View on GitHub

nezix.github.io

Practical UnityMol tutorial

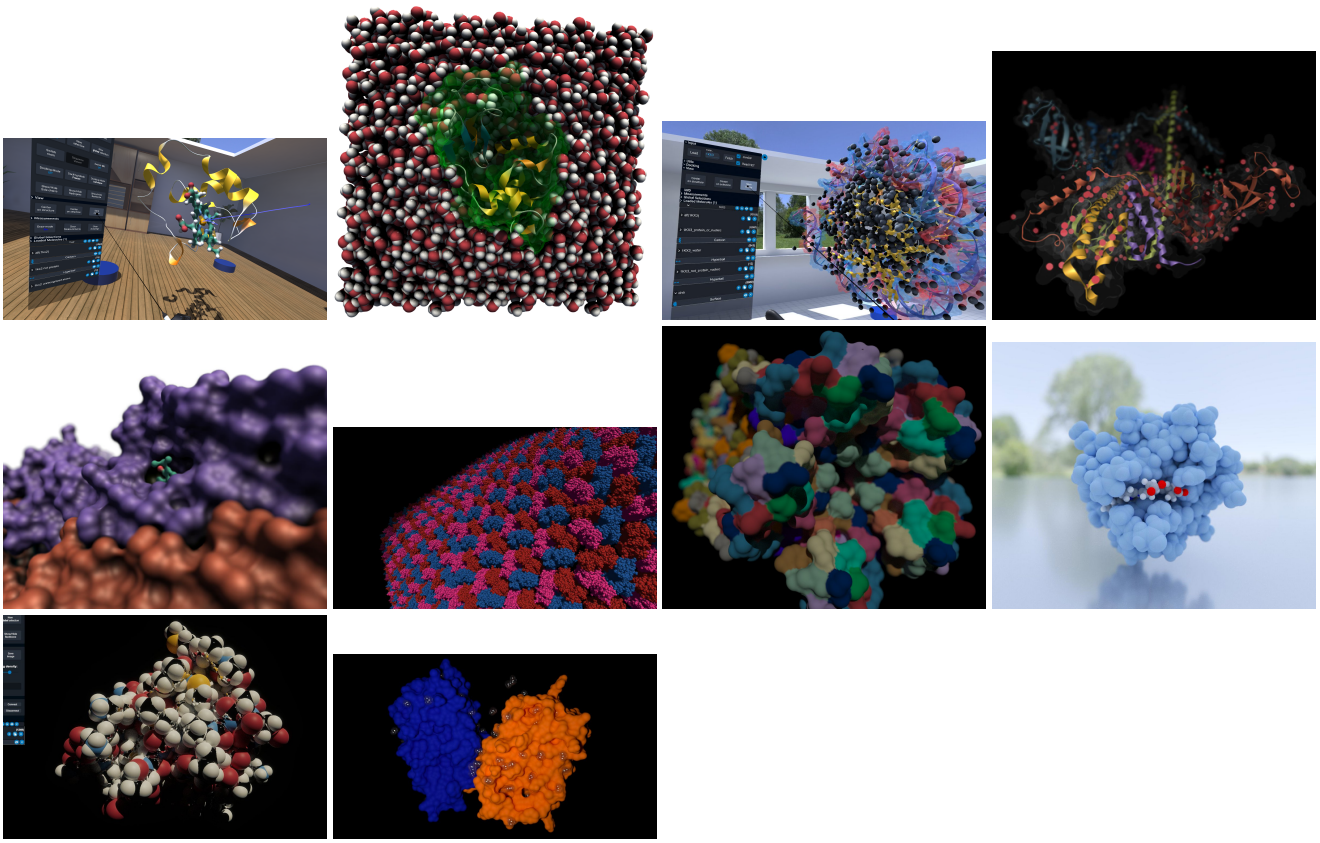
UnityMol Executables can be downloaded here

UnityMol is a molecular viewer that can be used in a standard desktop context or in a Virtual Reality (VR) context. For now, there are one executable for the VR version and one for the desktop version but the same code base is used.

