Niagara 3D Widget Guide

Technical Document

baudrate.io

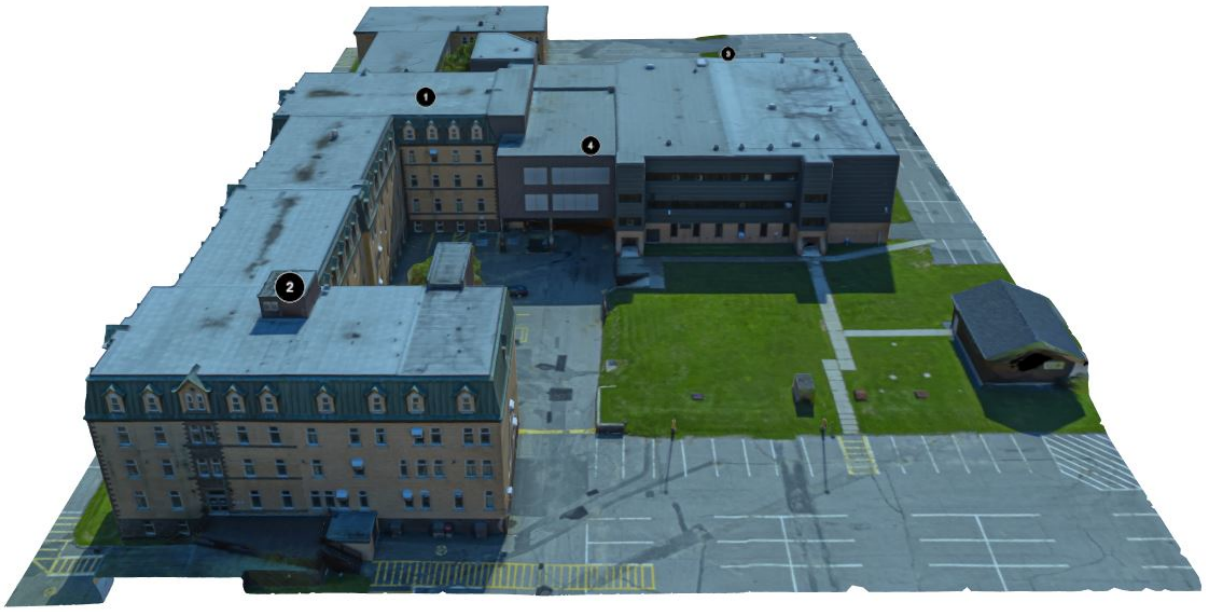
niagara4

February 09, 2024

Contents

1 Introduction

The Baudrate 3D Widget allows you to visualize and control building management systems in a 3D environment. It renders an actual 3D model, allowing you to interact with the building's systems and devices in a virtual space. This widget is designed to provide an intuitive and interactive interface for managing various facets of BMS operations. The user can move, rotate and zoom the 3D model to view the building from different angles and perspectives. They can also interact with the building's systems and devices, such as HVAC, lighting, and security, by clicking on them within the 3D model.



Compared to traditional 2D interfaces, the 3D view offers the following advantages:

- **Enhanced Visualization**: 3D models offer a more detailed and realistic view than 2D diagrams, significantly improving the understanding of complex building layouts and equipment designs and enabling quicker decision-making
- **Improved Troubleshooting and Maintenance**: The ability to easily locate and identify components in a 3D space streamlines troubleshooting processes and maintenance operations, especially in intricate systems.
- **Efficient Training and Onboarding**: 3D models provide an intuitive and comprehensive way to train new staff or external contractors, helping them understand the system faster and more effectively.
- **Better Planning and Decision Making**: With a clear view of existing setups, 3D models facilitate accurate planning and decision-making for modifications, expansions, or new installations.

- **Enhanced Collaboration**: Utilizing 3D models can significantly improve collaboration among team members, stakeholders, or clients by providing a common, detailed visual reference, leading to better communication and shared understanding of the project.
- **Intuitive Control**: Control of various building systems becomes more intuitive when operators can interact with 3D representations, simplifying adjustments to HVAC, lighting, and security settings.

The widget can be used in various applications, including:

- Building Management Systems
- Facility Management Systems
- Smart Cities
- Industrial Automation
- Energy Management Systems
- Security and Surveillance Systems
- Smart Home Automation

1.1 Key Features

- The widget uses WebGL to render 3D models directly in the browser or in Niagara Workbench, providing a realistic and interactive view of the building.
- Users can move, rotate, and zoom the 3D model to view the building from different angles and perspectives.
- Users can click on hotspots within the 3D model to move to a different location, view additional information, or control the building's systems and devices.
- Multiple aspects of the widget can be customized, including the 3D model, hotspots, and the user interface.
- The widget is designed to be responsive and mobile-friendly, allowing users to access the 3D model from any device.
- The widget is designed to be lightweight and efficient, ensuring smooth performance even on low-end devices.
- Supports Niagara 4.8 and later, and is compatible with all modern web browsers.

1.2 Requirements

- Tridium Niagara 4.8 or later powered device such as JACE, Supervisor or their OEM versions
- 3d-ux.jar module and the license file
- 3D model of the building or the equipment in glTF or GLB format

The widget is not resource-intensive and can be deployed on JACE as all rendering is done in client's browser. It is recommended to have a high-speed internet connection to load the 3D model quickly.

