Tutorial

DEM: simulation and visualization

Putri Mustika Widartiningsih

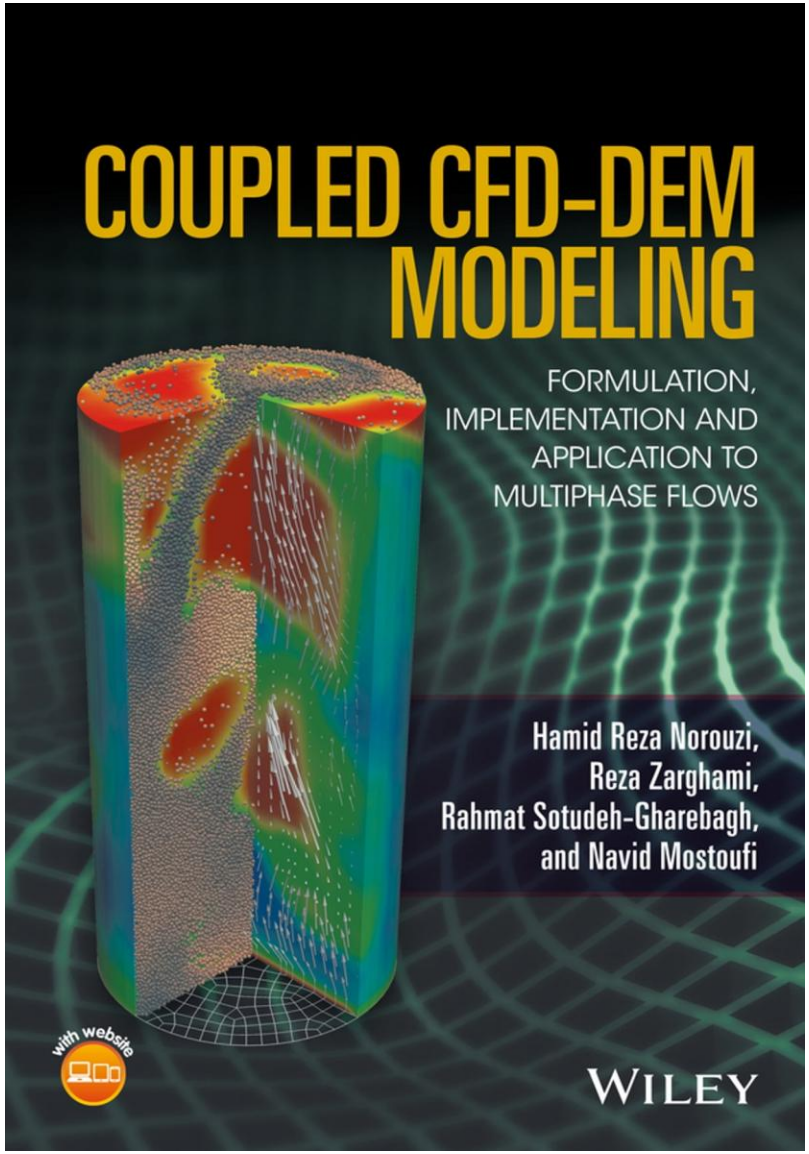
v.2.0

Agustus 2025

Prerequisites

- Visual Studio Code
- Blender 2.79b
- Paraview (any versions)

Reference



Coupled CFD-DEM Modeling: Formulation, Implementation and Application to Multiphase Flows

Hamid Reza Norouzi, Reza Zarghami, Rahmat Sotudeh-Gharebagh, Navid Mostoufi

ISBN: 978-1-119-00513-1
John Wiley & Sons, 2016

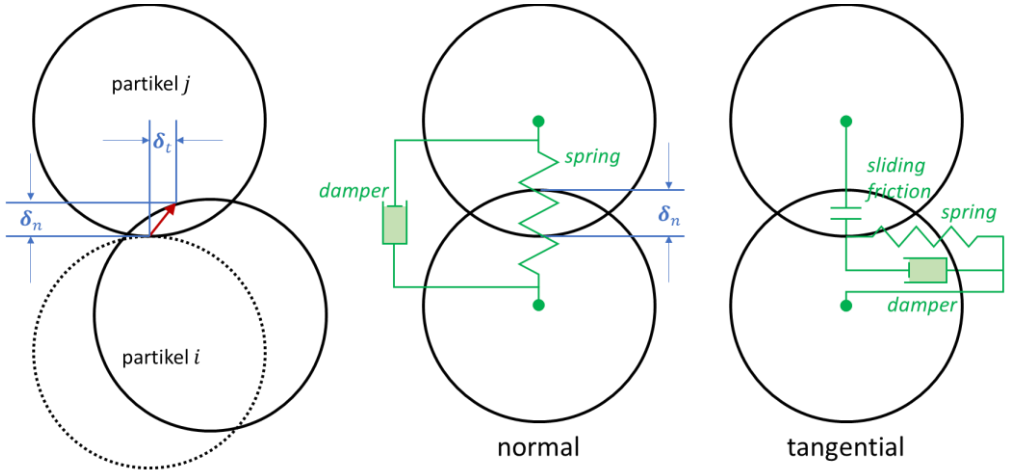
Governing equations

❑ Equations of motion

$$\textit{Translational} \quad m\ddot{\boldsymbol{x}} = \sum\left(\boldsymbol{F}_{C_n} + \boldsymbol{F}_{C_t}\right) + m\boldsymbol{g} + \boldsymbol{F}_{Ext} \qquad (1)$$

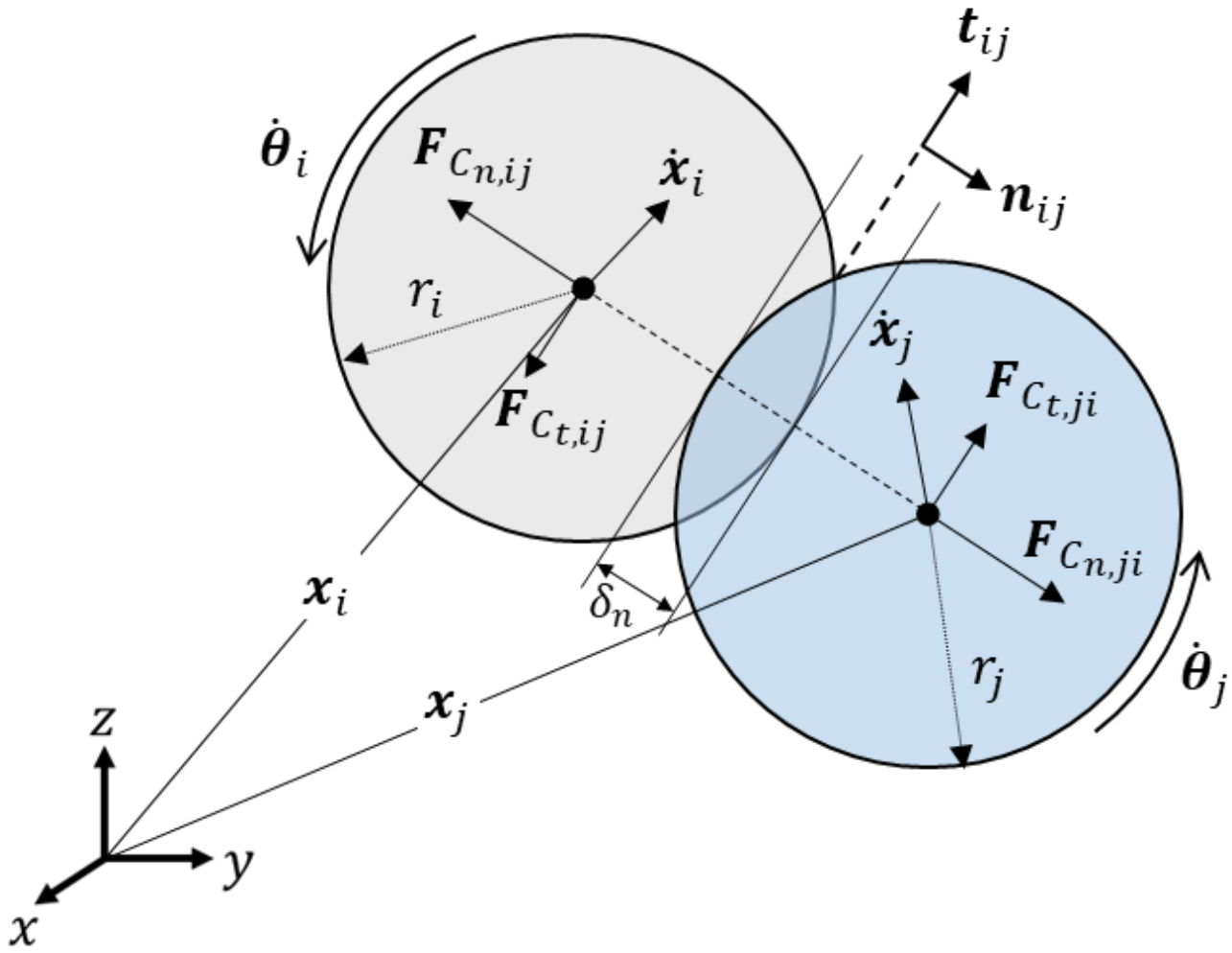
$$\textit{Rotational} \quad I\ddot{\boldsymbol{\theta}} = \sum\left(\boldsymbol{r} \times \boldsymbol{F}_{C_t}\right) + \sum \boldsymbol{M} \qquad (2)$$

❑ Contact forces



$$\textit{Normal} \quad \boldsymbol{F}_{C_n} = (-k_n\boldsymbol{\delta}_n - \eta_n\dot{\boldsymbol{x}}_n) \qquad (3)$$

$$\textit{Tangential} \quad \boldsymbol{F}_{C_t} = \begin{cases} -k_t\boldsymbol{\delta}_t - \eta_t\dot{\boldsymbol{x}}_t & \left(\left|\boldsymbol{F}_{C_t}\right| < \mu\left|\boldsymbol{F}_{C_n}\right|\right) \\ -\mu\left|\boldsymbol{F}_{C_n}\right|\dfrac{\dot{\boldsymbol{x}}_t}{|\dot{\boldsymbol{x}}_t|} & \left(\left|\boldsymbol{F}_{C_t}\right| \geq \mu\left|\boldsymbol{F}_{C_n}\right|\right) \end{cases} \qquad (4)$$



Forces acting on particle i and particle j during collision.

- **Normal component of contact force**

$$\boldsymbol{F}_{C_n} = (-k_n \boldsymbol{\delta}_n - \eta_n \dot{\boldsymbol{x}}_{rn})$$

- **Relative normal velocity**

$$\dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_j + \left(r_i \dot{\boldsymbol{\theta}}_i - r_j \dot{\boldsymbol{\theta}}_j\right) \times \boldsymbol{n}_{ij}$$

$$\dot{\boldsymbol{x}}_{rn} = \left(\dot{\boldsymbol{x}}_r \cdot \boldsymbol{n}_{ij}\right) \boldsymbol{n}_{ij}$$

- **Normal vector**

$$\boldsymbol{n}_{ij} = \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\left|\boldsymbol{x}_j - \boldsymbol{x}_i\right|}$$

- **Overlap**

$$\boldsymbol{\delta}_n = r_i + r_j - \left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|$$

- **Damping coefficient**

$$\eta_n = -2 \ln e \sqrt{\frac{m k_n}{\pi^2 + (\ln e)^2}}$$

*e coefficient of restitution

- **Tangential component of contact force**

$$\boldsymbol{F}_{C_t} = \begin{cases} -k_t \boldsymbol{\delta}_t - \eta_t \dot{\boldsymbol{x}}_{rt} & \left(\left|\boldsymbol{F}_{C_t}\right| < \mu \left|\boldsymbol{F}_{C_n}\right|\right) \\ -\mu \left|\boldsymbol{F}_{C_n}\right| \dfrac{\dot{\boldsymbol{x}}_{rt}}{|\dot{\boldsymbol{x}}_{rt}|} & \left(\left|\boldsymbol{F}_{C_t}\right| \geq \mu \left|\boldsymbol{F}_{C_n}\right|\right) \end{cases}$$

- **Relative tangential velocity**

$$\dot{\boldsymbol{x}}_{rt} = \dot{\boldsymbol{x}}_r - \dot{\boldsymbol{x}}_{rn}$$

- **Overlap (at time=s)**

$$\boldsymbol{\delta}_t^s = \boldsymbol{\delta}_t^{s-1} + \dot{\boldsymbol{x}}_{rt} \Delta t$$

