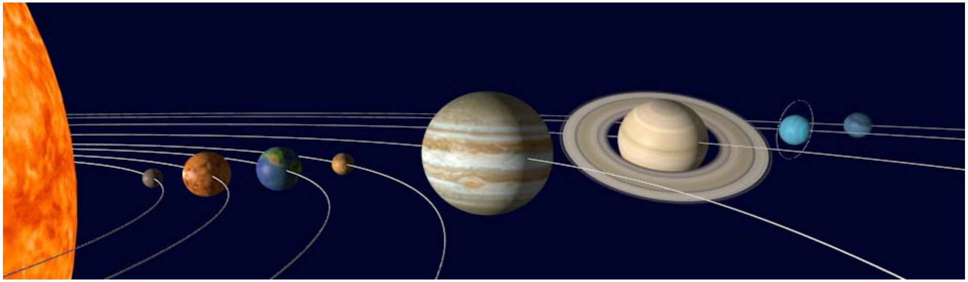
Modelling the Solar System using Blender & Python



Pyladies Workshop, 08.09.2015

Kristin Riebe

Agenda

1. Introduction
2. Getting started with Blender (GUI)
3. Blender with Python
4. Planets

Hands-on session: Write your own planets-script!

1. Introduction

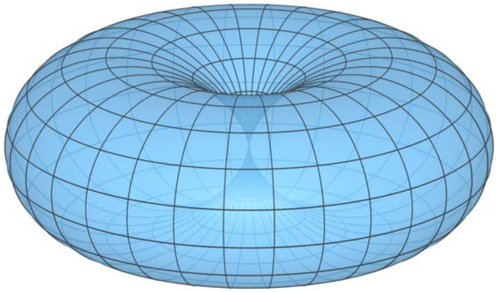
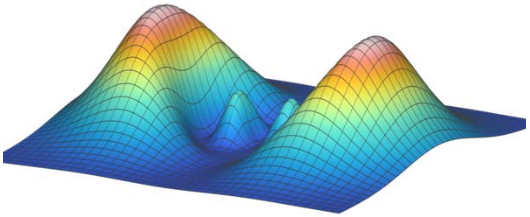
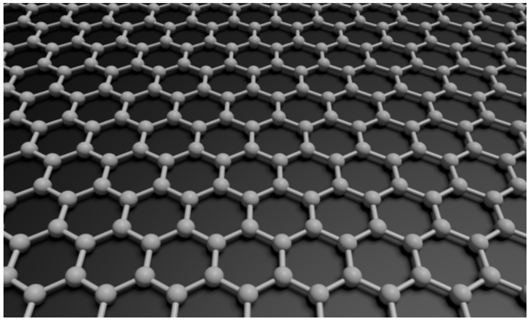
About me

- Leibniz-Institute for Astrophysics Potsdam (AIP)
    - E-Science group: data publication, web services
    - Data visualisation
- Hobby: computer graphics
    - create images 2D/3D
    - mainly for scientific articles/books or just for fun

Blender

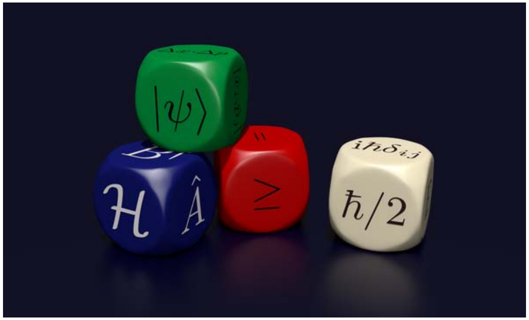
- Powerful 3D creation suite
- Wide range of applications: geometric objects in 3D, modelling landscapes, animating characters, but also compositing, video editing, ...
- Open Source, released by Blender Foundation, www.blender.org
- Works on Linux, Mac OS and Windows
- Interactive use, but also Python API for scripting
- Full control over light and camera settings, also 3D stereoscopic cameras
- Can be quite overwhelming at first, but:
    - No need to know everything!
    - Can get very far with only some basic knowledge.

