO.FileStream(Application.persistentDataPath + "/VoxelandData.txt",
System.IO.FileMode.Open))
      using (System.IO.StreamReader reader = new System.IO.StreamReader(fs))
            land.dataholder.data.LoadFromString( reader.ReadToEnd() );
land.Clear();
```

How can I synchronize terrain changes over the net?

In order to sync terrain over the net you need to monitor calling SetBlocks function on all the players computers. Once one of the players calls SetBlocks you need to call it on other computers with the same parameters with RPC.

If you are calling SetBlocks function from your custom scripts then you should call RPC along with calling SetBlocks. The alternative way is to add RPC to SetBlocks function itself (it is not the best way thought - you will have to write RPC in every new version of Voxeland).

How can I send changed terrain to newly connected clients?

You can save already connected client's terrain data to byte list with `IntervalTree.SaveToByteList()`. This function will return List<byte>. Then you can send this list over the net and load it on new client with `IntervalTree.LoadFromByteList(List<byte> byteList).`

Preparing Textures

This is the reference page devoted to old-style Voxeland mapping. It describes how to prepare textures for non-triplanar mapping.

Voxeland textures have 4x4 elements, each of them could be tiled in any direction. To prepare a texture you have to create 16 tilable elements. All elements have to be tiled among themselves regardless the tile side.

The easest way to do prepare a texture is:

1. Open the square texture source (for example, it will be 1024*1024), select 1/16 of it (256*256) and crop.
2. Mirror the cropped texture vertically.
3. Select triangle with two sides on the edge of cropped texture, and the last one is in the center. Note that the mirror seam should be inside the triangle.
4. Copy the triangle four times, each time rotating on 90 degrees around image center.
5. Mask or erase all the central part of the image. Actually, only thin borders will remain which are gradually fade. The thinner the borders - the better will be a result, but the sharper will be a transition between borders and base texture.
6. Tile the resulting frame four times in length and four times in height - so we get back 1024*1024 texture, made of 16 frames
7. Apply 16-frames texture over the base texture. Now you got tiled and prepared for use in Voxeland texture.