- **Damping coefficient**

$$\eta_t = -2 \ln e \sqrt{\frac{m k_t}{\pi^2 + (\ln e)^2}}$$

❑ Moment of inertia (sphere)

$$I = \frac{2}{5} m r^2$$

❑ Rolling friction (DCT model)

$$M_{ij} = -\mu_{roll} r_{eff} \left| \boldsymbol{F}_{C_n} \right| \frac{\dot{\boldsymbol{\theta}}_j - \dot{\boldsymbol{\theta}}_i}{\left| \dot{\boldsymbol{\theta}}_j - \dot{\boldsymbol{\theta}}_i \right|}$$

$$r_{eff} = \left(\frac{1}{r_i} + \frac{1}{r_j} \right)^{-1}$$

❑ Update particle's position

$$\ddot{\boldsymbol{x}}^t = \frac{\sum \boldsymbol{F}^t}{m}$$

$$\dot{\boldsymbol{x}}^{t+1} = \dot{\boldsymbol{x}}^t + \ddot{\boldsymbol{x}}^t \Delta t$$

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \dot{\boldsymbol{x}}^t \Delta t$$

❑ Update particle's orientation

$$\ddot{\boldsymbol{\theta}}^t = \frac{\sum \boldsymbol{T}^t}{I}$$

$$\dot{\boldsymbol{\theta}}^{t+1} = \dot{\boldsymbol{\theta}}^t + \ddot{\boldsymbol{\theta}}^t \Delta t$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \dot{\boldsymbol{\theta}}^t \Delta t$$

DEM program (1 particle)

Get the code:
https://github.com/putrimustika/DEM-1-particle/blob/main/initial_velocity_x

```
import math
import matplotlib.pyplot as plt
import os

# particle properties
radius                = 0.01
density               = 1000
spring                = 10000
restitution           = 0.9
sliding_friction_coef = 0.3
rolling_friction_coef = 0.2
gravity               = -9.81
output_frequency      = 100

# Initial conditions [x, y, z]
position = [[0.0, 0.01, 0.0]]
velocity = [[0.5, 0.0, 0.0]]
rotSpeed = [[0.0, 0.0, 0.0]]
rotAngle = [[0.0, 0.0, 0.0]]
displacement_t = [0.0, 0.0, 0.0]
particle = []

# ------------------------
mass    = 4/3 * math.pi * radius**3 * density
inertia = 2/5 * mass * radius**2
damping = -2 * math.log(restitution) * math.sqrt(mass*spring/(math.pi**2 +
(math.log(restitution)**2)))

# simulation properties
dt = 0.00001
iteration = 30000
```

Set particle properties

Number of output files = iteration / output_frequency

Set particle's initial conditions (x, y, z)

Set time step

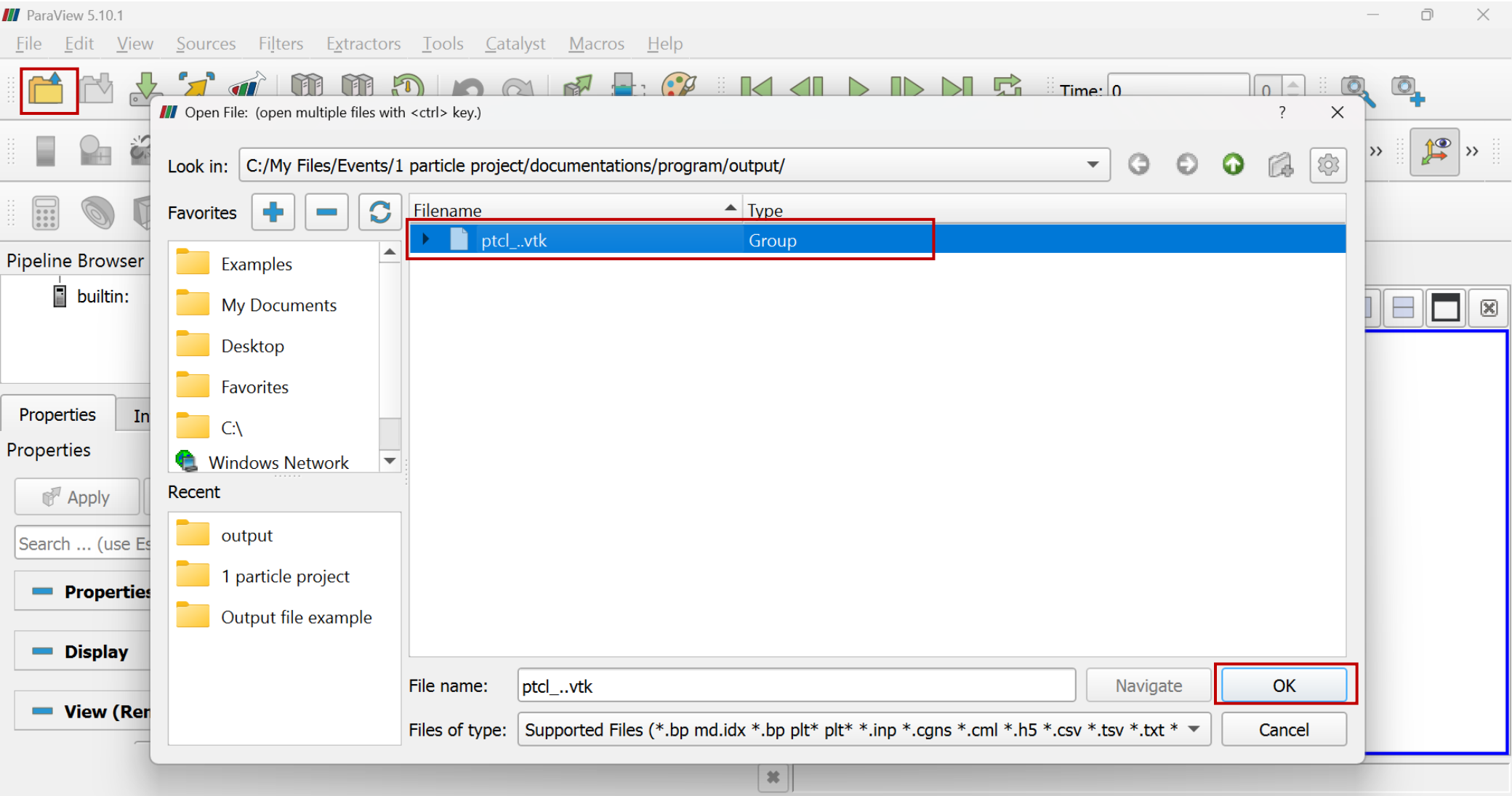
❑ Run the program

Once the program has finished running, confirm that:

1. A file named motion_data.txt is generated.
2. A folder named ‘output’ is created, containing .vtk files.
    - The number of files must equal $\frac{iteration}{output\_frequency}$

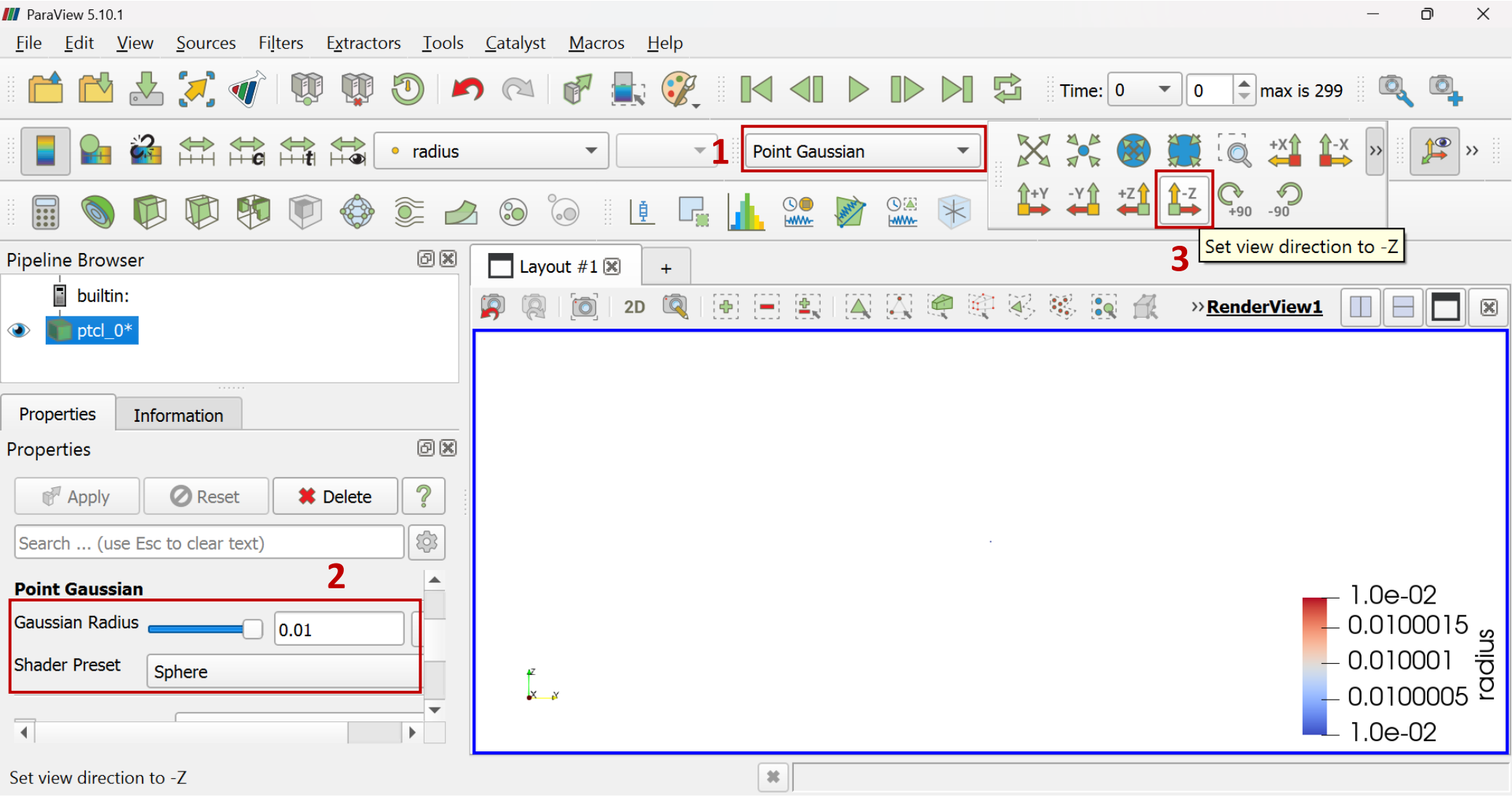
Quantitative and qualitative analyses using ParaView