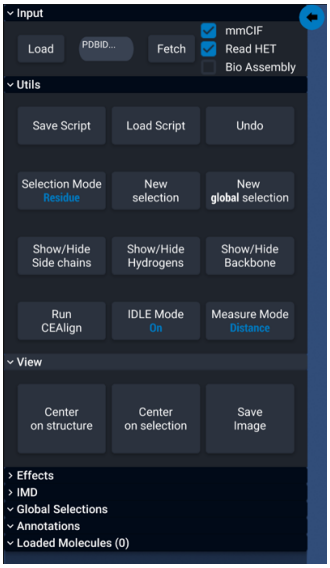
UnityMol is also a molecular visualization platform to prototype and develop new molecular representations and custom interaction metaphores. It is developed mainly in C# in the Unity game engine.

Molecular input files currently supported are: PDB/mmCIF/GRO/Mol2/XYZ/SDF and XTC for trajectory files, OpenDX for potential maps.

This tutorial gives a short introduction to UnityMol 1.1.2 in the form of a simplified documentation / tutorial.

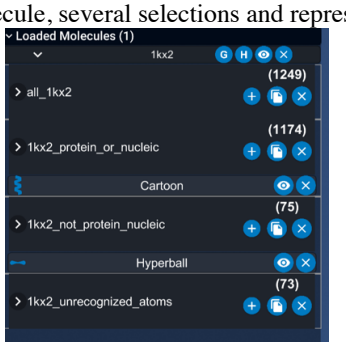


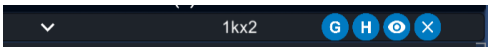
Interface

UnityMol is currently based on one window menu with retractable sub-menus: 

The top right button with an arrow hides the whole menu.

Loaded molecule buttons

When loading a molecule, several selections and representations will be created by default (if the showDefaultRep flag is on for the load and fetch API functions): 

Each loaded molecule will have this line : 

Use the left arrow to hide all the selection buttons of this molecule.

- The 1st button (G) is to define groups to move them together in VR.
- The 2nd (H) is a utility tool to call Reduce (http://kinemage.biochem.duke.edu/software/reduce.php) to add hydrogens to the molecule. This can be slow, the best option is always to manage the PDB file outside of UnityMol.
- The 3rd button hides or shows every representations of the corresponding molecule.
- The 4th is to delete/unload the molecule

Selection / Representation buttons

The selection button can be unfolded to show two input fields :

- The first one is the name of the selection.
- The second one can be used to enter a selection query using the selection language (see Selection language)

The (+) button allows to add representations to the corresponding selection. The (x) button deletes the selection and all the representations of this selection.

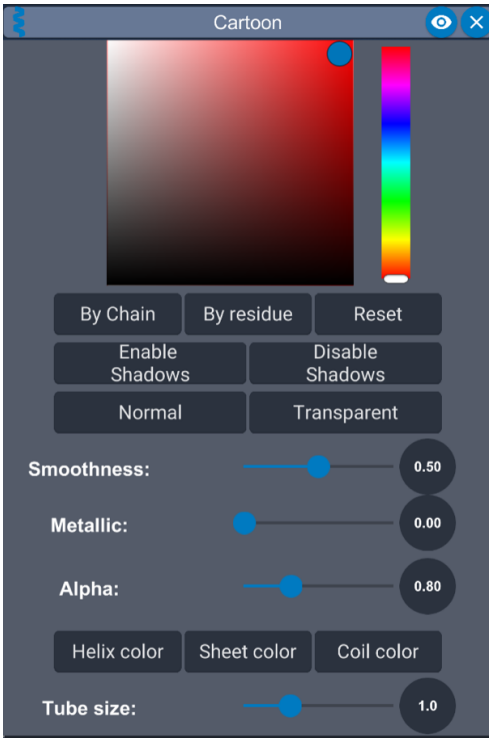
You can set or unset the current selection by clicking on the selection button, it will make the button darker and will show an outline around atoms of the selection. This outline is not updated when reading a trajectory for performance reasons for now.

Representations

First, the eye button hides/shows the corresponding representation. You can also delete a representation using the (x) button.

Note that if you modify the selection, you have to click on the eye button to update the representation.

