Catalog Rendering with Blender

Dr. Brian R. Kent, Scientist, National Radio Astronomy Observatory
Twitter, Instagram, YouTube: @VizAstro
https://www.cv.nrao.edu/~bkent/blender

Visualization Tutorial

October 2021 Flatiron Institute

Requirements: Linux, Windows, or Mac computer

A laptop is sufficient, but a mouse with scroll capability is preferred to a trackpad.

If you have a full keyboard with a right-side numeric keypad, it makes navigation easier.

Blender download (https://www.blender.org/)

Data: Extragalactic Distance Database (http://edd.ifa.hawaii.edu/)

Any catalog of 3D x, y, z positions in a readable format will work fine.

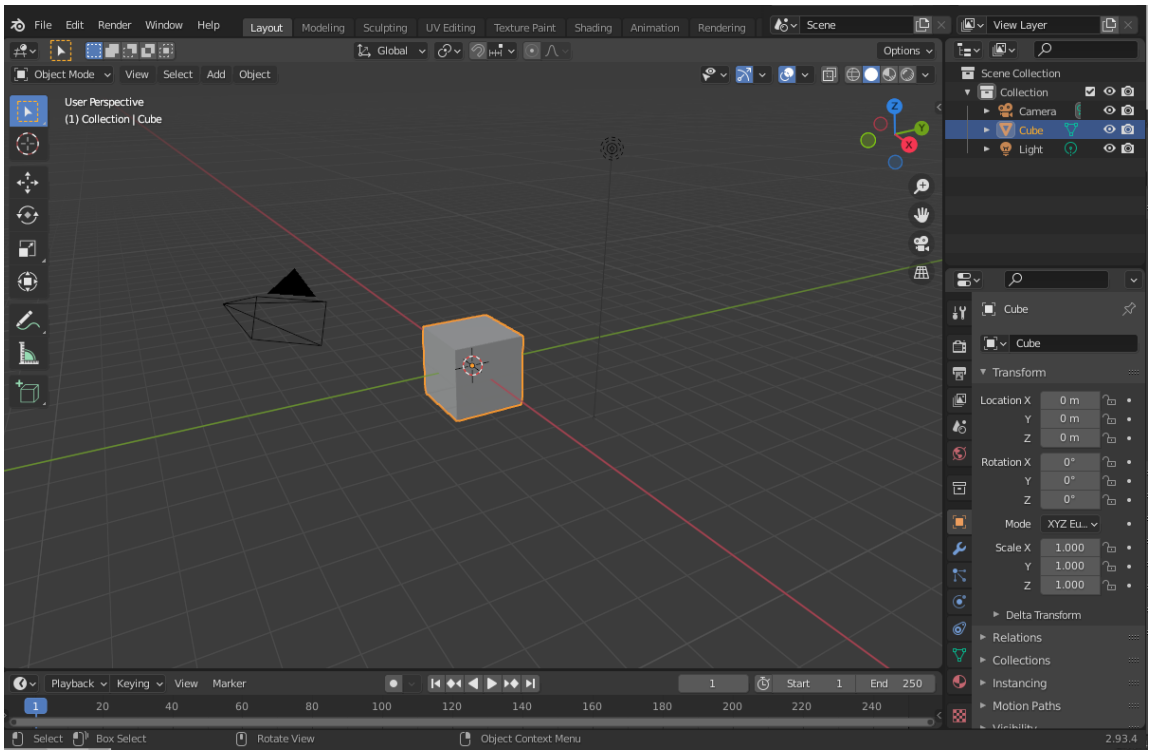
Basic knowledge of Python

Git repository: **git clone https://github.com/brkent/flatiron-sciware-blender-2021.git**

We can render the positions of objects in 3D space - in this case the locations of galaxies in the nearby Universe from the Extragalactic Distance Database (Tully et al.). This tutorial will use the following concepts:

- Working with the Blender interface, keystrokes, and mouse usage
  https://www.giudansky.com/images/articoli/2021/03/blender-infographic-sm-2500.png
- Adding mesh objects to our visualization space
- Understanding modifiers, constraints, and the particle generation system
  https://docs.blender.org/manual/en/latest/physics/particles/index.html
- Importing formatted text data into the Blender GUI with Python code and using the API
- Setting up a Cartesian grid for motion reference or a coordinate system
- Determining the camera positioning, movement, field-of-view, and keyframes
- Render output as a still image or animation.

