2006

Molecular viewer using Spiegel

Pavani Baddepudi

Recommended Citation

Baddepudi, Pavani, "Molecular viewer using Spiegel" (2006). Thesis. Rochester Institute of Technology. Accessed from

Molecular Viewer Using Spiegel

Pavani Baddepudi

Agenda

- Introduction & Background
- Project Overview
- Implementation Details
- Demo
- Future work
- Q&A Session

Introduction

- **3-D molecular viewer using Spiegel framework**
- **Distinctive representations of the molecules**
- **Interactive features**

Background Knowledge

- **Molecular Biology**
- **ProteinDataBase files**
- **CPK Coloring scheme**
- **Covalent and Vanderwaal’s radii**
- **Java3D**
- **Spiegel framework**

Related Work

- **JMOL**
- **Rasmol**
- **Chime**
- **JMV**

Algorithm

-Forms the basis of the structure

-Used to determine the bonding between the atoms

-Project uses same algorithm as Chime and Rasmol

```
if (distance(coords[i], coords[j]) between min_max)) {
            Bond exists
 }  else  {
             Bond does not exist
            }
```

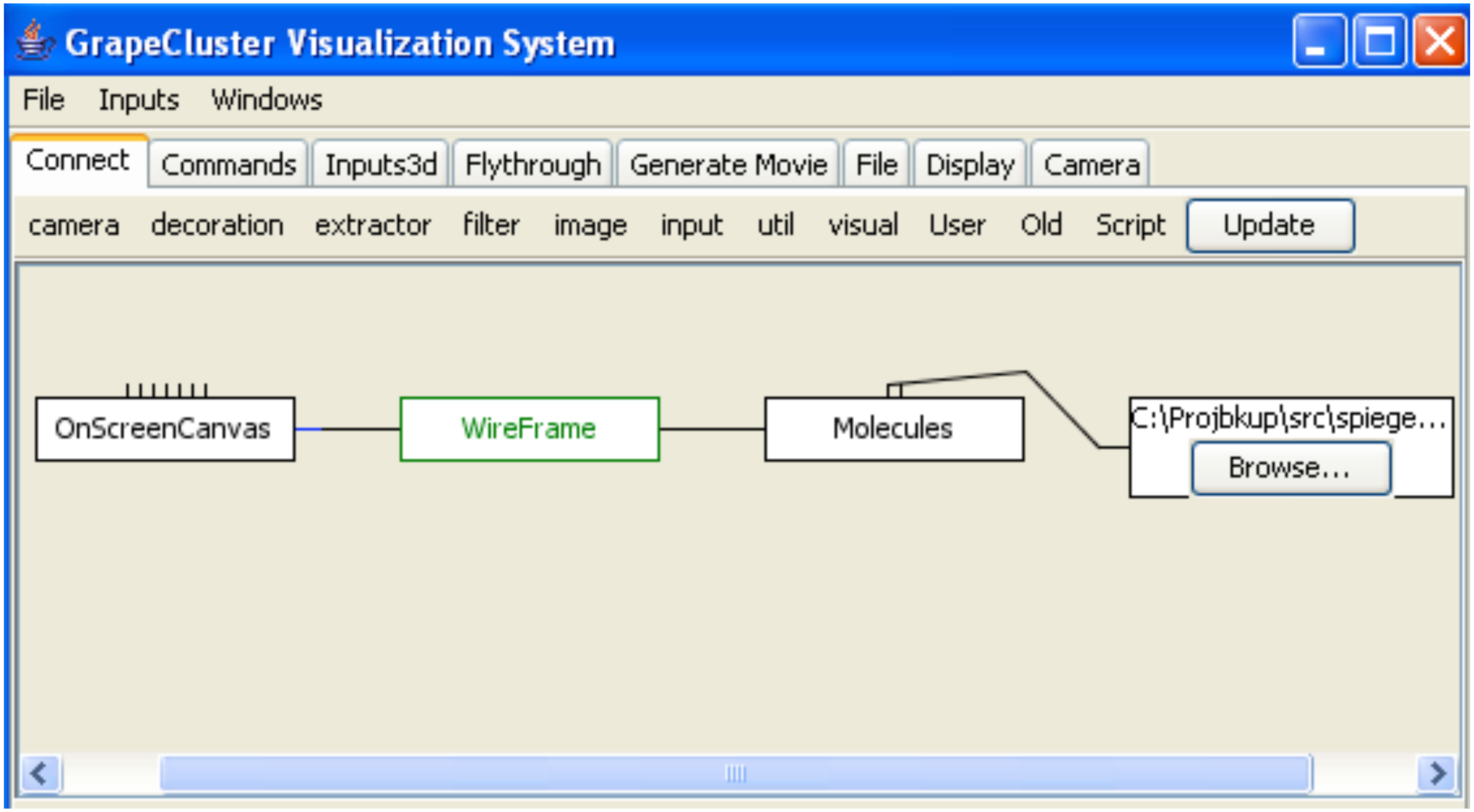
Where coords[i] and coords [j] are the coordinates of atom1 and atom2 respectively and   min = 0.6f max= 1.2 f.

