View Programs

A View entity defines a way to visualize several datasets together. In order to decide which datasets should be used, which information should be filtered, and how the results should be visualized, the user can supply a View Program, which will run a custom logic to construct a view layer.

A View program will typically go over the following stages:

1. Ask user input. Here the program prompts the user information to select which datasets and attributes should be used, visual options for rendering, and any other piece of information the view program requires in order to run.
2. Based on user input, create and configure a mashup of datasets from the available datasets in the project.
3. Set up rendering of the elements produced by the dataset mashup.

The following program creates a simple view layer to visualize a terrain mesh:

```
import voxelfarm as vf

terrain = vf.input_entity("terrain", "Terrain", vf.type.voxel_terrain)

mesh = vf.load_mesh(terrain, 0, None)

material = vf.new_material()

material.gradient = "dirt"

vf.render(mesh, material)
```

The program calls “input_entity()” to prompt the user for a terrain model. Next it asks the system to load the mesh data for the terrain using “load_mesh()”.

The following two lines set up a material that will be used to render the terrain mesh. The material is created by calling the “new_material()” function. The “gradient” property of the material is set to “dirt”, instructing the rendering system to ready this gradient texture and pass it as the gradient to be used by the GPU shader.

The last line calls the “render()” function, which instructs the mesh must be rendered with the material that was created in the previous lines.

Working with materials

A view program is expected to make at least one call to the “render()” function. This function takes two arguments, the last argument is the material that will be used to render the element. If no material is provided (“None” passed as material), the system will use the default material.

In this version, materials contain only two properties:

color	Defines an RGBA value as a four-member tuple. The RGB values will be used to tint the visual output for the component, while the alpha value will be used to make the surface transparent.
gradient	Can be one of the three available gradient modes: 1. “color” – Selects a vibrant color gradient 2. “dirt” – Selects a dirt-looking gradient 3. "difference" - Selects a red-to-green gradient

Revision #1
Created 17 March 2025 19:10:59 by admin
Updated 17 March 2025 19:11:59 by admin