Cy3D Documentation

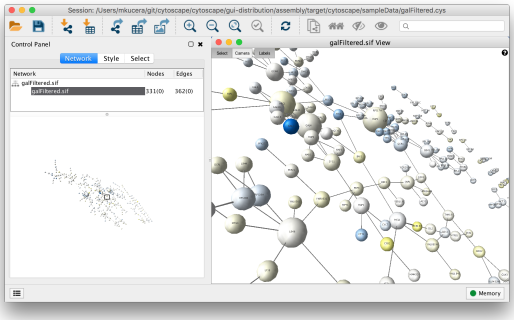
Release 1.0

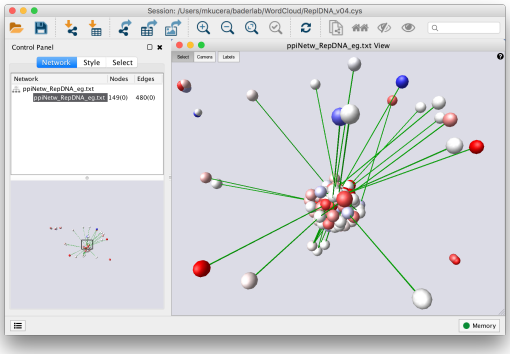
Mike Kucera

Feb 01, 2018

Contents

A 3D Renderer App for Cytoscape





CHAPTER 1

Installing

1.1 Requirements

Requires Cytoscape 3.3 or higher.

1.2 Installation

Install Cytoscape

- If you don't have Cytoscape please download and install the latest release from http://www.cytoscape.org/download.php.

Install Cy3D

- Open Cytoscape
- In the main menu select **Apps > App Manager**
- In the App Manager select Cy3D in the list of All Apps and click the Install button.

Alternatively Cy3D can be installed from the App Store.

- http://apps.cytoscape.org/apps/cy3d

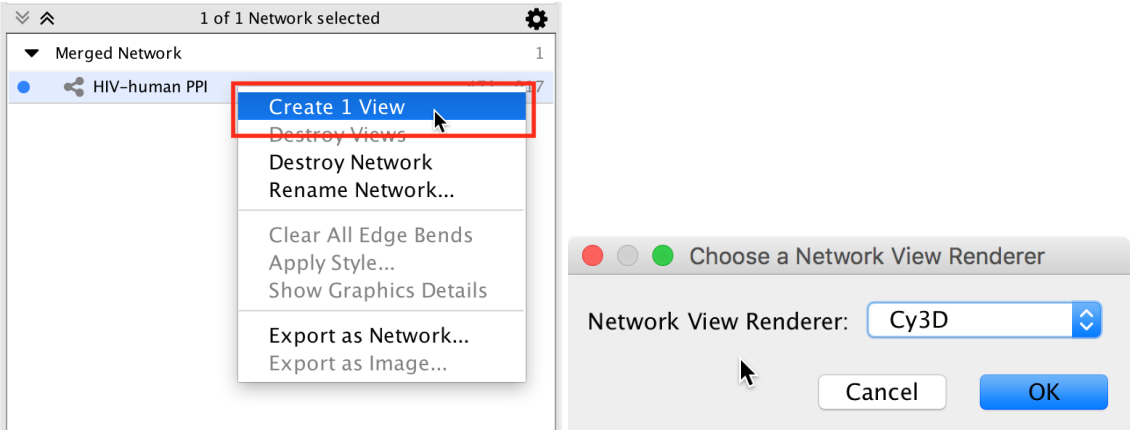
CHAPTER 2

User Guide

2.1 Creating a 3D View

To create a 3D view:

- Select a network in the Network panel.
- Right click on the network and select Create View (you may need to select Destroy View first).
- In the "Choose a network view renderer" pop-up select Cy3D.