1.3 Installation

We recommend to start exploring 3D Widget using the test station provided in the package. The test station includes sample 3D models and a few pre-configured hotspots to help you get started quickly. To install the widget and the test station, follow these steps:

1. Copy **3d-ux.jar** and **3d-doc.jar** into **/modules** folder of Niagara Workbench and **threeDemo** folder into /stations.
2. Restart Workbench and start the **threeDemo** station.
3. Open the **Services / LicenseService**. Press import icon » near License property and import the supplied *.license* file. Press the **Save** button and make sure Status Message shows *License ok*.
4. Open Campus folder – you shall see a 3D model. Navigate in the 3D view using mouse: left-click and drag to rotate the camera, right-click and drag to pan, scroll to zoom in and out. Click on the hotspot to see the popup.
5. Try other folders as well.

If you are adding 3D Widget to an existing station, follow these steps:

1. Copy **3d-ux.jar** and **3d-doc.jar** into **/modules** folder of Niagara Workbench.
2. Restart Workbench and start your station.
3. Open 3d palette and add **LicenseService** to **Services**.
4. Open the **Services / LicenseService**. Press import icon » near License property and import the supplied *.license* file. Press the **Save** button and make sure Status Message shows *License ok*.
5. Copy the 3D model file in glTF or GLB format into the station's **/shared** folder. The model in glTF format should be accompanied by a BIN file and texture image files. The model in GLB format is a single file.
6. Open a px file and drag & drop **Building3D** widget from the **3d** palette.
7. Double-click the widget and set **scene.model** attribute in **config** property to the path of the glTF or GLB file in the station's folder.
8. After a few seconds the model shall appear in the 3D view.
9. Follow the next section to customize the view and add hotspots.

The widget utilizes WebGL API to render 3D models directly in the browser. All modern web browsers support WebGL, but in Workbench browser WebGL is disabled by default. The widget will still work, but it might be slower when the camera moves around. To improve 3D rendering performance in Niagara Workbench, open **defaults/system.properties** file in the Niagara folder, uncomment this line and restart Workbench:

```
niagara.jxbrowser.lightweight.accelerated=false
```

1.4 Next Steps

To understand the main 3D Widget concepts and components read the Component Guide section.

The widget could be configured with lots of attributes. To understand how to edit them, see Attribute Editor section. For a full list of all attributes and their descriptions see Attribute Reference section.

An example of how to build a simple 3D view with hotspots is provided in Step-by-step Guide section.

By the end of this guide, you will have a thorough understanding of how to use the 3D Widget to create a 3D view of your building and add annotations to it. You will also learn how to customize the widget to suit your specific requirements.

2 Component Guide

This section describes main concepts one should know about in order to create new 3D views: 3D Model, Scene, Lights, Camera, Controls, GUI and Annotations.

2.1 3D Model

The 3D model is the main component of the 3D Widget. It is a representation of the building or equipment in a 3D space. The model is rendered using WebGL directly in the browser, providing a realistic and interactive view of the building.

There are various ways to obtain a 3D model:

- **Create a 3D model**: You can create a 3D model using 3D modeling software such as Blender, 3ds Max, or SketchUp. This process involves creating a 3D representation of the building or equipment from scratch. There are many freelance 3D modelers who can create a 3D model for you based on your specifications.
- **Use BIM model**: If you have a building information model (BIM) of the building developed in software such as Revit, ArchiCAD, and Navisworks, you can use it as a starting point. Normally you still need to use software such as 3ds Max to make your model more attractive, as BIM models are usually optimized for engineering and construction purposes, not for visualization.
- **Use a photogrammetry model**: If you have a series of photographs of a building taken from a drone, you can use photogrammetry software to create a 3D model from the photographs. This process involves taking multiple photographs of the building or equipment from different angles and using software to stitch the photographs together to create a 3D model.
- **Use a laser scan model**: One can make laser scans of a building interior or equipment using a regular iPhone with built-in LiDAR sensor and use software to convert the laser scan data into a 3D model. This process involves importing the laser scan data into the software and converting it to a 3D representation.
- **Download a 3D model**: For testing purposes you can look on many online repositories where you can find 3D models of buildings, equipment, and other objects. Websites such as Sketchfab, TurboSquid, and CGTrader offer a wide range of 3D models that can be used in the 3D Widget. The models of existing building can sometimes be found on Google Earth as well.

The 3D model should be saved to glTF (GL Transmission Format) or GLB (binary GL Transmission Format) file. These formats are widely supported and can be easily exported from multiple software packages. The model in glTF format should be accompanied by a .bin file and texture image files. The model in GLB format is a single file. Although the GLB format is more compact, it will not necessarily load faster than the glTF format, as the glTF parts can be loaded in parallel.

It is important to note that the 3D model should be optimized for real-time rendering in a web browser. This means that the model should be lightweight and efficient – known as *low poly* – ensuring quick loading and smooth performance even on low-end devices.

After the model is created, it can be tested online using the glTF Viewer or Windows 3D Viewer.

2.2 Scene

The scene is the container for all 3D objects loaded from glTF file. It is the main component that holds the 3D model and provides the environment for rendering the 3D view.

2.3 Camera

The camera is the viewpoint from which the 3D model is rendered. It defines the position, orientation, and field of view of the view. The camera can be moved, rotated, and zoomed to view the model from different angles and perspectives. The camera can be controlled using the mouse or touchpad by Controls.

2.4 Lights

Lights are used to illuminate the 3D model and create realistic lighting effects. Each light has color and intensity values that can be adjusted to create different moods and atmospheres. There are several types of lights that can be used in the 3D Widget:

- **Ambient Light**: This light illuminates the entire scene with a constant color equally. It is used to simulate the light that is reflected off surfaces and fills in shadows.
- **Hemisphere Light**: A light source positioned directly above the scene, with color fading from the sky color to the ground color. It is used to simulate the light that is reflected off the sky.
- **Directional Light**: This light is used to simulate sunlight. It is a powerful light that shines in a specific direction, like the sun.
- **Point Light**: This light is a point in space that emits light in all directions. It is used to simulate light bulbs and other small light sources.
- **Spot Light**: This light is a point in space that emits light in a cone shape. It is used to simulate flashlights and other focused light sources.
- **Rect Area Light**: This light is a rectangle in space that emits light in a specific direction. It is used to simulate light panels and other large light sources.