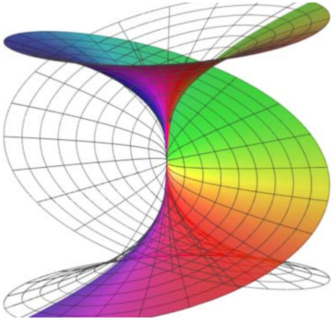
Examples: Science

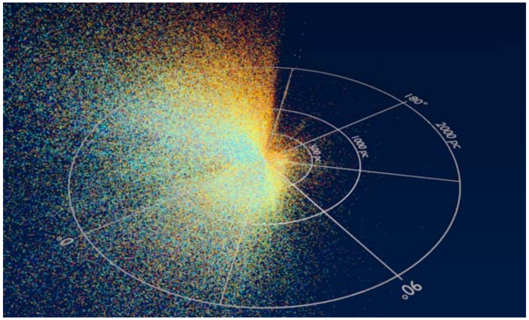




(C) Kristin Riebe, Springer/Spektrum Verlag

Examples II: Science & Fun

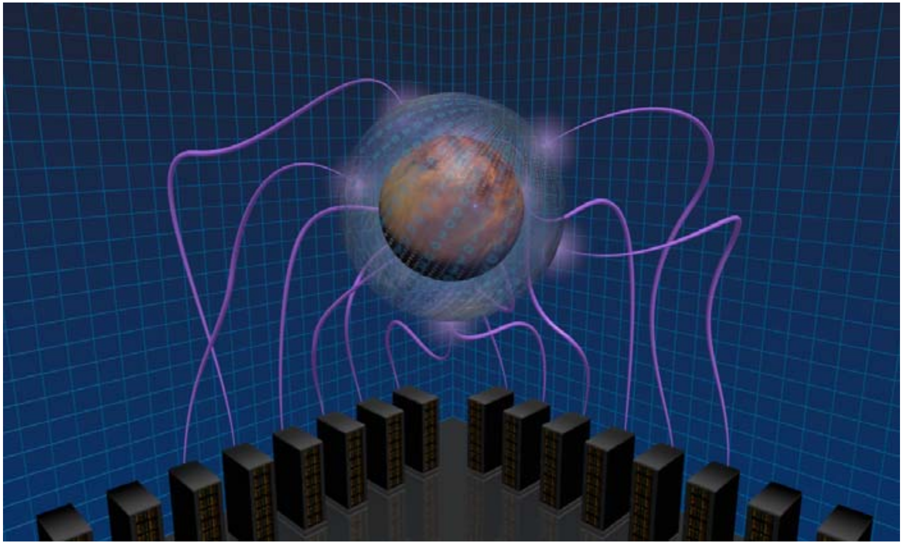






(C) Kristin Riebe, top right: Kristin Riebe, Springer/Spektrum Verlag

Examples III: Just for fun



(C) Kristin Riebe

See Blender Artists for great examples what else you can do with Blender!

General tips for Blender

- Save early, save often.
- There are *hotkeys* for nearly everything.
- Use a 3 button mouse (with scroll-wheel).
- Use keyboard with num block.
- Getting help:
  - Blender manual
  - Blender API documentation
  - Blender StackExchange
  - Blender Python Blog
  - Blender Artists Forum
  - There are many, many video tutorials out there!

2. Getting started with Blender (GUI)

Start Blender **now**,

preferably from the command line.

Installing Blender

Window layout

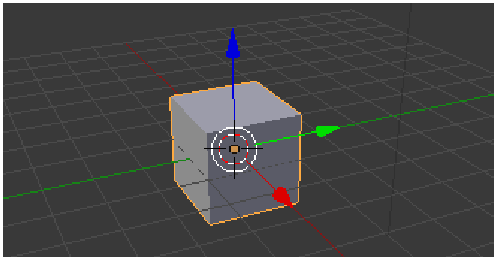
- Different *areas*: 3D view, Outliner, Properties, ...
    - 3D view = main working area!
- Each area can have a *region* attached (toolbar (`T` key) and properties (`N` key)).
- Highly customizable, can switch every area to any other area, define own hotkeys etc.
- Last resort if something breaks with window layout/GUI: `File` -> `Load Factory Settings` or just restart Blender
- **Important:** Anything you type only affects the window with the mouse pointer!
  Thus be careful to **put the mouse pointer in the correct area!**

Navigation

- Zooming in and out: `mouse wheel`
- Rotate around current center:
    - click middle mouse button (`MMB`) and drag
    - for setting the view center to currently selected object: choose `View` -> `View Selected` from menu or hit `Numpad .`
- Pan (move sideways): hold `Shift` key and `MMB` together and move the mouse sideways.