Spiegel Architecture

Consists of four plugins that form the building blocks of the system

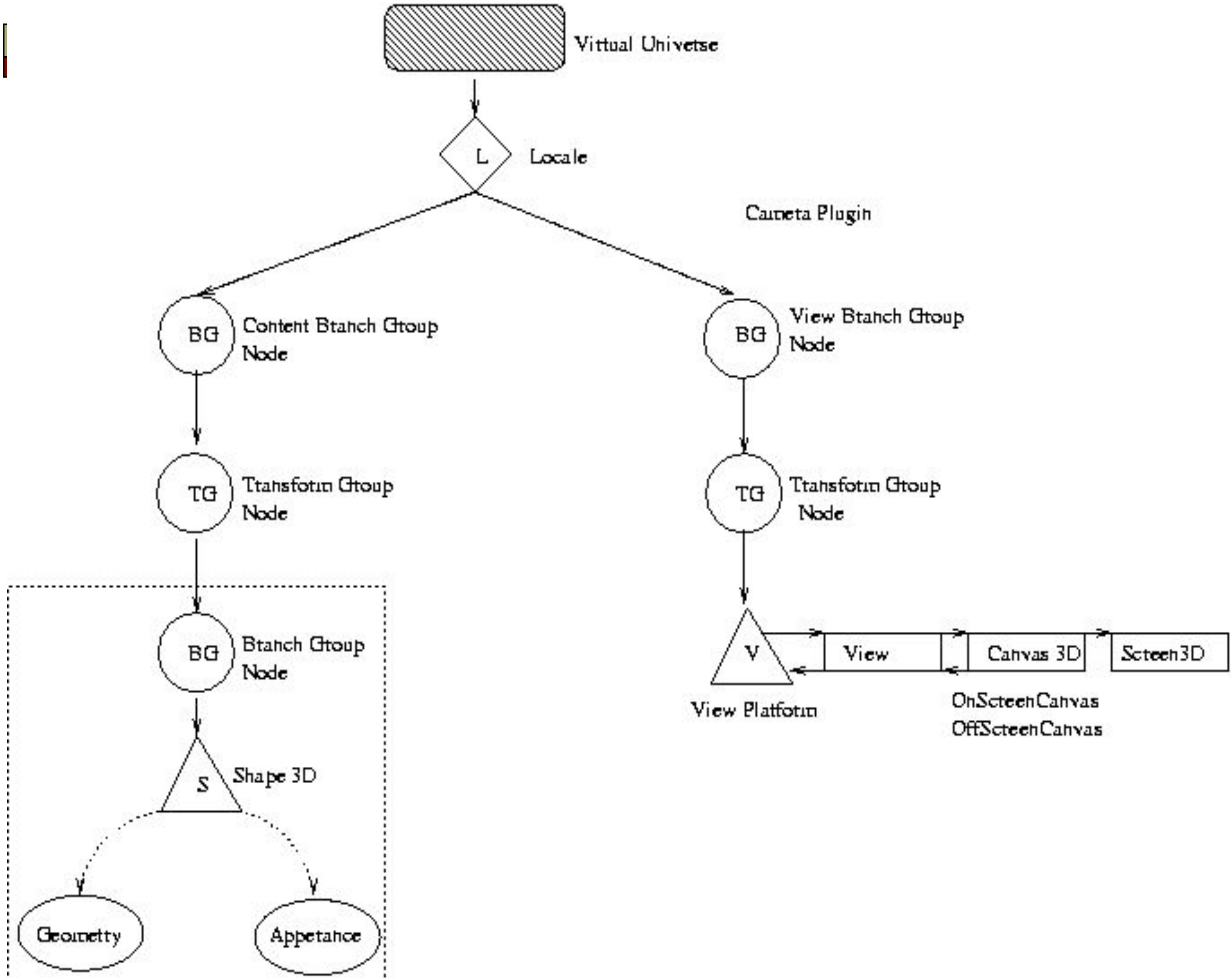
- **Extractor**
- **Visual**
- **Camera**
- **Filter**

Spiegel framework (as implemented in this project)



Java Scenegraph

Overview of the code flow


Virtual Universe
L
Locale
Camera Plugin
BG
Content Branch Group
Node
BG
View Branch Group
Node
TG
Transform Group
Node
TG
Transform Group
Node
BG
Branch Group
Node
V
View
Canvas 3D
Screen3D
View Platform
OnScreenCanvas
OffScreenCanvas
S
Shape 3D
Geometry
Appearance

Implementation Details

Extractor Plugin

- **Extractor plugin (Molecules)** performs the function of extracting the relevant information from the pdb files.

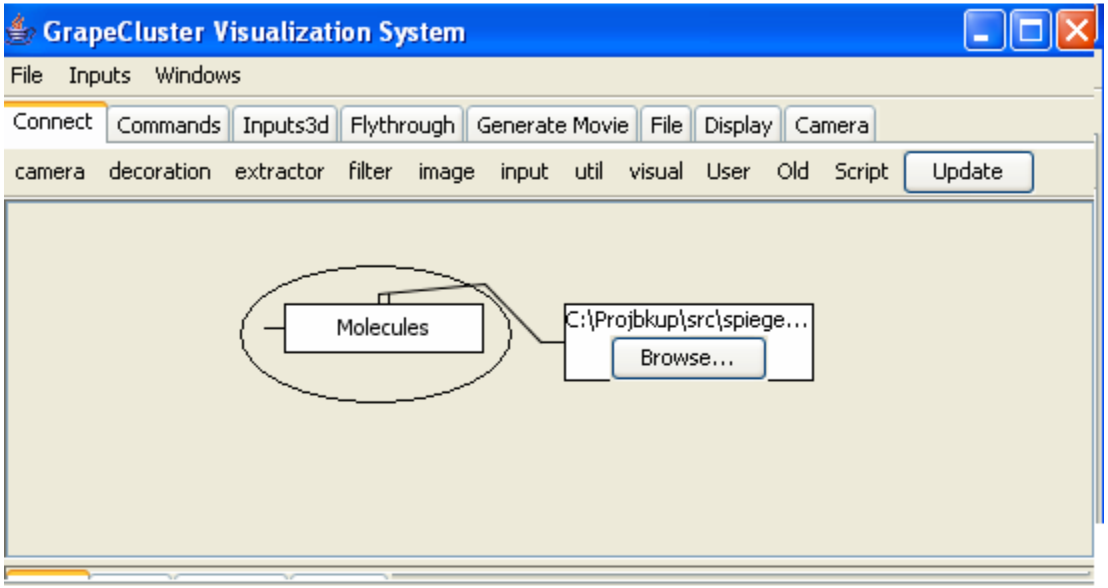
Data Structures

- HashTable **chainIds** stores the chain information
- HashTable **PositionList** holds the position of the atoms to be placed in the scene.
- The class **NeighborList** calculates determines their bonding with the atom using the algorithm
- HashMap **MoleculeIDMap** contains the serial number and name of the atoms.