1. Open **Paraview 5.10.1** (or later versions)
2. Open the **.vtk files**
3. Click **'Apply'**

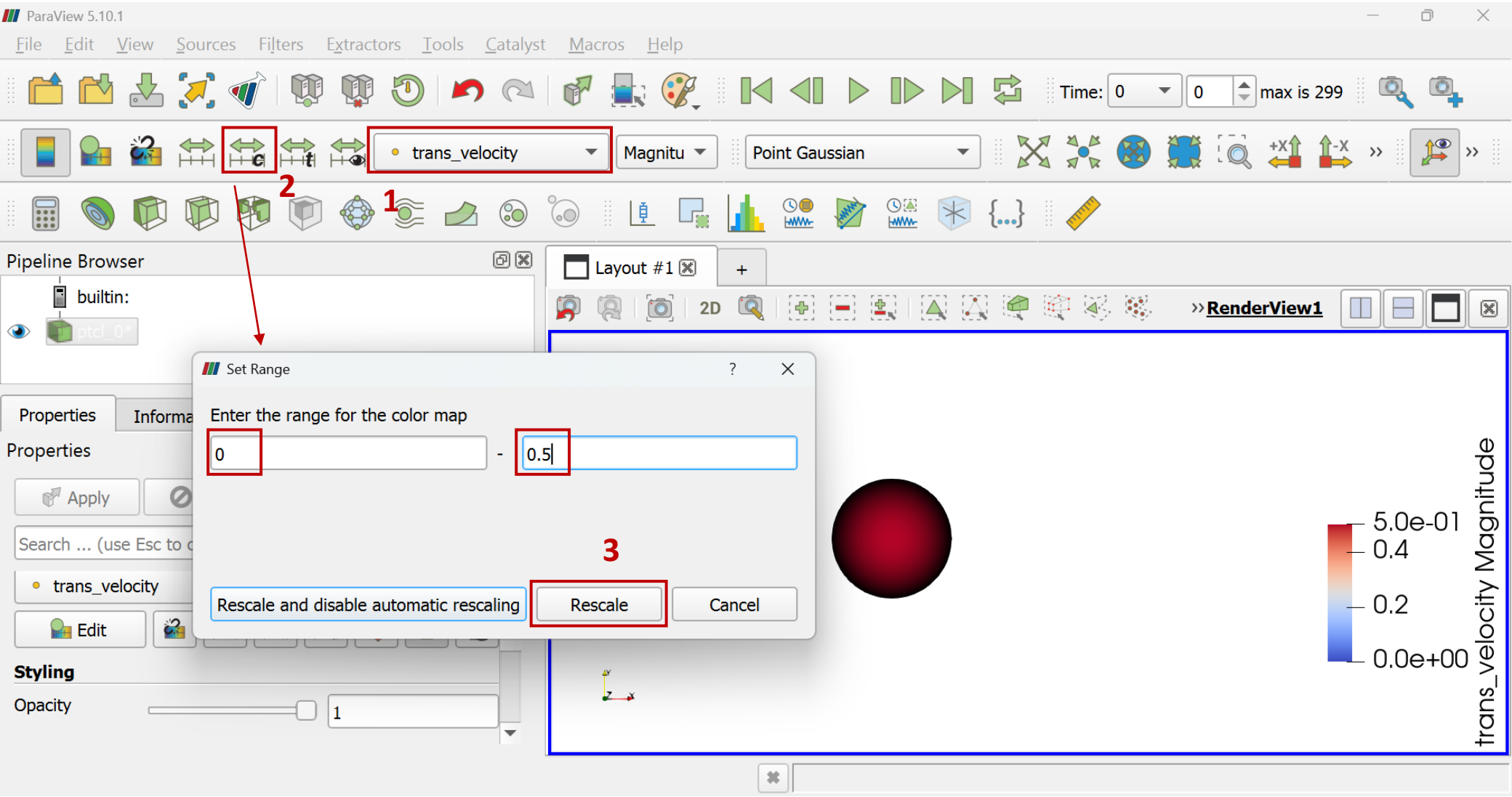


1. Representation → **Point Gaussian**
2. Gaussian radius → **set to the actual particle radius**, Shader preset → **Sphere**
3. Set view direction to **–Z**

   Tip: scroll up to zoom-in

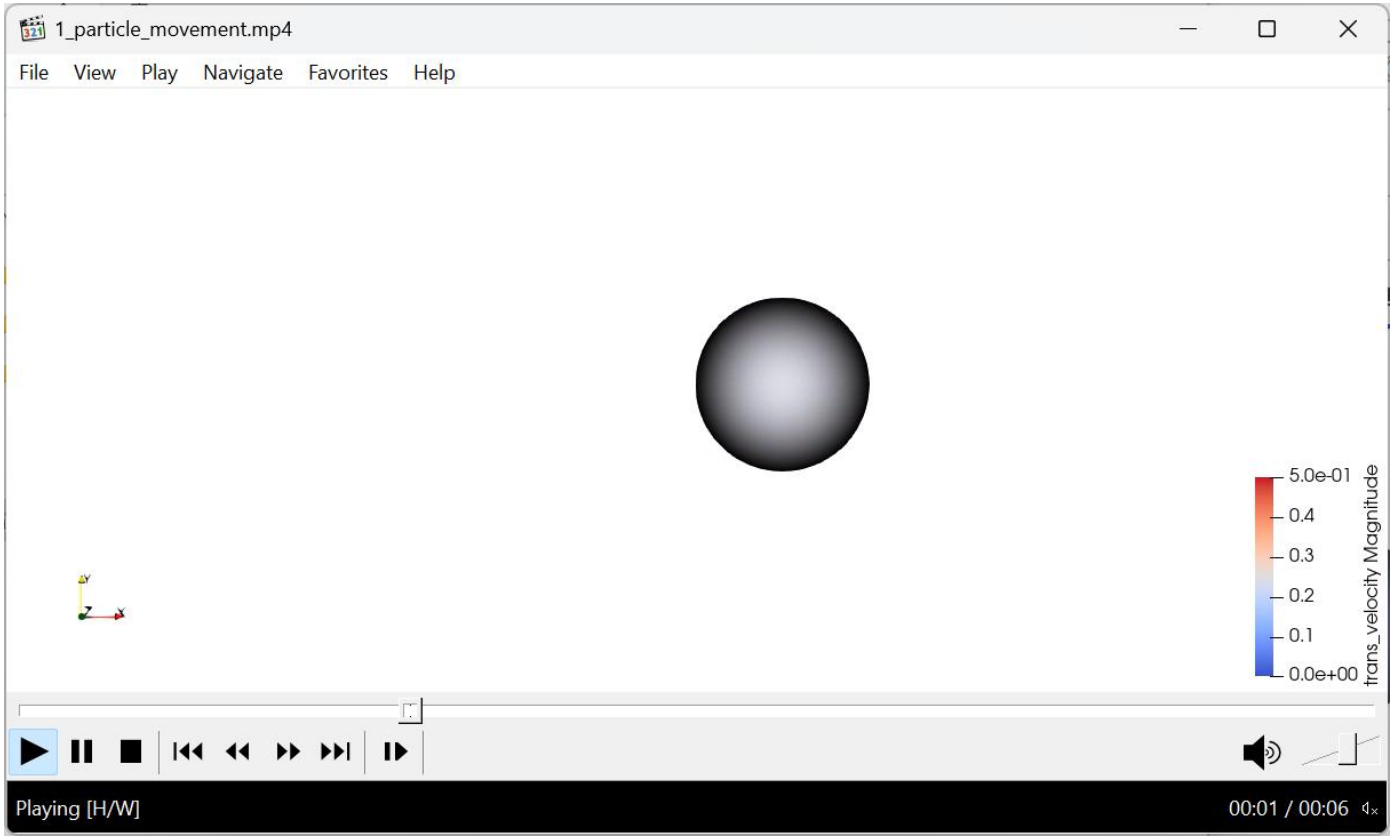


1. Color the sphere based on its **velocity magnitude**
2. Adjust the scale (e.g. 0 to 0.5 m/s)
3. Click **Rescale**
4. Click Play button ▶ to run the animation