By clicking on the selection label, the menu unfolds and you can tweak the different parameters of the representation :



Here is a list of representations available in UnityMol:

- **“hb”**: hyperball (licorice/balls&sticks/VDW/smooth)
- **“bondorder”**: bondorder (based on hyperball and uses bond order information to display double/triple bonds on proteins or XML custom bonds)
- **“c”**: cartoon (works for protein and DNA. Martini implementation for protein is still a work in progress)
- **”s”**: surface (EDTSurf and MSMS, QuickSES is a WIP)
- **“dxiso”**: iso-surface (based on opendx maps)
- **“sphere”**: sphere (mesh based, slower than hyperball)
- **“l”**: lines (higher quality when number of atoms is < 10000)
- **“hbond”**: hydrogen-bonds (animated dotted line computed based on shrodinger distance/angle thresholds: distance(H-Acceptor) = 2.8A / angle(Don-H-Acc) = 120 / angle(H-Acc-B) = 90)
- **“hbondtubes”**: hydrogen-bonds tubes (still tube meshes between atoms, same as hbonds)
- **“fl”**: fieldlines (based on opendx maps and its gradient)
- **“trace”**: trace (alpha carbon trace)
- **“sugarribbons”**: sugar representation (WIP)
- **“sheherasade”**: sheherasade representation (WIP, these Loic Nolin)
- **“ellipsoid”**: ellipsoid (using slow hyperball shader)
- **“p”**: points (fast 2D representation)

Python console

To make the python console appear (or disappear), you have to use the `CTRL + <space bar>` key combo or use the top left arrow.

To use it see Python Console

- **Every action done with UI buttons correspond to a function or a set of functions of APIPython**
- **There is a LOT of options that can be done in the console but are not possible from the UI**

The console is using IronPython, a C# interpreter that compiles python but also gives you access to C# objects. This means that you can call C# / Unity functions from the console like: `GameObject.Find("Directional Light")`

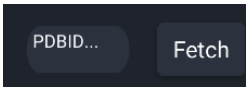
Getting started

Loading

Let's load a molecule, do some selections, create some representations and modify their properties.

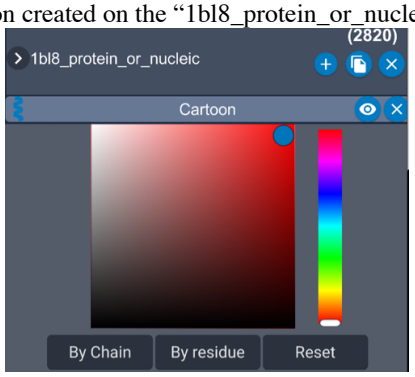
In the console, type :

```
s = fetch("1bl8")
print(s)
```

You can also use the PDBID input field of the menu in the Input sub-menu: 

This should show you something close to this: 

Coloring

Now let's change the color of the cartoon representation created on the "1bl8_protein_or_nucleic" selection by clicking on the cartoon line to expand the options and clicking on "By chain" button: 

This is equivalent to do `colorByChain("1bl8_protein_or_nucleic", "c")` in the console.

You can also color the full representations using the color picker in the menu that is equivalent to calling `colorSelection("1bl8_protein_or_nucleic", "c", Color.red)`.

UnityMol provides a lot of different color functions that are not always available in the UI:

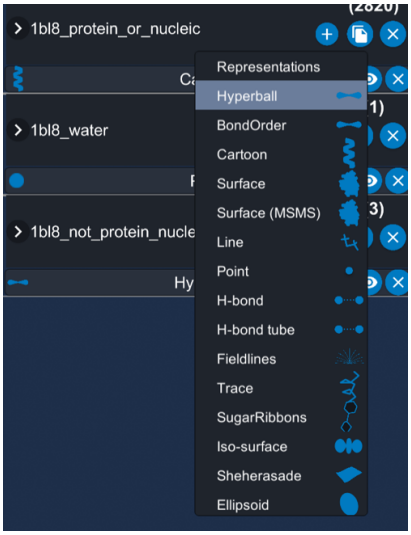
```
colorByAtom("selName", "c")
colorByResidue("selName", "c")
colorByChain("selName", "c")
colorByHydrophobicity("selName", "c")
colorBySequence("selName", "c")
colorByCharge("selName", "c")
colorByResidueType("selName", "c")
colorByResidueCharge("selName", "c")
colorByBfactor("selName", "c")
```

To modify default values for UnityMol colors, you can modify the file that is loaded when UnityMol starts: "StreamingAssets/customUnityMolColors.txt". This files overwrites "defaultUnityMolColors.txt" that is also loaded at start.

The format is: `AtomName #HexadecimalColor radius`. For example: `H #ffffff 1.2`

Adding representations to selections

We can add a "all-atoms" representation by adding the "Hyperball" representation: click on the (+) button of the "1bl8_protein_or_nucleic"

selection button and click on “Hyperball” in the dropdown menu: 

This is equivalent to do `showSelection("1bl8_protein_or_nucleic", "hb")` in the console.

