[WIP] IPythonConsoleIntegration.py and MolView3D folder added #1484

Open **malithakabir** wants to merge 4 commits into `rdkit:GSoC2017` from `malithakabir:master`

Conversation 32 Commits 4 Files changed 2 +989 −0

malithakabir commented Jun 30, 2017 • edited

Trying to embed 3DMol.js and ipywidgets in Jupyter notebook for 3D conformer browsing. Expecting feedback from RDKit community. Many thanks for your time in it.

Reviewers

gedeck

Assignees

Features:

- Multiple molecules supported while rendering 3DMol.js viewer and ipywidgets for conformer scrolling
- Existing properties from SD file can be viewed
- RDKit supported properties (descriptors) can be calculated and viewed for each molecule (I would like to improve it in next version)
- Multiple representation styles (line, cross, stick, sphere, ball-stick, surface)
- Many coloring scheme
- Conformer labeling in 3DMol.js viewer
- Atom labeling in 3DMol.js viewer

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

2 participants

```
IPythonConsoleIntegration.py and MolView3D folder added                e3eb41a
```

greglandrum changed the title from **IPythonConsoleIntegration.py and MolView3D folder added** to **[WIP] IPythonConsoleIntegration.py and MolView3D folder added** Jul 8, 2017

malithakabir added some commits Jul 15, 2017

```
Updating conf browser                                                  19836e9
multiple mol/conf render enabled …                                     1780453
multiple mol/conf render enabled …                                     d987de9
```

gedeck reviewed
Jul 22, 2017

View changes

Hello,

Welcome to the RDkit community. Greg asked me to have at your code and give you feedback. I hope you find my comments helpful.

At the end, I suggest a different way of structuring the code.

Please reach out to me with any questions.

Peter

```
rdkit/Chem/Draw/IPythonConsoleIntegration.py
```

```
+import time
+
+
+bgcolors_3d = ['0xeeeeee', '0x000000', '0xffffff']
```

gedeck Jul 22, 2017 Collaborator

The convention for constants is all capital letters. This applies to all constants.

You use a list here. Lists can be changed, so it would be safer to declare this as an immutable tuple.

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

Later on when you use this constant, you set the second value as default `bgcolors_3d[1]`. I prefer to use the first value; this is less surprising. Reordering would also give the values a more natural order (black, grey, white).

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

```
rdkit/Chem/Draw/IPythonConsoleIntegration.py
```

```
+
+bgcolors_3d = ['0xeeeeee', '0x000000', '0xffffff']
+
+prop_rdkit=sorted([prop for prop,obj in Descriptors._descList])
```

gedeck Jul 22, 2017 Collaborator

Python has a place holder for values that are not required; it's the `_` character. In this statement, you don't use the `obj` variable, so you could this `_` character here.

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

Please note that I removed the square brackets here as well. `sorted` expect an iterable. This could either be something like a list, but also a generator (e.g. `x[0] for x in anotherList`).

`rdkit/Chem/Draw/IPythonConsoleIntegration.py`

```
+
+def ProcessMolContainingObj(Mol):
+    """This function checks whether the object type fits the requiremen
+    If the oject doesn't have necessary attributes, it takes action to
```

gedeck Jul 22, 2017 Collaborator

what are the requirements? `Mol` should be a dictionary or being convertible to a dictionary?

`Mol` - again a convention. First letter capital is used for naming classes and functions. Please use lower letters.

`rdkit/Chem/Draw/IPythonConsoleIntegration.py`

```
+    If the oject doesn't have necessary attributes, it takes action to
+    """
+    try:
+        moldict = dict(Mol)
```

gedeck Jul 22, 2017 Collaborator

If you want to check if something is a dictionary, test for `isinstance(mol, dict)`.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+        if type(Mol) is tuple:
+            moldict=list()
+            moldict.append(Mol)
+            moldict = dict(moldict)
```

gedeck Jul 22, 2017 Collaborator

In general it is recommended to use `isinstance` instead of `type` (see this answer on stackoverflow).

Your code is equivalent to `moldict = dict([mol])`. I tried out a few things and it seems to me that this only works if `mol` has only two elements. In this case, it's clearer to write `moldict = {mol[0]: mol[1]}`

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+        elif hasattr(Mol, '__iter__') is False:
+            moldict=list()
+            moldict.append(('m0', Mol))
+            moldict = dict(moldict)
```

gedeck Jul 22, 2017 Collaborator

```
moldict = {'m0': mol}
```

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+        elif type(Mol) is list:
+            Mol_keys=['m'+str(x) for x in range(len(Mol))]
+            Mol_2=[(Mol_keys[i],Mol[i]) for i in range(len(Mol))]
+            moldict = dict(Mol_2)
```

gedeck Jul 22, 2017 Collaborator

May I suggest that you read up on dict comprehension and the `enumerate` function.