3D Widget adds one Hemisphere Light to the scene by default. This light can be adjusted or additional lights can be added to create the desired lighting effects.

2.5 Controls

Controls are used to interact with the 3D model. They allow users to move, rotate, and zoom the 3D model to view the building from different angles and perspectives. The controls can be customized to suit the specific requirements of the application. The 3D Widget uses Orbit Controls by default, which allows users to rotate the camera around the 3D model (left-click and drag), pan the camera (right-click and drag) and zoom in and out (scroll). The controls can be disabled or customized to change the rotation speed, damping factor, and other parameters.

2.6 GUI

The GUI (Graphical User Interface) is the small user interface windows that is displayed on top of the 3D model. It can be used to display additional information, determine camera and hotspot positions, and provide a way for users to interact with the 3D model. The GUI can be customized to suit the specific requirements of the application.

2.7 Annotations

Annotations are used to add additional information to the 3D model using Niagara real-time data. Annotations consist of hotspots and popups. Hotspots are small markers that are placed on the 3D model to indicate specific points of interest. When a user clicks on a hotspot, the camera moves to a certain position and an optional popup is displayed. Popups are small windows that can be displayed when a user clicks on a hotspot and contain additional information about the point of interest, including real-time or historical information and hyperlinks. The popups can be customized to suit the specific requirements of the application.

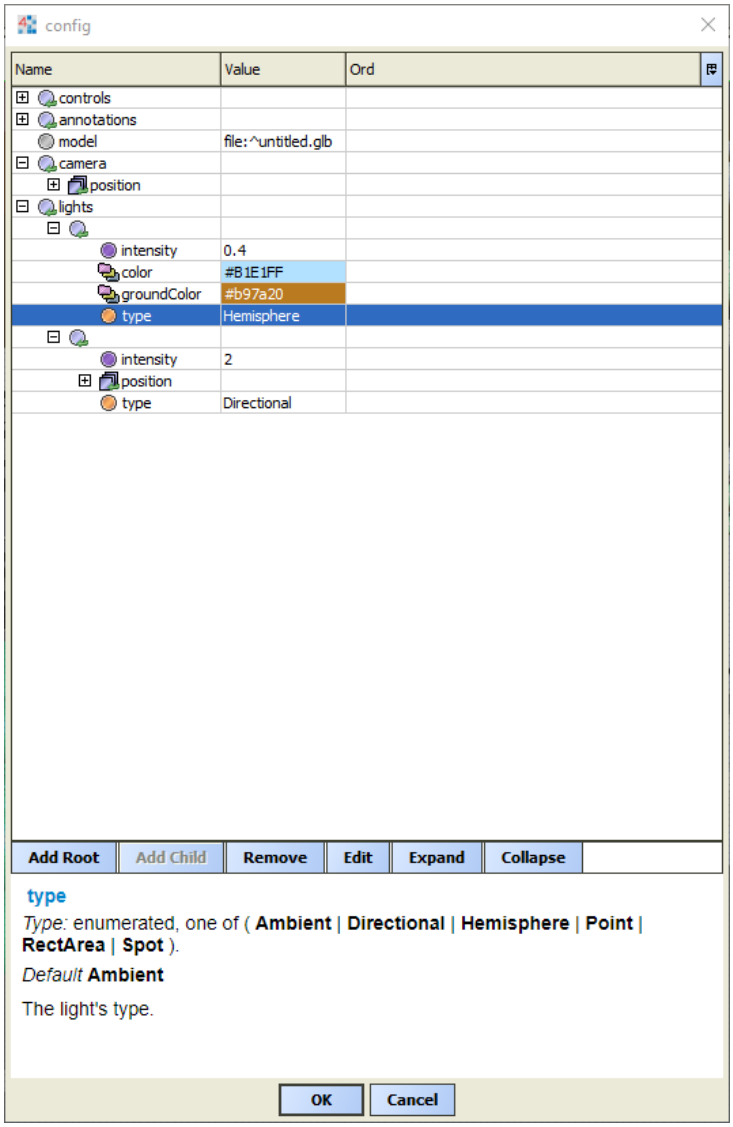
2.8 Helpers

Helpers are used to add additional visual elements to the 3D model for debugging and engineering purposes. Helpers can be used to display the bounding box, axes, and grid of the 3D model.

3 Attribute Editor

3D Widget is configured in the Niagara Workbench PX Editor. The widget is dragged and dropped from **3d** palette to the px file first and then configured by setting its property **config**. This property is a collection of attributes describing the widget stored in JSON format. Attributes are organized in a tree structure: one or multiple root attributes, which might have sub-attributes or children, which in turn might also have sub-attributes etc.

When clicking on **config** property, a new dialog box appears. Its top area contains added attributes, middle area contains buttons and bottom area has dynamic text, which describes selected attributes. This text helps to understand attribute purpose and possible values.