Let’s color carbon atoms in the hyperball representations in gray: `colorAtomType("1bl8_protein_or_nucleic", "hb", "C", Color.gray)`.

Show/hide buttons for representations are calling `showSelection("selName", "repType")` and `hideSelection("selName", "repType")` functions. Delete buttons are calling `deleteRepresentationInSelection("selName", "repType")` or `deleteRepresentationInSelection("selName")`.

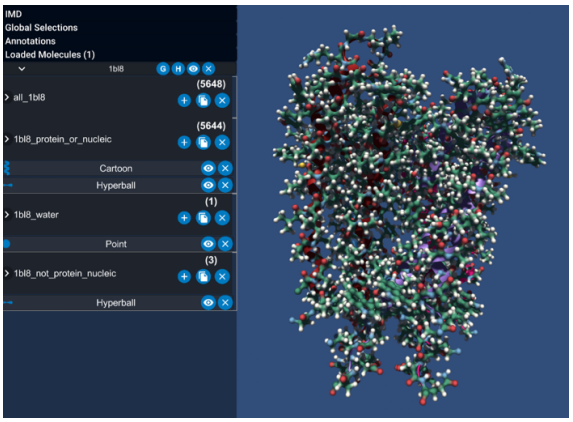
To hide all the representations of a selection you can call `hideSelection("selName")`.

To hide all representations of a structure you can call `hideStructureAllRepresentations("structureName")`, the opposite is `showStructureAllRepresentations("structureName")`.

You can also hide representations in all the selections based on the representation type: `hide("c")` will hide all cartoon representations.

Adding hydrogens with Reduce

Let’s add hydrogens to this molecule with the Reduce method by clicking on the “H” button next to the name of our protein. This is equivalent to do `addHydrogensReduce("1bl8")` or if you don’t know the name of the molecule you can use the ‘s’ variable storing the UnityMolStructure when we loaded the molecule :`addHydrogensReduce(s.name)`. We can also use the last loaded molecule by doing: `addHydrogensReduce(last().name)`.

This should update the representations to this: 

Note that the number of atoms in some selections has changed.

Zooming / Center

To zoom on a part of the molecule, you can double click on one atom of the residue if it is displayed. You can also double click the selection button (since UnityMol 1.1.1). From the console, you can do `centerOnSelection("1bl8_water", lerp=True)`.

You can also click on a selection to make it the “current selection” and click on the “Center on Selection” button under the “View” menu.

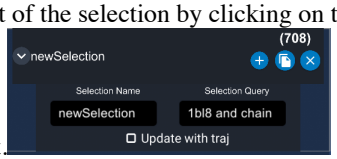
Translation / Rotation (desktop)

On desktop, you can use the mouse button to rotate all molecules with the left button, zoom in/out with the right button and translate.

If several molecules are loaded, you can only rotate one molecule by clicking on “R” + left button. To only translate a molecule press “T” and left click on the molecule.

Selection language

Let’s select only a part of the molecule and display it as a transparent surface. There are several ways to create selections:

- duplicate one by clicking on the copy/paste button  + edit the content of the by clicking on the left arrow to expand the selection options and enter `1bl8 and chain A` in the selection query part.
- click on the “New selection” button in the “Utils” menu + edit the content of the selection by clicking on the left arrow to expand the selection options and enter `1bl8 and chain A` in the selection query part. 

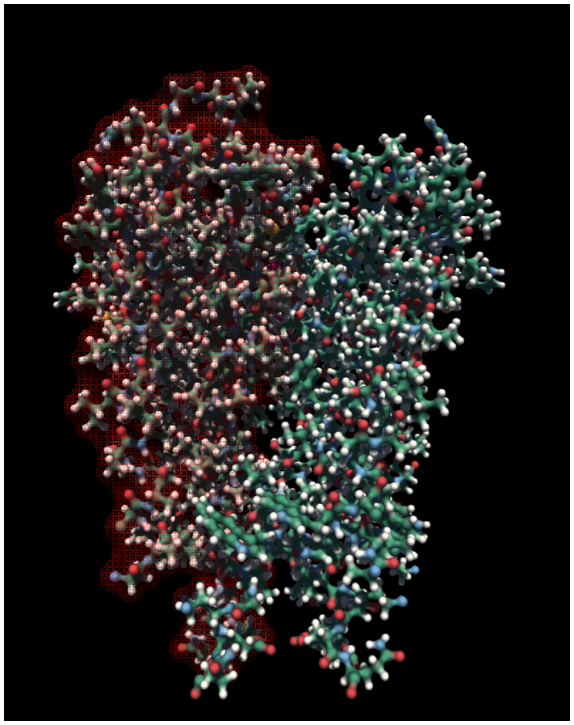
- use the console to directly create the selection by doing `select("1bl8 and chain A", "chainA")`

Now let’s show the newly created selection as a surface. Click on the (+) button and select Surface (MSMS) or enter `showSelection("chainA", "s", SurfMethod.MSMS)`.

