C3DWV
Collaborative 3D Web Viewer

User Guide

Author(s): Jan Sutter DFKI

Release Date: 04-08-2015
Revision: 1.0

Table of Contents

Introduction

This document describes the use of the Collaborative 3D Web Viewer (C3DWV, Codename: C3DWV) as a tool to edit and annotate XML3D scenes. For installation please consult the Administration Guidelines.

1. Overview

C3DWV, codenamed COMPASS ("Collaborative Modular Prototyping And Simulation Server"), is a framework for developing web applications with which elements of 3D scenes can be arranged and enriched with metadata.
It is collaborative in the sense that multiple users are able to see and interact with the same scene at the same time. It is modular because it's easy to assemble customized applications with the building blocks of the framework, and that it is easy to extend the metadata model to accommodate a given usage scenario. Its main usage scenario is prototypical layouting (one might also say simulation) of a scene.

2. Architecture and Specifications

The Collaborative 3D Web Viewer is an HTML-5 client application build on top of the XML3D GE from the FIWARE project and a JavaEE back-end application, based on EJB 3.1 and CDI, and uses JPA as a storage backend. Its main data model is an Entity-Component Architecture (ECA), where its Entities are nodes in a Scene Tree, and metadata concerning these Entities are the Components. C3DWV provides a REST API to access the scene tree, based on JAX-RS.

The shared scene graph structure is based on the entity component attribute design principle. This design favors composition over inheritance and is a popular design among interactive applications, such as games. In contrast to many entity component systems the Collaborative 3D Web Viewer uses a hierarchical entity (SceneNode) system, which means every entity can have child entities. This allows for a structure similar to a typical scene graph. The domain model for C3DWV can be seen in Figure 1.

A SceneNode in C3DWV is everything that is part of the virtual 3D scene, e.g. light or 3D mesh. Typically lights, camera, and meshes are distinct objects in a system. The entity component system deliberately refrains from this distinction and handles every object as an entity. The differentiation is made on the components attached to the entity. A component is a set of attributes and values, e.g. a light or geometry component. Depending on the components an entity has assigned the systems treats it as a light, a mesh, or both.
Currently the system does not provide a way to define new components using plug-ins. This is planned for a future release. Currently four predefined components exist:

- Transform: used to define an objects position and orientation in 3D space.
- Render Geometry: used to provide a URL to the XML3D mesh to render
- Directional Light: makes the entity a directional light source.
- Preview Image: can be used to provide a thumbnail for the entity
- Annotation Component: can be used to annotate an entity

Because C3DWV uses the XML3D GE for client-side rendering only XML3D files are supported. To import 3D models in other formats into the scene the C3DWV has an interface to communicate with the AssetStorage SE from FI-CONTENT, which can convert COLLADA files into XML3D (see Section 4.2.3.2).

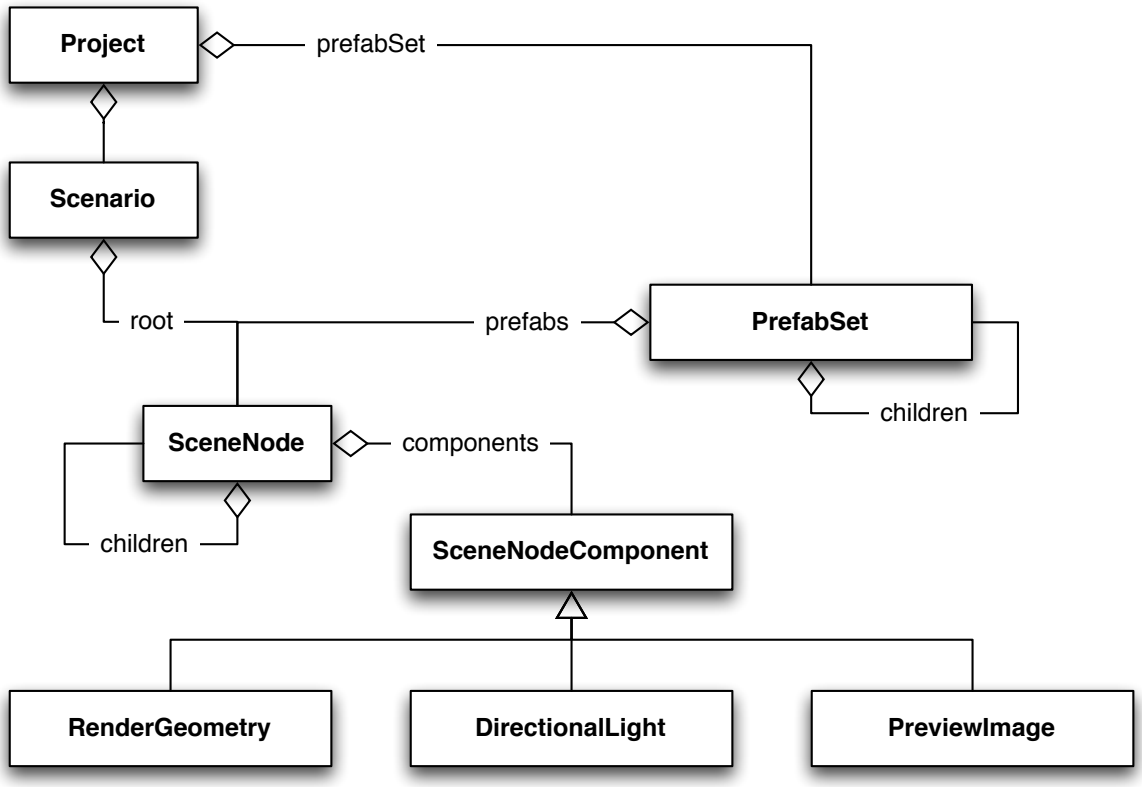


Figure 1: Domain Model, simplified view.

3. Getting Started

To get started with the C3DWV SE a prepared virtual machine can be used [link missing]. This Virtual Box VM is meant for testing and evaluation purposes only. Please do not use this VM inside your business critical infrastructure. To install the SE inside you infrastructure please consult the Administration and Installation guidelines.
The virtual machine starts the SE as well as an instance of the AssetStorage SE (ATLAS) during the boot process. The C3DWV can be accessed under the following URL: http://<vm-ip>:80808/c3dwv. The prepared instance of C3DWV comes with two example models provided by AIDIMA[1].

4. User Interface

4.1. *Projects* and Scenarios

C3DWV comes with a structure of *projects* and *scenarios*. The project is the topmost entity and holds a set of scenarios. A scenario is the entity holding all the information for a scene.

[1] http://www.aidima.eu

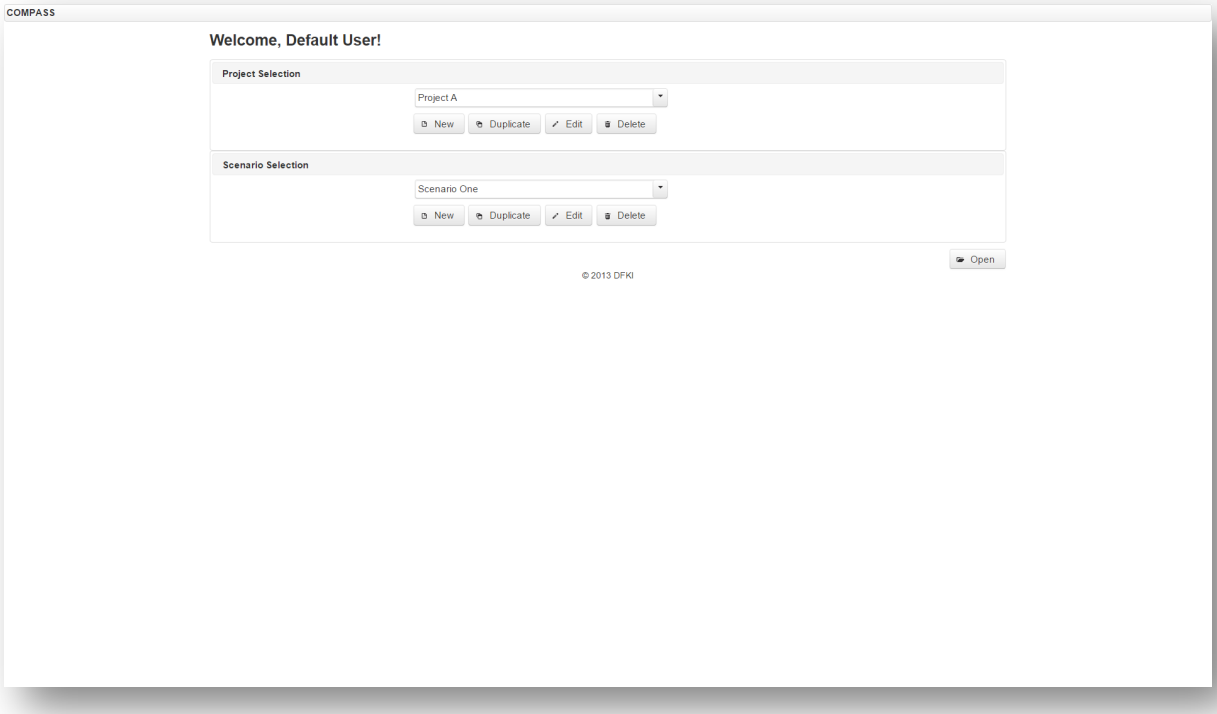


Figure 2: Project and scenario Project and Scenario management screen.