Buttons:

- Add Root [Ctrl+Insert] – to add trace root attribute – the ones on the top of a tree
- Add Child [Insert] – to add sub-attributes to selected attribute
- Remove [Delete] - to delete attribute
- Edit [Ctrl+E] - to modify attribute value
- Expand [Ctrl+Down] - expands all root properties so that sub-attributes are visible
- Collapse [Ctrl+Up] - collapses all root properties and their sub-attributes

Initially the top area is empty. To add a new attribute, click **Add Root** button. This opens a selector with all root attributes available for this object. The bottom area of the dialog box displays information of a selected attribute: its type, default value and description. In case of complex attributes – objects and arrays – it is necessary to select the attribute and **Add Child** first. That will open the attribute selector containing all attribute children. After the attribute is added, it is possible to **Edit** its value. Unused attributes can be deleted with **Remove** button.

The attributes have various types: string, number, boolean, object, array, color, enum, and others. The attribute type is displayed in the bottom area of the dialog box. Objects and arrays have children, which can be added with **Add Child** button. The children can be of certain type, including objects and arrays.

For example, the **camera** attribute is an object, which has children: **position**, **fov**, **near**, **far**. The **position** is an array, which has three number children: x, y, and z coordinates.

The **lights** attribute is an array of light objects. Each light has children: **type**, **color**, **intensity**, **position**, **target** etc.

An engineer shall not specify all attributes manually. The widget has a set of default values, which are usually sufficient to start with. The attributes are added and modified only when necessary.

4 Attribute Reference

This section describes all attributes available for the 3D Widget.

4.1 Scene

The **scene** object contains attributes that control the scene, which is the container for all 3D objects:

- **model** - the ord to the 3D model file in .glb or .gltf format.
- **background** - the background color of the scene.

4.2 Camera

The **camera** object contains attributes that control the viewpoint from which the 3D model is rendered:

- **position** - the position of the camera as an array of three numbers: x, y, and z coordinates. If not specified, the widget calculates the position automatically based on the bounding box of the 3D model. If automatic position is not suitable, move camera where desired in user mode and use GUI window to determine the position and the target.
- **target** - the point the camera is looking at as an array of three numbers: x, y, and z coordinates. If not specified, the widget calculates the target automatically as the center of the 3D model.
- **fov** - the field of view of the camera in degrees. The default value is *50*.
- **near** - the distance to the near clipping plane – nothing closer to the camera than this will be rendered. The default value is *0.1*.
- **far** - the distance to the far clipping plane – nothing further from the camera than this will be rendered. The default value is *1000*.

4.3 Lights

The **lights** array contains light objects that are used to illuminate the 3D model and create realistic lighting effects. If not specified, the widget adds one Hemisphere light with light blue sky color and brownish ground color to the scene by default. Each light has the following attributes:

- **type** - the type of the light. Possible values are: Ambient, Hemisphere, Directional, Point, Spot, RectArea. The default value is *Directional*. See Lights for more details.
- **color** - the color of the light. The default value is *white*.
- **groundColor** - the color of the ground for Hemisphere light. The default value is brownish-orange *#B97A20*.

- **intensity** - the intensity of the light. The default value is *1*.
- **position** - the position of the light as an array of three numbers: x, y, and z coordinates. This attribute is used for Hemisphere, Directional, Point, Spot, and RectArea lights. The default value is *[0, 1, 0]*. Use GUI window to determine the position and the target of the light.
- **target** - the point the light is looking at as an array of three numbers: x, y, and z coordinates. This attribute is used for Directional, RectArea and Spot lights. The default value is *[0, 0, 0]*.
- **power** - the luminous power of the light measured in lumens (lm). Changing the power will also change the light's intensity. This attribute is used for Point and RectArea lights. The default value is *1000*.
- **distance** - the distance from the light where the intensity of the light is zero. This attribute is used for Point and Spot lights. The default value is *0*.
- **decay** - the amount the light dims along the distance of the light. This attribute is used for Point and Spot lights. The default value is *2*.
- **angle** - the angle of the light in degrees. This attribute is used for Spot lights. The default value is *60*.
- **penumbra** - the angle of the penumbra of the light in degrees. This attribute is used for Spot lights. The default value is *0*.
- **width** - the width of the light. This attribute is used for RectArea lights. The default value is *10*.
- **height** - the height of the light. This attribute is used for RectArea lights. The default value is *10*.

4.4 Controls

The **controls** object contains attributes that control the interaction with the 3D model:

- **enabled** - enable/disable controls. The default value is *true*.
- **enableZoom** - enable/disable zooming. The default value is *true*.
- **enableRotate** - enable/disable rotating. The default value is *true*.
- **enablePan** - enable/disable panning. The default value is *true*.
- **enableDamping** - enable/disable damping (inertia), which can be used to give a sense of weight to the controls. The default value is *false*.
- **dampingFactor** - the damping factor for the controls. The default value is *0.05*.
- **enableKeys** - enable/disable keyboard controls. The default value is *true*.
- **keyPanSpeed** - the speed of panning when using the keyboard. The default value is *7*.
- **autoRotate** - enable/disable auto-rotation. The default value is *false*.
- **autoRotateSpeed** - the speed of auto-rotation. The default value is *2*.
- **minPolarAngle** - the minimum polar angle to orbit vertically. The default value is *0*.
- **maxPolarAngle** - the maximum polar angle to orbit vertically. The default value is *90*.
- **minAzimuthAngle** - the minimum azimuth angle to orbit horizontally. The default value is

-Infinity.

- **maxAzimuthAngle** - the maximum azimuth angle to orbit horizontally. The default value is *Infinity*.
- **minDistance** - the minimum distance of the camera from the target. The default value is *0*.
- **maxDistance** - the maximum distance of the camera from the target. The default value is *Infinity*.