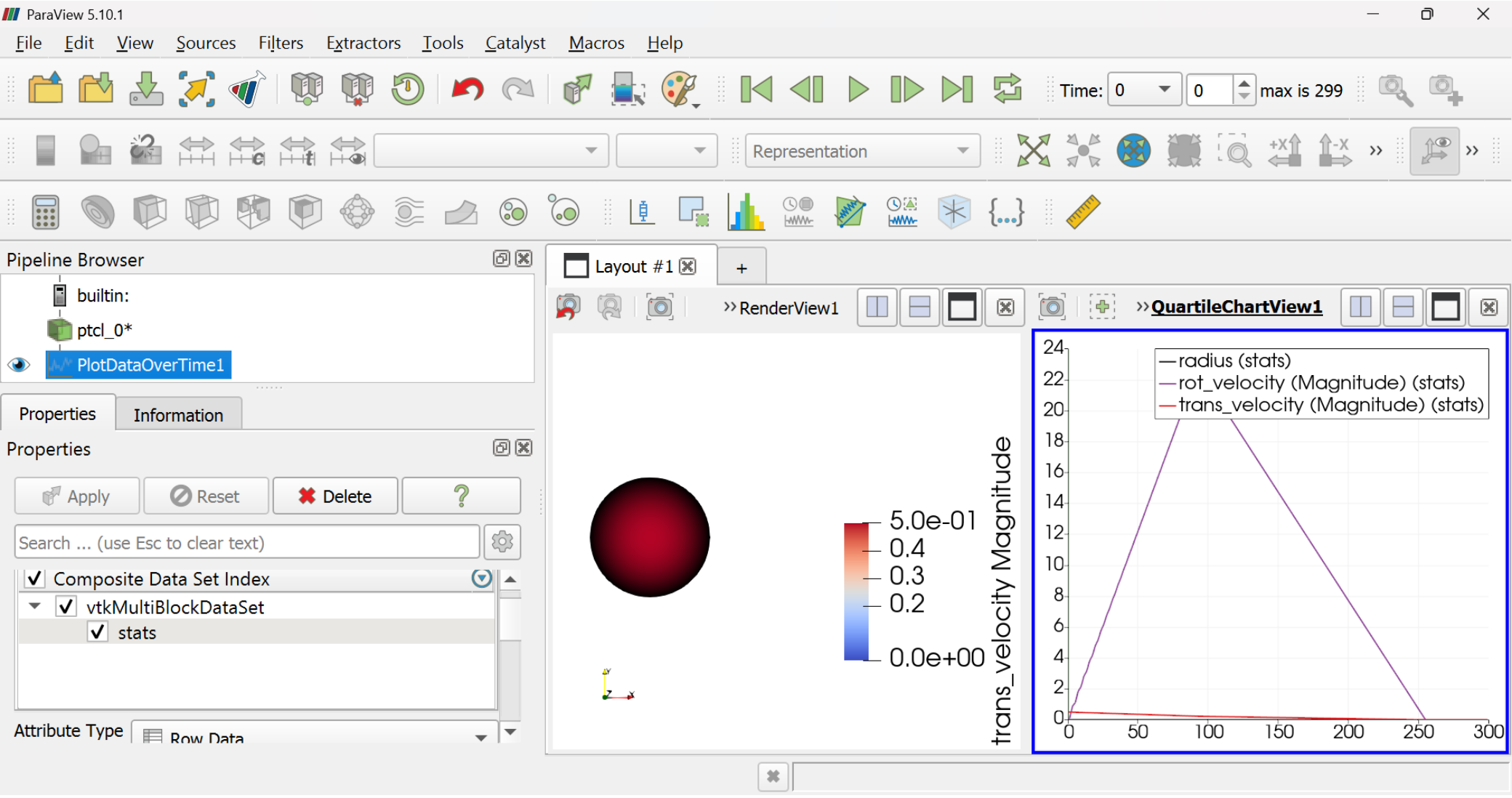
❑ Save the animation

1. Go to **File → Save Animation**
2. Set frame rate (e.g. 50 fps) → **OK**
3. Animation successfully created



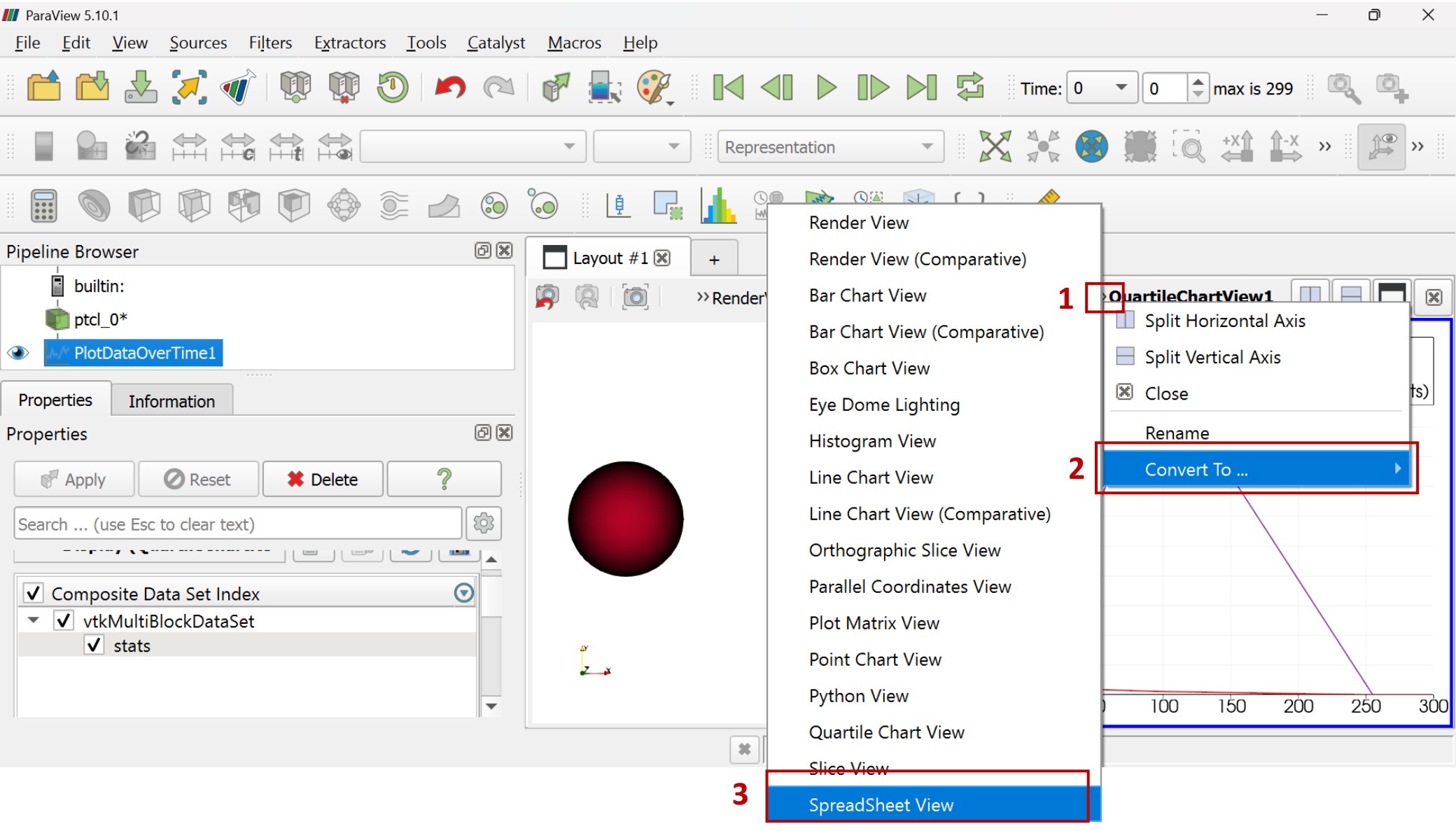
❑ Quantitative analysis: Plot data

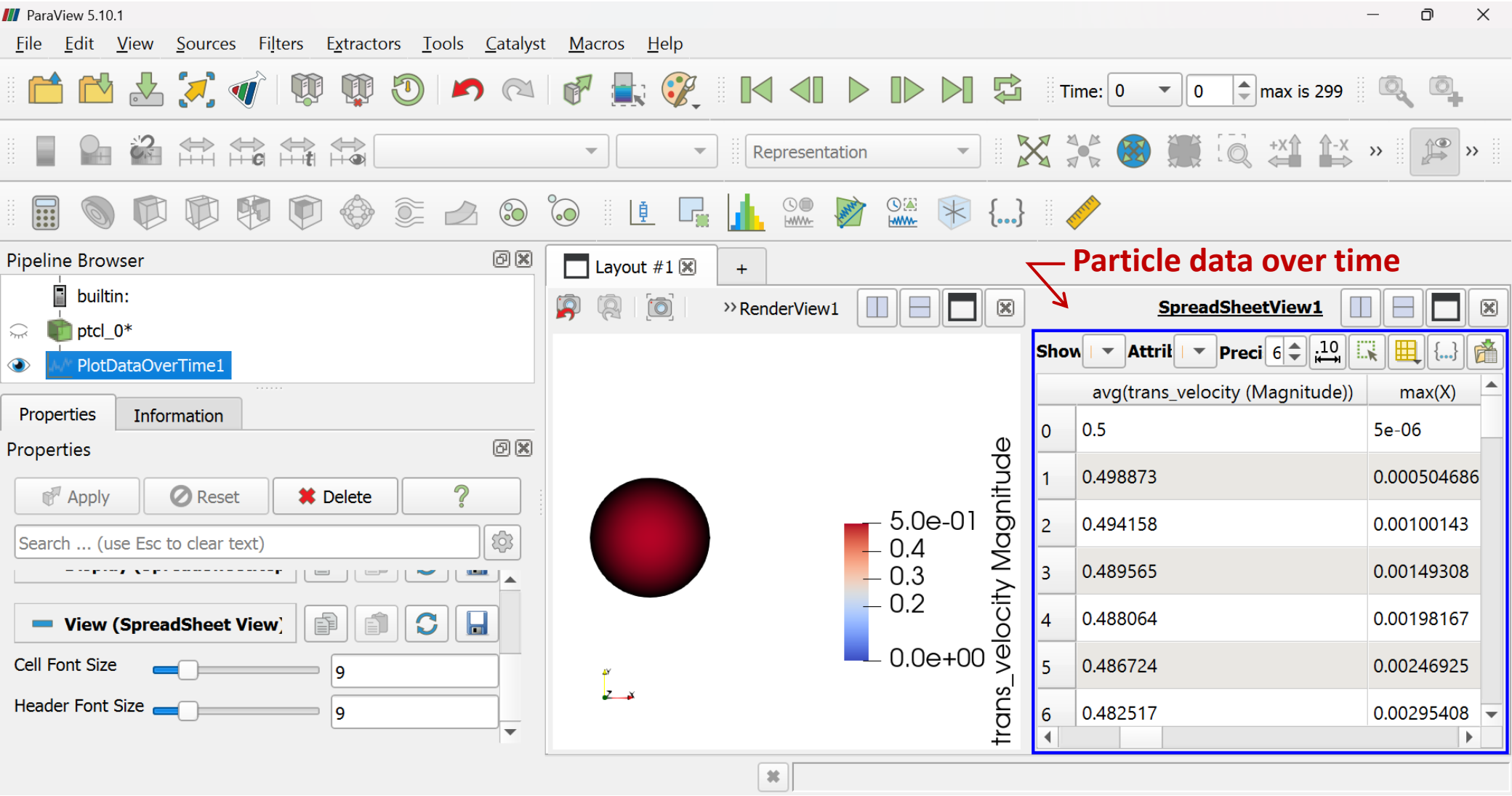
Go to **Filter → Data Analysis → Plot Data Over Time → Apply**



❑ Quantitative analysis: Extract data

1. Right click ›› **QuartileChartView1**
2. Convert to → SpreadSheetView

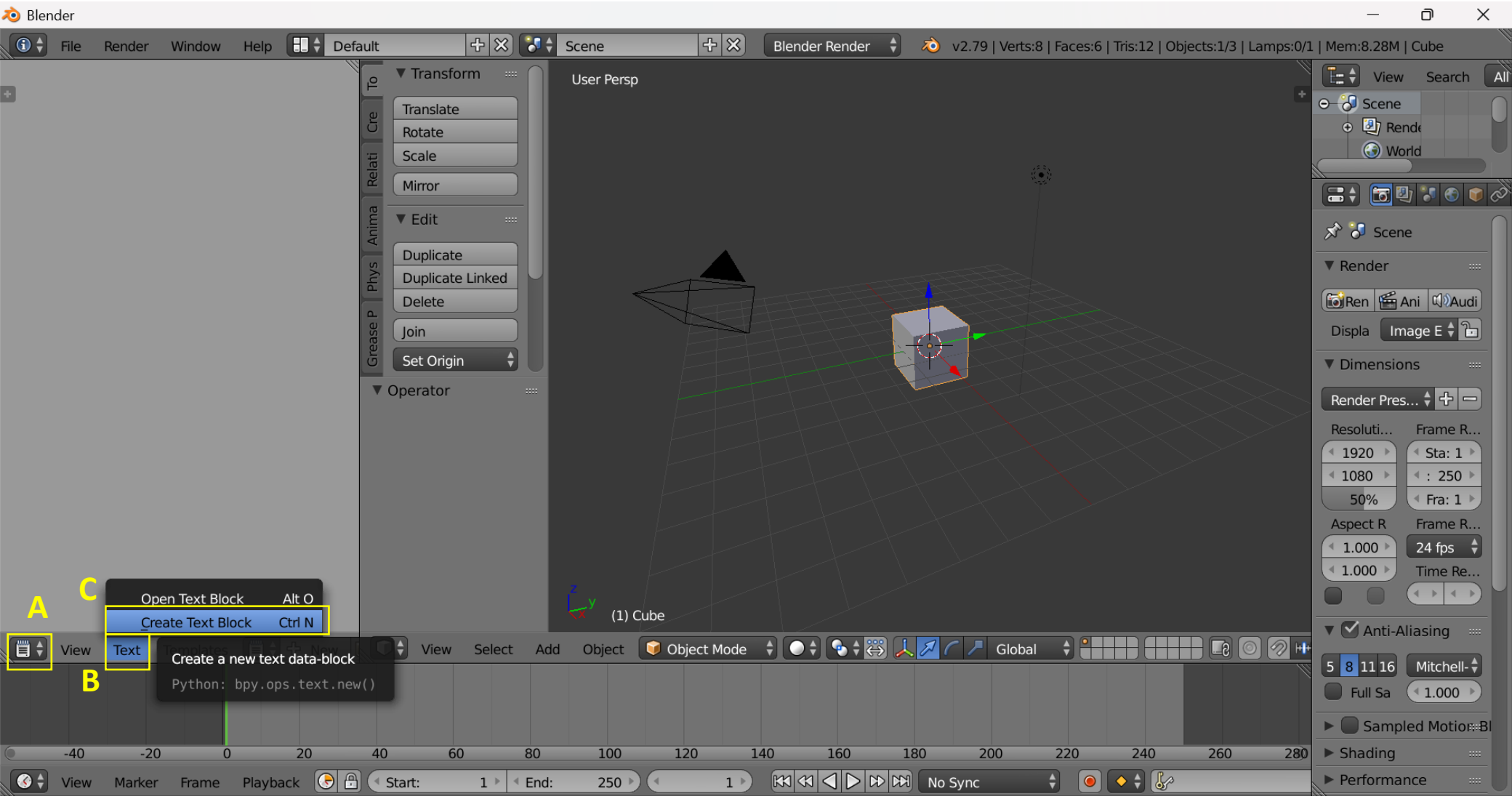



ParaView 5.10.1
File Edit View Sources Filters Extractors Tools Catalyst Macros Help
Time: 0 0 max is 299
Representation
Pipeline Browser
builtin:
ptcl_0*
PlotDataOverTime1
Properties Information
Properties
Apply Reset Delete
Search ... (use Esc to clear text)
View (SpreadSheet View)
Cell Font Size 9
Header Font Size 9
Layout #1
RenderView1
5.0e-01
0.4
0.3
0.2
0.0e+00
trans_velocity Magnitude
Particle data over time
SpreadSheetView1
Show Attrib Preci 6
avg(trans_velocity (Magnitude)) max(X)
0 0.5 5e-06
1 0.498873 0.000504686
2 0.494158 0.00100143
3 0.489565 0.00149308
4 0.488064 0.00198167
5 0.486724 0.00246925
6 0.482517 0.00295408

Realistic visualization using Blender

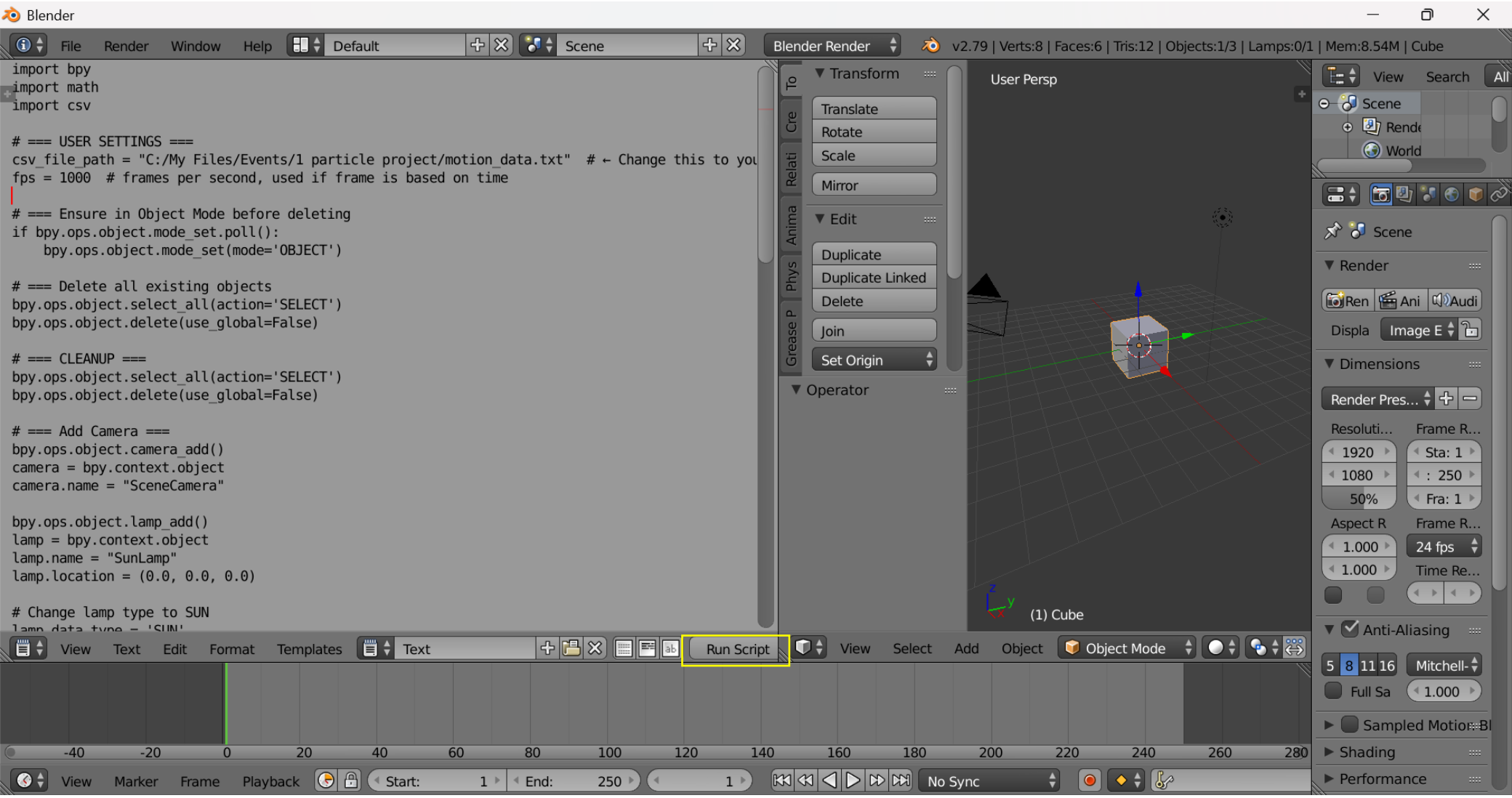
Get the code:
https://github.com/putrimustika/DEM-1-particle/blob/main/1_particle_visualization