```
moldict = {'m' + str(i): m for i, m in enumerate(mol)}
```

I tend to use the format string method to combine strings with variables, but that is just a preference. Your solution here seems clearer.

```
'm{}'.format(i)
```

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+            Mol_keys=['m'+str(x) for x in range(len(Mol))]
+            Mol_2=[(Mol_keys[i],Mol[i]) for i in range(len(Mol))]
+            moldict = dict(Mol_2)
+    return moldict
```

gedeck Jul 22, 2017 Collaborator

If `mol` meets none of the conditions, the method will return `None`. This implicit here, and I think it would be good to make it explicit. The user of the method then knows that `None` could be returned. This happens for example if you pass in a string as an argument.

```
rdkit/Chem/Draw/IPythonConsoleIntegration.py
```

```
+
+
+
+def update3D(model_id):
```

gedeck Jul 22, 2017 Collaborator

This is a very long function. Consider splitting it into chunks.

```
rdkit/Chem/Draw/IPythonConsoleIntegration.py
```

```
+    It runs first time through handle_button() when the start button cl
+    """
+
+    uid=globals()['rdkit_wg_dict'][model_id]
```

gedeck Jul 22, 2017 Collaborator

It's ok call `globals()` every time, however each function call costs time. It's maybe still fast enough, however it's good practice to assign the dictionary that `globals()` returns to a local variable.

```
rdkit/Chem/Draw/IPythonConsoleIntegration.py
```

```
+
+    globals()['view_'+uid].removeAllModels()
+    globals()['view_'+uid].removeAllSurfaces()
+     globals()['view_'+uid].removeAllLabels()
```

gedeck Jul 22, 2017 Collaborator

The expression `globals()['view_'+uid]` is used more than 10 times. Consider assigning it to a local variable

```
 view = globals()['view_'+uid]
view.removeAllModels()
view.removeAllSurfaces()
view.removeAllLabels()
```

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+     molDictKey=globals()['molTupleId_'+uid].value
+
+     if globals()['selectAllMols_'+uid].value is True:
```

gedeck Jul 22, 2017 Collaborator

The `is True` is not required.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+     if globals()['selectAllMols_'+uid].value is True:
+         keys=list(globals()['moldict_'+uid].keys())
+         for i in keys:
+             globals()['rdkit_mol_select_'+uid].append(i)
```

gedeck Jul 22, 2017 Collaborator

`globals()['moldict_'+uid]` is a dictionary. If you want to iterate over the keys, you can just use it in the for-statement.

```
for i in globals()['moldict_'+uid]:
    globals()['rdkit_mol_select_'+uid].append(i)
```

You append to the existing list here. This may mean that keys entries occur multiple times. You could consider using a set instead of a list.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+            globals()['rdkit_mol_select_'+uid].append(molDictKey)
+        else:
+            globals()['rdkit_mol_select_'+uid] = list()
+            globals()['rdkit_mol_select_'+uid].append(molDictKey)
```

gedeck Jul 22, 2017 Collaborator

Use `if... elif ... else` instead.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+
+    molNameList = list(set(globals()['rdkit_mol_select_'+uid]))
+    globals()['selected_mols_view_'+uid].value = ', '.join(molNameList)
```

gedeck Jul 22, 2017 Collaborator

After replacing `globals()['rdkit_mol_select_'+uid]` with a `set`. This would become:

```
 molDictKey=globals()['molTupleId_'+uid].value
if globals()['selectAllMols_'+uid].value:
  globals()['rdkit_mol_select_'+uid] = set(globals()['moldict_'+uid])
elif globals()['selectMultiMols_'+uid].value:
  globals()['rdkit_mol_select_'+uid].add(molDictKey)
else:
  globals()['rdkit_mol_select_'+uid] = {molDictKey}

molNameList = globals()['rdkit_mol_select_'+uid]
globals()['selected_mols_view_'+uid].value = ', '.join(globals()['rdkit_mol_sele
```

There is no need to convert `molNameList` to a list, a set is just fine and the more appropriate data structure here. you can iterate over it and get it's length. order is not important, uniqueness of values is.