The surface can be displayed as a wireframe or with a transparent material by clicking on the surface label to expand the properties sub-menu then clicking on the “Transparent” button. You can also enter `setTransparentSurface("chainA")` or `setTransparentSurface("chainA", 0.30)` in the console.

Let’s color the surface in red: `colorSelection("chainA", "s", Color.red)` or `colorSelection("chainA", "s", Color(1.0,0.0,0.0,1.0)`

Hide the cartoon representation with `hideSelection("1bl8_protein_or_nucleic", "c")` and change the background color with `bg_color(Color.black)` or `bg_color("black")` or `bg_color(Color(0,0,0,1))`

You should have something close to this: 

For a better overview of all the possibilities of the selection language please see here and here.

Select in range

To select atoms around 5.0 Angstrom of the water molecule of 1BL8 type:

```
select("not water and within 5.0 resname HOH", "closeToWat")
showSelection("closeToWat", "l")#show this selection as lines
```

You can now re-use this selection to create new selections:

```
select("not closeToWat", "AllButCloseToWat")
```

Saving command history

To save the current selections and representations to be able to share it or reload the commands you can click on the “Utils” -> “Save script” button. This will save all the commands that we executed, using the UI or the console. Some optimization work can be done to clear unnecessary commands by editing this file by hand.

You can also call the `saveHistoryScript("mypath.py")` command from the console to save the history of commands.

You can reload the file containing a list of commands by doing `loadHistoryScript("mypath.py")` or clicking on the “Load Script” from the “Utils” menu.

Taking pictures/videos

You can take a screenshot by clicking on the “Save image” button in the “View” menu. This will create a folder called UnityMolScreenshort on your desktop and save the image to it with a 1080p resolution. You can also use `screenshot("mypath.png", 1920, 1080, transparentBG=False)` command.

Videos can be taken via the command `startVideo("mypath.mp4", 1920, 1080, 60)` and stopped with `stopVideo()`. To have an example of a script to render an animation take a look at this script.

Please see this page for more information.

Scripting

The UnityMol console combined to the python API provides a way to perform simple actions but also complicated and convoluted scripts.

For example, let’s clean all loaded molecules by doing `reset()` and load 3EAM without displaying a representation `fetch("3eam", showDefaultRep=False)`.

We are going to create a selection by chain and color each chain.

```
s = last() #Get the lastly loaded structure
colors = [Color(1,0,0,1), Color(1, 1, 0, 1), Color(1,1,1,1), Color(0,1,1,1), Color(0,0,1,1)] #Create a list of colors
idC = 0
for c in s.currentModel.chains.Values: #Loop over the chains of the molecule
    sel = select(s.name+" and chain "+c.name, "Chain"+c.name) #Create a selection for the chain
    showSelection(sel.name, "c") #Show the selection as a cartoon
    colorSelection(sel.name, "c", colors[min(idC, len(colors)-1)])
    idC+=1
```

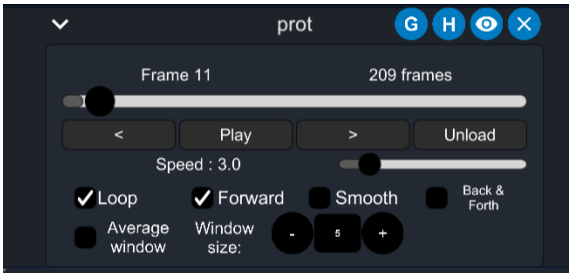
For a complete set of all API commands please read this page !

Trajectory / Model Player

UnityMol supports loading XTC trajectories using libxdrfile2. The first time a trajectory is loaded, a binary .offset file is created to store the position of each frame in the file. This allows to “jump” in the file **without loading the whole trajectory in memory**.

To load a trajectory, press the “Load” button from the Input menu and set the file filters to “Trajectory files” or use: `loadTraj("structureName", "mypathToXTCFile.xtc")`.