Open **Blender 2.79**

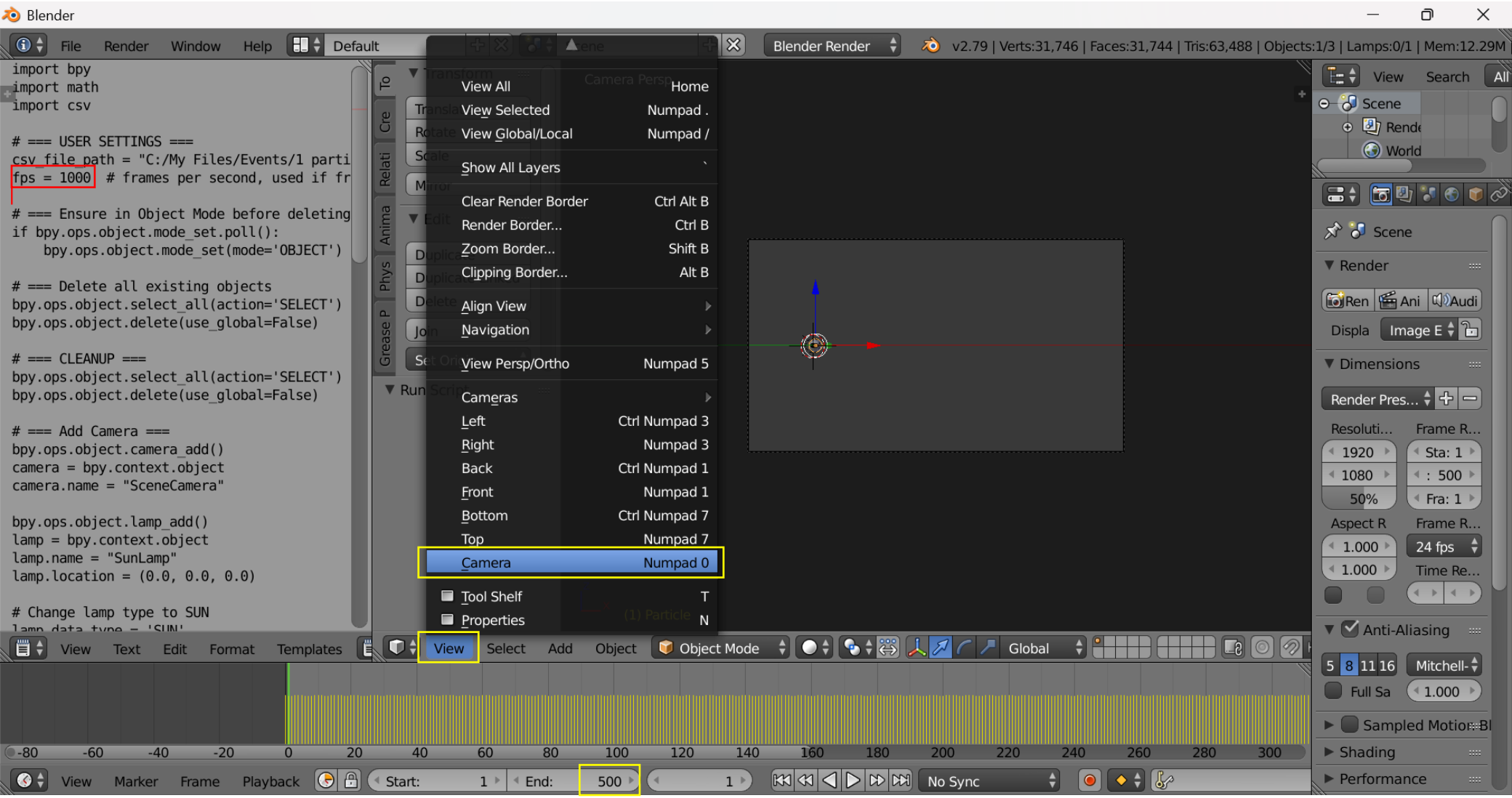


1. Split the window into 2.
2. Change the left window view to **Text Editor**
3. Click **Text → Create Text Block**

1. Paste the code.
2. Set ‘csv_file_path’ to the directory on your PC that contains ‘motion_data.txt’ → **Run Script**.



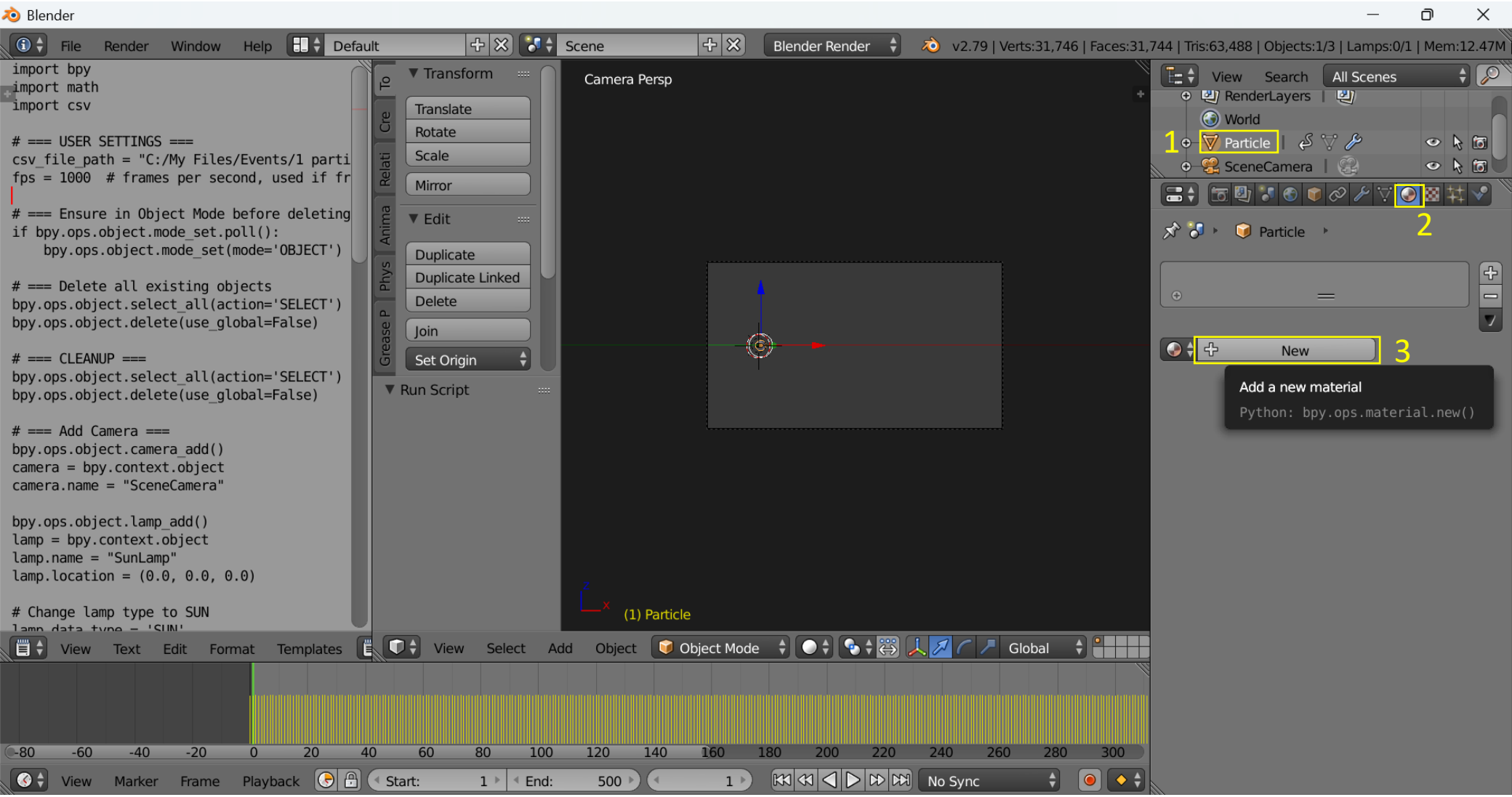
1. Change ‘end’ frame to the actual frame number.
2. Frame number = iteration number / fps (e.g. 500000 / 1000 = 500)



3. Click **View → Camera**
or press Numpad 0

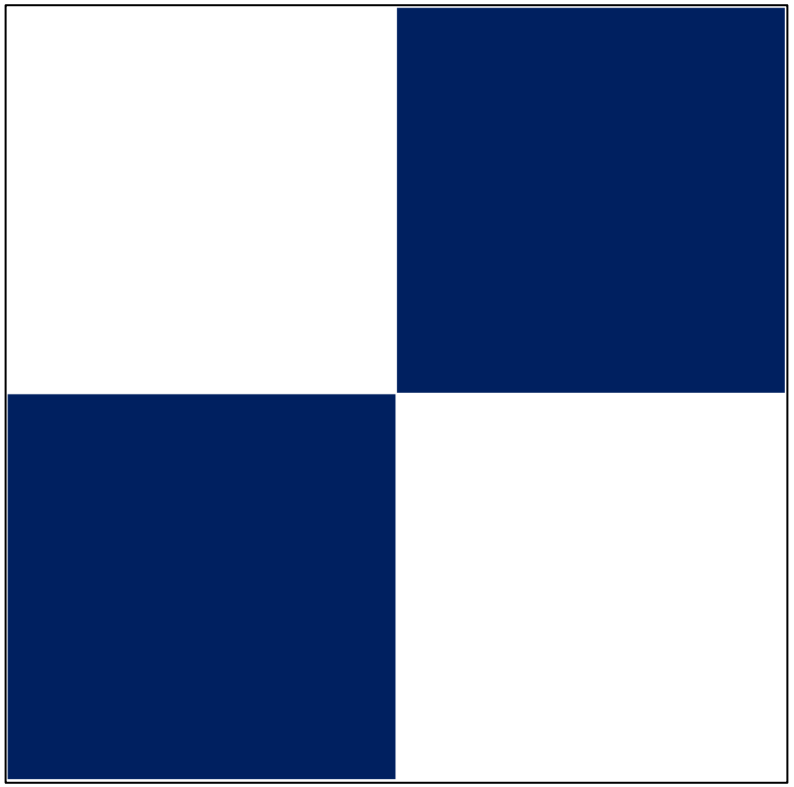
Assign particle material:

Select **Particle → Material → New** (Add a new material)



Prepare an image to color the particle.
Tip: Including patterns in the image helps visualize the rotational motion of particles.