4.1.1. Projects

First let us look at the project. As mentioned beforehand, the project is the topmost entity of C3DWV' hierarchic entity structure. It manages a set of scenarios as its children and, moreover, it is the place the *PrefabSets* reside.
Just to explain briefly, a PrefabSet, as the name indicates, is a set of *Prefabs*. Prefabs, short for "Pre-Fabricated Elements", are used as templates for parts of a scene tree and can be used to store reusable parts of a scene. Thus all prefabs organized within a project are available in all scenarios of that project. We will explain this in more detail in Section 4.2.3.
For project management the UI provides four functions: Creation, Copying, Editing and Deletion of a Project (see Figure 2).

Creation

When creating a new project you need to specify a name, which may only contain letters, numbers, dashes, braces, colons and dots; No leading or trailing spaces. Moreover the name must be unique. There must not be a Project with the same name. After creating a new project it has neither Scenarios and nor PrefabSets.

Copying

When a project should be duplicated, a new name for the copy must be pro- vided. The new name follows the same naming rules, which apply when creating a new project. The copy of the project contains copies of the original's data, including PrefabSets, Scenarios, and the content of those. These copies are completely separate from and retain no link to the original data.

Editing

Since the name is the only basic attribute of a project, it is the only thing one can edit in the edit view. Editing the name will also enforce the aforementioned naming conventions. PrefabSets can be managed within the Editor View, see Section 4.2.3.

Deletion

Deleting a project will remove it, as well as its associated Scenarios. PrefabSets and containing Prefabs will only be deleted if they are no longer referenced by any other Project. However, there is currently no frontend for associating PrefabSets with different Projects.

4.1.2. Scenarios

As stated previously, the Scenario represents the entry point for a given scene. The scenario has a name, a preview image, and a reference to the scene tree, which will be shown in the 3D editor view after opening the scenario.
For managing scenario C3DWV comes with the same four operations also available for Projects: Creation, Copying, Editing, and Deletion.

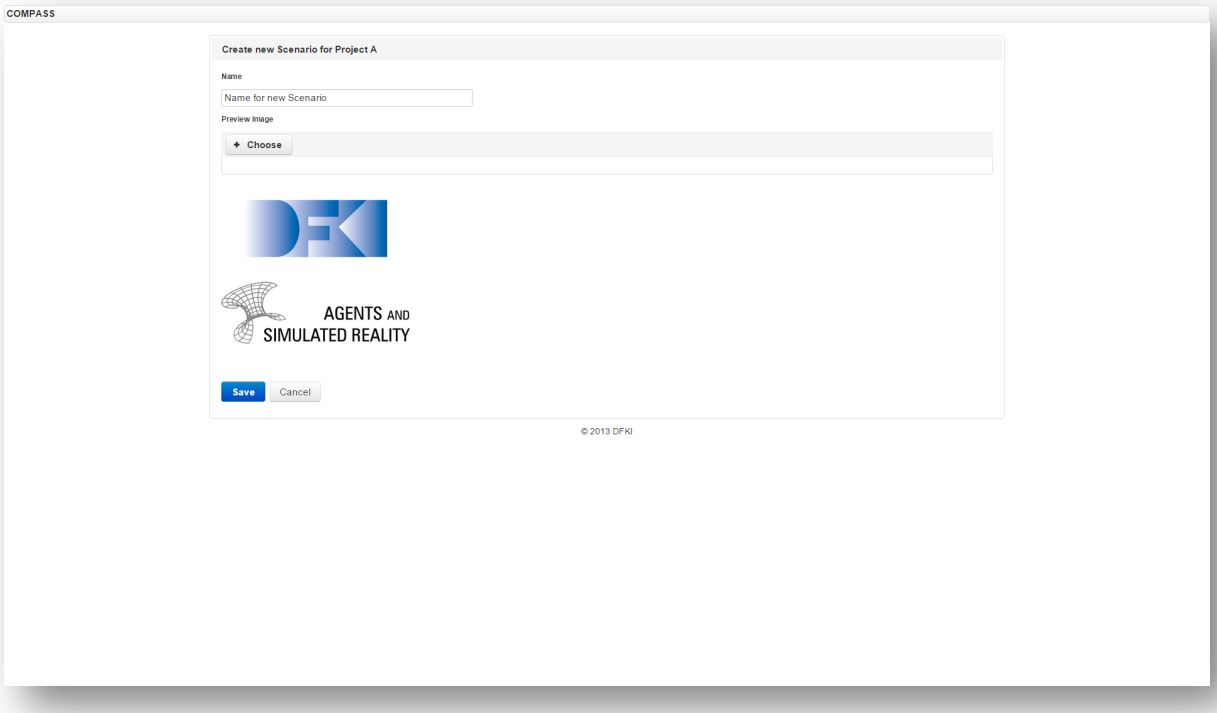


Figure 3: Creating a scenario, with an example preview image uploaded already.

Creation

When creating a new scenario you need to specify a name, which may only contain letters, numbers, dashes, braces, colons and dots; No leading or trailing spaces. Moreover the name must be unique within the set of scenarios of a project. When creating a scenario you may also want to upload a preview image, if no preview is uploaded a default image – the grey question mark – will be added to the scenario. After creating a new scenario it has an empty scene. The new scenario is automatically added to the currently selected project, see figure Figure 2 and Figure 3.

Copying

When you duplicate a scenario, a new name for the copy must be provided. The new name follows the same naming rules, which apply when creating a new scenario. The copy of the scenario is, except for the name, completely identical. If you want to change the preview image as well you can edit the copy afterwards. The copy has the same parent project as the original; currently there is no way to move a scenario to another project.

Editing

When you edit the scenario you can either choose a new name, following the aforementioned naming conventions, or upload a new preview image.

Deletion

Deleting a scenario will delete the scenario as well as the containing scene data and the preview image.

4.2. Scenario Editor

The following sections describe the Scenario Editor, or 3D Editor, the part where you can organize your 3D scene. As seen in Figure 4 the editor view is assembled from 5 different areas: the scene tree, the 3D view, the property and component panel, the PrefabSet tree, and the list of Prefabs.