Selecting and moving objects

- Select object: right mouse click (`RMB`)
- Add object to current selection: `Shift` + `RMB`
- Active object = last selected object! (white highlight)
- Moving objects:
    - Place mouse pointer on one of the 3 arrows, click with left mouse button (`LMB`), drag it along the line, release `LMB`
    - Hit `G` key ("grab"), move the mouse pointer until ready, then click with `LMB`



Typical steps

- **Add object**, adjust location, size, shape
  - Use *Add* menu at bottom of `3D view` area, e.g.:

    `Add` -> `Mesh` -> `UV Sphere`
  - Move using arrows and `LMB`
- **Add material** (color, transparency, reflection etc.)
  - At *Properties* area, *Material* tab: click `New` button
- **Add texture**
  - At *Properties* area, *Textures* tab: click `New`
- **Add contraints or modifiers** (also in *Properties*)
- **Add keyframes** for animation (using timeline and `I` key)
- **Adjust light and camera** for the scene
- **Render** the scene

Rendering

- = Take a picture of your scene
- Need a camera and a light source (lamp)
- Object must be visible for camera
    - Check with `View` -> `Camera` (`Numpad 0`)
- Render:
    - `Render` -> `Render Image` (at top menu, close to `File`)
    - or `F12`
    - or in *Properties* area, *Render* tab, press *Render* button
    - or directly from command line:

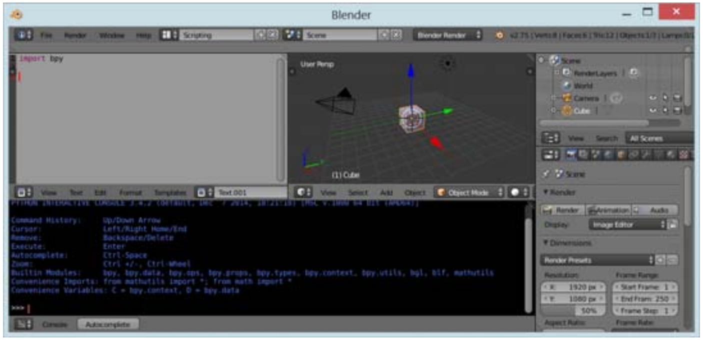
    ```
    blender -b <yourfile.blend> -f 1
    ```

- Quit *Render view* using `Esc` key.

3. Blender with Python

Basics

- Nearly everything done in the GUI can be scripted via Python.
- Blender uses **Python 3**, bundled
- Start Blender from the **command line**, otherwise no error output!
- Use **Scripting layout** with *Text Editor* and *Python Console*
- Load/write scripts inside **Text Editor**, `Text` -> `Run Script`
- Use **import bpy** inside scripts to import Blender functions
- Use `Text` -> `Save As` or `Save` to save text as external file.



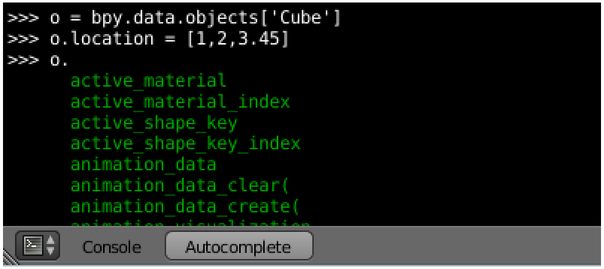
```
import bpy
```

```
PYTHON INTERACTIVE CONSOLE 3.4.2 (default, Dec  7 2014, 18:21:18) [MSC v.1800 64 bit (AMD64)]

Command History:     Up/Down Arrow
Cursor:              Left/Right Home/End
Remove:              Backspace/Delete
Execute:             Enter
Autocomplete:        Ctrl-Space
Zoom:                Ctrl +/-, Ctrl-Wheel
Builtin Modules:     bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>>
```

From GUI to Python

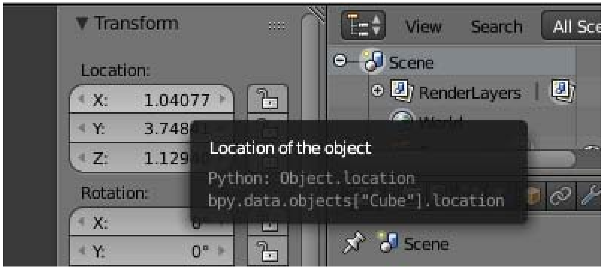
- Graphical interface supports you with:
  - **Info** window:
    - at the top, above or below *File* menu
    - shows log of the applied functions
  - **Tool tips**:
    - when hovering with the mouse over a button, field, object or other elements
  - **Right mouse-click** context menu:
    - often contains link to Blender documentation
  - **Python Console**:
    - test functions and properties immediately
    - use `Ctrl` + `Space` for auto-completion