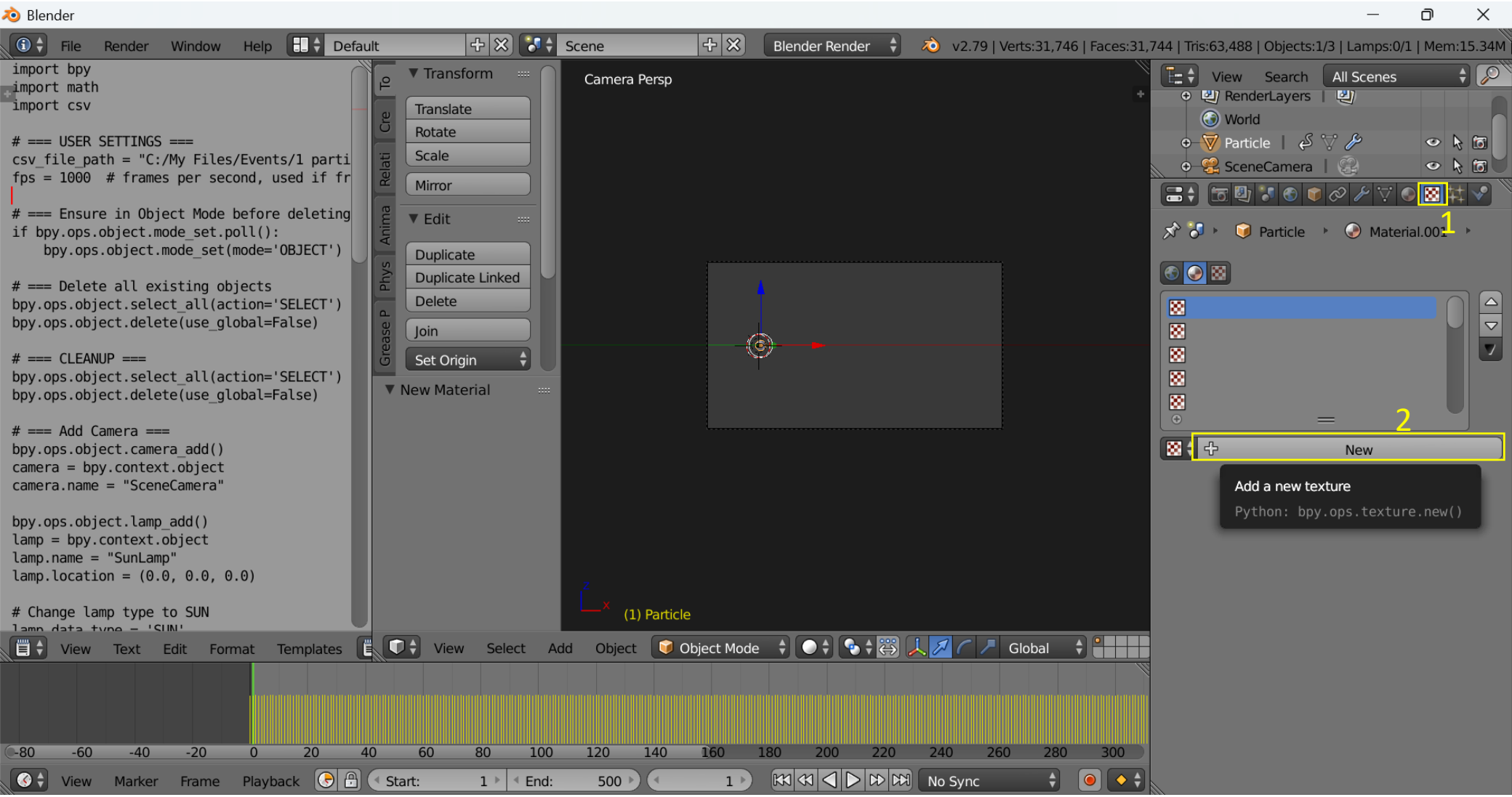
Example:



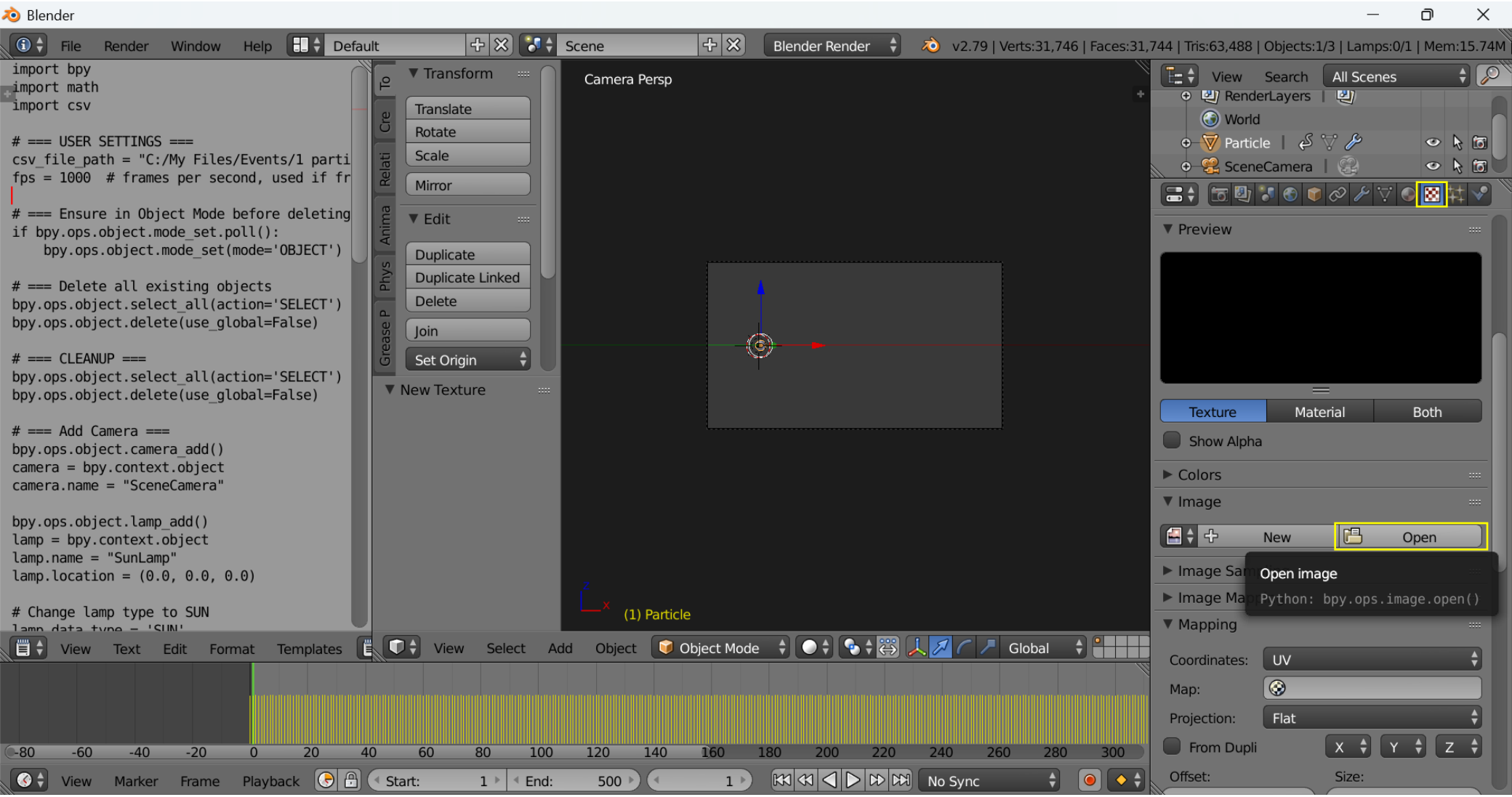
Save image as:
ptcl_pattern.png

Paint the sphere with the image.

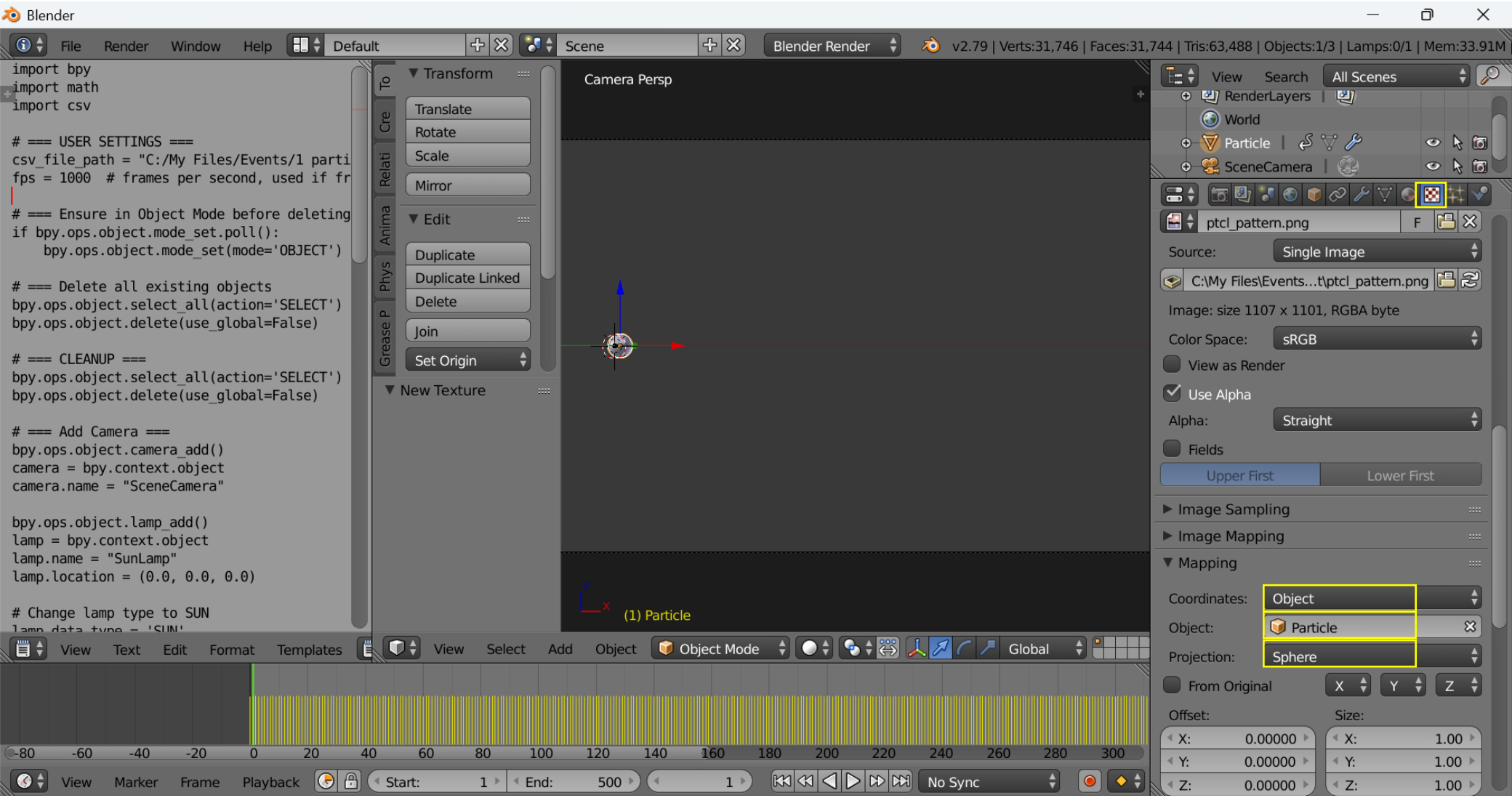
Go to **Texture → New** (Add a new texture)



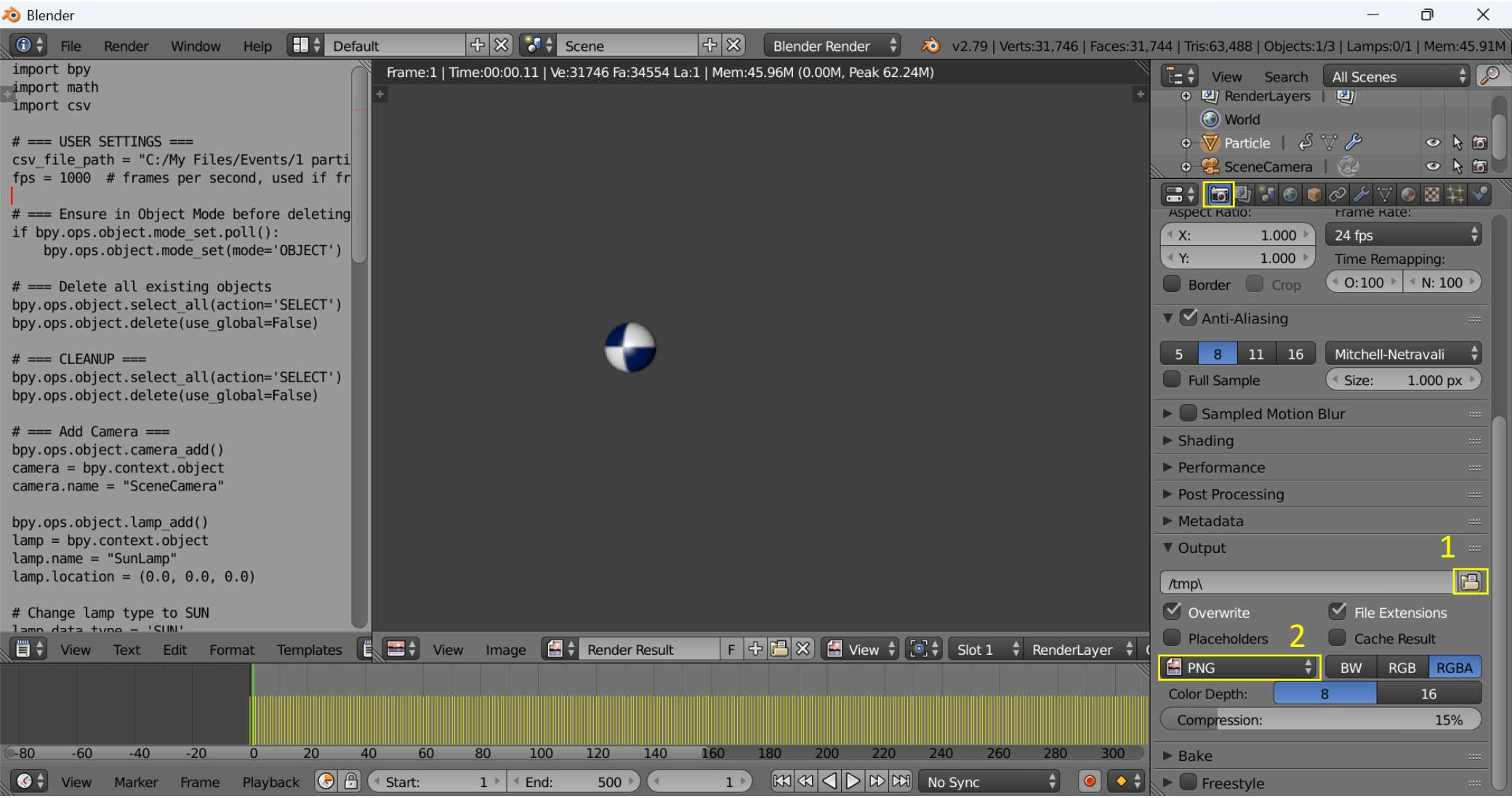
Scroll down → **Open** → ptcl_pattern.png



