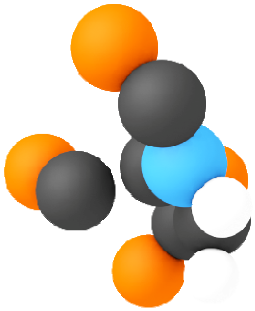
BioBlender

Mike Pan | Summer 2010 | IFC CNR Pisa



Overview

- For end-users
  - Packaging and installation
  - General workflow
  - User interface
  - Troubleshooting and Limitations

- For developer
  - Method of integration with Blender
  - Code architecture and design
  - Conventions and Tidbits
  - Bugs and workarounds
  - Limitations and roadmap

end-user

Bioblender is a platform based on Blender 2.5 for:

- Molecular Visualization
- Molecular Dynamics Simulation
- Complex protein visualizations

Packaging and Installation

- BioBlender is simply a regular Blender 2.53 Beta binary release with a few custom scripts and data files.
- BioBlender is an extension of Blender, and thus, like Blender, does not require a formal installation process. To run BioBlender, simply extract the compressed file and launch blender.exe.
- In the console window, if you see the line “Hello from BioBlender”, then BioBlender launched correctly.
- BioBlender uses a few other free third-party softwares (PyMLP, APBS, etc), these softwares and their licences are included in the BioBlender package.
- For full functionality, BioBlender requires the installation of Python 2.6, and PyMol (Free or Licensed). These two packages must be installed separately by the user.
- BioBlender runs on Windows and Linux (with reduced feature set).

General Workflow: Basics

- Once Blender is launched, it is recommended to load template.blend. This Blender file is a good starting point for user unfamiliar with BioBlender. It contains a ready-made lighting and camera setup for rendering, and an optimized interface layout for working with BioBlender. Avoid saving over this file.
- The BioBlender user interface is contained in the vertical Scene Property Panel. Here, you can import, export, view, and manipulate the molecular data.
- Each interface element (buttons, sliders, toggles) has help text associated with it. Mouse-over them to see a full description of what it does.
- Errors and progresses are displayed in the console window. Check this window frequently if you suspect something is wrong. Critical errors will appear in the main BioBlender Interface as a popup.
- Once a PDB model is successfully imported into BioBlender, a .cache file with the same name as the Blender file will be written to disk beside the blender file. This file contains the serialized data needed by the blender file.

General Workflow: Step by Step

1. Select a PDB file and set the desired import options.
2. Click on Import PDB, a 3D representation of the molecule should appear in the 3D viewport in less than 10 seconds. One can:
   - Manipulate the camera around the molecule with the middle mouse button
   - Select individual atom or select amino acid as a group by right clicking
   - Change the amount of visible atoms in the viewfilter bar. (e.g. α-carbons only; Main & Side Chain)
   - View the solvent accessible surface of the protein
3. Show MLP on Surface calculates the lipophillicity of the molecule on the surface. The Grid Spacing and Formula setting are used in the calculation.
   - The MLP is mapped to the surface from black to white. The Contrast and Brightness slider can be used to obtain an visually pleasing result.
4. Render MLP to Surface bakes the mapping obtained in step 3 into a novel visualization technique where the lipophillicity is mapped as roughness across the surface. Where smooth denotes low hydrophillicity, and rought denotes high hydrophillicity. Press F11 to view the output.
5. Show EP calculates and displays the electric potential of the surface as field lines using all the settings located to the left.
   - Use Clear EP to delete all the object created for the EP visualization.

General Workflow: Animation

- If two or more conformations are loaded into Blender from a single PDB file, an animation is created automatically in which BioBlender will automatically interploates between the initial and final position of each atom using linear interpolation.
- The timeline at the bottom of the BioBlender window shows the progress of the animation.
- Animation will start playing automatically once the import process is complete.

  length of animation = the number of conformations **x** the keyframe interval
- Running the EP visualization will also animate in the 3d viewport.
- Basic protein motion is done using LERP, more physically accurate result can be obtained by running the animation using the game engine which resolves collision problems between atoms, while keeping bonds relatively rigid.

