I know what you did last summer: GSOC

Malitha Kabir, Greg Landrum, Paul Czodrowski
RDKit UserGroupMeeting, Berlin, September 20th 2017

MERCK

How does GSoC work?

- Organizations associated with open-source projects (potentially multiple projects) apply to Google to participate.
  In our case: OpenChemistry, organized by Geoff Hutchinson and David Koes

- Once selected, organization post project ideas along with mentors for those projects
  http://wiki.openchemistry.org/GSoC_Ideas_2017

- Students express interest in projects and submit applications

- The organization ranks the student applications

- Google decides how many of those highly-ranked students to fund for each project

- Work starts

GSoC: Project proposal

There were a number of RDKit proposals made as part of the OpenChemistry organization. This is the one we foun
http://wiki.openchemistry.org/GSoC_Ideas_2017#Project:_RDKit_-_3Dmol.js_Integration

Project: RDKit - 3Dmol.js Integration

Brief explanation: 3Dmol.js (http://3dmol.csb.pitt.edu/) is a JavaScript library for visualizing molecular data. The goal of this project is to enable ligand modifications of a protein-ligand complex (http://www.nature.com/nprot/journal/v11/n5/fig_tab/nprot.2016.051_F1.html)

Expected results: Python functionality allowing RDKit molecules to be sent to the force fields available in the RDKit to perform ligand modification and energy minimisation inside the binding pocket. Integration of this with a Jupyter-notebook UI based on 3Dmol.js

Prerequisites: Python and Javascript

Mentor: Paul Czodrowski (paul.czodrowski at merckgroup dot com)

Short documentation of Malitha's work (static HTML)

https://github.com/malithakabir/RDKitGSoC2017/blob/master/updateAfterGSoC/README.md

Pull request link (updated on 12th September 2017):

- https://github.com/rdkit/rdkit/pull/1562

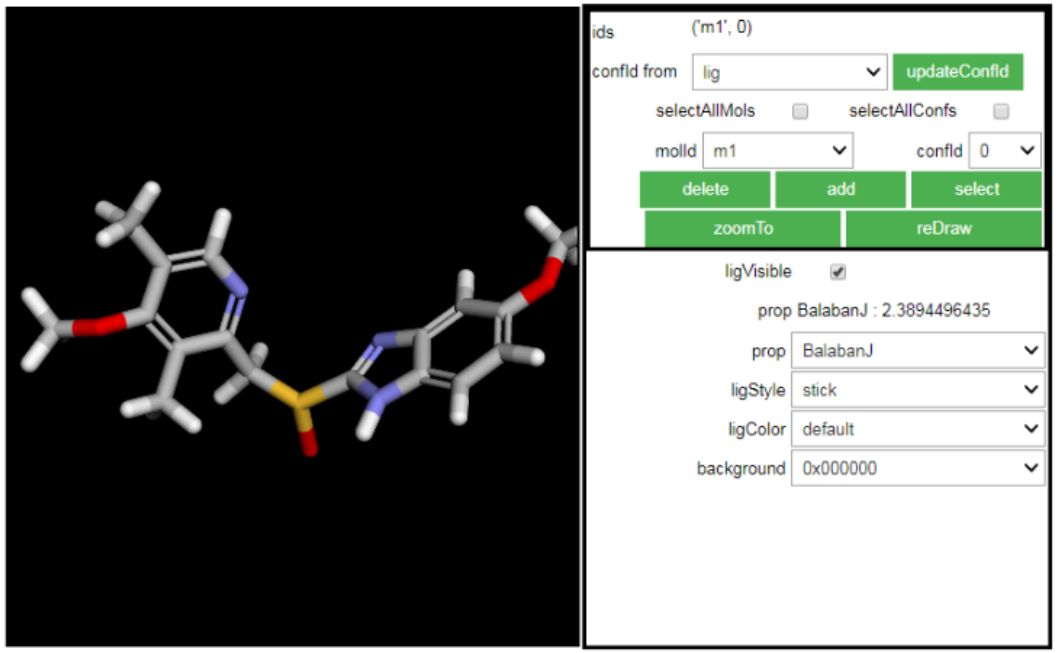
Gallery

notebook 1

```
view = ipy.MolView3D(ligandDict = moldict_2, protein = nm.rest)
```

notebook 5

```
view = ipy.MolView3D(ligandDict = moldict, propertyPanel = True)
```



Acknowledgements

- Malitha Kabir
- Peter Gedeck
- Brian Kelley
- David Koes
- Geoff Hutchison & openchemistry.org
- Google