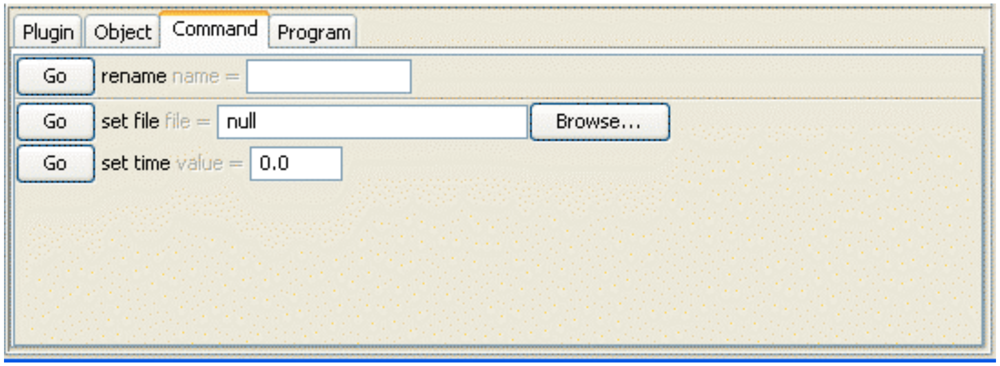
Input and Output variables

- **Input** – Files with pdb extension only.
- **Output** - passed onto **Filter** or **Visual** Plugin. (MoleculeIdMap)

Representation in Spiegel



C:\Projbkup\src\spiege...



rename name =

set file file = null

set time value = 0.0

Filter plugin

- Prunes the data obtained from the pdb files to focus on particular strands. Has been applied to the secondary structure only.

DataStructure

Depending on the input string from the user filter plugin stores the serial number and the relevant chainId in a **chainIdMap** which is a SortedMap

Input and Output variables

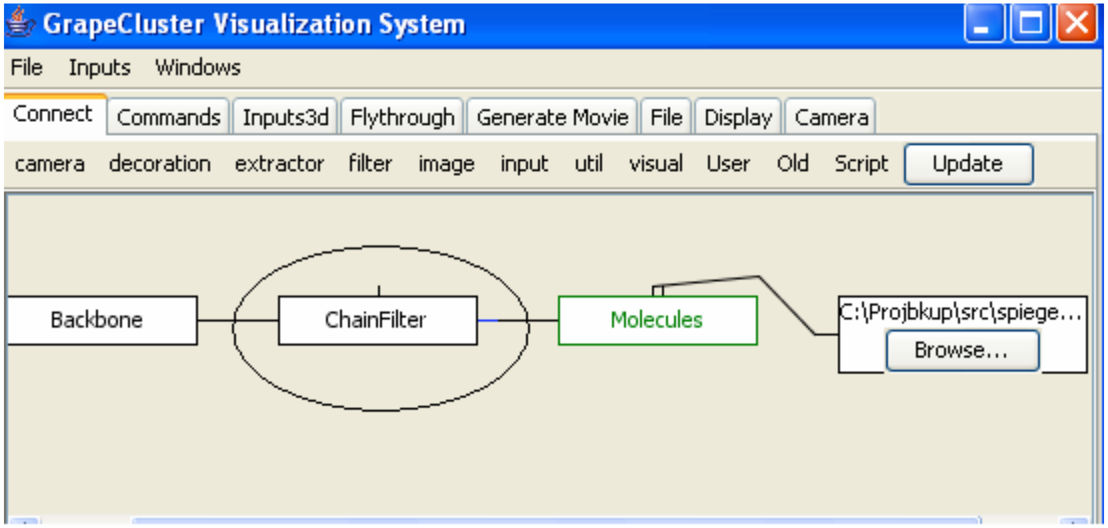
- **Input** – HashMap **MoleculeIDMap** from Extractor

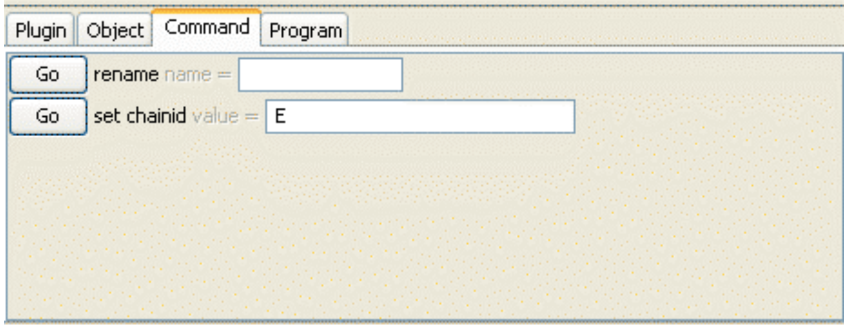
  Strings predefined for DNA file 1d66.pdb

  P – Amino Acid chains **N** – Nucleic Acid chains **D, E** – Individual DNA strands
  A, B- Individual Protein strand