User Interface: Import

- Import File Path: To see a molecule displayed in 3D in Blender, one only needs to provide an input file in the format of a PDB, CSV or TXT file. To load a model, click on the small folder icon (under the Import region) to open a file browser.
  - If a PDB file is selected, BioBlender will count the number of conformations within the file and gives the user the choice to load any arbitrary subset of the available conformations. (See Import Model List)
  - If a CSV or TXT file is selected, BioBlender will load ALL the files of the same extension within the directory. So it is crucial to make sure the CSV or TXT sequence is places in a file folder with no other files.
- Model Name Tag: This editable textbox allows you to give a unique name to the model to be imported.
- Import Model List: This editable textbox shows the user which conformation BioBlender will import when the Import button is clicked. Only visible when multiple conformations are detected in a PDB.
  - One can inspect and edit this list to skip loading of unwanted models or files. Note that this list is comma separated.
- Verbose: Toggle display extra information in the console for debugging
- Space Fill: Toggle use of VdW radius or covalent radius for atom geometry.
- SideChain: Toggle loading of side chain atoms. (Faster when disabled)
- Hydrogen: Toggle loading of Hydrogen. If SideChain is disabled, H is never loaded. (Faster when disabled)
- Make Bonds: Toggle generation of physics constraints to represent bonds between atoms. Useful for physics simulation of molecular motion. (Faster when disabled)
- High Quality: Toggle use of higher quality atom and surface geometry. (Faster when disabled)
- Single User: Toggle use of shared mesh data for atoms. Enable for Game Engine since shared material does not work well in game. (Faster when disabled)
- Keyframe Interval: Set the interval (in frames) at which animation keyframes are inserted. Only visible when multiple conformations are detected.
- Import PDB: Loads the PDB file and generates the 3D geometry in Blender, taking into account all the settings mentioned above.

User Interface: View

- Currently Selected Model: Is the name of the protein currently active. The name of the model is the tag name the user used in the import process. All other operations (change view, EP|MLP visualization, export) will operate on the selected model only.
- PDB Data: This line shows the associated PDB information for the selected atom.
- View Mode: This selector allows the user to change the view mode of the molecule. Clicking on surface will invoke PyMOL to generate the solvent accessible surface of the molecule.
  - Solvent Radius controls the solvent radius parameter used by PyMol.
  - To force a regeneration of the surface, click on the REFRESH button.
- Interactive Mode: Enter the Blender Game Engine and to see the molecule interactively.
- Collision: This setting changes the behavior of the physics engine:
  - 0. Physics Engine is disabled, no collision evaluation is carried out
  - 1. Physics Engine is enabled
  - 2. Physics Engine is enabled, and the resulting new animation is written to the Fcurve, the game automatically ends when the simulation is complete.

User Interface: Motion

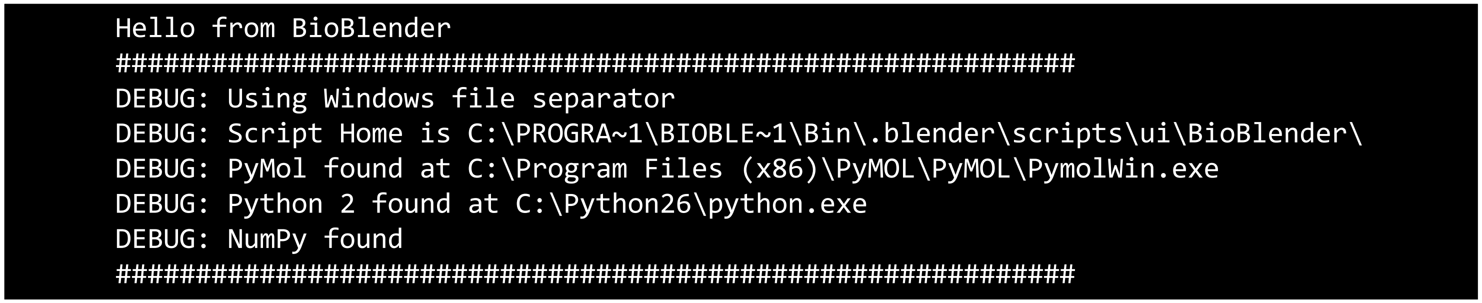
- Since linear interpolation of the protein motion left a lot to be desired, the Blender physics engine can be used to elaborate the movement of the atoms, taking into account collision and rigid body bonds.
- New motions are recorded 200 frames to the right of the original motion, so as not to interfere with the old motion.
- The behavior of the atoms is highly dependent on the following settings:
  - Collision radius of atom (Physics Property Panel)
  - Rigid body constraints (Object Constraints Property Panel)
  - Physics Engine Settings (World Property Panel)
  - Initial position of atoms (intersecting atoms at time0 is bad)
  - Speed of the transition (slower movement might be more stable)
  - The size of the molecule (Smaller molecules might be more stable)