More details

High level view: `IPythonConsoleIntegration.py`

```python
from IPython.display import HTML as scriptHTML
import time
import sys
import copy

PY3 = sys.version_info[0] >= 3

BGCOLORS_3D = ('0x000000', '0xeeeeee', '0xffffff')

PROP_RDKIT = tuple(sorted(prop for prop, _ in Descriptors._descList))

DRAWING_LIGAND_3D=('line', 'cross', 'stick', 'sphere', 'surface', 'ballstick')

DRAWING_PROTEIN_3D=('line', 'cartoon', 'surface')

LIGAND_COLOR_SCHEME_3D=('default', 'greenCarbon', 'cyanCarbon', 'magentaCarbon',

def Check3Dmolpromise():


# I think this function is not required
# User should be able to supply appropriate input
# A notebook showing 3Dmol visualization should be present in rdkit doc
def ProcessLigandDict(ligandDict, keyForParentMol = 'parent'):

def MinimizeLigand(ligandDict,


def AddPropToLigandDict(ligandDict, keyForParentMol = 'parent'):

class MolViewState(object):




class MolView3D(object):
```

Major parts

1. core calculation engine (moldict creation that includes property and minimization)
2. molstate class for handling selected molecule and conformers
3. visualization class

Molecule dictionary formatting for MolView3D function

```python
def ProcessLigandDict(ligandDict, keyForParentMol = 'parent'):
    """This function adds another key to the dictionary of the molecule."""

    if isinstance(ligandDict, dict) is False:
        raise TypeError("ligandDict must be a dictionary")

    keys = list(ligandDict.keys())
    firstKey = keys[0]

    if isinstance(ligandDict[firstKey], dict):
        raise TypeError("ProcessLigandDict doesn't support nested ligandDict")

    newLigandDict = {}

    for molId in keys:
        newLigandDict[molId] = {}
        newLigandDict[molId][keyForParentMol] = ligandDict[molId]

    return newLigandDict
```

MolView3D input arguments

```
class MolView3D(object):

    def __init__(self,
                 ligandDict = None,
                 protein = None,
                 keyForParentMol = 'parent', keyForMinimizedMol = 'minimized', energyDataKey = 'energy',
                 ligStyle = 'stick', protStyle='cartoon', emLigStyle = 'stick',
                 ligSelPanel = 'full',
                 stylePanel = None,
                 labelPanel = False,
                 propertyPanel = False,
                 emPanel = False):
        """This function initiates required widgets and 3Dmol.js viewer"""
```

How a molecule dictionary looks like for MolView3D

```
moldict_2_em = ipy.MinimizeLigand(ligandDict = moldict_2, molAndConfIds = 'allConfs', maxIters = 100)
moldict_2_em
```

```
{'0': {'energy': {0: 58.633313732423886,
   1: 44.20194969784161,
   2: 73.35949543197519,
   3: 63.213297694165654,
   4: 58.45552390009194,
   5: 64.46032077676604,
   6: 54.410833986384276,
   7: 69.46089943696155,
   8: 61.58846805801912,
   9: 48.59804768491881},
  'minimized': <rdkit.Chem.rdchem.Mol at 0x7f87c56d7440>,
  'parent': <rdkit.Chem.rdchem.Mol at 0x7f87c56d73d0>}}
```

Add property to molecule

```python
def AddPropToLigandDict(ligandDict, keyForParentMol = 'parent'):
    """ Add property to the mol """
    for molId in ligandDict:
        mol = ligandDict[molId][keyForParentMol]
        for prop_name in PROP_RDKIT:
            calculator = Descriptors.__dict__[prop_name]
            mol.SetProp(prop_name, str(calculator(mol)))

    return ligandDict
```

```python
moldict_2 = ipy.AddPropToLigandDict(ligandDict = moldict)
moldict_2
```

```
{'0': {'parent': <rdkit.Chem.rdchem.Mol at 0x7f4d5b27a3d0>}}
```

Ligand extraction

```
nm = LigandExtract.ExtractMolFragment(Mol,ResName = 'IRE')
nm
```

```
ExtractResult(match=<rdkit.Chem.rdchem.Mol object at 0x7f4d5b27a3d0>, rest=<rdkit.Chem.rdchem.Mol obje
ct at 0x7f4d5b27a280>)
```

```
nm.match
```

```
<rdkit.Chem.rdchem.Mol at 0x7f4d5b27a3d0>
```

```
nm.match.GetNumConformers()
```