The following steps will set up this visualization.



Your default screen in Blender will show the scene above - a solid cube mesh, camera, and lighting element. One can switch to different topical views with the tabs at the top. In the central viewer, the left mouse button selects objects (surrounded in an orange outline), and the middle mouse wheel can zoom, or rotate with a click. Shift-middle mouse wheel allows translation of the view.

To lock the user view to a camera, left click to select the camera and highlight it in orange. Pop out the Transform controls on the right hand side of the viewer, select View, and check Lock Camera to View.

Importing your Data

The most efficient way to create a large number of objects is to use a *Particle System* modifier. Left click to select the Cube mesh. The mesh will act as the particle emitter - think of a bunch of marbles being released

from our mesh cube. We control which of the six faces the marbles originate from, what their initial trajectory will be, any jitter randomness, the count number, longevity (think half-life) through the visualization, and what material properties will be assigned to them.

Creating thousands of objects mesh objects makes such a task impossible. Looping over objects is also rather inefficient. It is easier to use the Blender scripting interface and leverage a bit of Python code to control a particle system for our 3D catalog. Below shows a single function and boilerplate code for creating a particle system, and moving the particles to their appropriate locations. Click on the Scripting tab at the top of the screen, start a new file and give it a unique identifier name,, and copy the code below into the new file. You will need to change the file path to your local data.

```
import math
import glob
import csv
import numpy as np

import bpy

def particleSetter(self):
       particle_systems = object.evaluated_get(degp).particle_systems
       particles = particle_systems[0].particles
       totalParticles = len(particles)

       scene = bpy.context.scene
       cFrame = scene.frame_current
       sFrame = scene.frame_start

       # at start-frame, clear the particle cache
       if cFrame == sFrame:
           psSeed = object.particle_systems[0].seed
           object.particle_systems[0].seed = psSeed

       # Define the local copy of your text data file
        # Note that Windows will still use forward slashes like
        #    Mac and Linux.
       filepath = '/<path>/<to>/<your-file>/edd.txt'
       fields = ['name', 'dist', 'x', 'y', 'z']
```

```
        reader = csv.DictReader(open(filepath), fields, delimiter=' ')

        cataloglist = []

        for row in reader:
            cataloglist.append([float(row['x']), float(row['y']), float(row['z'])])

        catalog = np.array(cataloglist)

        #  However you choose to read in your data,
        #    it should end up being a numpy array
        #    and be put into a flattened list.
        flatList = catalog.ravel()

        # Set the location of all particle locations to flatList
        particles.foreach_set("location", flatList)

# Prepare particle system
object = bpy.data.objects["Cube"]
object.modifiers.new("ParticleSystem", 'PARTICLE_SYSTEM')
# This is the number of catalog elements in your data file.
object.particle_systems[0].settings.count = 3529
object.particle_systems[0].settings.frame_start = 1
object.particle_systems[0].settings.frame_end = 1
# We don't want our catalog to disappear, so set it to some arbitrary large value
#  that exceeds your number of anticipated rendered frames.
object.particle_systems[0].settings.lifetime = 10000
object.show_instancer_for_viewport = False
degp = bpy.context.evaluated_depsgraph_get()

#clear the post frame handler
bpy.app.handlers.frame_change_post.clear()

#run the function on each frame
bpy.app.handlers.frame_change_post.append(particleSetter)

# Update to a frame where particles are updated
bpy.context.scene.frame_current = 2
```

Run the script by clicking the Play (▸) button at the top of the screen. The data should load quickly and appear in the 3D viewer space.