- **Output** – SortedMap **ChainIDMap**

Representation in Spiegel





rename name =

set chainid value = E

Visual Plugin

- **Visual plugin** imparts shape, size and color of the atoms

DataStructure

- Wireframe **(Wireframe)** Representation

  Line Array

- Ball and Stick Representation **(Molecules3D)**

  Line Array and Point Array

- Backbone Representation **(Backbone)**

  LineStrip Array

- Spacefill Representation **(CPKmodel)**

  Sphere

Data Structure

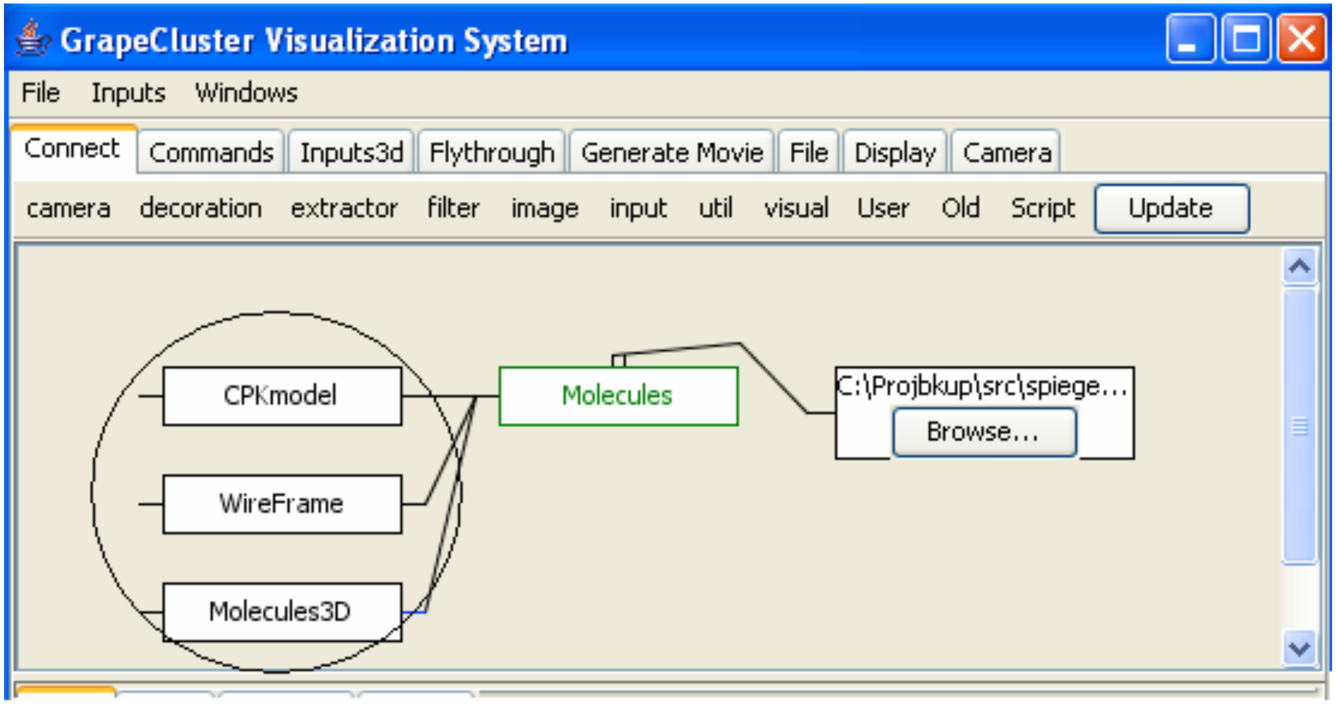
- Output from the representations mentioned above are added to a **BranchGroup** which represents the Content BranchGroup and is passed on to the next plugin