By default, UnityMol loads the trajectory on the last loaded molecule when using the “Load” button.

This sub-menu should be displayed under the loaded molecule UI row: 

A similar menu is displayed when a molecule with several models is loaded. There are two ways to load a molecular file with several models. By default, UnityMol parses a PDB/mmCIF/GRO file with `modelsAsTraj = True` meaning that the first model is read and only the new positions of the next frames are stored. The second way is to really store each model in a different UnityMolModel.

UnityMol allows to have models with different number of atoms for the same molecule only if the `modelsAsTraj` flag is false.

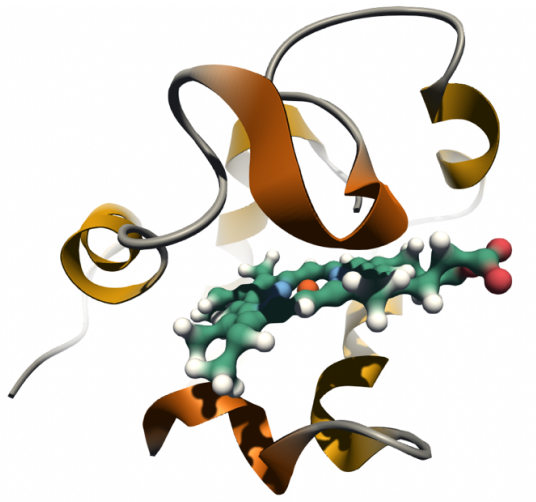
The smooth button interpolates between 2 frames using a linear interpolation. Cubic interpolation uses 4 frames to interpolate between n-1 to n+2 frames. `last().xdr.cubicInterpolation = True`.

The average window button activates averaging all the frames between the current frame n and the n+WindowSize frames.

Note that smoothing and averaging should be fast because it's done using C# Burst compiler, multi-threaded and SIMD accelerated computation.

Effects (desktop)

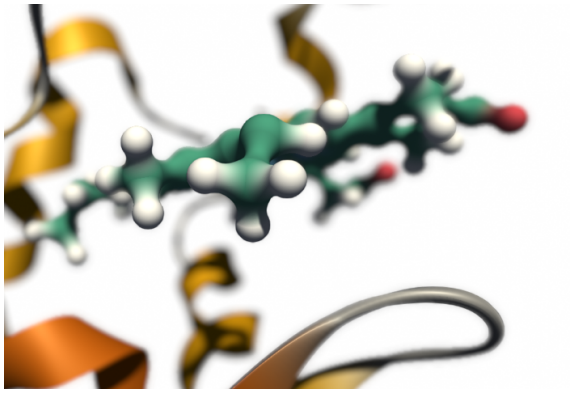
Currently, UnityMol has 3 effects you can activate in desktop mode:



- Depth cueing / Fog :

Related functions are

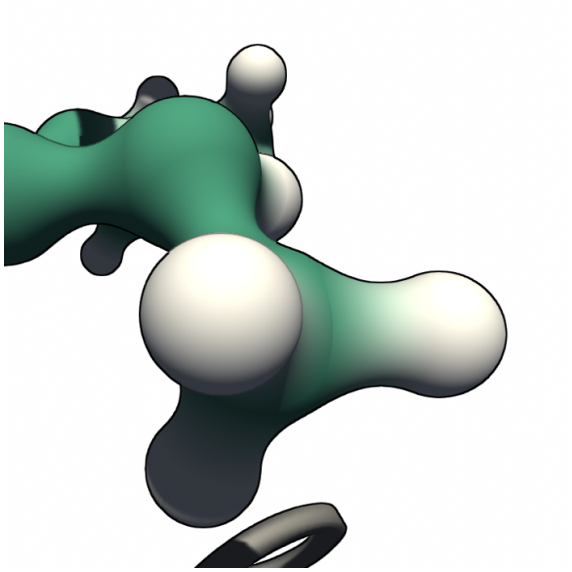
```
enableDepthCueing()
disableDepthCueing()
setDepthCueingStart(0.5)
setDepthCueingDensity(0.1)
```

- Depth of field / Bokeh effect: 

Related functions are

```
enableDOF()
disableDOF()
setDOFAperture(0.1)
setDOFFocalLength(0.1)
setDOFFocusDistance(0.1)
```

Note: when the DOF effect is activated, clicking on a molecule does not trigger the selection but changes the DOF focus distance.