- Click on the *Layout* tab at the top of the screen on the far upper left.
- Switch to View → Area → Toggle QuadView to see the 3D View space from multiple angles.
- Switch the upper right hand panel with View → Camera → Active Camera View

The particles are rendered as non-shaded point-like halos in the 3D view space, which is useful for graphic effects like smoke and fire. However, this does not translate into a rendered object. We need to create a single mesh object that will be duplicated among the particle points representing our catalog.

- Add a UV Sphere Mesh object to the 3D scene. It's position and scaling are irrelevant as it will be made invisible in both the scene and render, and scaled appropriately in the particle control menu.
- Add a new material (second icon from the bottom on the lower right panel), change the Surface to Emission, and a bright color of your choosing (See references for Colorimetry).
- Next, switch back to the Cube mesh object, and select the Particles panel on the right hand side. Open up the Render sub-menu. Choose Render as Object, uncheck Show Emitter, and choose the Sphere as the Instance object. This will put low-resolution spheres at every particle point. The default scaling under the Render menu is 0.05 - this should be fine, but adjust accordingly.
- Finally, for the particles, we need to return to the Modifier on the Cube object that we created with our Python script and Convert the particles to objects.

Adding a Reference Plane

- Add a plane with Add → Mesh → Plane. Scale the plane with the **S** key and press **TAB** to enter Mesh Edit mode. Subdivide the plane five times or more and press **TAB** one more time to again return to Object mode.
- Add a material to the plane mesh on the Properties panel and add a Wireframe Modifier. Scale the thickness to 10^{-4} . Choose a Material and Emitter color that contrasts well with the background—blue on black usually works well.

- Set the ‘Emission’ value to 1.5 or higher.
- Set the World tab background horizon color on the Properties panel to black.

Setting a Camera Path and Movement

We can now use the default camera object to point at an Empty Object. This will allow us to control where the camera points when we set our camera keyframes.

- Add an empty object with Add → Empty → Plain Axes for the camera to track.
- Left-click to choose the camera object and set the position and rotation values on the Transform toolbar to zero.
- Click the Constraints tab on the right-hand side Properties panel.
- Choose ‘Track To’ and select the target as ‘Empty’.
- The camera should correctly orient the upward vector in the +Z direction and the camera plane normal vector should always be looking at the Empty object.

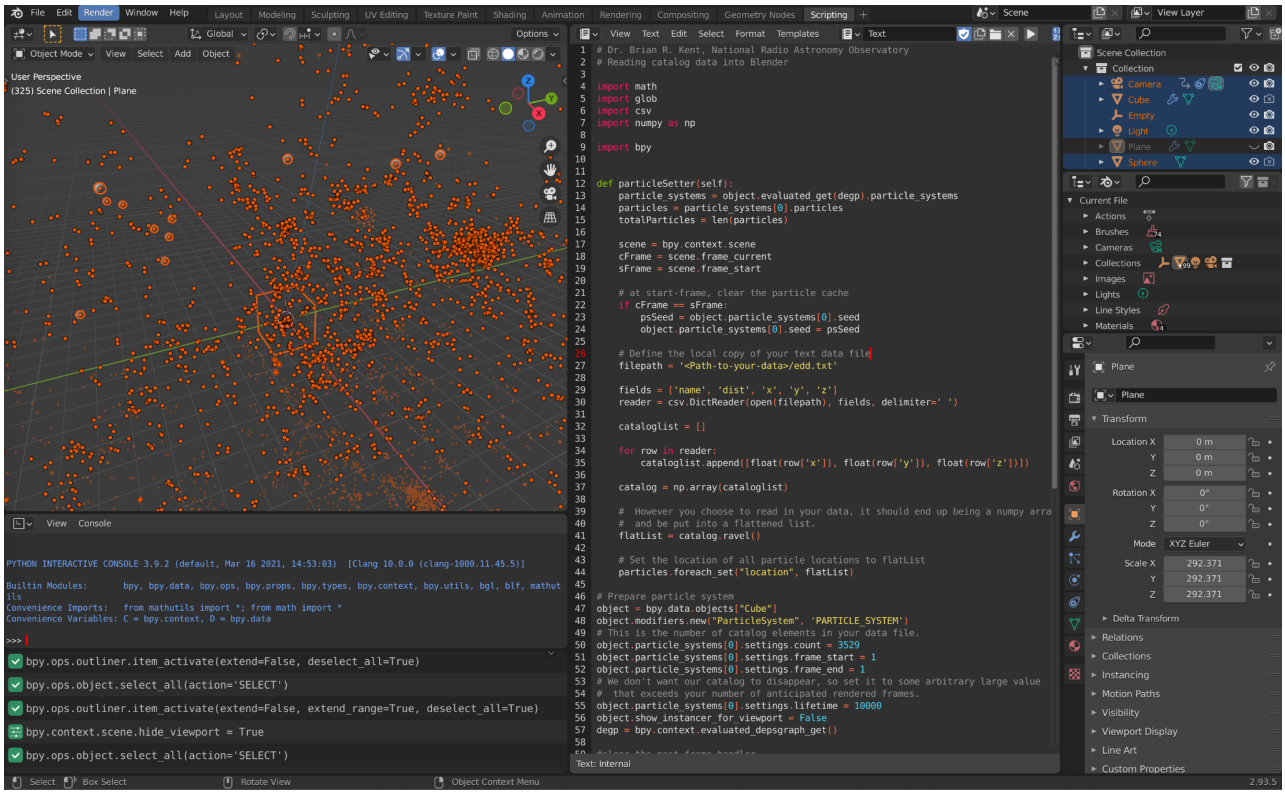
Animate the visualization by keyframing the camera.