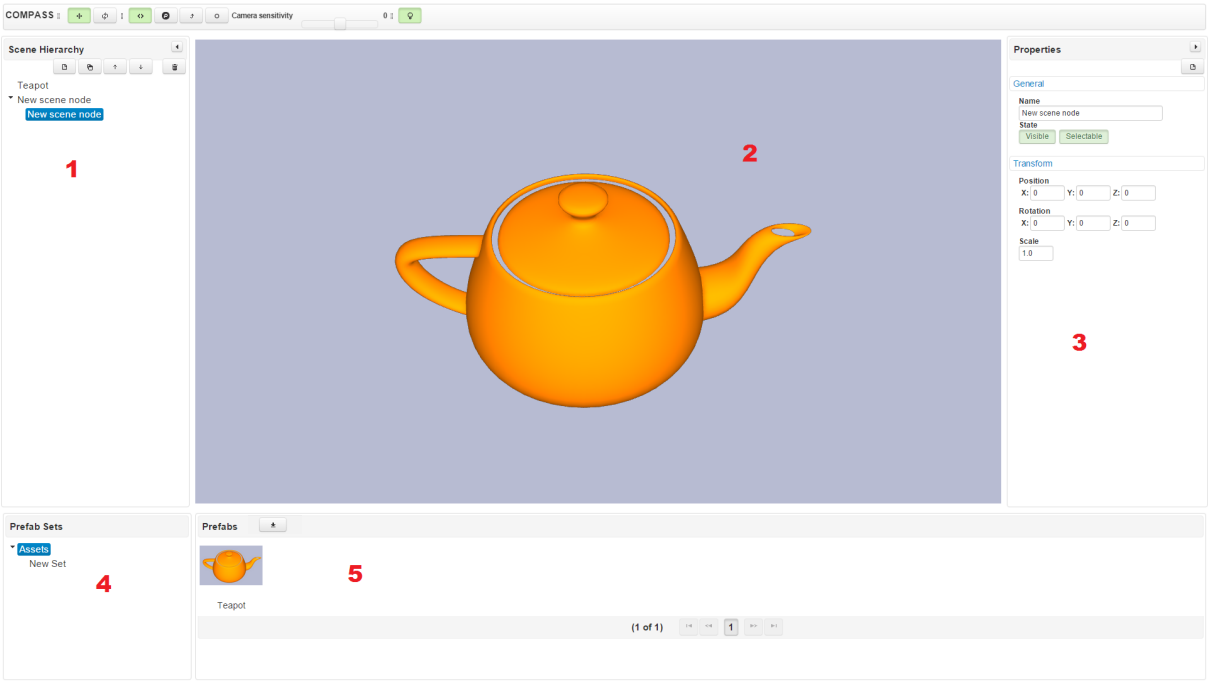


Figure 4: The Scenario editor, with (1) scene tree, (2) 3D View, (3) property and component editor panel, (4) Prefab set tree and (5) Prefab list

4.2.1. Scene Tree

The Scene Tree part of the editor view allows you to have an overview of the scene hierarchy, select *SceneNodes* and manage the scene hierarchy. The panel shows the scene tree; initially none of the nodes are expanded. You can open and close part of the tree by clicking on the disclosure triangle icons in front of the nodes name, just like you do in any other tree view, e.g. a file manager. You can create new SceneNodes, select a SceneNode, copy a SceneNodes or delete them and you can move them within the hierarchy. Most of those operations are available via the buttons at the upper end of the Scene Hierarchy panel or via the SceneNodes context menu Figure 5.

Select a SceneNode

You can select a SceneNode in the hierarchy just by left clicking it. You can also select it by clicking onto a 3D object in the 3D View. By doing so the SceneNode related to the 3D objects gets highlighted and the hierarchy gets expanded so that the SceneNode is visible in the Scene Hierarchy panel.

Create a new SceneNode

If you want to create a new SceneNode you can choose weather you want a new sibling or a new child. A new sibling appears alongside the currently selected Node on the same hierarchy level. If you create a new child it will be added as a child of the currently selected SceneNode. The new SceneNode has a default name, no transformations applied, and no components attached to it.

Copy a SceneNode

There is also the possibility to create a new node by duplicating an existing one. A duplicate of the selected node is created and will be added as a sibling (i.e., within the same parent as the original node). The duplicate has the same name, transformation, and also copies of the same SceneNodeComponents. This feature is only accessible via the context menu (see Figure 5).

Delete a SceneNode

To delete a SceneNode select it and click the delete button – the one with the trash can. Or use the context menu to do so.

Move a SceneNode in the Hierarchy

You can use the up and down arrow buttons at the top to move a node a position up or down within the hierarchy level. Alternatively you can just drag and drop a node to the place you want it to be. As you drop it into another level of the hierarchy – by this re-parenting it – the transform of the node will be adapted in such a way, that the node is still at the same spot in the global coordinate system.

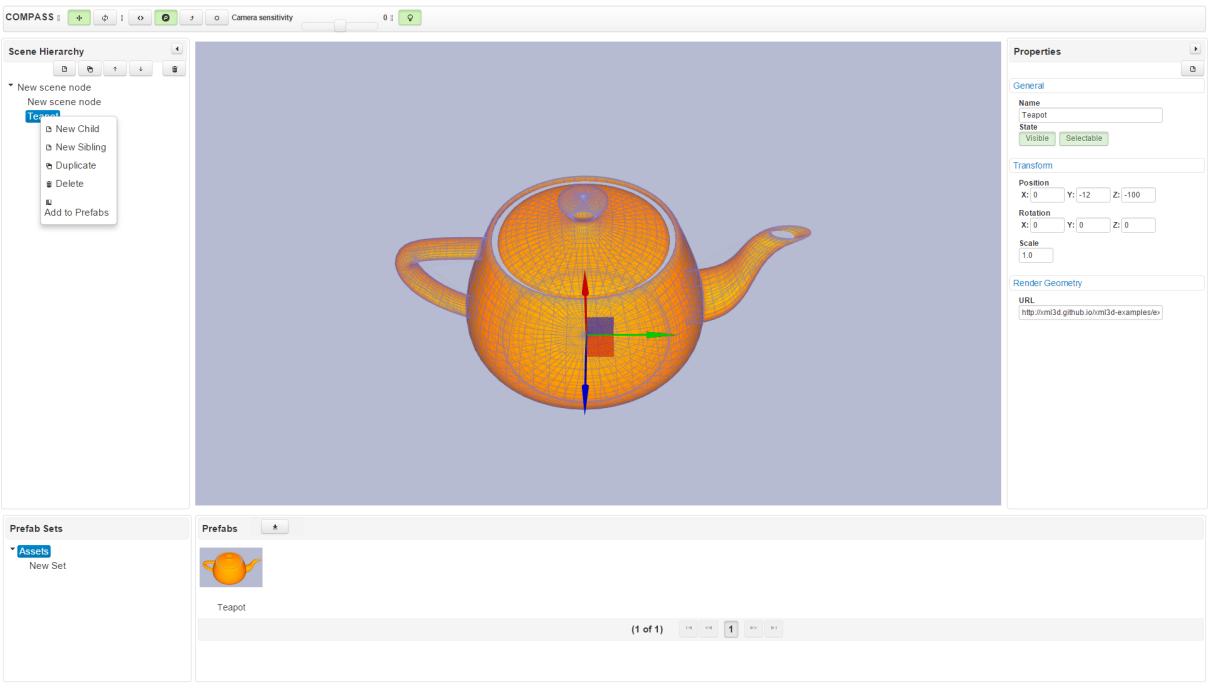


Figure 5: The Scene tree allows manipulation of the Scenarios Hierarchical structure.

4.2.2. Scene Node Properties and Components

On the right hand side of the Editor is the SceneNode's Properties Panel (see Figure 6). It allows you to view and manipulate the SceneNode's base attributes and to manage SceneNodeComponents.