I would call the variable `molNames`.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+        if mol.GetNumConformers()>1:
+
+            allConfIds = range(mol.GetNumConformers()-1)
```

gedeck Jul 22, 2017 Collaborator

I think this should be `range(mol.GetNumConformers())`

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+                        globals()['rdkit_conf_select_'+uid].append(confId)
+                    else:
+                        globals()['rdkit_conf_select_'+uid] = list()
+                        globals()['rdkit_conf_select_'+uid].append(confId)
```

gedeck Jul 22, 2017 Collaborator

This code is very similar to the handling of selected molecules. A set would be more suitable here.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+        for confId in confIdsList:
+            mol = globals()['moldict_'+uid][molName]
+            mb = Chem.MolToMolBlock(mol,confId=confId)
+            globals()['view_'+uid].addModel(mb,'sdf')
```

gedeck Jul 22, 2017 Collaborator

There is a constraint in this code here that may or may not be a problem in the future. It depends on the requirements. This code requires that every molecule that is shown has the same number of conformers.
@greglandrum or **@pzc** may want to clarify this.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+                # update widget
+                globals()['prop_precalc_wg_'+uid].options=all_prop_from
+                # Get selected property
+                prop_name=eval('prop_precalc_wg_'+uid+'.value')
```

gedeck Jul 22, 2017 Collaborator

Here is a good blog post on the dangerousness of eval. What do you try to achieve?

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+            # Descriptor calculation schema eval("Descriptors.TPSA(mol)
+            prop_name=globals()['prop_calc_wg_'+uid].value
+            prop_calc_cmd="Descriptors."+ prop_name + "(mol)"
+            prop_str = prop_name + ' : ' + str(eval(prop_calc_cmd))
```

gedeck Jul 22, 2017 Collaborator

same comment as above. Here you can use:

```
 calculator = Descriptors.__dict__[prop_name]
prop_str = prop_name + ' : ' + str(calculator(mol))
```

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
172 +        color = 'default'
173 +
174 +
175 +    if drawAs is 'surface':
```

gedeck Jul 22, 2017 Collaborator

For string comparison use `==`

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
192 +        try:
193 +            labelConf=globals()['confLabel_'+uid].value
194 +            labelAtom=globals()['atomLabel_'+uid].value
195 +            if labelConf is True:
```

gedeck Jul 22, 2017 Collaborator

`is True` is not required

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
252 +
+
+
+def ShowConformers3D(Mol, protein = None,
```

gedeck Jul 22, 2017 Collaborator

Again a very long function. Would be good to split it into chunks.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+
+def ShowConformers3D(Mol, protein = None,
+                          useDrawAs = False, drawAs=None,
```

gedeck Jul 22, 2017 Collaborator

Set default value for `drawAs` to `'stick'`

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+    # Required global objects
+    globals()['rdkit_mol_select_'+uid] = list()
+    globals()['rdkit_conf_select_'+uid] = list()
```

gedeck Jul 22, 2017 Collaborator

For the reason outlined above, these should be sets.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+    # Right hand panel (widgets)
+    wgListBox=list()
+    itemLayout=Layout(display='flex', flex_flow='row', justify_content=
```

gedeck Jul 22, 2017 Collaborator

Good to use flex_box for easy layout.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+    # molecules and conformers
+    globals()['moldict_'+uid] = ProcessMolContainingObj(Mol)
+    keys=list(globals()['moldict_'+uid].keys())
```

gedeck Jul 22, 2017 Collaborator

You cannot be sure that dictionaries preserve the order. It might be better to ensure that any order is preserved by explicitly using `OrderedDict`. It's still necessary to convert to a list of keys to get the first element.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+    globals()['molTupleId_'+uid] = Dropdown(description='', options=key
+    globals()['selectMultiMols_'+uid] = Checkbox(description='selectMul
+    globals()['selectAllMols_'+uid] = Checkbox(description='selectAllMo
+    globals()['confId_'+uid] = Dropdown(description='', options=range(9
```

gedeck Jul 22, 2017 Collaborator

`list(range(9))` for Python 3.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+
+    wgListBox.append(cbConfSelect)
+
+    globals()['rdkit_wg_dict'].update({globals()['molTupleId_'+uid]._mo
```

gedeck Jul 22, 2017 Collaborator

This is very hard to read for adding one key value pair to a dictionary.

```
 rdkitWG = globals()['rdkit_wg_dict']
rgkitWG[globals()['molTupleId_'+uid]._model_id] = uid
```