Python Console, Tooltips and Info



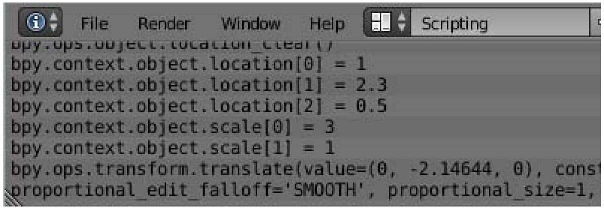
```
>>> o = bpy.data.objects['Cube']
>>> o.location = [1,2,3.45]
>>> o.
      active_material
      active_material_index
      active_shape_key
      active_shape_key_index
      animation_data
      animation_data_clear(
      animation_data_create(
```

Python Console



```
Location of the object
Python: Object.location
bpy.data.objects["Cube"].location
```

Tool tip



```
bpy.ops.object.location_clear()
bpy.context.object.location[0] = 1
bpy.context.object.location[1] = 2.3
bpy.context.object.location[2] = 0.5
bpy.context.object.scale[0] = 3
bpy.context.object.scale[1] = 1
bpy.ops.transform.translate(value=(0, -2.14644, 0), cons
proportional_edit_falloff='SMOOTH', proportional_size=1,
```

Info log

Active object/choose object

- `bpy.context.object` = active object (last selected)
- Can be replaced by another object like this:

  `myobj = bpy.data.objects['CameraPath']`

- Example:

```python
bpy.context.object.location[0] = 1.0

myobj = bpy.data.objects['CameraPath']
myobj.location[0] = 1.0
```

Mode

- Available functions and operators change with current mode!
- Adjust mode in bottom menu of *3D view* area
- Only need `Object Mode` and `Edit Mode` for this session:
  - `Object mode`: change global properties
  - `Edit Mode`: change data, e.g. handles of Bézier curves, vertices and faces of a mesh
- Always switch back to `Object Mode` before running scripts in this workshop!
- Via Python: `bpy.ops.object.mode_set(mode='OBJECT')`

Operators

- Start with `bpy.ops.`
- Operate on the current context, i.e. usually on currently **active** object, in the current mode
- Example: smooth a sphere (*Object* mode):

```
sph = bpy.data.objects['Sphere']
bpy.context.scene.objects.active = sph
bpy.ops.object.shade_smooth()
```

- Generally slow, but encapsulate more complex steps
- Use underlying low-level functions, when possible and useful

Adding objects I

- Using (primitive) operator:

```
bpy.ops.mesh.primitive_uv_sphere_add(...)
obj = bpy.context.object
mesh = obj.data
```

Adding objects II

- Using low level functions:

```
mesh = bpy.data.meshes.new(meshName)
obj = bpy.data.objects.new(objName, mesh)
scn = bpy.context.scene
scn.objects.link(obj)
scn.objects.active = obj
obj.select = True
mesh.from_pydata(verts, [], faces)
mesh.update()
```

- See Three ways to create objects

Data collections

- Module `bpy.data` gives access to data in currently loaded Blender file
- Examples:
  - `bpy.data.objects` : collection of all objects in the scene
  - `bpy.data.materials` : collection of all materials
  - `bpy.data.textures` : collection of all textures
- Blender's collections allow use of index or string for accessing elements:
  - `bpy.data.objects['Camera']`
  - `bpy.data.objects[0]`

More advanced usage

- Blender allows to integrate Python scripts directly:
    - via own defined operators
    - by defining menus and panels for custom scripts, own add-ons
    - by inserting new buttons into existing panels
- Look at *Text Editor*, `Templates` menu for such examples.
- There's a number of useful `Add-ons` available, see `File` -> `User Preferences` , `Add-ons`
- We won't use these in this workshop.