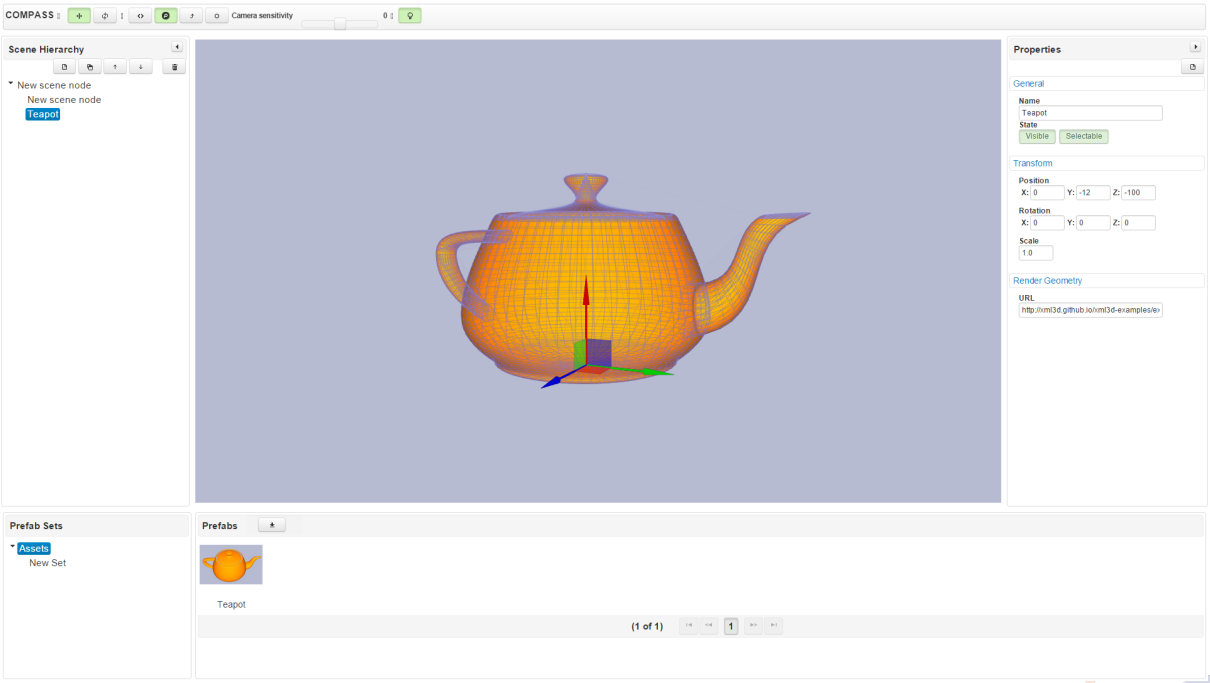


Figure 6: The selected scene node's properties can be edited in the right-hand side pane. As an example, the node selected has a RenderGeometry component, whose URL points to an XML3D asset.

Basic Attributes

Each SceneNode has a name, which can be altered in the properties view. The names don't have to be unique and the same conventions as for the other entities apply; they are limited to a length of 128 characters. This panel also allows toggling the nodes visibility and selectability in the 3D view. Unselectable nodes can still be selected in the scene tree, but will not be selected when clicking their 3D representation.

Transform

The properties panel also allows changing the transform of the SceneNode. In the transform section the transform is listed as a 3D position, a rotation in Euler angles, and a single uniform scale factor.

Manage SceneNodeComponents

In addition to the basic attributes, SceneNodeComponents can be attached to a SceneNode. To do so, use the button in the upper right corner of the Properties panel to get to the Component Selection dialog (displayed in Figure 7); there you can add and remove SceneNodeComponents. After adding a component, its editor will be rendered as new expandable section in the properties view. For more details on the available components, see Section 4.2.2.

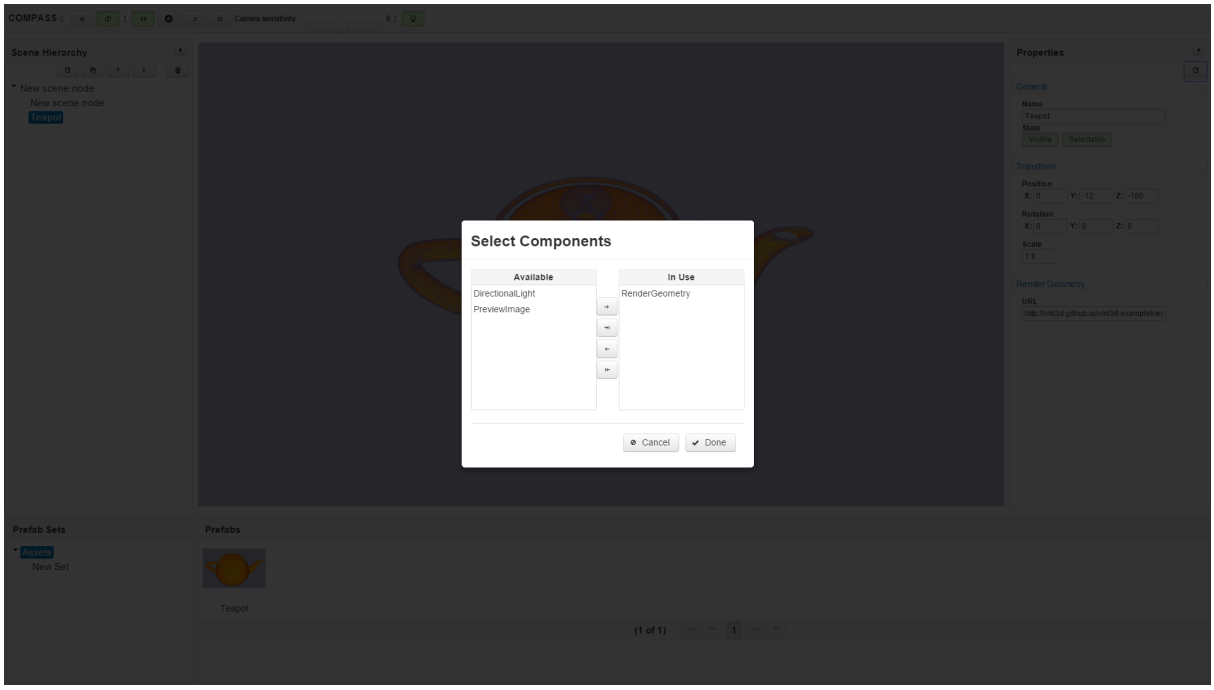


Figure 7: You can choose which SceneNodeComponents you want to add to a SceneNode.

4.2.3. PrefabSets and Prefabs

This section is about PrefabSets and Prefabs, the "Pre-Fabricated Elements". Those are useful if you have any re-usable models you need in a series of scenarios. Imagine you have a number of scenarios with the same topic, e.g. kitchens, where each scenario is a different example of how you might want to arrange the kitchen pieces. Then your Prefabs would probably be the different cupboards, shelves, stoves, sinks, fridges and maybe more. After creating the Prefabs once, you can use them in each of the scenarios.
Moreover, you can even sort those Prefabs in different sets. In our example you might have a set for dark furniture and a light one, or sets from different manufacturers. Or even both combined, since you can organize PrefabSets in a hierarchical tree. The tree is rendered in the Prefab Sets panel in the lower left corner. You can open and close subtrees, just like you do in the Scene Hierarchy tree.
Be aware, that any change you do to the PreabSets or Prefabs in the Scenario Editor will affect all Scenarios of that Project!

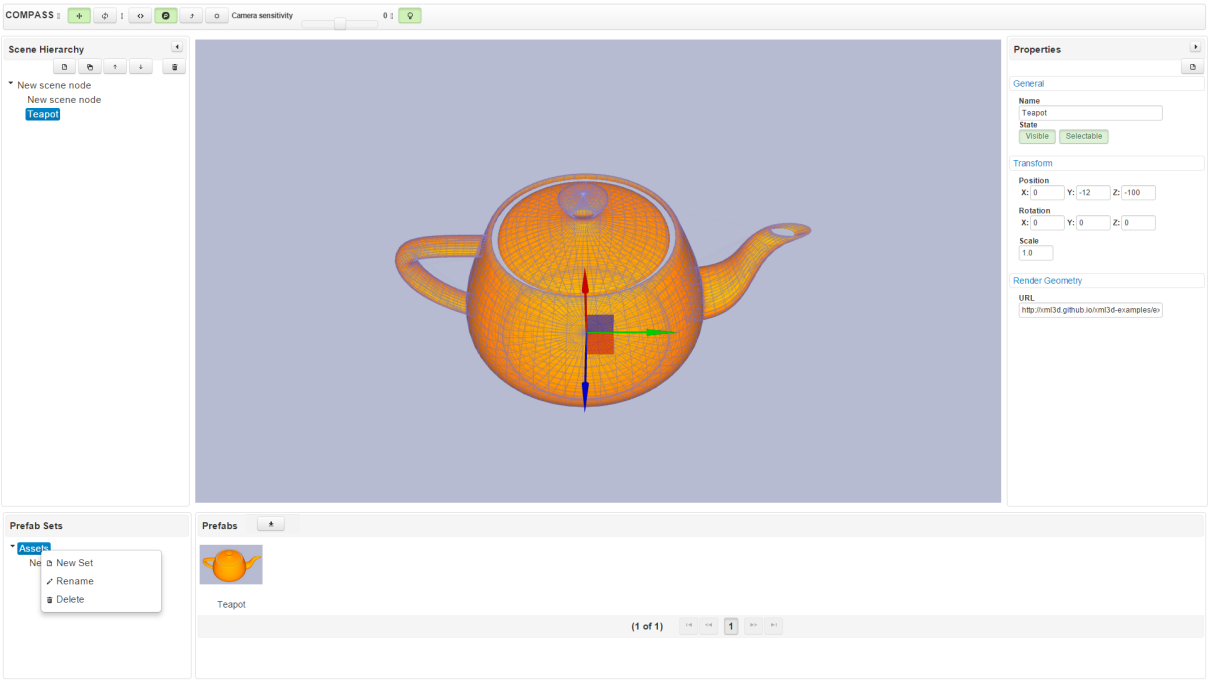


Figure 8: Prefabs are organized into Prefab sets. That way, Prefabs can be grouped by common themes and/or uses. All scenarios in the same Project share PrefabSets.