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+    if drawAs is None:
+        drawAs = 'stick'
+    else:
+        drawAs = drawAs
```

gedeck Jul 22, 2017 Collaborator

Last two lines redundant code. `drawAs == drawAs` already before the if statement. If you set the default value in the keyword arguments to this function, you don't need this at all.

rdkit/Chem/Draw/IPythonConsoleIntegration.py

```
+    globals()['view_'+uid].zoomTo()
+
+    display(globals()['view_'+uid].insert(uid))
+
```

gedeck Jul 22, 2017 Collaborator

Suggestion for combining everything that describes the molecule view state:

```python
class MolViewState(object):
  def __init__(self, molecules):
    """ Molecules is dictionary of molecules """
    self.uid = str(time.time()).replace('.','')
    self.moldict = molecules

    # These should have reasonable initial values
    self.rdkit_mol_select = set()
    self.rdkit_conf_select = set()
    self.allConfIDs = []

  def selectMolecules(self, selectAllMols, selectMultiMols, selectMol):
    """ Select either all moleculs or add selectMol or show only selectMol """
    if selectAllMols:
       self.rdkit_mol_select = set(self.moldict)
    elif selectMultiMols:
      self.rdkit_mol_select.add(selectMol)
    else:
      self.rdkit_mol_select = {selectMol}

   def selectConformations(self, selectAllConfs, selectMultiConfs, selectConf):
    """ For all selected molecules, select either all conformations or add selec
    for mol in self.selectedMolecules:
      nConformers = mol.GetNumConformers()
      if nConformers > 1:
        if selectAllConfs:
          self.rdkit_conf_select = set(range(nConformers))
        elif selectMultiConfs:
          self.rdkit_conf_select.add(selectConf)
        else:
          self.rdkit_conf_select = {selectConf}
      elif mol.GetNumConformers() == 1:
        self.rdkit_conf_select = {0}

  @property
  def selectedMolNames(self):
    """ Return the names of all selected molecules """
    return self.rdkit_mol_select

  @property
  def selectedConfIds(self):
    """ Return the names of all selected molecules """
    return self.rdkit_conf_select

  @property
  def selectedMolecules(self):
    """ Return the selected molecules """
    return [self.moldict[name] for name in self.selectedMolNames]

  @property
  def selectedModels(self):
    """ Iterator over all selected models (molecules/conformations) """
    for mol in self.selectedMolecules:
      for confId in self.selectedConfIds:
        yield Chem.MolToMolBlock(mol, confId=confId)

  @property
  def allConfIds(self):
    """ Return the number of conformations - use the first selected molecule to
    nconfIds = self.selectedMolecules[0].GetNumConformers()
    return list(range(nconfIDs))
```

and initialize like this:

```
 molViewState = MolViewState(ProcessMolContainingObj(Mol))
globals()['mol_views'][molViewState.uid] = molViewState
```

The class is now independent of the viewer and you can easily write tests for it. It will also be easier to modify it should it be necessary to support different number of conformers. I would also add the property calculation to this class and actually move it into a separate module. As the ipython functionality becomes more and more important, consider having a separate directory for it (**@greglandrum** please comment on this).

Using this class, it would lead to the following change in `update3D`:

```
 uid = globals()['rdkit_wg_dict'][model_id]
```

```
molViewState  = globals()['mol_views'][uid]

molViewState.selectMolecules(globals()['selectAllMols_'+uid].value,
                             globals()['selectMultiMols_'+uid].value,
                             globals()['molTupleId_'+uid].value)
globals()['selected_mols_view_'+uid].value = ', '.join(molViewState.selectedMolN

molViewState.selectConformations(globals()['selectAllConfs_'+uid].value,
                                 globals()['selectMultiConfs_'+uid].value,
                                 globals()['confId_'+uid].value)
globals()['confId_'+uid].options = molViewState.allConfIds()
globals()['selected_confs_view_'+uid].value = ', '.join([str(x) for x in confIds

# Add models (molecules/conformations) to viewer
for model in molViewState.selectedModels:
  globals()['view_'+uid].addModel(model, 'sdf')
```

This should give you an idea how to continue with this. I would also combine everything that has to do with controls in one class and the 3D view of the molecule into one class as well.