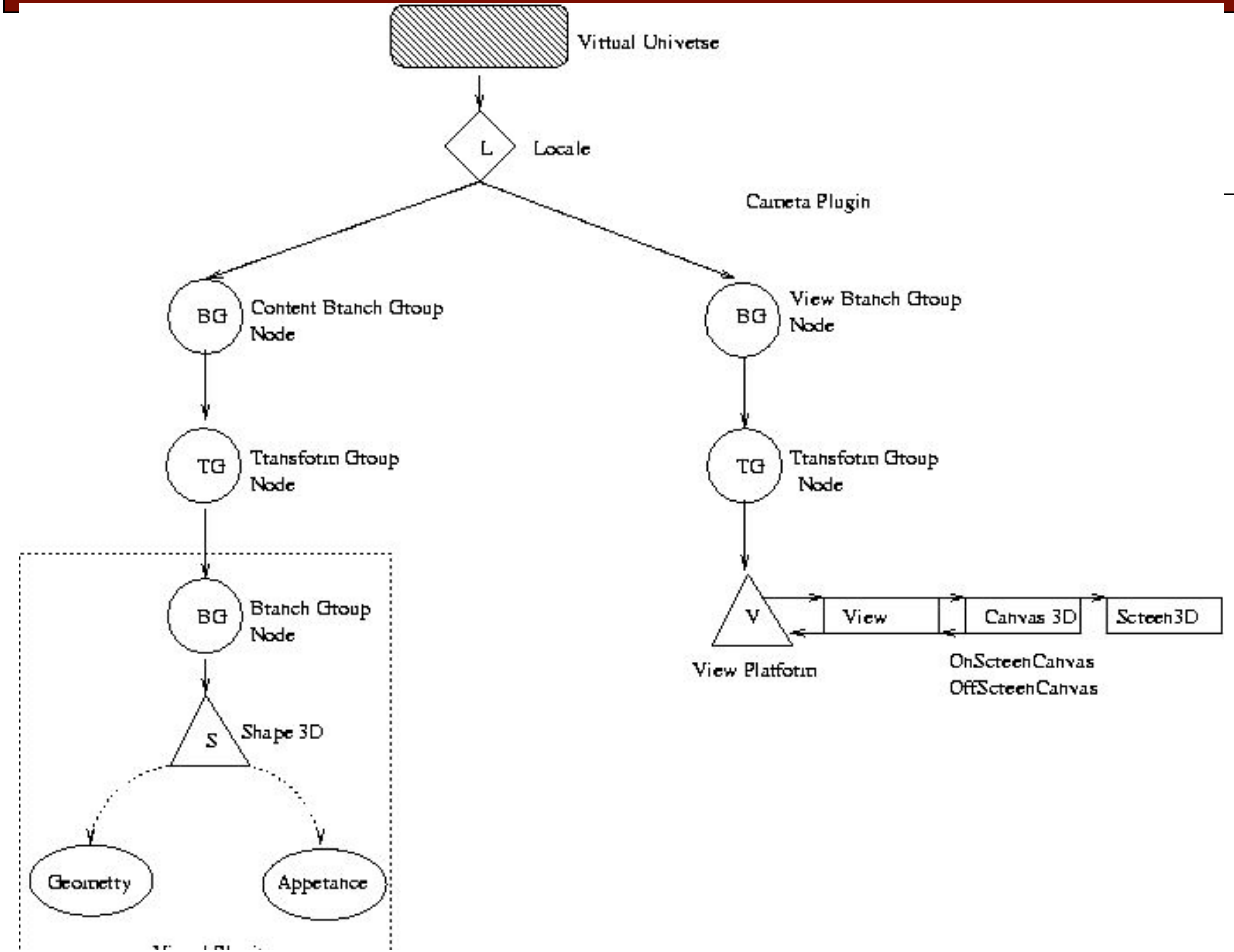
Input and Output variables

- **Input -** HashMap **MoleculeIDMap** from the Extractor or SortedMap **chainIdMap** from Filter plugin

- **Output - BranchGroup** object

Representation of Spiegel




Virtual Universe
L
Locale
Camera Plugin
BG
Content Branch Group Node
BG
View Branch Group Node
TG
Transform Group Node
TG
Transform Group Node
BG
Branch Group Node
V
View
Canvas 3D
Screen3D
View Platform
OnScreenCanvas
OffScreenCanvas
S
Shape 3D
Geometry
Appearance

Camera plugin

- Displays the final image on the screen and provides control parameters to change the display.

Data Structure

- View **BranchGroup** containing **TransformGroup** which has **ViewPlatform** as its node
- ViewPlatform contains the **Canvas** which is the placeholder for the image and the **View** object contains the viewing transform matrix.
- Two types of renderings based the type of **Canvas** object used **- OnScreen (DirectCanvas**) and **OffScreen (Camera3D)**canvas

Data Structure (cont'd)

- Offscreen canvas -batch rendering
- Onscreen canvas -to incorporate all the interactive features
- Both Content BranchGroup **(from Visual Plugin)** and View BranchGroup are tied together under the **Locale** object and finally the **VirtualUniverse** in Camera Plugin

Input and Output variables

- **Input – BranchGroup** from Visual Plugin

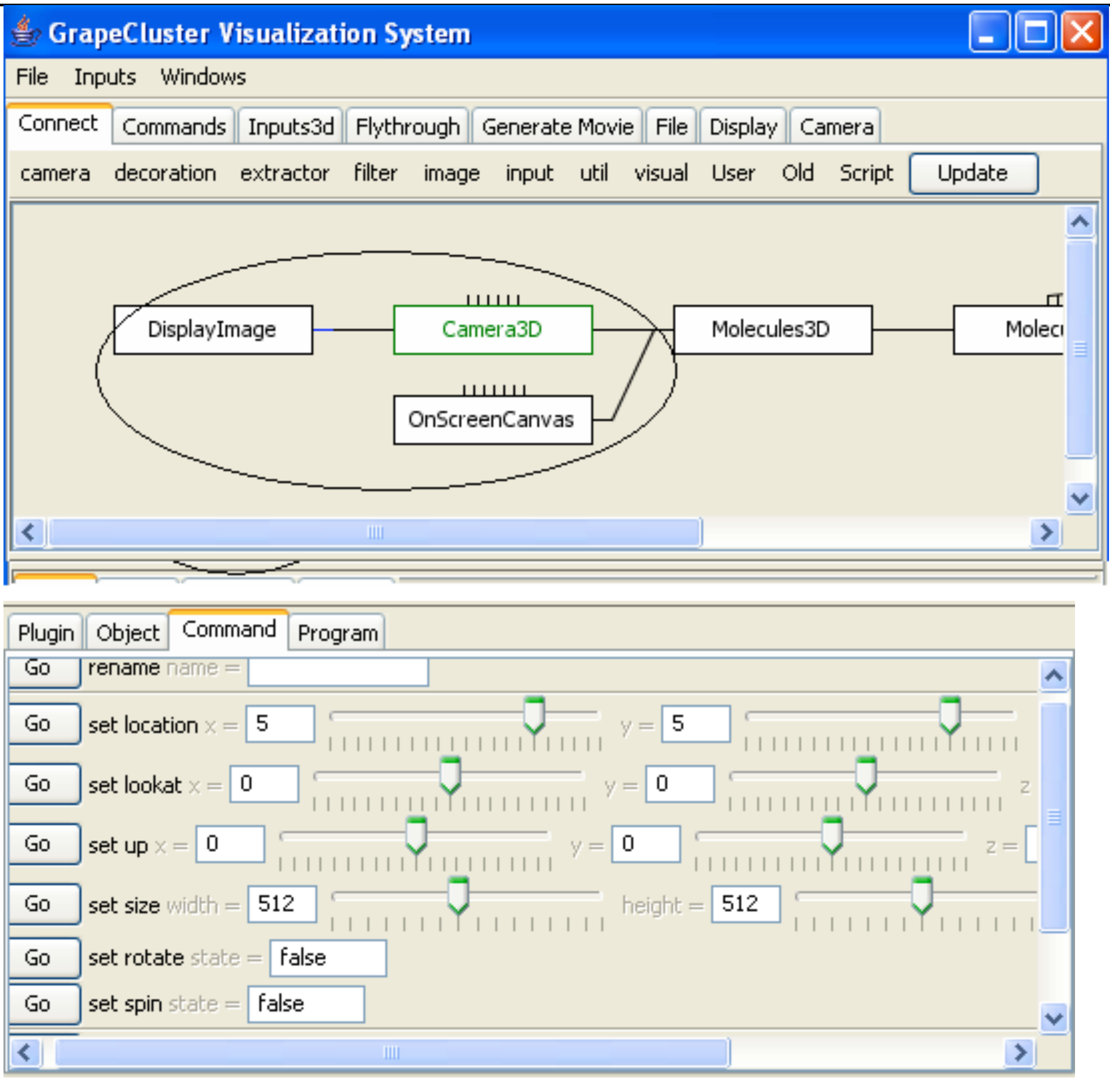
  set location(x,y,z) **,lookat** (x,y,z) and **up** (x,y,z) together form the viewing transform determine how and where the image is displayed on the screen.