- This test animation will be 20 s long at 30 frames per second. Set the number of frames to 600 and set the current frame to 1.
- Left-click to select the camera and press the I key to keyframe the position, rotation, and scale of the camera.
- On the Animation toolbar, set the current frame to 600. Move the camera in the 3D view port to a different location and orientation.
- Keyframe the camera position and rotation one final time with the I key.
- Set up as many keyframes as you like to “film” the angles of your catalog. Blender will automatically interpolate between the keyframes and accelerate/decelerate the movement accordingly.

The visualization can now be exported in a 1080p HD video.

- On the Output Properties tab the Properties panel and make sure the resolution is 1920x1080p. Set the frame rate to 24 or 30 frames per second.

- Set the output to AVI JPEG, quality 100 percent, and specify a unique filename on your local disk. Click the Render → Render Animation menu at the top of the tab to render the visualization.

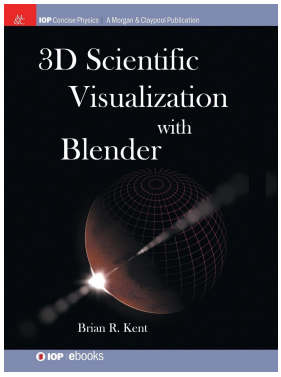


General Blender resources:

BlenderGuru
Blender Materials
BlenderDiplom
BlenderNation
CG Cookie
Tutorials and Modelling
Wiki Book
Blendtuts
Blender Models

Videos and Tutorials:

https://www.youtube.com/user/VisualizeAstronomy/videos



Publications:

3D Scientific Visualization with Blender:
https://iopscience.iop.org/book/978-1-6270-5612-0

Visualization in Astronomy:
https://iopscience.iop.org/journal/1538-3873/page/Techniques-and-Methods-for-Astrophysical-Data-Visualization

Extragalactic Distance Database:
https://ui.adsabs.harvard.edu/abs/2009AJ....138..323T/abstract

360 Video and Spherical Panoramas:
https://ui.adsabs.harvard.edu/abs/2017PASP..129e8004K/abstract

Visualization Colorimetry

- https://seaborn.pydata.org/
- https://seaborn.pydata.org/tutorial/color_palettes.html#general-principles-for-using-color-in-plots
- See references at the bottom of this page: https://medium.com/hipster-color-science/a-beginners-guide-to-colorimetry-401f1830b65a
- Talk by Dr. Michael Waters: https://www.dropbox.com/s/7s9seplrnw3ea7p/Practical_Colorimetry_for_Scientific_Visualization_-_Michael_J_Waters_-_2021_3_14.pdf