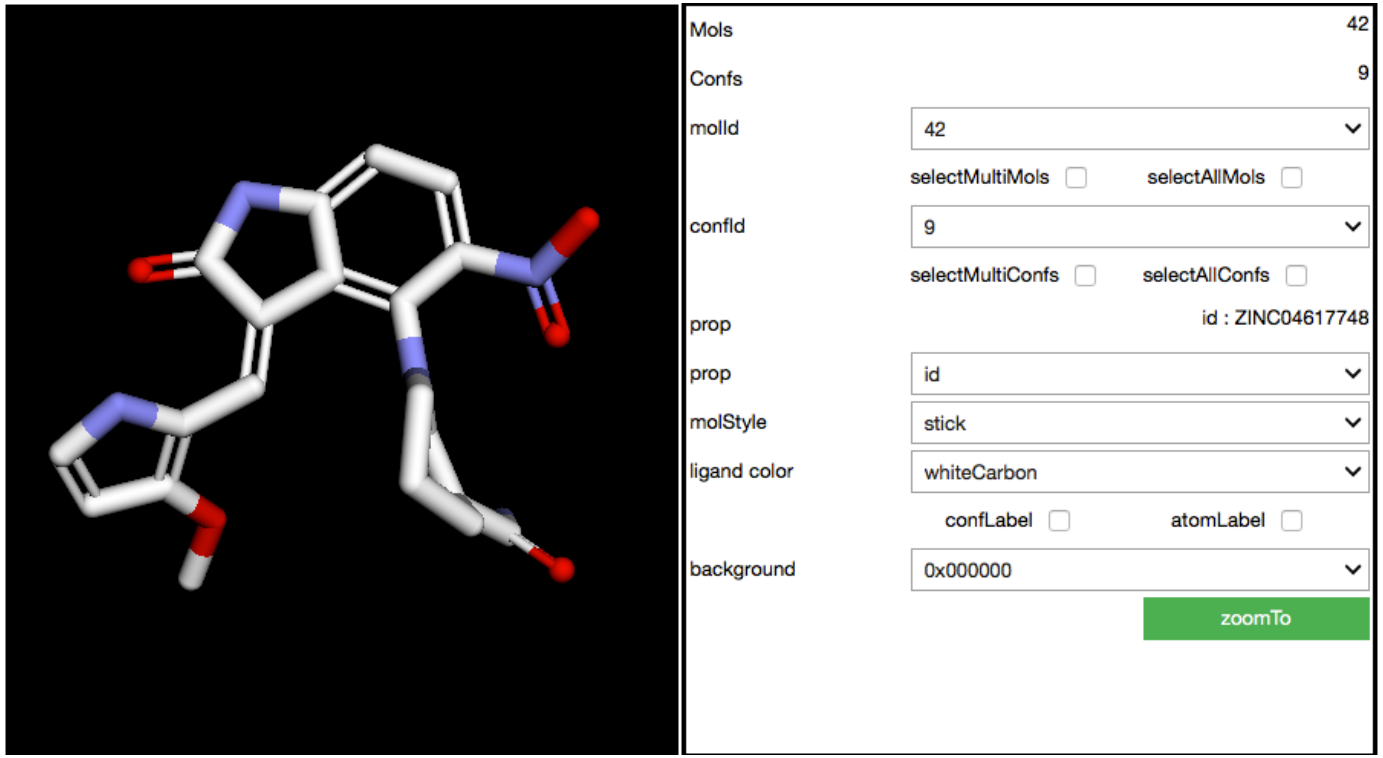
```
1
```

object nm contains two elements. nm.match is the ligand and nm.rest is the protein.

Conformer browser (with and without protein)

https://github.com/malithakabir/RDKitGSoC2017/blob/master/GSoC2017_notebook_1_ConformerBrowse_panels_and_confSelection.ipynb

https://github.com/malithakabir/RDKitGSoC2017/blob/master/GSoC2017_notebook_2_ConformerBrowse_with_proteins.ipynb



https://github.com/malithakabir/RDKitGSoC2017

13 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Find file | Clone or download

malithakabir committed on GitHub notebook 3 energy minimization — Latest commit 07a00ad 8 hours ago

BrowseMultimolsV8.ipynb	submitted for evaluation	2 days ago
GSoC2017_notebook_1_ConformerBrowse_pane...	notebook 1 (conf browse - simple)	22 hours ago
GSoC2017_notebook_2_ConformerBrowse_with_...	notebook 2 conf browser (protein and ligand)	19 hours ago
GSoC2017_notebook_3_energy_minimization_ov...	notebook 3 energy minimization	8 hours ago
IPythonConsoleIntegration.py	submitted for evaluation	2 days ago
LigandExtract.py	submitted for evaluation	2 days ago
README.md	updated readme	17 hours ago

README.md

RDKitGSoC2017

RDKit - 3Dmol.js integration

Mentors: Paul Czodrowski and Greg Landrum
Acknowledgement: Peter Gedeck reviewed codes, provided advice on code restructuring, and wrote initial MolViewState class.
Date: 29th August 2017
Email: malitha12345@gmail.com

To visitor: We appriciate your ideas on adding new features in it. So, please don't hesitate to drop your words.

Tasks listed in GSoC 2017

- A simple conformer browser (completed)
- Energy minimization of ligand extracted from PBD file (completed)
- Molecule editing (incomplete at the time of writing this - date : 29th August 2017)