4.5 GUI

The **gui** object contains attributes that control the graphical user interface windows displayed on top of the 3D model and used to display additional information, determine camera and hotspot positions, and provide a way for users to interact with the 3D model:

- **hidden** - enable/disable GUI. The default value is *false*.
- **expanded** - show GUI expanded. The default value is *true*.
- **title** - the title of the GUI window. The default value is *Controls*.
- **style** - the object containing CSS styles for the GUI window.
  - **position** - the position of the GUI window. The default value is *fixed*.
  - **top** - the top position of the GUI window. The default value is *30px*.
  - **left** - the left position of the GUI window. The default value is *30px*.
  - **bottom** - the bottom position of the GUI window. The default value is *auto*.
  - **right** - the right position of the GUI window. The default value is *auto*.
- **values** - the array of the GUI pane as a list of object.property strings. The default value is *[camera.position, controls.target, light.color, scene.background, controls.autoRotate, info.object, info.point]*.

The default GUI window allows to see and modify these attributes: * the position of the camera and its target point – useful to determine the position and the target of the camera and the lights; * the colors of the light and the background – useful to experiment with the colors; * the auto-rotation mode of the camera – allows to enable camera rotation; * the object name and mouse click coordinates – useful to find where to make hotspots.

4.6 Annotations

The **annotations** array contains attributes that control the hotspots and popups. Each annotation is an object with the following attributes:

- **position** - the position of the hotspot as an array of three numbers: x, y, and z coordinates.
- **text** - the text of the popup. The default value is *1*.

- **camera** - the position of the camera to move to when the hotspot is clicked as an array of three numbers: x, y, and z coordinates.
- **size** - the size of the hotspot. The default value is *5*.
- **duration** - the duration of the camera movement in milliseconds. The default value is *2000*.
- **font** - the font of the popup text as a CSS font property. The default value is *40px Arial*.
- **lineColor** - the color of the popup border. The default value is *white*.
- **fillColor** - the color of the popup background. The default value is *black*.
- **fontColor** - the color of the popup text. The default value is *white*.
- **lineWidth** - the width of the popup border in pixels. The default value is *2*.
- **easing** - the easing function of the camera movement animation. The default value is *Quadratic.InOut*.
- **popupBorder** - the border of the popup as a CSS border property. The default value is *1px solid #ccc*.
- **popupOffset** - the offset of the popup from the hotspot center as two numbers: x and y coordinates. The default value is *[40, 40]*.
- **popupSize** - the size of the popup as two numbers: width and height. The default value is *[400, 300]*.
- **popupOrd** - the ord of a px file or a view, which will be shown in a popup window. If not set, no popup will be shown, so the hotspot will be used only for camera movement.

4.7 Helpers

The **helpers** object contains attributes that control the helper objects, which are used to visualize the position and orientation of the 3D model:

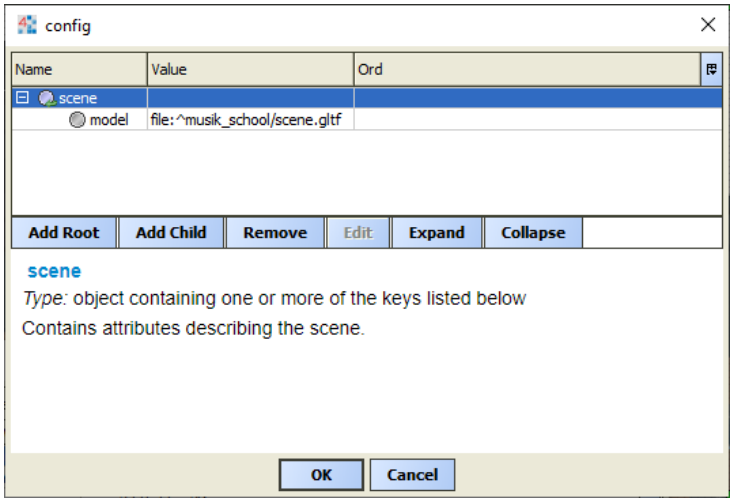
- **showAxes** - shows axes. The default value is *false*.
- **showBoundingBox** - shows bounding box. The default value is *false*.
- **showGrid** - shows grid. The default value is *false*.

5 Step-by-step Guide

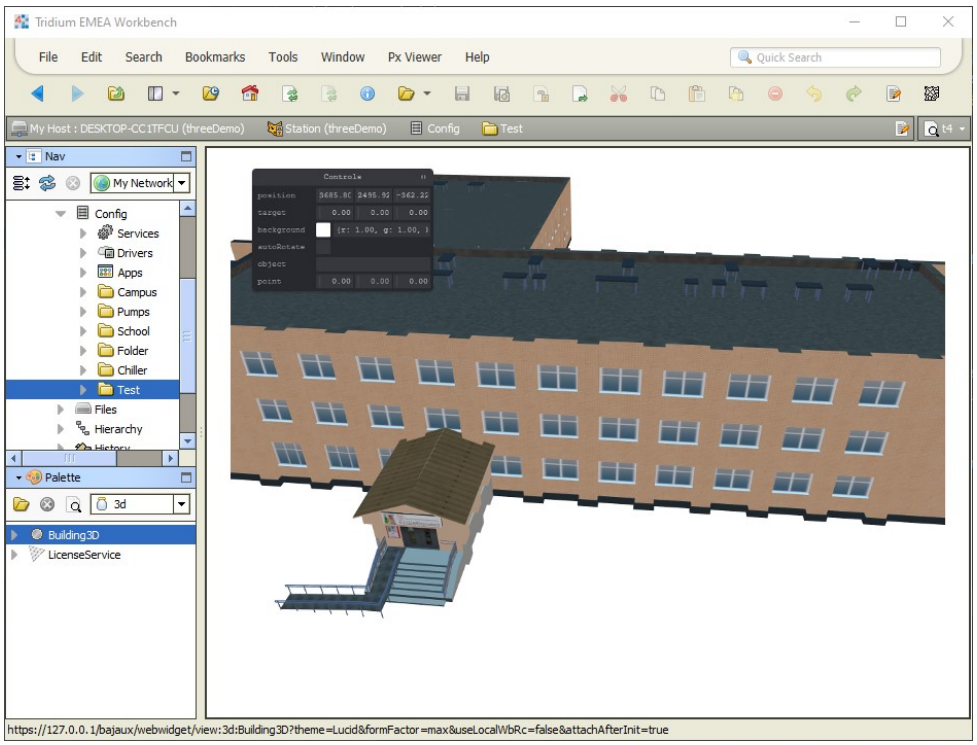
In this guide we will build a 3D view with a simple 3D model and a hotspot with a popup displaying real-time and historical data.