User Interface: Render

- If two or more PDB conformations from a single PDB file are loaded into Blender, an animation can be made.
- Simply select a visualization type, set the frame range and step size, and select Export Movie.
  - Step size allows the user to render only every *n*th frame, this value is ignored when rendering EP because particles do not behave properly when intermediate frames are skipped.
- The animation sequence is saved to the export path as defined by the user.
- Animation is exported as a series of still images, this ensures that the rendering can be resumed should the rendering process be disrupted.

Troubleshooting and Limitations

- The console (that black window that opens with Blender) is generally a very helpful tool in seeing what is going on.
- When BioBlender starts, this is what you should see in the console:



```
Hello from BioBlender
############################################################
DEBUG: Using Windows file separator
DEBUG: Script Home is C:\PROGRA~1\BIOBLE~1\Bin\.blender\scripts\ui\BioBlender\
DEBUG: PyMol found at C:\Program Files (x86)\PyMOL\PyMOL\PymolWin.exe
DEBUG: Python 2 found at C:\Python26\python.exe
DEBUG: NumPy found
############################################################
```

- Settings such as application path (to Python and PyMol) can be found in bin/.blender/scripts/ui/settings.py
- On WinXP, one need to install the Visual C++ 2008 SP1 redistributable (x86) found in the installer folder for Blender to launch correctly.

developer

Bioblender extends Blender 2.5 in the following ways:

Python Scripts

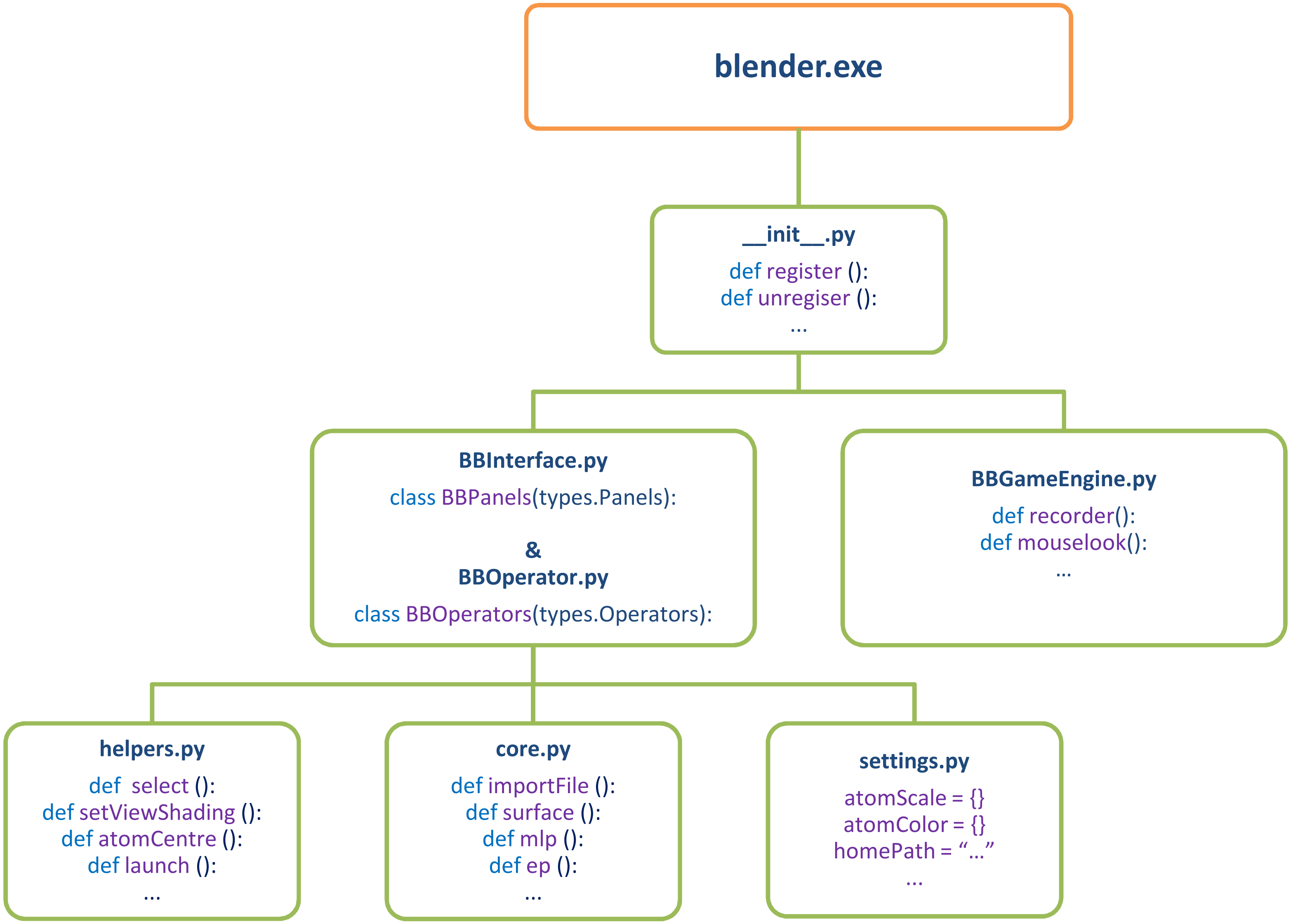
Third Party Applications

Patches to Blender base

Method of integration with Blender

- The BioBlender core is a basically a collection of scripts and data files located in bin/.blender/scripts/ui/.
- BioBlender also makes use of apbs, pdb2pqr, pyMLP, pyMOL and Python 2.
- Because atom generation method uses a custom feature that is not available in the official release (duplicate with multiple copies at once), BioBlender requires a patched version of Blender. The official Blender 2.53 Beta release from Blender.org will work, but is hundred of magnitudes slower when dealing with molecules with more than a few hundred atoms.
- Since there is no way to declare new spaces, the user interface for BioBlender is drawn to the scene property panel. Existing items in the scene panel are removed.
- The template.blend file contains also the setup for the game engine.

Code architecture: overview



blender.exe

__init__.py

```
def register ():
def unregiser ():
…
```

BBInterface.py

```
class BBPanels(types.Panels):
```

&

BBOperator.py

```
class BBOperators(types.Operators):
```

BBGameEngine.py

```
def recorder():
def mouselook():
…
```

helpers.py

```
def  select ():
def setViewShading ():
def atomCentre ():
def launch ():
…
```

core.py

```
def importFile ():
def surface ():
def mlp ():
def ep ():
…
```

settings.py

```
atomScale = {}
atomColor = {}
homePath = "…"
…
```

Code architecture: data

Data

Temporary Files (disk)	**Python internal types (store.py)**	**Blender RNA entries (.blend)**
Intermediate files needed by third party applications. Stored to disk.	Created at script runtime. Not saved to the Blender file. Can be manually saved by serialization.	Statically typed. Stored in the blender file.

Examples:
tmp.pdb
tmp.wrl
tmp.dx
tmp.obj
	Examples:
	Saved to disk:
	`modelContainer = { ... }`
	Not saved to disk:
	`sideChainCache = [ ... ]`
	`mainChainCache = [ ... ]`
		Example:
		`String(BBImportPath)`
		`Bool(BBImportHydrogen)`
		`Bool(BBMakeBonds)`
		`Bool(BBHighResAtom)`
		`Enum(BBViewFilter)`