  set size (height, width) changes the size of the canvas

  set spin (boolean) starts the animation feature

- **Output** – final image on the Screen

Representation in Spiegel



rename name =

set location x = 5 y = 5

set lookat x = 0 y = 0 z

set up x = 0 y = 0 z =

set size width = 512 height = 512

set rotate state = false

set spin state = false

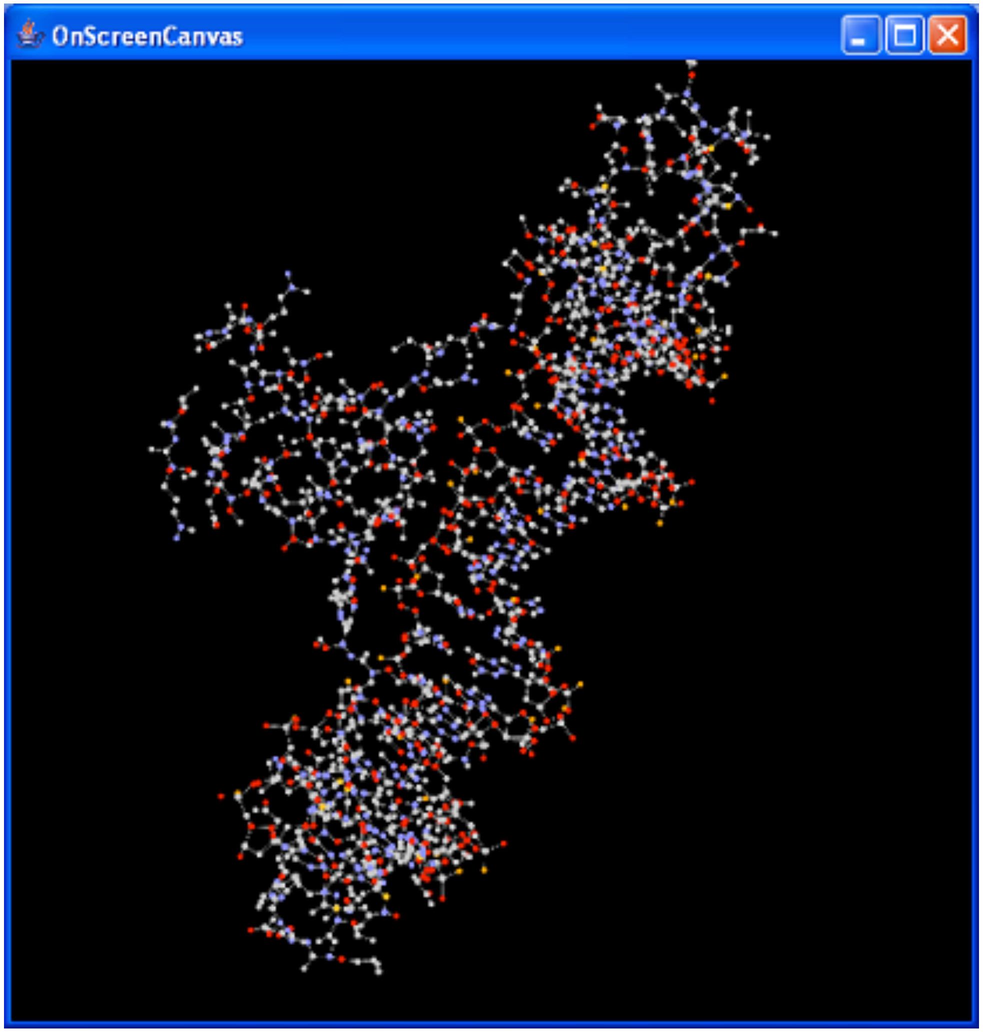
Features added

- **MouseListener** was added to the Onscreen canvas to achieve the following interactive features
  - Rotation - mouse is moved with the left mouse button pressed
  - Translation - mouse is moved with the right mouse button pressed
  - Zoom - mouse is moved with the middle mouse button pressed