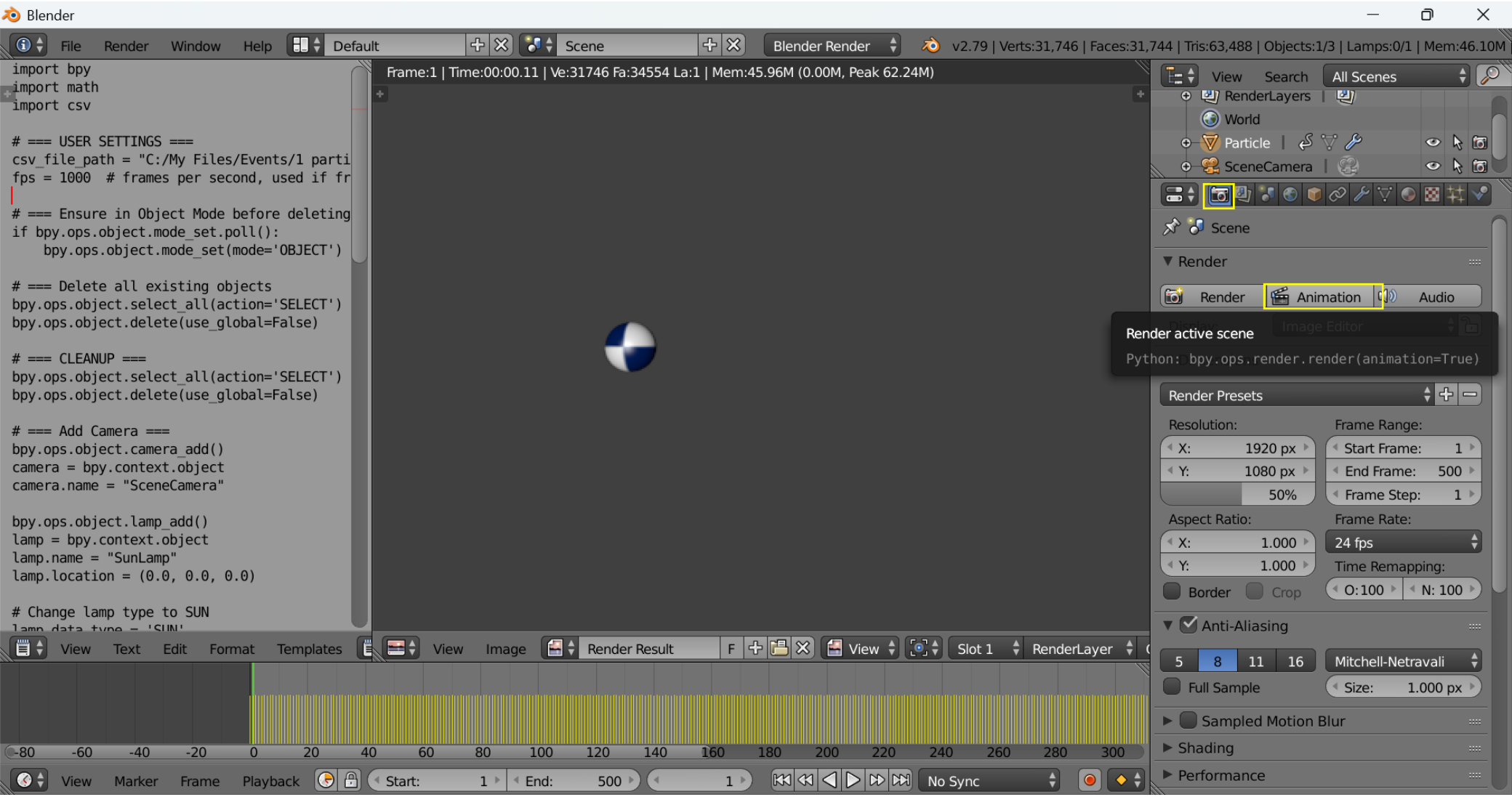
Coordinates → Object
Object → Particle
Projection → Sphere
Press **F12** to render image



1. Click **Open** and select a folder to locate the rendered images → **Accept**
2. File format → **PNG**

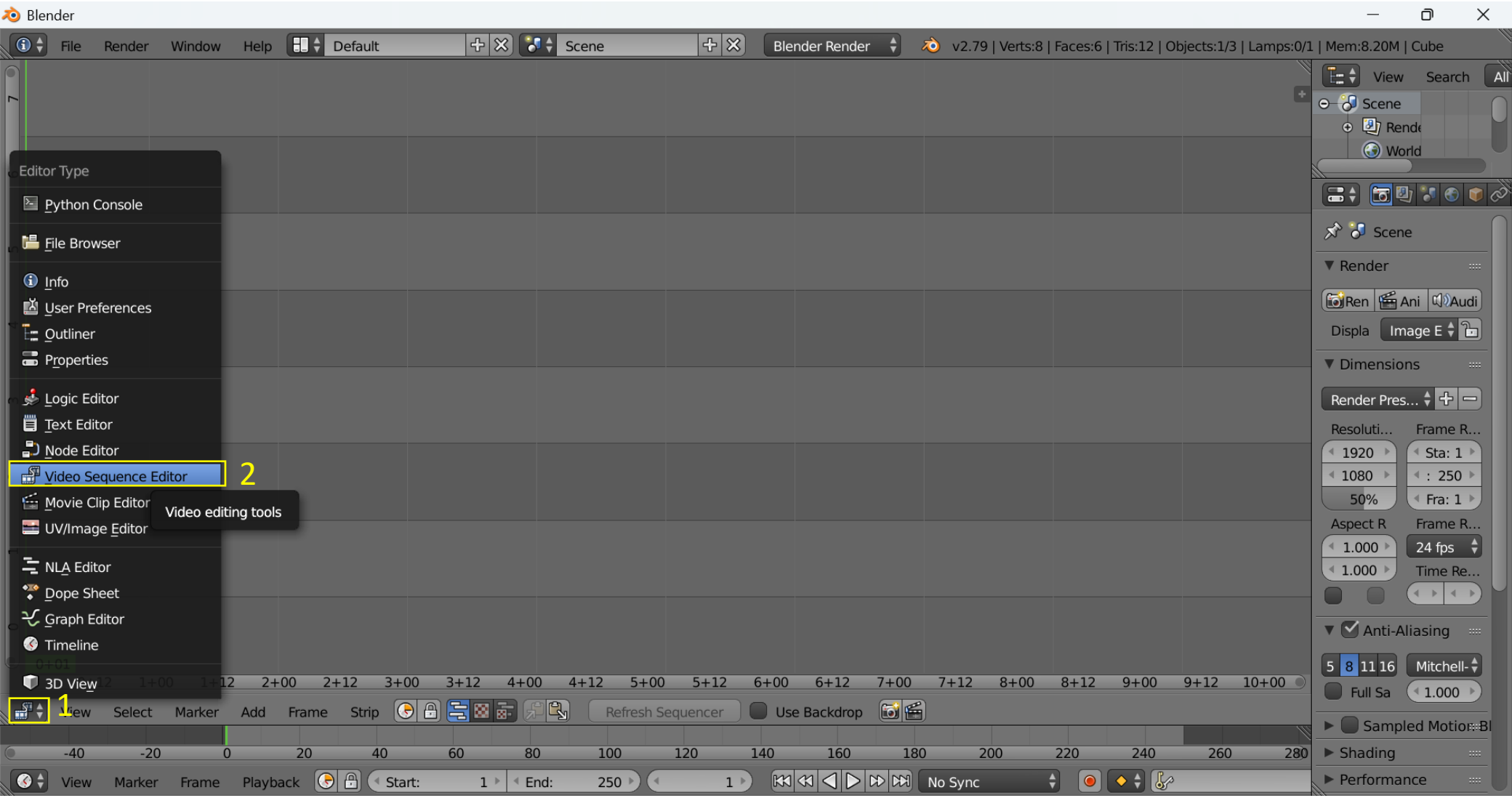


Click **Animation** to render all frames (1 to 500).
The rendered images are saved in the folder selected in the previous step.