Code architecture: detail

- PDB reader uses a column based parsing method, which should be more standard-compliant than relying on whitespaces.
  - Has options to skip hydrogen or sidechain
  - Able to generate bonds as physics constraints (using a lookup table to build the links)
  - Each atom is represented by a Blender sphere object, the visual size is either the covalent radii or the VdW radii, object has collision radius (always VdW), color(element type), and is associated with the full PDB data line (amino acid group, location, charge, etc)
  - The list of atoms for each loaded molecule is stored in a Python list for easy access. (this list is stored to disk as the cache file)
- For multiple conformations, animation key frames are inserted at the initial and final position.
- Once a model is imported, user can
  - Hide or show atom groups based on user selected filters
  - Animate protein from one conformation to the other (linear interpolation)
  - Enter the game engine mode and elaborate the molecular motion using the physics engine.
  - Run EP and MLP visualizations on the model

sidenote: RNA

- All blender datablocks (object, materials, images) is stored internally in Blender as “DNA”.
- RNA is the data API that exposes the DNA to a higher level programming layer (Python)
- When coding in Python, one can only access data through the RNA path. An example of an RNA path:

  data.scenes[“Scene”].objects["Cube"].data.verts[7].co

- To create a Python String:
  - BBImportPath = “Hello World”
- To create an RNA String in Scene:
  - types.Scene.StringProperty(attr=“ImportPath", name=“File Path", description="", default=“Hello World")

sidenote: operators class

```
class BBOperatorImport(types.Operator):
    bl_idname = "BBOperatorImport"
    bl_label = "Import PDB"
    bl_description = "generate 3D Model"
    def invoke(self, context, event):
        try:
            context.user_preferences.edit.use_global_undo = False
            importFile()
            context.user_preferences.edit.use_global_undo = True
        except Exception as E:
            self.report('ERROR', "Import Failed: " + str(E))
            print ("Error: Import Failed: " + str(E))
            return {'CANCELLED'}
        else:
            return {'FINISHED'}
```

sidenote: interface class

```
class BBPanel():
    bl_space_type = 'PROPERTIES'
    bl_region_type = 'WINDOW'
    bl_context = "scene“
    def poll(cls, context):
        return (context.scene)

class BBPanelFile(BBPanel, types.Panel):
    bl_label = "BioBlender Select PDB File"
    def draw(self, context):
        scene = context.scene
        layout = self.layout
        split = layout.split(percentage=0.7)
        split.prop(scene, "BBImportPath", text="")
        split.prop(scene, "BBModelRemark", text="")
```