- Outline : 

Related functions are

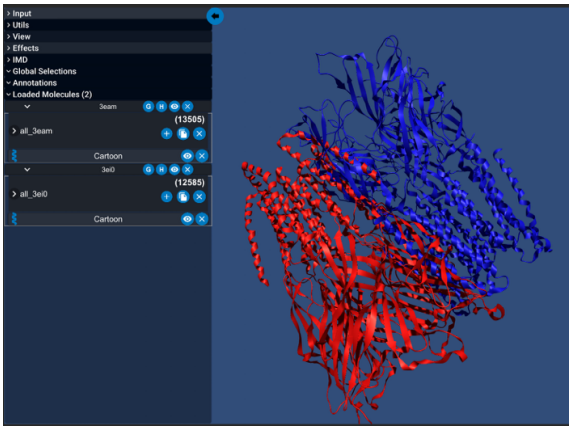
```
enableOutline()
disableOutline()
setOutlineThickness(10.0)
```

Aligning 2 proteins

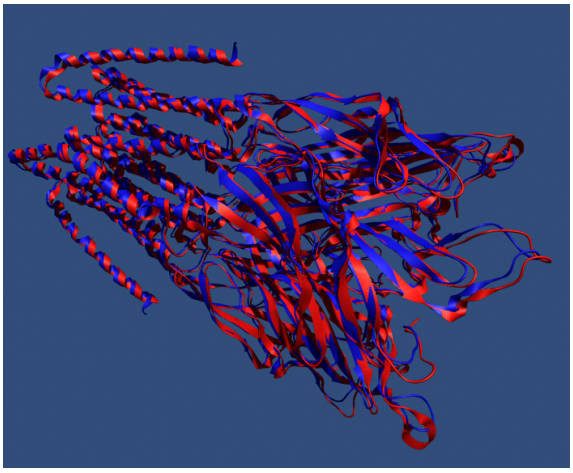
UnityMol uses CEAlign from here, an algorithm that performs well for low sequence similarity and still fast. It only uses alpha carbons to align the second argument (mobile) of the function `cealign` on the first argument of the function (target).

As an example, load 2 proteins and display them as cartoon:

```
s1 = fetch("3eam", showDefaultRep=False)
s2 = fetch("3ei0", showDefaultRep=False)
show("c")
colorSelection(s1.ToSelectionName(), "c", Color.red)
colorSelection(s2.ToSelectionName(), "c", Color.blue)
```

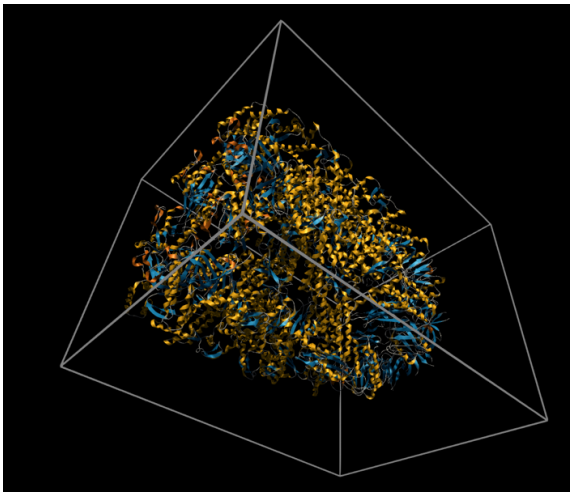
You should have something like this: 

Clicking on “Run CEAlign” from the Utils menu applies CEAlign on the 2 lastly loaded molecules. This is equivalent to `cealign(s1.ToSelectionName(), s2.ToSelectionName())`.



Bounding boxes

You can show the bounding box of the loaded molecule by doing `showBoundingBox("structureName")` / `hideBoundingBox("structureName")`.



Overwrite bonding information (PSF / TOP / XML)

PSF

You can load bounding information from a PSF file (NAMD protein structure file)

```
loadPSFTopology("structureName", "mypathToPSFfile.psf")
```

TOP

Similarly, you can load a TOP file (Gromacs topology file) to overwrite bounding information and update all representations:

```
loadTOPTopology("structureName", "mypathToTOPfile.top")
```

XML

UnityMol provides a simple way to load covalent and non-covalent bonds using a XML format:

```
<Bonds>
<Bond from="0" to="1" order="2.0" type="covalent"/>
</Bonds>
```

This links the first atom of the loaded molecule in the order of the loaded file to the second one by a double covalent bond.