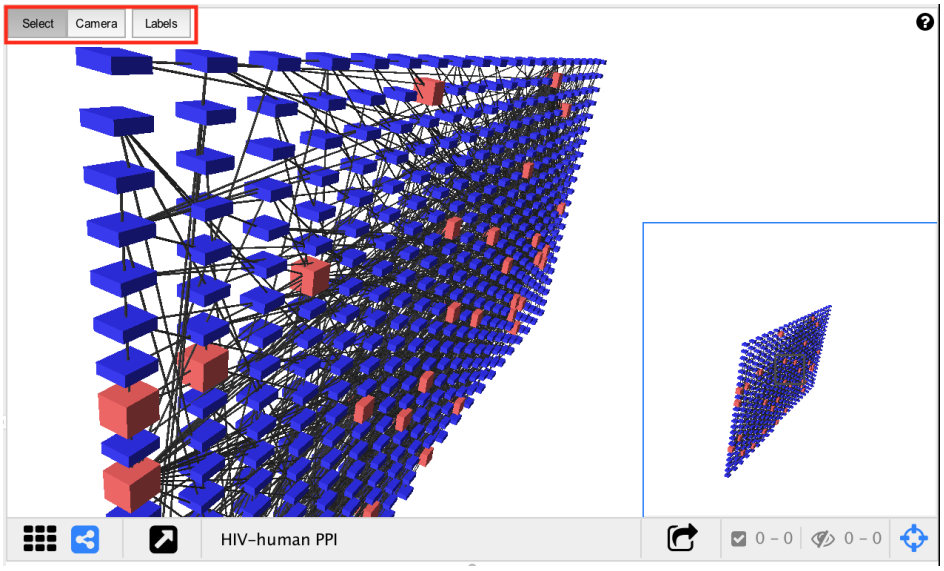
To create a 3D view without destroying the existing view:

- Open the network view in the standard 2D renderer.
- Select all the nodes (Select > Nodes > Select All Nodes).
- Click the New network from selection toolbar button.
- Close the network view that was just created.
- Select the newly created network in the Network panel.

- Right click on the network and select Create View.
- In the “Choose a network view renderer” pop-up select Cy3D.

2.2 Controlling the Camera

- Cy3D has 2 modes: camera mode and select mode.
- The modes are toggled using the toolbar in the 3D renderer view.



- In camera mode the mouse rotates the camera around the network. Mouse wheel moves the camera in and out.
- In select mode the mouse is used to select nodes and edges, and to activate the context menu.
- For a detailed list of controls click the (?) icon at the top right of the 3D renderer view.

3D Renderer Controls

Toolbar	Buttons toggle between Select Mode and Camera Mode
Shift	Hold to force Select Mode
Alt	Hold to force Camera Mode

Select Mode

Left Click	Select Node or Edge
Left Click and Drag	Selection box
Ctrl + Left Click	Add to current selection
Right Click	Context menu

Camera Mode

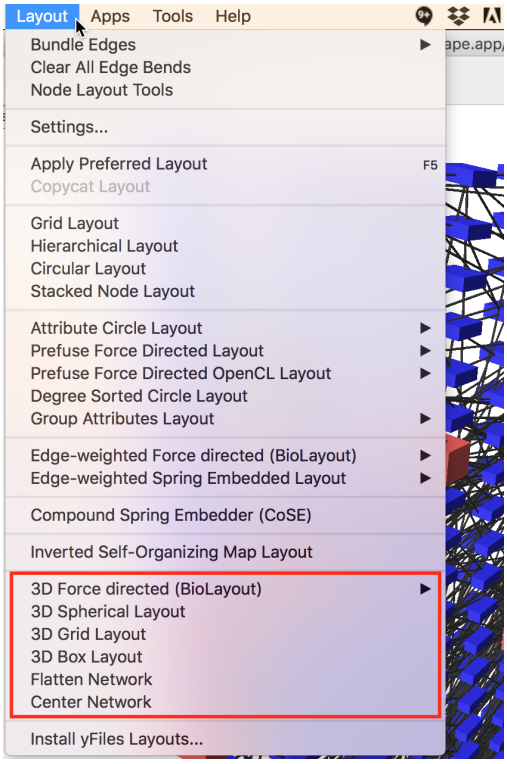
Click and Drag (Near Center)	Orbit Camera
Click and Drag (Around Perimeter)	Roll Camera
Mouse Wheel	Zoom In/Out

Keyboard Controls

Up, Down, Left, Right	Orbit camera (at constant speed)
R	Reset camera to default position

2.3 Layouts

Currently the only way to move nodes around in 3D is to apply a layout.