Niagara station should be licensed for 3D Widget as described in Installation section. For the purpose of this guide we will use a free building 3D model available on Sketchfab. Download glTF file and unzip its into /shared/musik_school folder of your Niagara station.

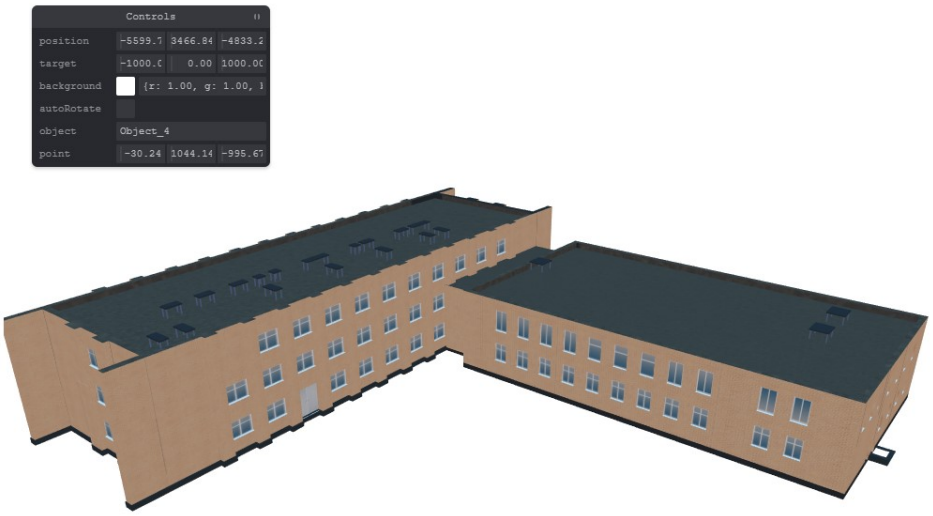
1. Drag & drop **Building3D** widget from the **3d** palette to the px file.
2. Open widget properties and click on config property to open attribute editor.
3. The object **scene** with attribute **model** is added by default – click Expand button to see it. Change **model** value to the ord of glTF file in the station's folder.

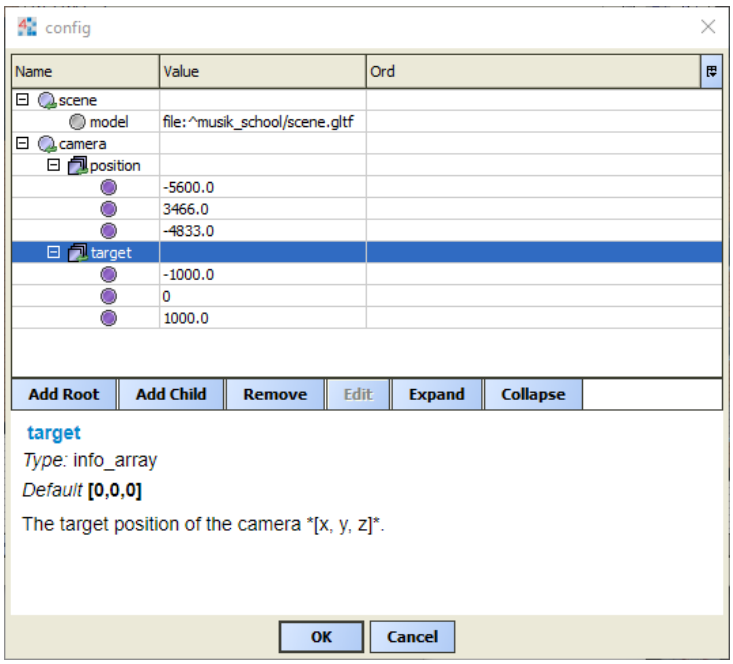


4. You shall be able to see the 3D model rendered on the screen! Switch to view mode and try to navigate around using mouse or touchpad. Left-click and drag to rotate the camera, right-click and drag to pan, scroll to zoom in and out.
5. There is a small black GUI window in the top left corner of the screen. We will hide it later, but for now it will provide us useful information about the model.

