Voxel Generator Programs

A spatial entity will go through several stages in its lifetime. It is possible to provide custom logic for some of these stages. This version of the Voxel Farm platform allows running custom Python programs to produce volumetric spatial output. This type of program is known in the system as a Voxel Generator program.

Typically, a Voxel Generator program will go through the following stages:

1. Ask the user for input values, in case the generated object is parametric, and the user can provide values to these parameters.
2. Set the spatial entity bounds. This step is optional but should be always included because it allows the system to evaluate the spatial entity only within its known bounds, which is more efficient.
3. For each voxel in the spatial entity, emit a field density o other voxel attribute.

Initialization

Before anything else, the Python program must include the Voxel Farm Python library:

```
import voxelfarm
```

Voxel Generator programs must make sure to initialize the Voxel Farm Python library by calling “voxelfarm.init()”:

```
voxelfarm.init()
```

The program can then ask the user information about the entity using the “voxelfarm.input()” function:

```
h = voxelfarm.input(“altitude”, “Altitude”)
```

At this point, if the program knows of the spatial bounds of the entity, the bounds can be set by calling “voxelfarm.set_entity_bounds_x”, “voxelfarm.set_entity_bounds_y” and “voxelfarm.set_entity_bounds_z”.

```
vf.set_entity_bounds_z(h - 1000, h + 1000)
```

The program can either output voxel data, or field data. Field data will be contoured on the fly by the system and it is best suited for smooth surfaces. If the program needs to produce sharp edges and points, it is better to output voxel data.

Field Data Output

To output field data, the Python program must iterate over each voxel in the field. The range of voxels in a field is found in the “voxelfarm.field” variable:

```
for v in voxelfarm.field

    # will execute for each voxel “v” in the field
```

To get the coordinates in project space of the voxel, use the “voxelfarm.get_field_origin” function:

```
for v in voxelfarm.field

    p = voxelfarm.get_field_origin(v) # returns tuple for xyz
```

Using these coordinates, the user program can make useful computations. The program will have access to any Python library that is included in the Content Server. These computations eventually will produce a value for the field in the voxel’s origin. Voxel Farm will produce a surface wherever the field becomes zero. Places where the field is positive, will be considered “outside” the surface. Places where field is negative, will be considered solid and inside the surface.

To set the field value for the voxel, call the “voxelfarm.set_field” function:

```
for v in voxelfarm.field

    p = voxelfarm.get_field_origin(v) # returns tuple for xyz

    field = MyFieldFunction(p) # a call to custom function that outputs the field
```

```
    voxelfarm.set_field(v, field)
```

The following program creates an infinite plane with bumps on top of it:

```
import voxelfarm as vf

import math

vf.init()

h = vf.input("h", "Altitude")

vf.set_entity_bounds_z(h - 1000, h + 1000)

for v in vf.field:

    p = vf.get_field_origin(v)

    dh = 300 * (math.sin(0.001 * p[0]) + math.sin(0.001 * p[1]) + math.sin(0.001 * p[2]))

    dh = dh + 100 * (math.sin(0.01 * p[0]) + math.sin(0.01 * p[1]) +  + math.sin(0.01 * p[2]))

    vf.set_field(v, p[2] - h - dh)
```

Voxel Data Output

Consider using Voxel Data output in cases where you need the resulting spatial entity to contain sharp features.

Voxel Farm's voxel format allows for both sharp and smooth features. Once a spatial entity is built to output voxel data, its program must at least output a material definition for each voxel. Additional optional channels are a 3D surface vector and a color.

To output voxel data, the Python program must iterate over each voxel in the request. The range of voxels in a request is found in the "voxelfarm.voxels" variable:

```
for v in voxelfarm.voxels

    # will execute for each voxel “v” in the request
```

To get the coordinates in project space of the voxel, use the “voxelfarm.get_voxel_origin” function:

```
for v in voxelfarm.field

    p = voxelfarm.get_voxel_origin(v) # returns tuple for xyz
```

Using these coordinates, the user program can make useful computations. The program will have access to any Python library that is included in the Content Server. These computations eventually will produce a value that will determine if the voxel is set as solid material, and which values should be used for color and the surface position inside the voxel.

To set the material for a voxel, call the “voxelfarm.set_material” function:

```
for v in voxelfarm.field

    p = voxelfarm.get_field_origin(v) # returns tuple for xyz

    field = MyFieldFunction(p) # a call to custom function that outputs a field

    if (field < 0.0)

        voxelfarm.set_material(v, 1)
```

The following program creates an infinite plane with bumps on top of it:

```
import voxelfarm as vf

import math

vf.init()
```

```
h = vf.input("h", "Altitude")

vf.set_entity_bounds_z(h - 1000, h + 1000)

for v in vf.voxels:

    p = vf.get_voxel_origin(v)

    dh = 300 * (math.sin(0.001 * p[0]) + math.sin(0.001 * p[1]) + math.sin(0.001 * p[2]))

    dh = dh + 100 * (math.sin(0.01 * p[0]) + math.sin(0.01 * p[1]) +  + math.sin(0.01 * p[2]))

    field = p[2] - h - dh

    if (field < 0.0):

        vf.set_material(v, 1)
```

Revision #1
Created 17 March 2025 19:01:21 by admin
Updated 17 March 2025 19:08:07 by admin