There are 4 3D layout algorithms available in the Layout menu:

- 3D Force Directed (BioLayout)
- 3D Spherical
- 3D Grid
- 3D Box

Additionally there are two other ways to manipulate the graph which are implemented as layouts:

Flatten Network Sets the Z coordinate of every node to zero. It is useful if you would like to apply a 2D layout, because most of the existing 2D layouts just ignore the Z coordinate. Apply the Flatten layout before or after applying a 2D layout to fix the results of the 2D layout.

Center Network Computes the centroid of all the nodes and translates the graph so that the centroid is the new origin.

CHAPTER 3

Cytoscape Renderer Implementation Guide

Cy3D can be used as an example for how to add a renderer to Cytoscape. The following provides an overview of the important parts of the Cy3D codebase that will help to create your own custom renderer App for Cytoscape.

Cy3D source on GitHub

Implement NetworkViewRenderer

- This is the starting point that Cytoscape uses to access your factories for creating renderers and network views.
- Provides the display name and unique ID of your renderer.
- Creates instances of RenderingEngineFactory.
- Creates instances of CyNetworkViewFactory.
- See Cy3DNetworkViewRenderer.java

Implement RendereringEngineFactory

- You should provide two rendering engine factories, one for the main view, and another for the smaller birds-eye view.
- The RenderingEngineFactory creates an instance of RenderingEngine for the given network view.
- The RenderingEngineFactory also provides access to your VisualLexicon.
- A container object is passed to the factory whenever a RenderingEngine is created. This is typically an instance of JComponent, and will be the parent for the RenderingEngine's drawing canvas.
- Your RenderingEngineFactory must also register the newly created RenderingEngine with the Cytoscape RenderingEngineManager.
- In Cy3D there is one class Cy3DRenderingEngineFactory that implements RenderingEngineFactory. Two instances are created, each is parameterized with a GraphicsConfigurationFactory which provides functionality that is specific to the main view or the birds-eye view.
- See Cy3DRenderingEngineFactory.java

Implement CyNetworkViewFactory

- Creates instances of CyNetworkView.

Implement RenderingEngine

- Creates and initializes a "canvas" for drawing.
- Attaches Mouse and Keyboard listeners to the "canvas" to handle input.
- In Cy3D the "canvas" is a GLJPanel which is a special panel that can render a 3D scene using OpenGL.
- See Cy3DRenderingEngine.java

Implement CyNetworkView, View<CyEdge> and View<CyNode>

- These are your "view model" objects that represent the visible network, nodes and edges.
- Each View object must have a unique SUID. These should be generated using SUIDFactory.getNextSUID().
- The main job of these objects is to store visual properties. For this reason Cy3D has a base class called Cy3DView for network, node and edge view which provides the visual property storage code.
- Important methods in CyNetworkView:
  - The constructor, this creates View instanes for all the nodes and edges in the network.
  - CyNetworkView.updateView(). This is called whenever something changed and your views should be updated. In Cy3D it just tells the main and birds-eye view canvases to repaint.
  - getRendererId(), this must return the same renderer ID that is defined in your NetworkViewRenderer.

Extend BasicVisualLexicon

- Cytoscape expects your VisualLexicon to provide all of the visual properties that are part of BasicVisualLexicon, even if your renderer does not use some or all of the properties. This is because existing Cytoscape code and Apps expect all of those properties to be available programatically.
- If your renderer does not support some of the visual properties from BasicVisualLexicon these can be hidden from the UI by overriding the isSupported() method.
- If your renderer does support a discrete visual property but does not support some of the values in the discrete range then they may be hidden from the UI by overriding the getSupportedValueRange() method.
- Again, visual properties from BasicVisualLexicon can be hidden from the user, but they must always be accessible in the code.

(Optional) Extend AbstractLayoutAlgorithm

- Your renderer may also provide additional layouts.

(Optional) Pop-up menus.

- Your renderer may provide access to some of the same context menu actions as the 2D renderer, however this is not easy to achieve.
- Keep in mind that many of the menu actions are not multi-renderer aware and will fail with a ClassCastException when used with a renderer that is not the default 2D renderer. These actions should be filtered out.