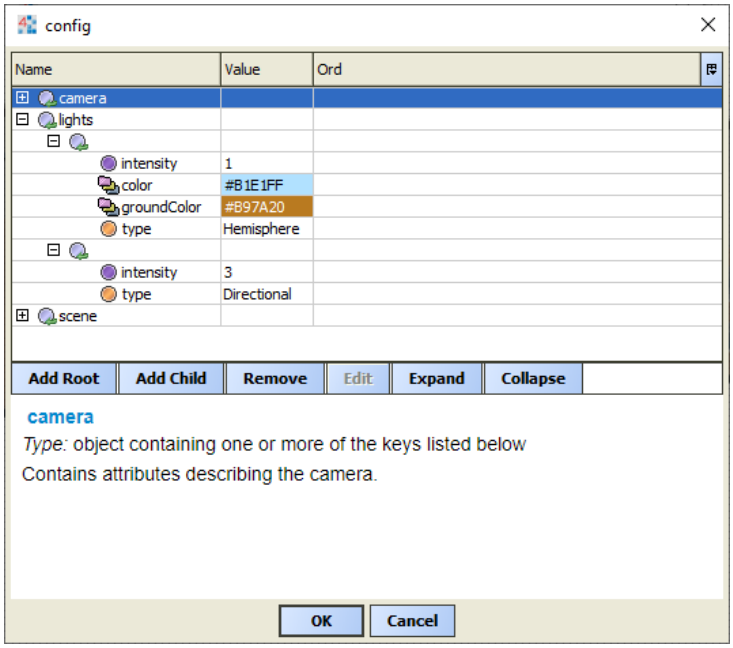


6. The widget automatically position the camera to fit most of the model into the view. We will override the position or the camera and its target (where it is looking at) to better display the model. In order to do that we need to move camera and target to the desired position using the mouse or GUI. Then we will copy the **position** and **target** coordinates from the GUI window to the widget attributes **camera.position** and **camera.target**.



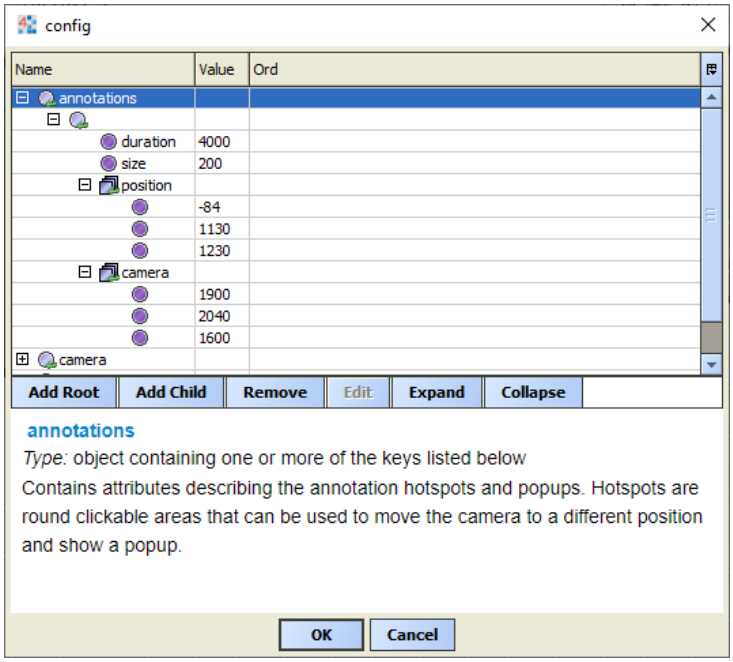


7. Lights can drastically change the look of the model. We will add a few lights to the scene. Click **Add Root** and select **lights** attribute. There will be one Hemisphere light added by default. Select **lights** and click **Add Child** to add one more light. Select it, click **Add Child** to add **type** and **intensity** attributes. Set **type** to **Directional** and **intensity** to **3.0**. Now the model should be well lit with one Hemisphere light and one Directional light coming from the top, as if it was a sun.

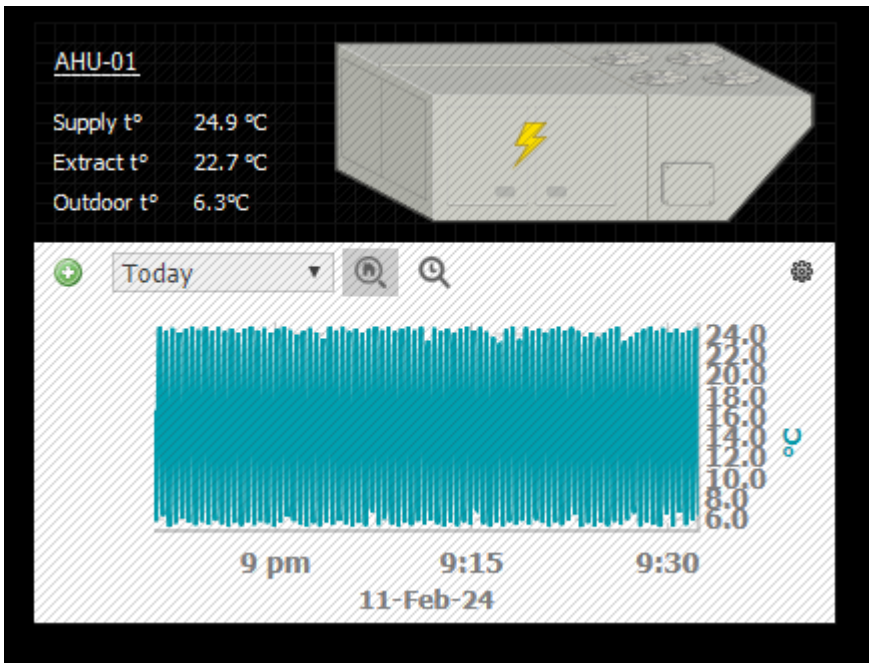


8. Now we will add an annotation to the model. To determine hotspot position switch to viewing mode and click where the hotspot should be on the model. The GUI will display the coordinates of the click in **point** line. Position camera where it shall move and copy **position** line coordinates on GUI.

