Creating Views

Views are special entities which can be visualized in the 3D viewer. A view will typically involve one or more spatial entities, which will be used in the visualization. The view also captures a number of parameters that will determine how the objects should be visualized like attribute filters and color legends.

View entities have the "type" property set to "FILE" and the "file_type" property set to "VIEW". To create Views, follow the same steps described in the Creating an Entity section. The Entity-specific Properties section outlines the properties expected for a View entity.

Using Views as containers

A single view may contain multiple children views. A children view could also contain other children views. The 3D viewer UI may display views with children as folders, and terminal views as objects.

A view that contains children, should have its "view_type" property set to "container", and should include a property named "entity_container" which lists the view children. Each children view is a different entity. This list contains the IDs of the children view entities separated by spaces.

If the view is not a container, its "view_type" must be set to the ID of a View Program entity. Please review the View Programs section for more information on these programs.

The following diagram illustrates the relationship between these different entity types and their properties:

Setting view object inputs

The use of user-supplied programs allows the organization to add new types of visual objects dynamically, after the spatial platform is deployed. This also implies the inputs and behavior of these objects may not be known in advance to the application developers.

In order to construct a view entity of a specific type, the application must know which inputs the object type requires. This can be achieved in two ways:

1. The application knows the inputs because it also knows the program (for instance, the application has first submitted the program and then it submits the view entity)
2. The application discovers the inputs for the program using the REST API. See Getting Program Inputs for this case.

To set a particular input, first lookup its type and review how values should be encoded. Then, proceed to set the input value as a property in the view entity. This property will have a special name, which will be in the form:

input_value_#id#

Where #id# should be replaced by the identifier of the input.

Predefined view object types

The platform includes a predefined set of view object types:

Type	Description
com.voxelfarm.program.view.terrain	A terrain surface. Has these inputs: 1. A terrain dataset entity. Use the "input_value_e" property to set this input. 2. A raster dataset entity. Use the "input_value_raster" property to set this input. 3. A boolean to toggle visualization of orthoimagery. Use the "input_value_colors" property to set this input. 4. A boolean to toggle pre-computed surface normal maps. Use the "input_value_normals" property to set this input. 5. A color legend. See the properties used for this type of input in the Entity-specific Properties section.

com.voxelfarm.program.view.mesh	An indexed mesh. Has these inputs: 1. The indexed mesh dataset that will be visualized. Use the "input_value_mesh" property to set this input. 2. A raster dataset entity. Use the "input_value_raster" property to set this input. 3. A color legend. See the properties used for this type of input in the Entity-specific Properties section.
com.voxelfarm.program.view.blockmodel	A voxelized block model. Has these inputs: 1. The voxelized blockmodel mesh dataset that will be visualized. Use the "input_value_bm" property to set this input. 2. A raster dataset entity. Use the "input_value_raster" property to set this input. 3. An attribute query that will be used to filter the block model. See the properties used for this type of input in the Entity-specific Properties section. 4. A color legend. See the properties used for this type of input in the Entity-specific Properties section.
com.voxelfarm.program.view.pointcloud	An indexed point cloud. Has these inputs: 1. The indexed point cloud dataset that will be visualized. Use the "input_value_pc" property to set this input. 2. A color legend. See the properties used for this type of input in the Entity-specific Properties section.
com.voxelfarm.program.view.drillholes	An indexed drill hole model. Has these inputs: 1. The indexed drillhole dataset that will be visualized. Use the "input_value_dh" property to set this input. 2. A raster dataset entity. Use the "input_value_raster" property to set this input.

Example: Using predefined types

This example will create two objects: a top level view that contains another view object. Before anything, we will use the call described in the Requesting a new ID section to obtain two new IDs for the objects.

First, we create a root view container. This will appear in the UI as a top level view.

```
Invoke-WebRequest -Uri
"http://localhost:58697/entity.ashx?id=1A5D65CBE25942B7A4594DD57C55F904&project=D9927BF62BFA4CCD930CBC7
16601D09C" `
-Method "POST" `
-Headers @{
"User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/85.0.4183.102 Safari/537.36"
  "Accept"="*/*"
  "Origin"="http://localhost:58697"
  "Sec-Fetch-Site"="same-origin"
  "Sec-Fetch-Mode"="cors"
  "Sec-Fetch-Dest"="empty"
  "Referer"="http://localhost:58697/cloud/project.html?id=D9927BF62BFA4CCD930CBC716601D09C"
  "Accept-Encoding"="gzip, deflate, br"
  "Accept-Language"="en-US,en;q=0.9,es;q=0.8"
} `
-ContentType "application/json; charset=UTF-8" `
-Body "{`"ID`":`"1A5D65CBE25942B7A4594DD57C55F904`", `"project`":`"D9927BF62BFA4CCD930CBC716601D09C`",
`"name`":`"New View`", `"type`":`"FILE`", `"file_type`":`"VIEW`", `"view_type`":`"container`", `"state`":`"COMPLETE`",
`"file_date`":`"1601166681260`", `"entity_container`":`"A20307F8C120444E9FBFAB9298B2F8D7`",
`"file_folder`":`"0`"}"
```

Then we create at least one visual object inside the view. In this example we will add a terrain surface, using the "com.voxelfarm.program.terrain" type:

```
Invoke-WebRequest -Uri
"http://localhost:58697/entity.ashx?id=A20307F8C120444E9FBFAB9298B2F8D7&project=D9927BF62BFA4CCD930CBC7
16601D09C" `
-Method "POST" `
-Headers @{
"User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/85.0.4183.102 Safari/537.36"
  "Accept"="*/*"
  "Origin"="http://localhost:58697"
  "Sec-Fetch-Site"="same-origin"
  "Sec-Fetch-Mode"="cors"
  "Sec-Fetch-Dest"="empty"
  "Referer"="http://localhost:58697/cloud/project.html?id=D9927BF62BFA4CCD930CBC716601D09C"
  "Accept-Encoding"="gzip, deflate, br"
  "Accept-Language"="en-US,en;q=0.9,es;q=0.8"
} `
-ContentType "application/json; charset=UTF-8" `
-Body "{`"ID`":`"A20307F8C120444E9FBFAB9298B2F8D7`", `"name`":`"Sept 2009 `",
`"view_type`":`"com.voxelfarm.program.view.terrain`", `"input_label_e`":`"Terrain`", `"input_filter_e`":`"8`",
`"input_type_e`":`"3`", `"input_value_e`":`"67448DFF1C2F4C3AA7F9D858CE30A455`", `"input_label_colors`":`"Use
Ortho-imagery`", `"input_filter_colors`":`"0`", `"input_type_colors`":`"6`", `"input_value_colors`":`"0`",
`"input_label_normals`":`"Use high resolution detail`", `"input_filter_normals`":`"0`", `"input_type_normals`":`"6`",
`"input_value_normals`":`"0`", `"input_type_colorlegend`":`"7`",
`"project`":`"D9927BF62BFA4CCD930CBC716601D09C`", `"type`":`"FILE`", `"file_type`":`"VIEW`",
`"state`":`"COMPLETE`", `"file_date`":`"1601166681446`", `"file_folder`":`"0`", `"virtual`":`"1`"}"
```

The new view object points to a terrain dataset with ID "67448DFF1C2F4C3AA7F9D858CE30A455".

Also note that the child view has the "virtual" property set to "1". This signals the application this entity is a building block for another entity and should not be listed as a top element in the UI.

Revision #2
Created 17 March 2025 16:11:07 by admin
Updated 17 March 2025 21:14:43 by admin