4.2.3.1 PrefabSets

As mentioned, the sets are organized in a hierarchical tree and are used to bundle Prefabs. Using the context menu on a PrefabSet (see Figure 8) you can create a new set, rename the current one or delete it.

Create a new Set
When you create a new set it will automatically be named "New Set" and is created as a child of the selected set.

Rename a Set
After creating a new set you might want to rename it. To do so, open up the context menu, select rename and enter a new name in the dialog box. The same restrictions as for projects and scenarios names apply: they can only contain letters, digits, dashes, braces, colons, and dots. No leading or trailing spaces. However, they do not need to be unique.

Delete a Set
Again using the context menu, you can delete a PrefabSet. As you delete a PrefabSet its child sets as well as all the Prefabs contained in those sets are removed. Note, that you are not allowed to delete the root PrefabSet.

Reparent a Set
Furthermore, PrefabSets can be re-parented to another parent set by just dragging and dropping them onto the new parent.

4.2.3.2 Prefabs

After explaining what prefabs are, we will now have look at how to create, organize, delete, and instantiate them. As you select a PrefabSet, its Prefabs are shown in the bottom bar. The Prefabs are listed and for each Prefab its Preview Image – if one is attached – is shown.
One more thing needs to be mentioned here: Prefabs are SceneNodes, which are not part of the scene tree, but are instead stored to be instantiated, i.e. to be copied to the scene tree, later. This also means, that Prefabs have the same properties and component mechanism as SceneNodes and can be altered the same way (see Section 4.2.2). Especially notable is the fact that Prefabs share their selection status with SceneNodes in the tree; when a Prefab is selected, the currently selected node in the scene (if any) becomes deselected.

Create a new Prefab
There are three ways to create a new Prefab: First, you can either drag and drop a SceneNode from the Scene Hierarchy panel into the Prefabs panel or onto a PrefabSet. Second, you can use the SceneNode's context menu entry "Add to Prefabs" in order to add it to the current set. Last but not least, a prefab can be created using the AssetStorage SE. The button with the downwards-pointing arrow can be used to import an existing 3D model from an AssetStorage SE instance (Figure 9). It possible to create a new asset from a COLLADA file using the "New Asset" button. This will upload the asset to the AssetStorage SE, which will convert the model into XML3D. Afterwards it will be available in the list of available assets. The button complete asset hierarchy will import the model as a full hierarchy of ScenNodes instead of a single SceneNode. This is useful if the hierarchy of the meshes inside the 3D model is important, e.g. for assembly models.

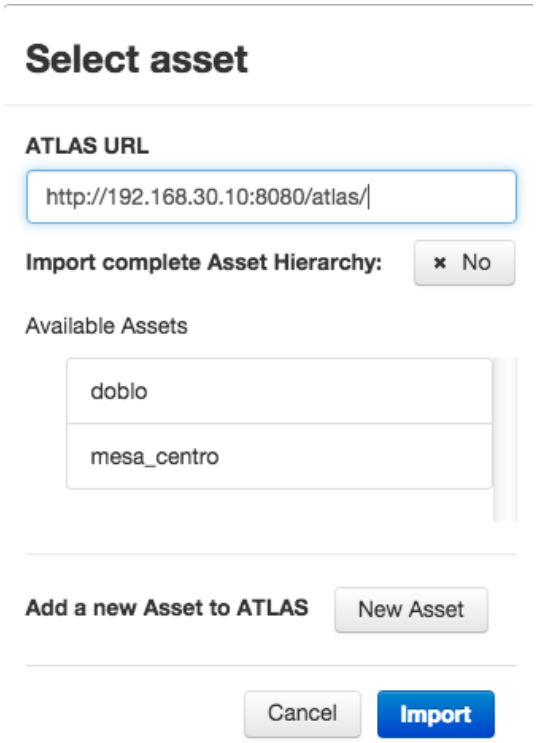


Figure 9: Dialog to import an existing or create a new asset using the AssetStorage SE.

Instantiate a Prefab

A prefab can be instantiated by either dragging it from the Prefab panel onto the 3D view and the prefab will be added to the root level of the scene hierarchy. Otherwise you can drag the Prefab onto a SceneNode entry in the Scene Hierarchy Panel, the Prefab will be instantiated as new child of the SceneNode.

Delete a Prefab

To delete a Prefab open its context menu, choose the delete option and confirm the deletion.

Move a Prefab to another Set

In order to move a Prefab to another PrefabSet, drag the Prefab and drop it onto the desired PrefabSet.

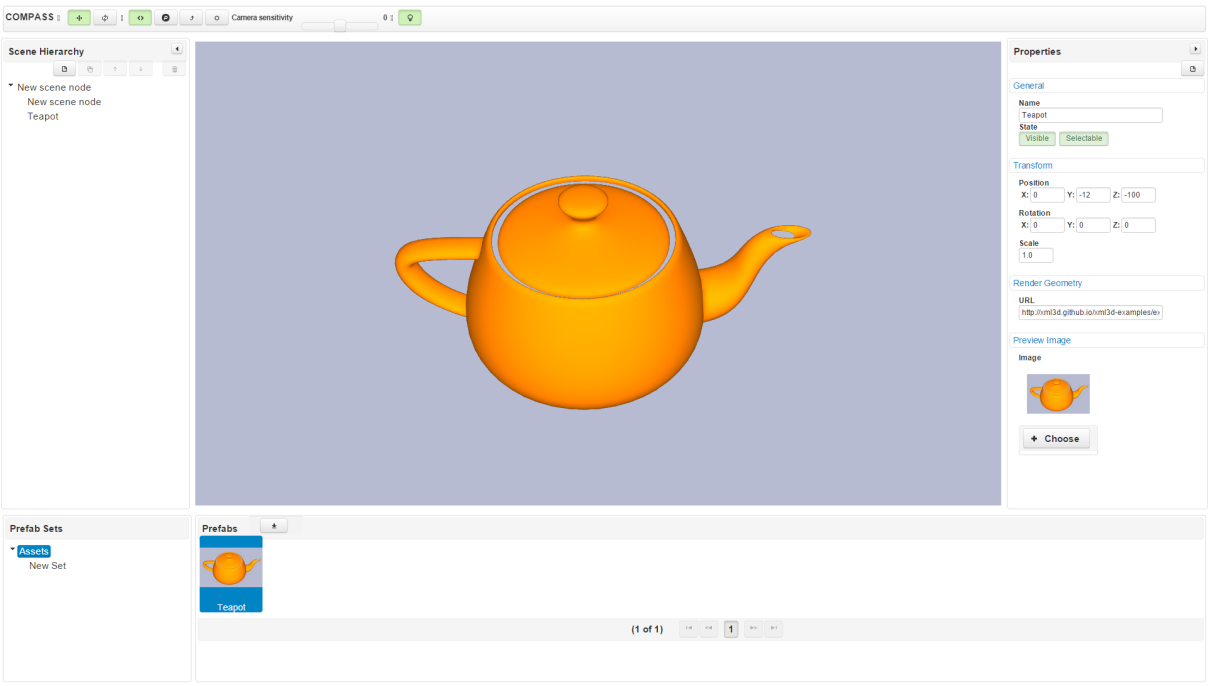


Figure 10: Prefab created from the Teapot in the scene. The property editor shows the PreviewImage component attached to the prefab.

4.3. 3D-Editor

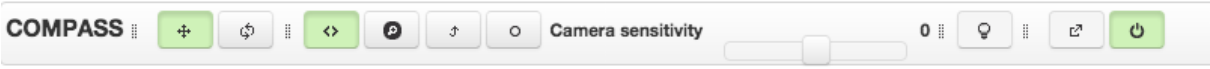


Figure 11: Upper button bar. From left to right: Translate widget, Rotate widget, Fly Camera, Inspect Camera, Reset to scene, Focus on selection, Camera speed slider, Default Light, Annotate Mode, Toggle Annotation Visibility.