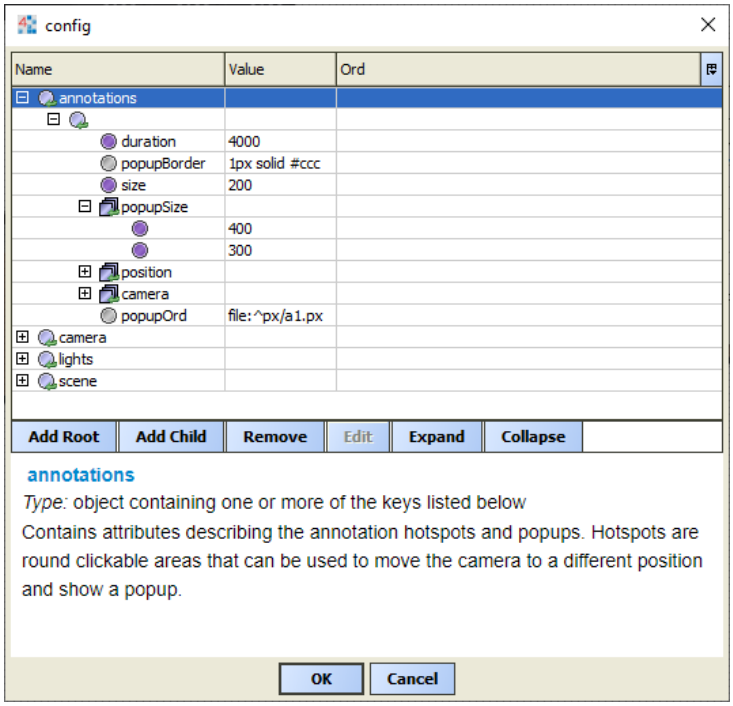
9. In edit mode click on **Add Root** to add **annotations**, add one child, then add child's attributes **position**, **camera**, **size**, and **duration**. Set hotspot **position** to GUI **point** coordinates, camera position to GUI **position** coordinates, size to **200** and duration to **4000**. That will create a round hotspot with text **1** and diameter of 200 – various models have different scales, so the hotspot size should be adjusted accordingly. When clicking on the hotspot, the camera will move to the required position in 4000 milliseconds and turn to the hotspot.

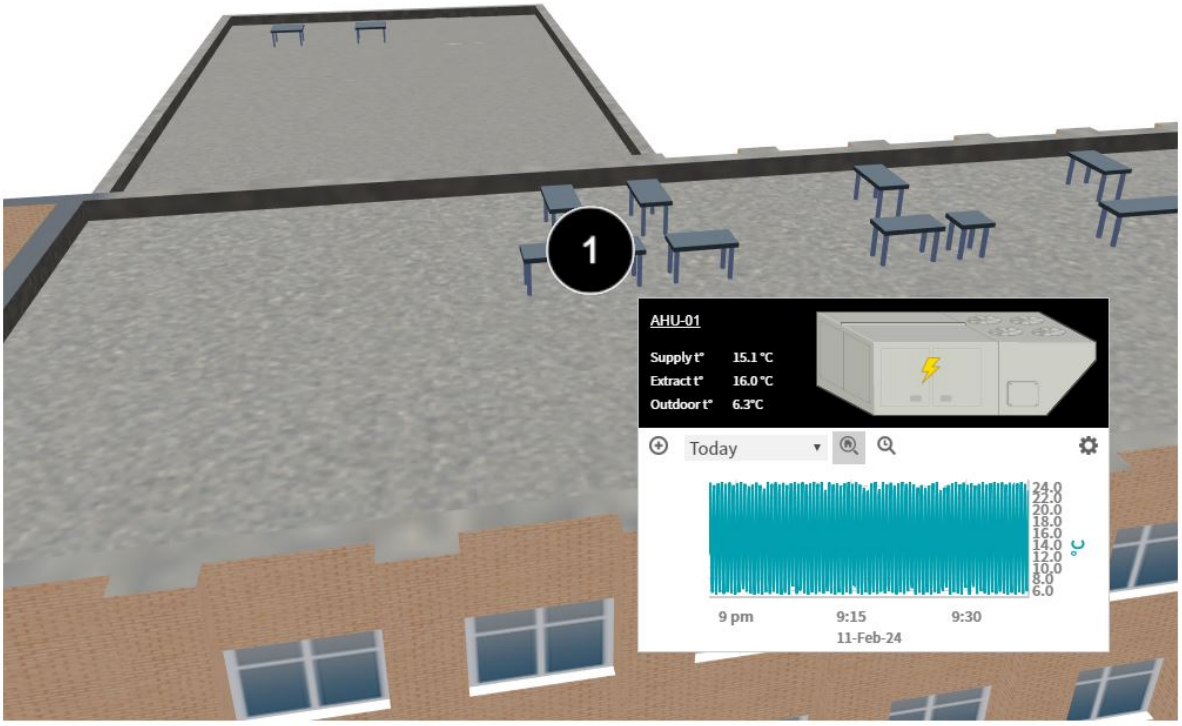


10. Now we will add a popup to the hotspot. First we make a regular px file sized, for example, 400 x 300 pixels. The px file can contain any Niagara components: labels, real-time values, hyperlinks, widgets etc.



11. Add attributes **popupOrd** and **popupSize** to the annotation object. Set **popupOrd** to the ord of the px file and **popupSize** to the size of the px file. Now when clicking on the hotspot, the popup will be displayed.





12. To remove the GUI window, set **gui.hidden** attribute to **true**.

Further improvements can be made by adding more lights of various types, colors and intensities, adding more annotations and popups. Scene background and light colors can be changed, camera field of view can be adjusted, controls behaviour and many other attributes can be modified to achieve the desired look and behavior of the 3D view.