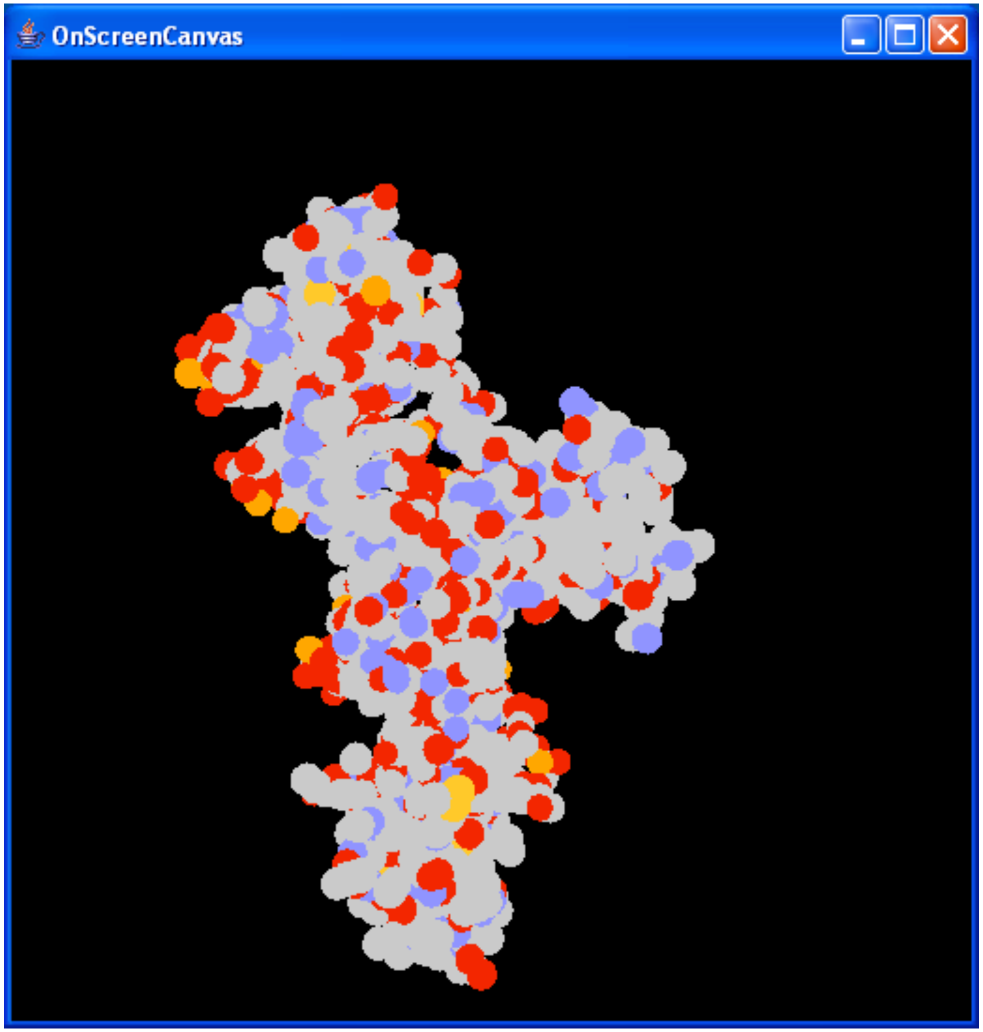
- **Rotation Interpolator** has been added to to achieve animation. When enabled the image completes 360 degrees rotation in 4 seconds

Final Output of Different Models (as seen on the screen)