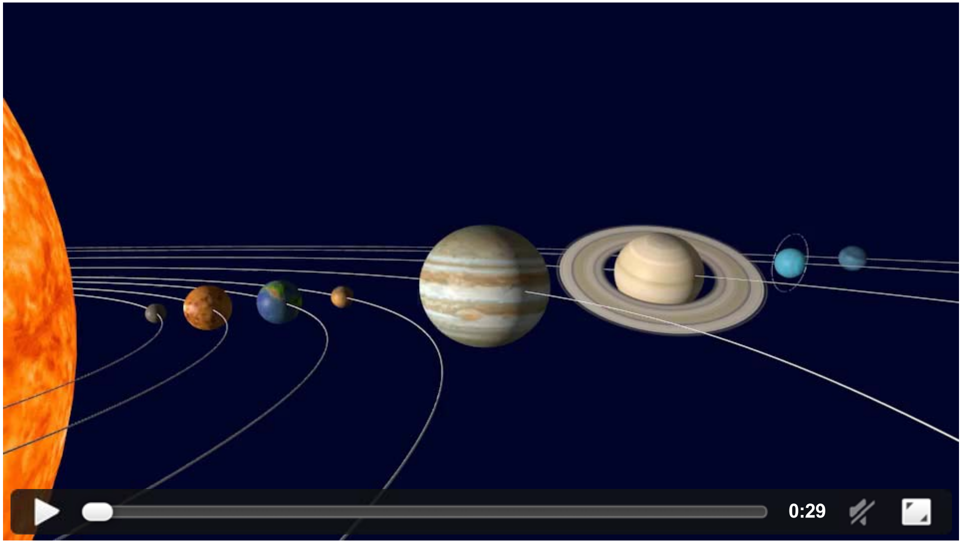
4. Planets

Planets?

- Non-glowing bodies, orbiting around a star
- 8 planets for the Sun
- Properties:
    - closely resemble spheres, a bit flattened
    - each planet has different surface or cloud structure
    - rotate around star on ellipses (mostly close to circles)
    - rotate around their own axis
    - own rotation axis is tilted
    - some planets have ring system
    - have very different rotation times and orbit periods
- Solar system in 3D should reflect this

Creating planets with Blender

- Materials on GitHub, linked at:
  http://kristinriebe.github.io/solarsystem-workshop/
- Contains instructions to guide you
- Example for final animation:

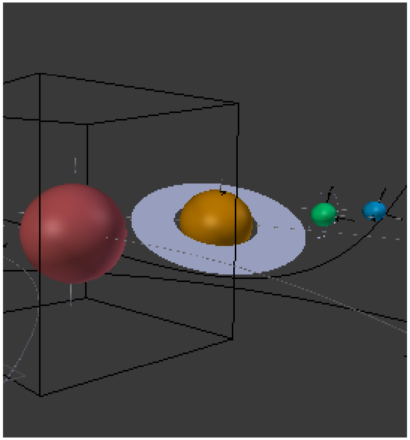


Getting started

- Start Blender from command line
- Open `planets-template.blend`
- In *Text Editor*, load `create_planet.py`
- Comment all functions except `add_sphere`
- Run the script: `Text` -> `Run Script` (`Alt` + `P`)
- Test e.g. changing location, size, color
- Add material and textures, check interface and script
- Render to see results

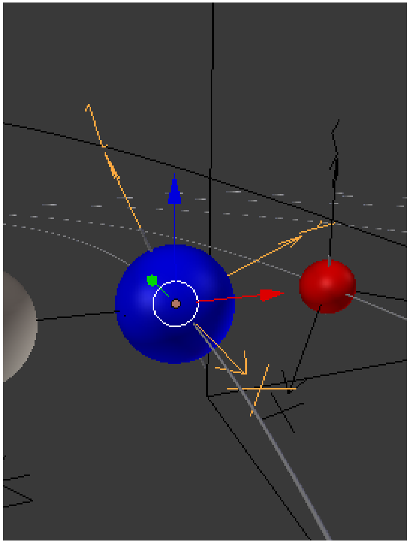
Main tutorial steps I

- Create spheres, add color and texture
  - use template script, adjust it
- Read planet properties from file
  - use csv module, dict reader
- Add flattening, axial tilt, rings
  - manipulate properties
  - use another script as module
- Orbit paths
  - add curves, adjust thickness
- Camera animation
  - constraints (Follow Path, Track To)
  - keyframe evaluation time



Main tutorial steps II

- Orbit animation
  - use constraint (Follow Path)
  - keyframe evaluation time
  - adjust animation curves (F-Curves)
  - apply transformations
- Rotation animation
  - parenting
  - set keyframes
  - adjust animation curves (F-Curves)
- Render
  - via interface or script



Have fun!

http://kristinriebe.github.io/solarsystem-workshop/