Possible bond types are: `covalent` or `db_geom`, `hbond` or `h-bond` or `hbond_weak`, `halogen`, `ionic`, `aromatic`, `hydrophobic`, `carbonyl`.

```
loadBondsXML("structureName", "pathToXMLFile.xml")
overrideBondsWithXML("structureName")
unloadCustomBonds("structureName", modelId)
```

Restoring UnityMol computed topology

After loading a topology from a PSF/TOP/XML file, you can restore the bonds computed by UnityMol at load time by doing:

```
restoreBonds("structureName")
```

Secondary structure assignation

When loading a protein with `load()` or `fetch()` the default behaviour is to use the secondary structure information from the file and use DSSP algorithm when there is none. You can force DSSP to be run: `fetch("1kx2", forceDSSP=True)`.

You can run DSSP on a loaded molecule by doing: `switchSSAssignmentMethod("structureName", forceDSSP=True)`.

Annotations

There are several types of annotation in UnityMol to add labels, objects, lines, measurements to atoms. To add a transparent sphere around an atom: `annotateAtom("structureName", atomId)` To add a transparent sphere not attached to an atom, use `annotateSphere(worldP=Vector3(0,0,0), scale=1.0)`

You can also add some label to an atom: `annotateAtomText("structureName", atomId, "My text")`, or not attached to an atom: `annotateWorldText(Vector3(0,0,0), 1.0, "My text", Color.white)`. To add text in 2D, you can use `annotate2DText(screenP=Vector2(0.5, 0.5), scale=1.0, text="My text", textCol=Color.white)` where the screen position is defined by (0,0) = bottom-left and (1,1) = top-right.

To add a line: `annotateLine("structureName", atomId, "structureName2", atomId2)` and not attached to an atom: `annotateWorldLine(Vector3(0,0,0), Vector3(3,0,0), 1.0, Color.white)`.

Measurements

Measuring is using annotations to display lines and text on atoms: In desktop mode, press “M” and click on atoms to compute a distance/angle/torsion angle based on `UnityMolMain.measureMode` that can be changed using the command `setMeasureMode(mode)` where mode is 0 = distance, 1 = angle, 2 = torsion angle. Clicking on the “Measurement mode” button is calling `setMeasureMode()` function.

In VR, pressing the touchpad button while aiming the target atom should trigger measurements.

You can also play a sonar sound at a position to localize an atom or an object using the `playSoundAtPosition(Vector3(10,0,0))` command.

Interactive Molecular Dynamics (IMD)

Connection to a running simulation using the IMD protocol is done via the IMD menu. You can also call `connectIMD("structureName", "127.0.0.1", 8888)` for a simulation running on the localhost on the 8888 port. Disconnection using the console:

`disconnectIMD("structureName")`.

By default, UnityMol uses the last loaded molecule when clicking on IMD connect button.

During IMD, you can grab an atom or several atoms by click & dragging the mouse in desktop or the controller in VR.

The IMD protocol is implemented using MDDriver since UnityMol 1.1.2.

Docking mode

UnityMol team collaborated with Levitt's lab at Stanford University to build a docking mode in UnityMol, available in VR. It uses the AMBER forcefield to interactively compute the potential energy of the system. For more information please read this page.

Export to OBJ/FBX

One can export the current representations of a structure using one of these functions: `exportRepsToOBJFile("structureName", "outputpath.obj")` or `exportRepsToFBXFile("structureName", "outputpath.fbx")`.

- Note that this currently ignores point, hbond and bond-order representations.
- For hyperball representations, a custom algorithm extracts a mesh based on current hyperball parameters, this can lead to a mesh slightly different from the actual representations in UnityMol. It also produces large meshes.

The FBX format file is a binary file format that is lighter than the OBJ text file format.

Startup commands

You can directly load a molecular file or a command file (previously saved from UnityMol or a custom script) when launching UnityMol from a terminal:

```
$> ./UnityMol -i /path/to/UMolState.py
```

```
$> ./UnityMol -i /path/to/1kx2.pdb /path/to/1bl8.pdb /path/to/1bl8_traj.xtc
```

To fetch a PDB when starting UnityMol you can do:

```
$> ./UnityMol -i 3eam
```

You can also set the directory to easily load local files with

```
$> ./UnityMol -d /path/to/folder
```

More information

If you need help or have any question regarding UnityMol and it's usage, please send an email to unitymol@gmail.com. We are also active on Twitter: @UnityMol

Published with GitHub Pages