At the very top of the C3DWV's interface is the toolbar. From left to right the controls are as follows:

- Activate translation gizmo. This disables the rotation gizmo.
- Activate rotation gizmo. This disables the translation gizmo.
- Activate fly camera controls.
- Activate examine camera controls.
- Reset camera in case you get lost in 3D space.
- Move camera to selected scene node.
- Camera sensitivity slider. Higher values mean more sensitive camera controls.
- Toggle default lighting. Use this if no lights have been placed in the scene, yet.
- Toggle annotation mode.
- Toggle annotation visibility.

4.3.1. Widgets

To manipulate the transform of a SceneNode two widgets are available in the 3D view. You can switch between them with the respective buttons in the top button bar. Holding the left mouse button and dragging the mouse will perform the transform associated with the widget element clicked.

Translation

The translation widget, shown in Figure 12, is selected by default. It is used to move a node along an axis, using the widgets' arrows. Nodes can also be moved within a plane using the small square between the arrows. While the SceneNode is manipulated using the widget, the transform values in the properties panel are updated as well.

Rotation

The rotate widget allows, just as its name indicates, to rotate a node around its local coordinate axes. The widget is shown in Figure 13.

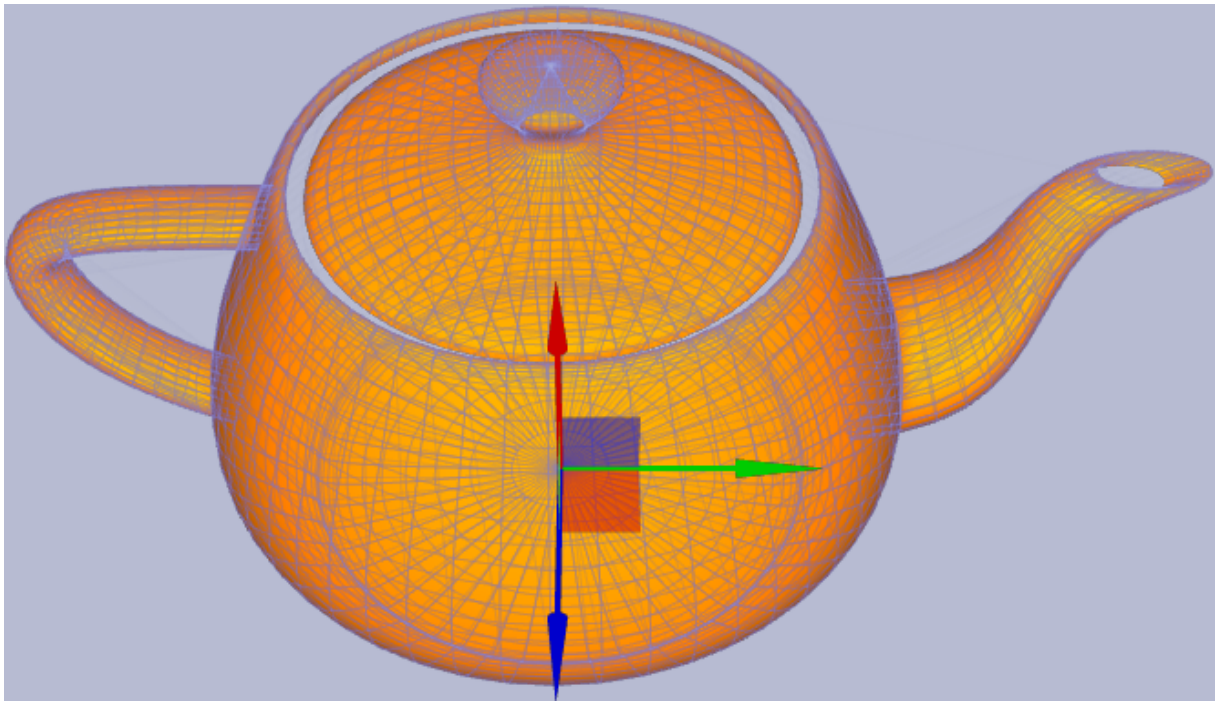


Figure 12: The Translate Widget.

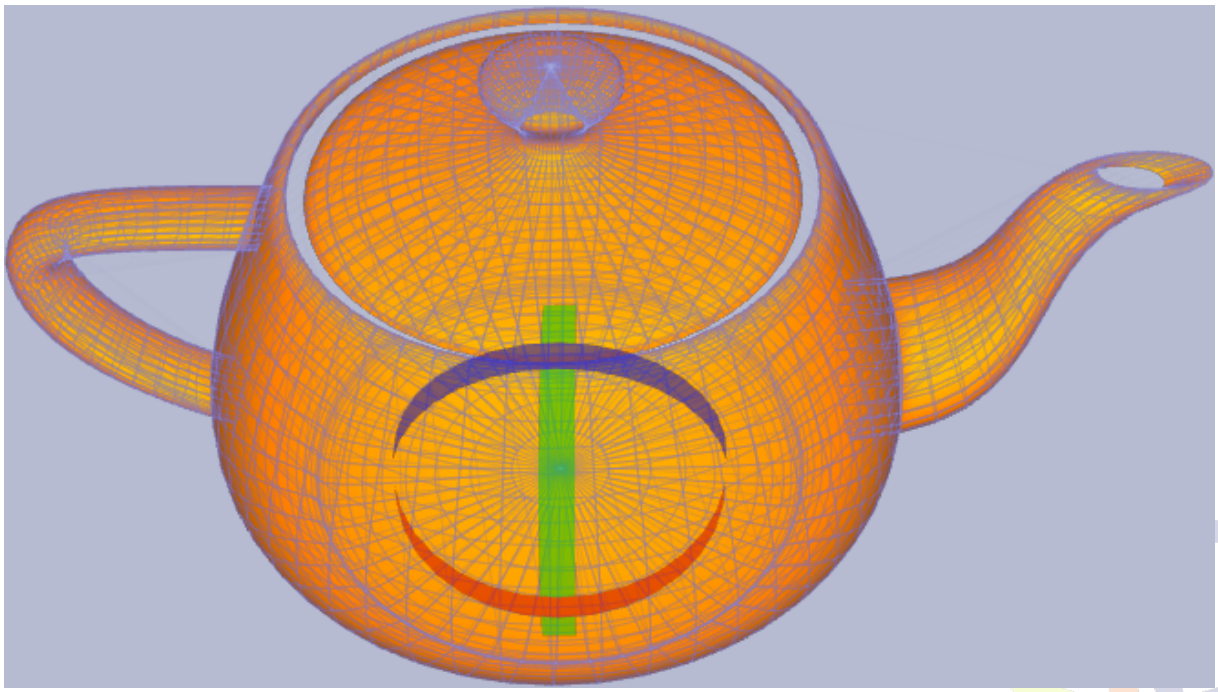


Figure 13: The Rotate Widget.

4.3.1.1 Camera

C3DWV offers two different modes for the camera controls: *Fly Camera* and *Inspect Camera*. You can use the buttons in the button bar at the top of the application to switch between the two modes. All other camera settings are also available via that button bar, shown in Figure 11.

Fly Camera
The *Fly Camera* allows the user to navigate the camera freely in the scene. You use the mouse and keyboard – WASD keys – to control the camera. The WASD keys move the camera in a plane, W forward, A to the left, S to the back and D to the right. Using the mouse, the camera can be rotated by keeping the right button pressed and moving the mouse.

Inspect Camera
In contrast to the Fly Camera the *Inspect Camera* is used to inspect a certain point or object. The camera looks at this point and is moved on a sphere around this point or zoomed towards this point. Keeping the middle mouse button pressed and moving the mouse rotates the camera around the inspect point. If you want to zoom, rotate the scroll wheel.

Camera Sensitivity
In both modes the slider in the upper tool bar allows you to adjust the camera speed. The value ranges from -30, very slow, to 30, very fast. The default is 0.

Resetting the Camera Position
There are two operations to reset the cameras position: The first will show the whole scene. The second one will focus on the currently selected SceneNode.

4.3.1.2 Default Lighting

Besides the camera sensitivity control is the button with the lamp icon. This buttons toggles a default light, just in case you are as lazy as the developers and you do not want to add your own light sources to the scene. The light will behave as if it was shining at the scene from behind the camera.