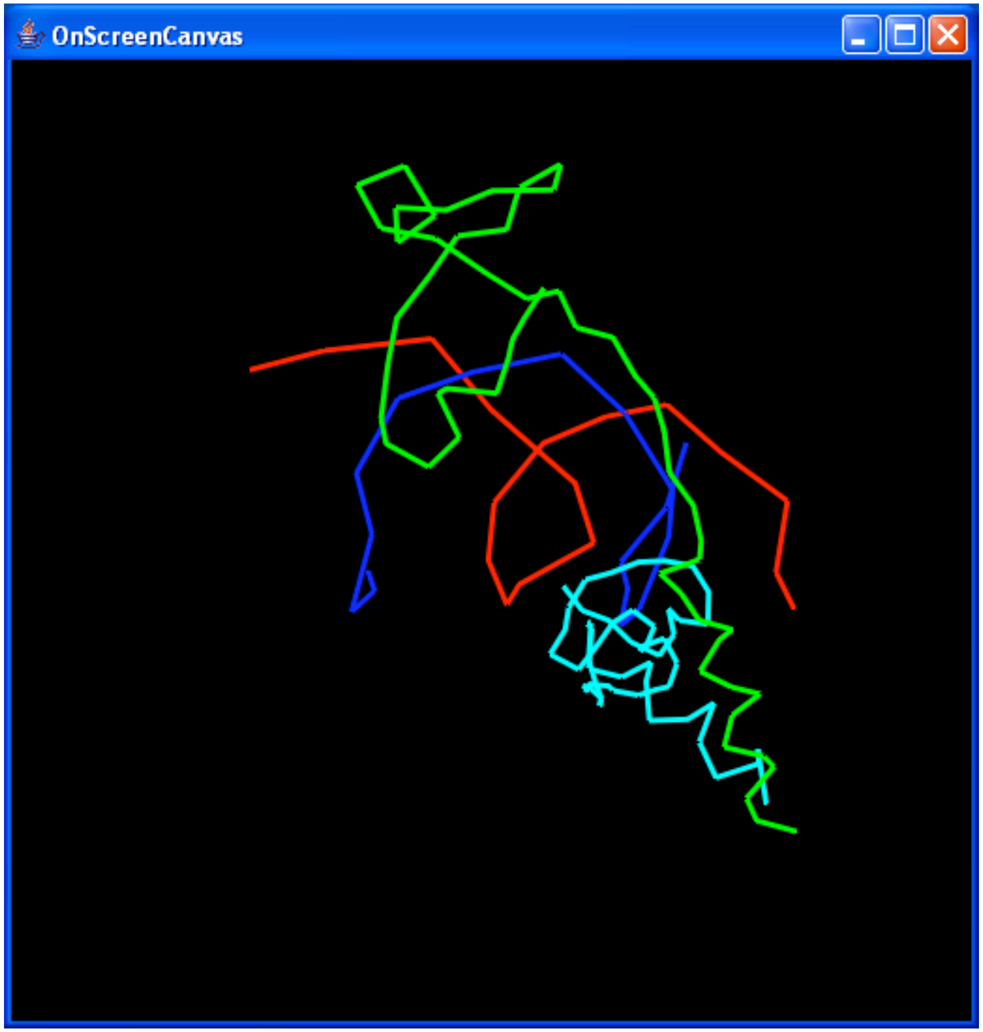
Ball and Stick Model



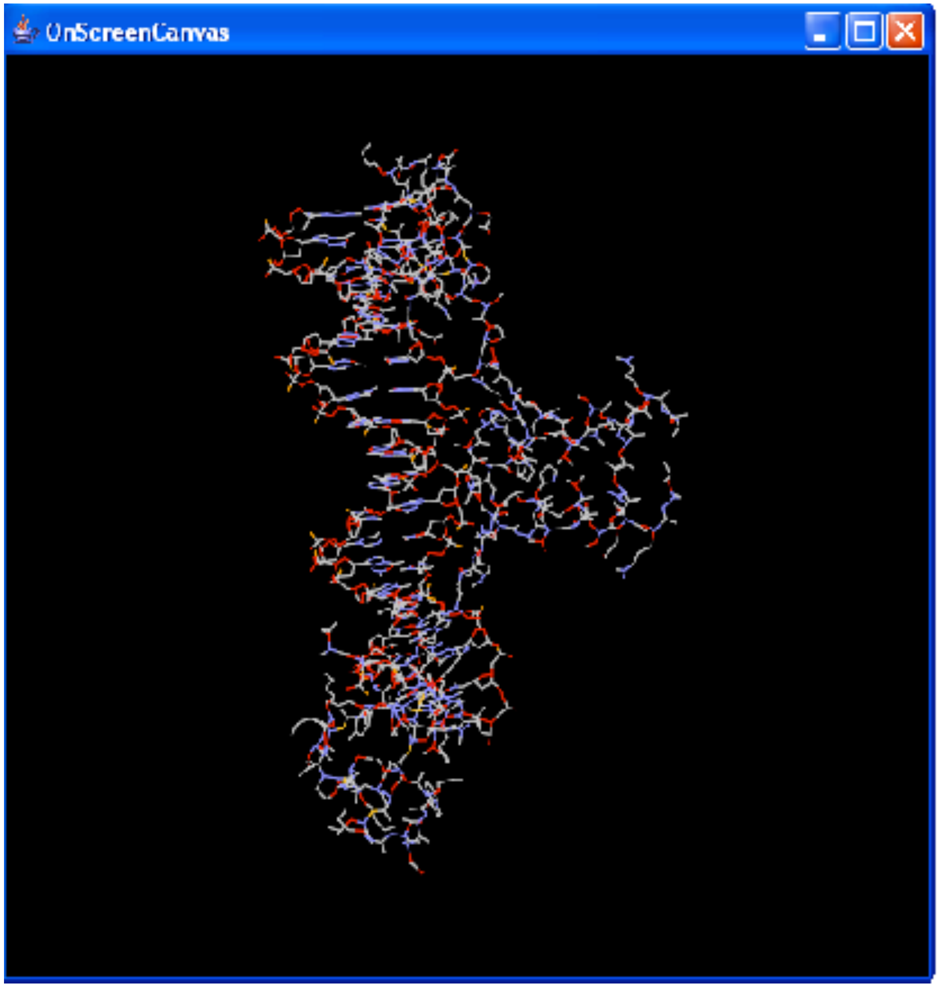
Spacefill Model



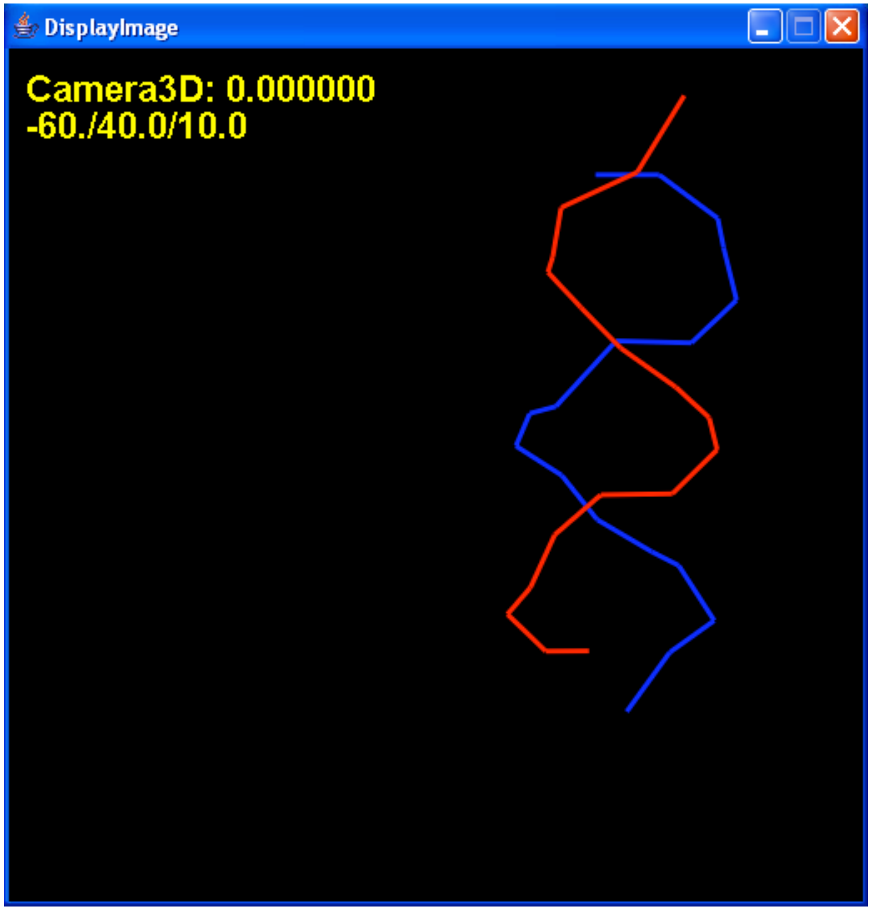
Backbone Model



Wireframe Model



Output from Filter Plugin



Camera3D: 0.000000
-60./40.0/10.0

Sprache Scripts

Spiegel also provides Gui2Script feature which converts the GUI configuration into **Sprache** script that can be executed from the command prompt.

Future Work

- Add support for multiple file types CIF,MOL,etc
- Future dynamic simulation projects could use these models
- Comparison of the models in this project with models created using various bond calculation theories

Q&A