sidenote: pickle

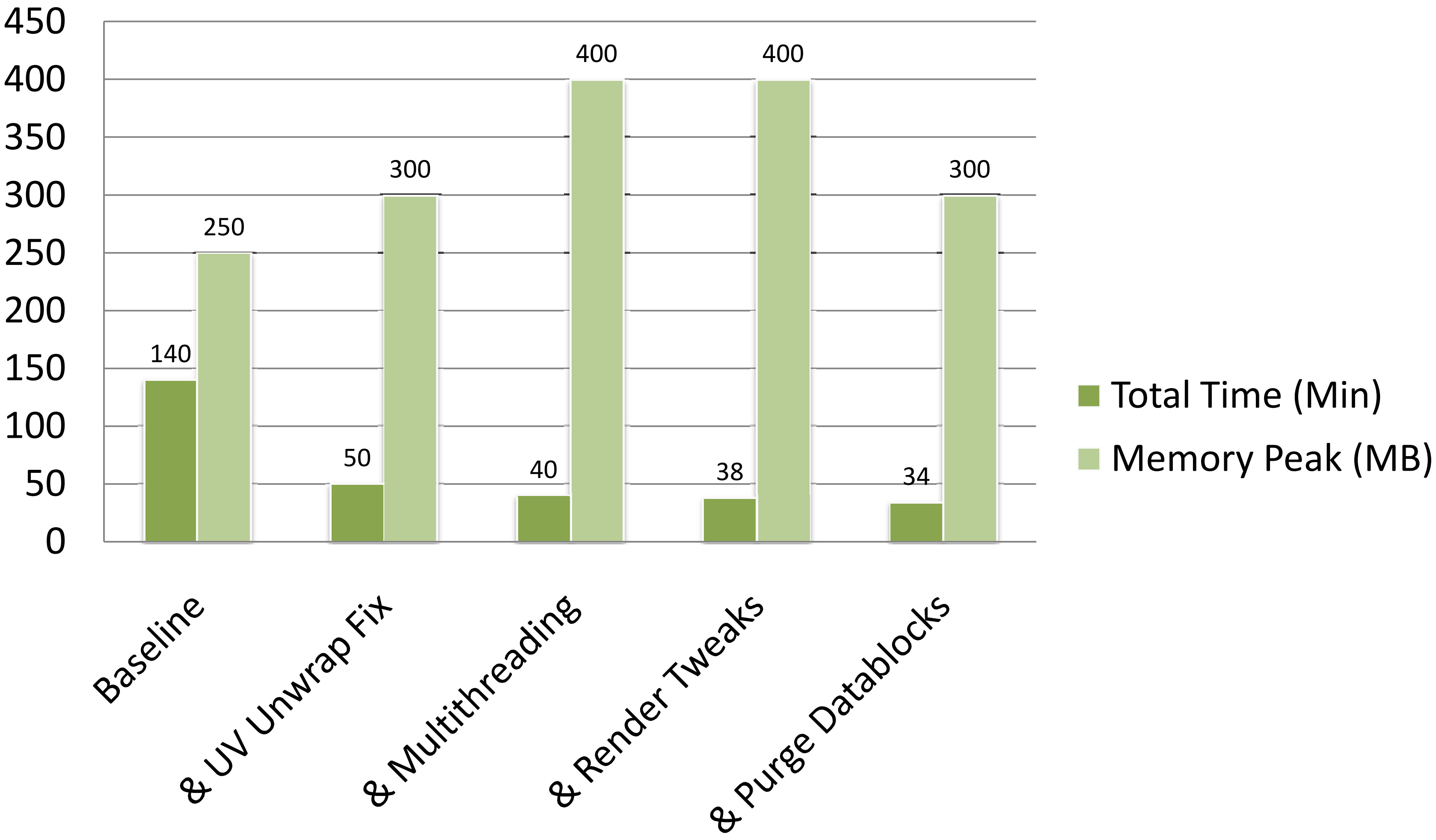
- serialization is a way to convert arbitrary variable from program memory into a string representation that can be communicated across a network or saved to storage.
- Python’s serialization module is called pickle.

```
myVar = [2,3,4]
pickle.dump(myVar)
b'\x80\x03]q\x00(K\x02K\x03K\x04e.‘
```

- Now we can store that string to disk and retrieve it later.

```
myVar = pickle.load(myFile)
myVar: [2,3,4]
```

Conventions and Tidbits: Performance



Bugs and workarounds

- Atoms are originally added via the ops.object.duplicate operator in Blender. But this has some significant performance problem as the number of atom increases.
  - Solution: A patch for Blender is introduced that avoids calling ops.object.duplicate multiple times.
- UV unwrapping of the surface using ‘smart-projection’ is very slow when there is a lot of objects in the scene (even when they have nothing to do with the object being unwrapped).
  - Solution: Import surface into a blank file, unwrap, and then append into main file.
- Certain operations such as surface-generation and MLP-calculation are do not rely on each other’s output and can be run at the same time.
  - Solution: independent tasks are carried out in parallel as separate processes, multicore processor can benefit from this.
- Certain operations such as adding particles system and materials cannot be done via Python, since the context is never correct.
  - Solution: Append a dummy object with the particle system already attached from another file to avoid having to create the particle system via script.

Limitations and roadmap

- All operations (except the most trivial ones) are wrapped in a try block to catch all runtime errors. Error-prone areas also use try blocks to catch specific errors. Attempt are made to give user precise feedback for the error and what they can do to remedy the situation. ("PDB file does not follow 80 column specification" rather than errors such as 'array overflow on line 938')
- Evaluation of protein motion using the physics engine is fully working, but the result needs to be evaluated for physiochemical feasibility.
- Code for interacting with the prodrg website (submitting PDB and fetching result) is fully working and can be found at bin/.blender/scripts/ui/prodrg.py. But it is not implemented into the main program.
- scivis.exe occasionally have some problem with floating point numbers due to regional settings. (e.g. 5,0 vs 5.0)

Where to get help

- Official API: http://www.blender.org/documentation/250PythonDoc/contents.html
- Sample code at bin/.blender/scripts/
- Me!