4.3.1.3 Annotations

Textually annotating virtual 3D objects is a feature found in many 3D applications. The implementation in C3DWV allows the user to place annotation everywhere on a 3D object (see Figure 14). A small pin in 3D space represents each annotation. A single click on the pin will show a dialog with the annotation (see Figure 6). This can dialog can also be used to edit the annotation.

Annotating Objects
To annotate a 3D objects the application has to be turned into annotation mode by clicking the second button from the right (see Figure 11). If this button is toggled clicking on an object will show the annotation dialog, as shown in Figure 15.

Hide/Show Annotations
The last button in the toolbar will hide or show all annotation pins in the scene.

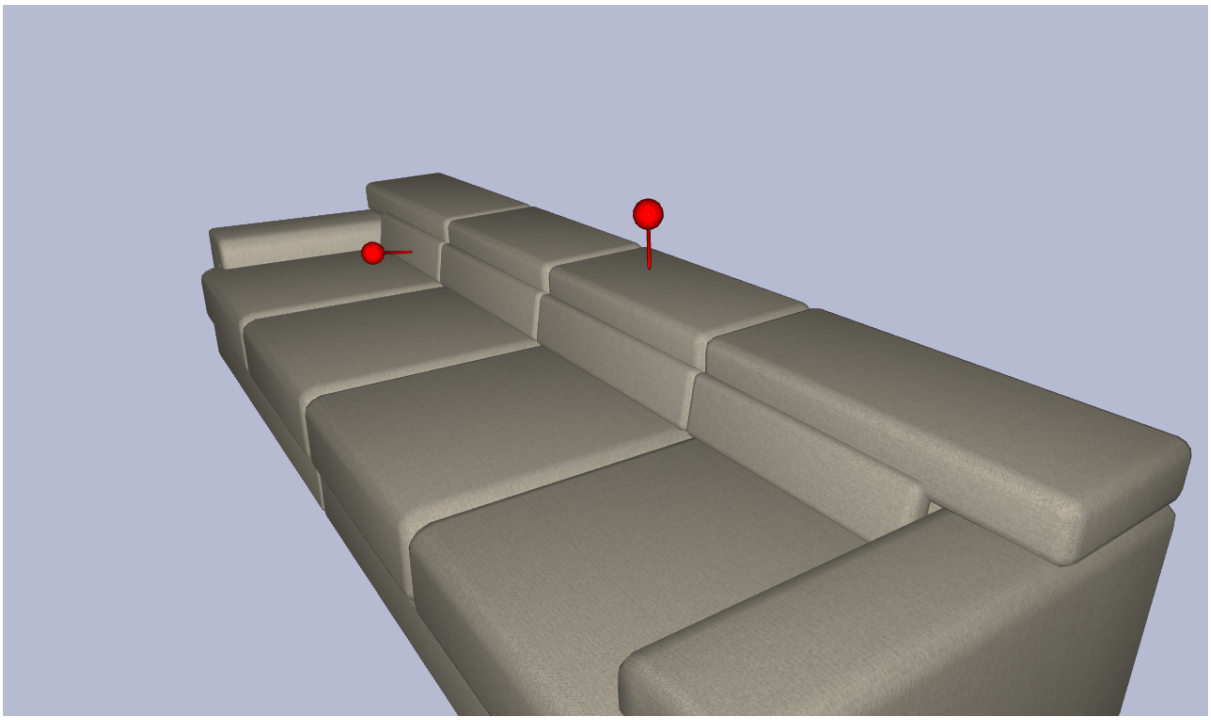


Figure 14: Annotations are represented as small pins in 3D space.

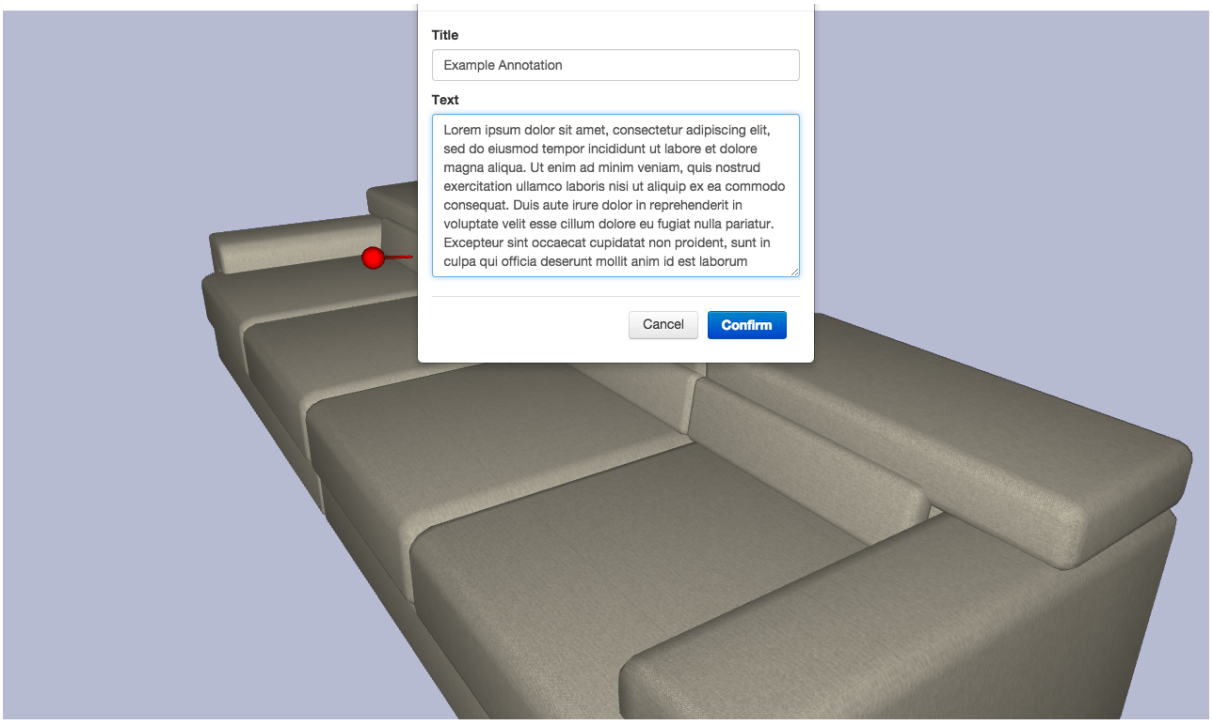


Figure 15: Annotation dialog.

4.4. Scene Node Components

C3DWV comes with three built-in SceneNodeComponents: *Render Geometry*, *Preview Image,* and *Directional Light*. How to manage SceneNodes and their Components was covered in Section 4.2.2.

Render Geometry

The *Render Geometry Component* has only a single attribute, which is a URL to an XML3D Asset (e.g. http://xml3d.github.io/xml3d-examples/examples/assets/res/assets.xml#teapot). As a result the asset referenced by this URL will be automatically rendered in the 3D view.

Preview Image

The *Preview Image* allows attaching an image to the SceneNode. This is really only useful for prefabs; those images are used as small previews in the Prefabs panel (see Section 4.2.3).

Directional Light

The *Directional Light Component* can be used to add a directional light source to the scene by attaching the component to a SceneNode. After adding the component a small sun icon right of the nodes name in the scene tree indicates that this component is attached to the node. The light sources color and intensity can be configured and the cast shadow option can be toggled. The light will shine in the direction the node it is attached to is oriented.

Annotations

The *Annotation Component* can be used to annotate a SceneNode. This component will be attached automatically if an annotation is placed in the annotation mode. Every annotation placed on the entity will be listed and can be edited or deleted.

Custom Components

Last but not least there is the possibility to create *Custom SceneNodeComponents*. These can be used to attach any type of metadata to a SceneNode. How to extend C3DWV and create your own SceneNodeComponents is explained in depth in Section 5.2.

5. Application Programming Interface

In the current Version no official API exists. For the next major release a REST-API is planned that allows other applications to operate on the internal shared scene graph structure. However, the REST-API, even though unstable, is already in the current version and used internally. Since it is already used by applications the following documentation explains the current **unofficial** API. Keep it mind that the API may change until the next major release even though we expect the API to only undergo minor changes.

5.1. Rest API

5.1.1. Introduction

C3DWV offers a REST API over which the scene trees managed by C3DWV can be accessed. A design goal of C3DWV, that a user or integrator should have access to all operations available in the client HTML application via REST as well.

C3DWV's REST API is designed to be self-documenting, by annotating the code with appropriate metadata. This is achieved through the use of Swagger, a REST documentation system. The documentation is extracted from the code and combined into a single document at compile time. This document, built as part of the C3DWV-rest module, is also put into the web application. After deployment, the REST API documentation is available at <C3DWV-path>/restapi-doc.html in the browser.

5.1.2. Example

To illustrate the usage of the REST API, let's have a look at a typical example. To retrieve a SceneNode, you may request a URL such as the following: http://localhost:8080/c3dwv/resources/restv1/scenenodes/4 Unless you forget the Accept: application/json header, C3DWV will reply with a JSON representation of that particular SceneNode. Figure 4.3 shows one such possible reply.
The JSON representation repeats the id, corresponding to the one at the end of the URL it has been retrieved from. Similarly to the web frontend, there are properties for the node's intrinsic properties. In contrast to the frontend however, the rotation is encoded as a quaternion, as opposed to degrees rotation about the principal axes.

```
{
  id: 4,
  name: "Teapot",
  parent: 3,
  components: [2],
  children: [ ],
  selectable3d: true,
  visible: true,
  localTranslation: { x: 0, y: 0, z: 0 },
  localScale: 1,
  localRotation: { x: 0, y: 0, z: 0, w: 0 }
}
```

Also differently to the frontend, the parent and children properties are there to traverse the scene tree. You can retrieve the corresponding scene nodes, by appending these ID values to the URL, e.g. /scenenode/3.
Finally, the components are the last and probably most interesting part of a given SceneNode. The corresponding SceneNodeComponent can be retrieved by getting the URL http://localhost:8080/C3DWV/resources/restv1/scenenodecomponents/2. Figure 4.3 shows the result of such a request.

```
{
  type: "de.dfki.asr.C3DWV.model.components.RenderGeometry",
  id: 2,
  owner: 4,
  meshSource: "http://xml3d.github.io/xml3d-examples/
               examples/assets/res/assets.xml#teapot"
}
```

5.2. Extending C3DWV

C3DWV comes with several possibilities to extend it: The most common ones are custom *SceneNodeComponents* and *Plugins*. There are also more advanced usages, such as integrating other Java Enterprise components. All of these cases are subsumed by the concept we call “Custom COMPASS”, which describes a combined package of custom parts and what we call “stock” COMPASS.
A custom C3DWV can be assembled manually, however this is a rather tedious task. Therefore, we offer a template that can be instantiated and contains the necessary groundwork. Since we use *Maven* as our build system, we chose the mechanic that Maven offers for this kind of task. Maven calls these project templates *Archetypes*.
In the following sections we will focus on the Archetype, and its extension by custom SceneNodeComponents or by Plugins. We will explain which functionality those two extension mechanisms provide and how to use them. The aforementioned advanced usages border on developing C3DWV itself. Thus, we would like to refer you to the respective part IV of this document.

5.2.1. Custom C3DWV

Before you can start to create a custom C3DWV, C3DWV itself and the *Custom* COMPASS *Archetype* need to be available to your Maven installation. Since the archetype is created and installed as part of the overall C3DWV build process, we recommend doing just that. Go to your local clone of the C3DWV git repository and run mvn install. If everything went well, the C3DWV packages and the archetype have then been installed to your local repository. Of course, if you built and installed the C3DWV parent project beforehand, this was already done for you.

Afterwards, you can instantiate the archetype by running:

```
mvn archetype:generate
    -DarchetypeGroupId=de.dfki.asr.C3DWV
    -DarchetypeArtifactId=C3DWV-custom-archetype
```

If you use an IDE to develop, you can use its inbuilt archetype instantiation mechanism. You will probably need to specify the above group and/or artifact ID.

In summary, the process of creating a custom C3DWV is surprisingly short and consists only of the following steps:

- Checkout C3DWV
- Build C3DWV
- Instantiate a custom C3DWV archetype
- Add the functionality to the custom modules
- Build the custom C3DWV
- Deploy and run

5.2.2. Anatomy of a Custom C3DWV

To keep things modular and nicely separated, your C3DWV-based application should follow the pattern of the archetype (that is very similar to C3DWV's own structure), which consists of these main modules:
Data Model: These are your Java (JPA, to be precise) classes that hold the data in your application. Most likely, you will have additional SceneNodeComponents in there, which extend C3DWV's Scene model.

Persistence Unit: Your custom application defines its own JPA persistence unit, which ties together the base C3DWV data model and your customized data model from the previous item. The base C3DWV business logic, which is added to your application in the deployment module will then pick up and work on the combined data model.
Business Logic Implementation: If you choose to add complex operations on your data model, this is the place to put it. Classes in this Package should probably be EJBs, to benefit from Container Managed Transactions. At the very least, you will need to register any custom SceneNodeComponent that you created in your Data Model with C3DWV' base business logic. An example how to do this is contained in the Archetype.
Web Applicaton Overlay: Custom user-visible interfaces to your Data Model and Business Logic go here. The Overlay module is configured such that it will take C3DWV' base web interface, and add your contents, or even replace parts of the base web interface with your variant.
Deployment: This module takes all of the above, and adds C3DWV' base business logic. All of this tied together will give you an *enterprise application archive* (EAR), which you can deploy to a container configured for C3DWV. With that, you should have a customized web application that extends C3DWV with your features!

5.2.2.1 SceneNodeComponents

This section will briefly go over the minimum requirements to add a custom *SceneNodeComponent*. We anticipate that this will be the most frequent oper- ation users of the archetype will do. For each of the modules presented in the previous section, we will show what needs to be modified.

Model

Modify and rename the SampleSceneNodeComponent class to suit your needs. Observe especially the contents of the @C3DWVComponent annotation, as that will be used by the base C3DWV web layer to provide an interface to your component.
Entity classes should not be very "smart", in the sense that they support com- plex semantic operations. These operations should be implemented in the

Business Logic module

The main concern of classes in the Data Model module is database storage and serialization. Therefore you should take care to get all of the annotations right (javax.xml.bind.annotation.* and com.fasterxml.jackson.annotation.*).
You can, of course, add multiple SceneNodeComponent derived classes at this point. They all need to follow the scheme laid out in the SampleScene- NodeComponent. You are limited mostly by JPA and the serialization frameworks.
Some IDEs check a project for the presence of JPA annotations. If your IDE warns you that "The project does not contain a persistence unit.", please ig- nore the warning. The persistence unit only gets added to the deployment later on.

Persistence Unit

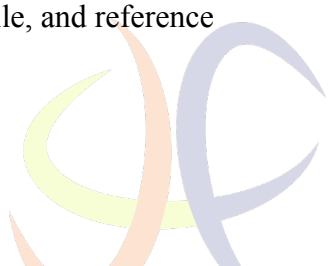
You don't really need to modify this, unless you rename your Data Model artifact. The final name of the Data Model module JAR file is referenced in the persistence.xml file. If you edit your persistence.xml file, remember to leave the reference to the C3DWV-model JAR intact.

Business Logic

Here, you need to add any SceneNodeComponents from your Data Model you want to use to the SampleComponentAnnouncer.announceComponents() method. The Announcer will be picked up by C3DWV' base business logic via CDI. The ComponentRegistry looks for all implementations of the ComponentAnnouncer interface.

Web Application Overlay

The samplecomponent.xhtml contains a predefined *Facelet UI composition* which will be displayed in C3DWV' web UI properties view. Modify this to suit your needs, i.e. to be able to edit all of your entity class' data fields.
If you decide to rename the file, don't forget to update the @C3DWVComponent annotation on your Entity class.
If nothing is displayed in the Properties View after adding your component to a scene node this usually indicates a problem with the JSF in your component's .xhtml file. Your JavaEE container's log should provide further information on what went wrong.
If you added more than one SceneNodeComponent, each should probably have an editor facelet. Add more .xhtml files modeled after the samplecomponent.xhtml file, and reference them in your component classes.



Deployment

The Deployment module is configured mainly via its pom.xml file. There, you can add further modules to the EAR (such as an external EJB or WAR module you want to integrate), or configure the root folder under which the web application(s) will be visible within the container. Please refer to maven's EAR plugin documentation for further details.

5.2.2.2 Building and Installing

Once your implementation is finished, go to the root of your archetype instance and run mvn package. If all goes well, the finished .ear can be found in your deployment/targetfolder.