Create movie / animation

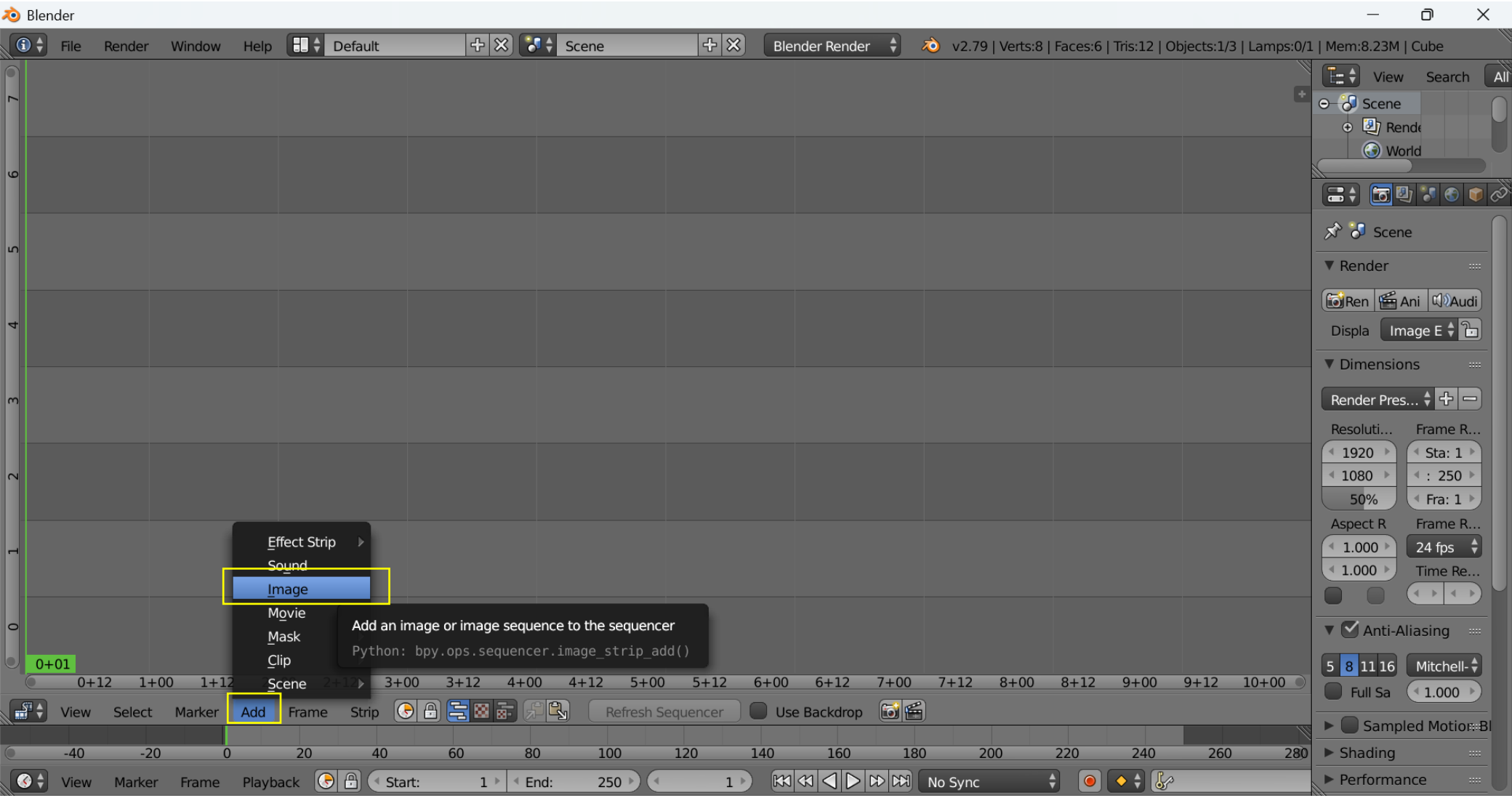
Open a new Blender window



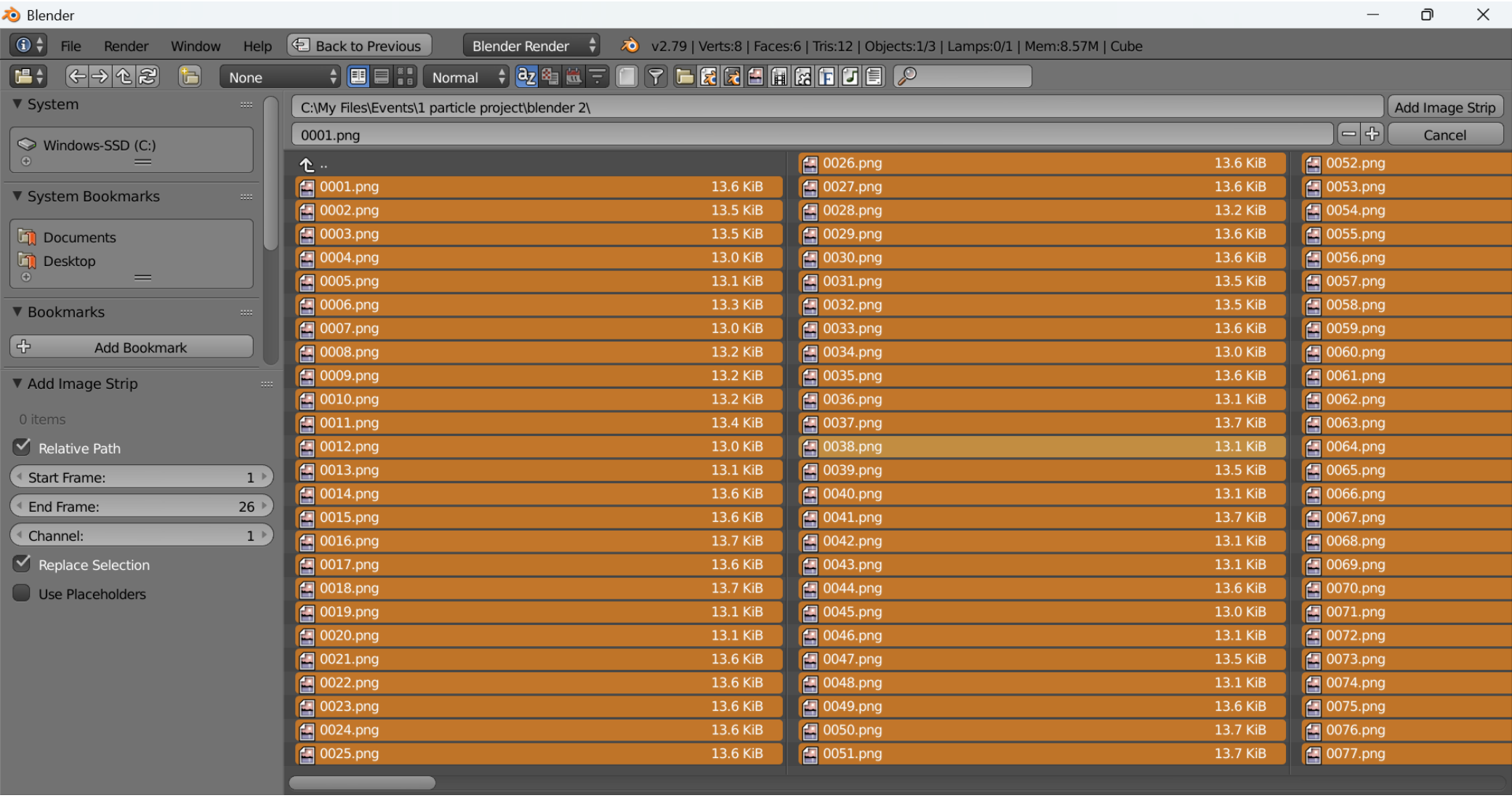
Environment → **Video Sequence Editor**

Click **Add → Image**

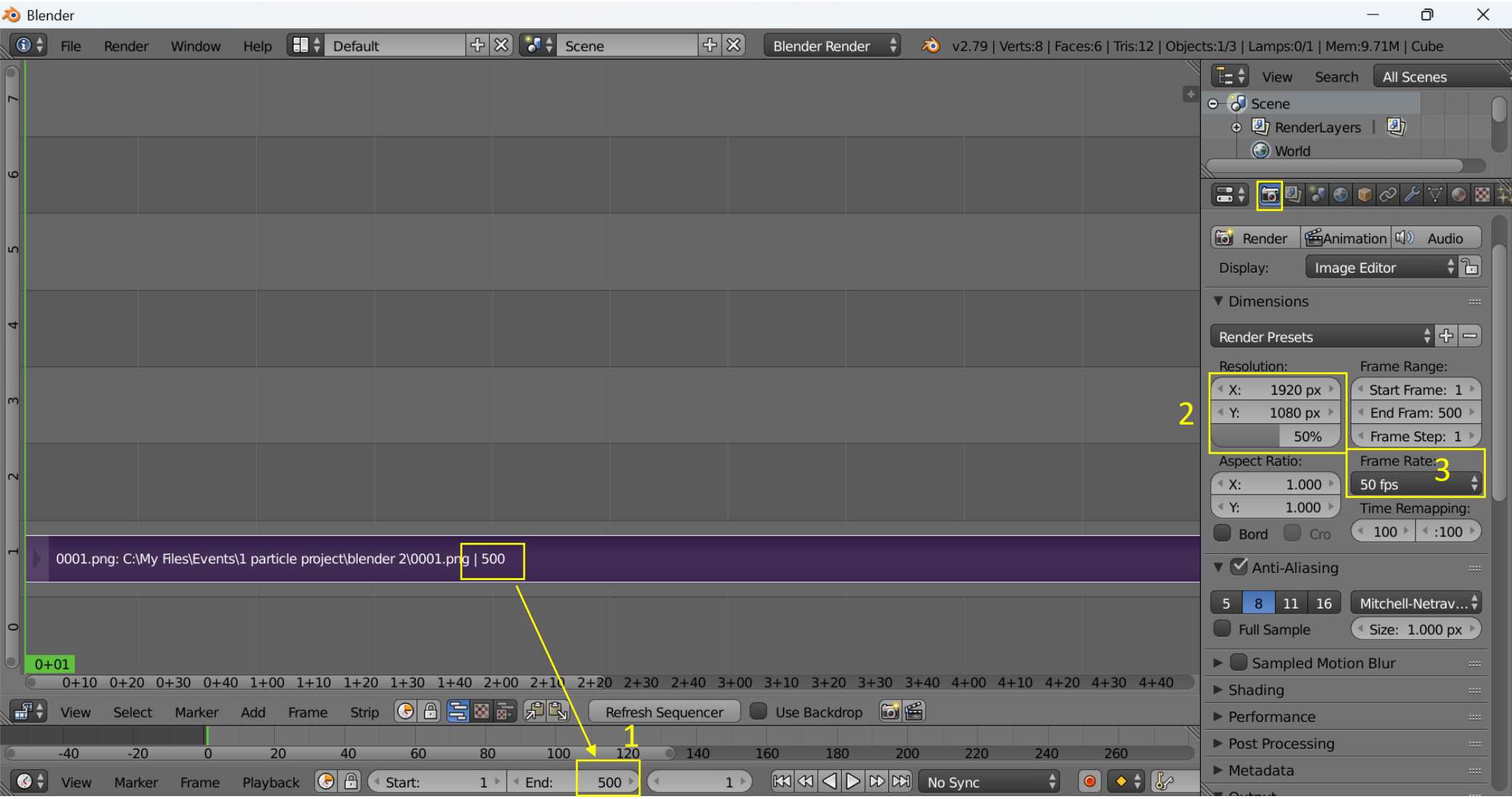
Load all images rendered in the previous step.



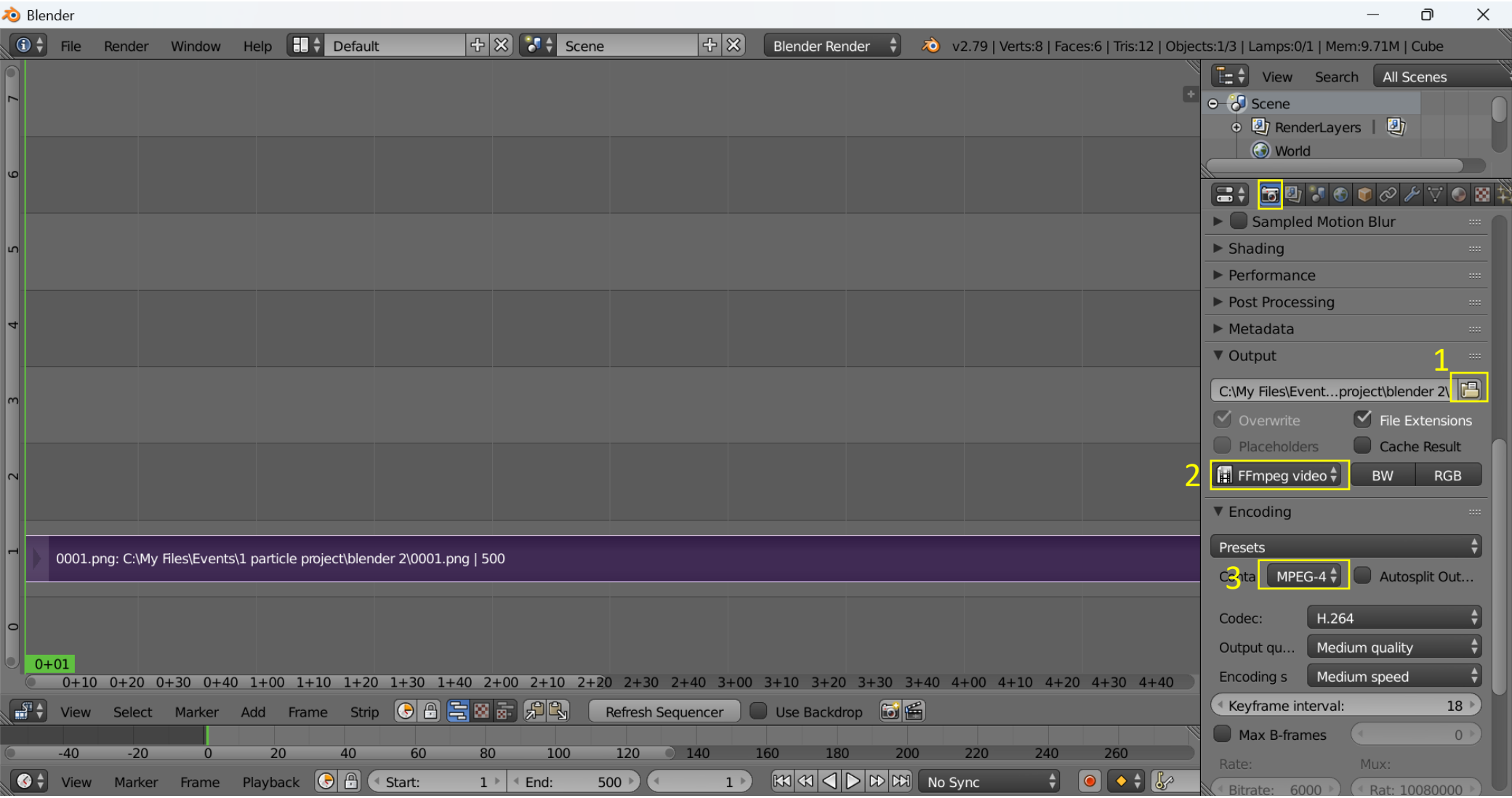
Tip: Press ‘A’ to select all images.
Click **Add Image Strip**



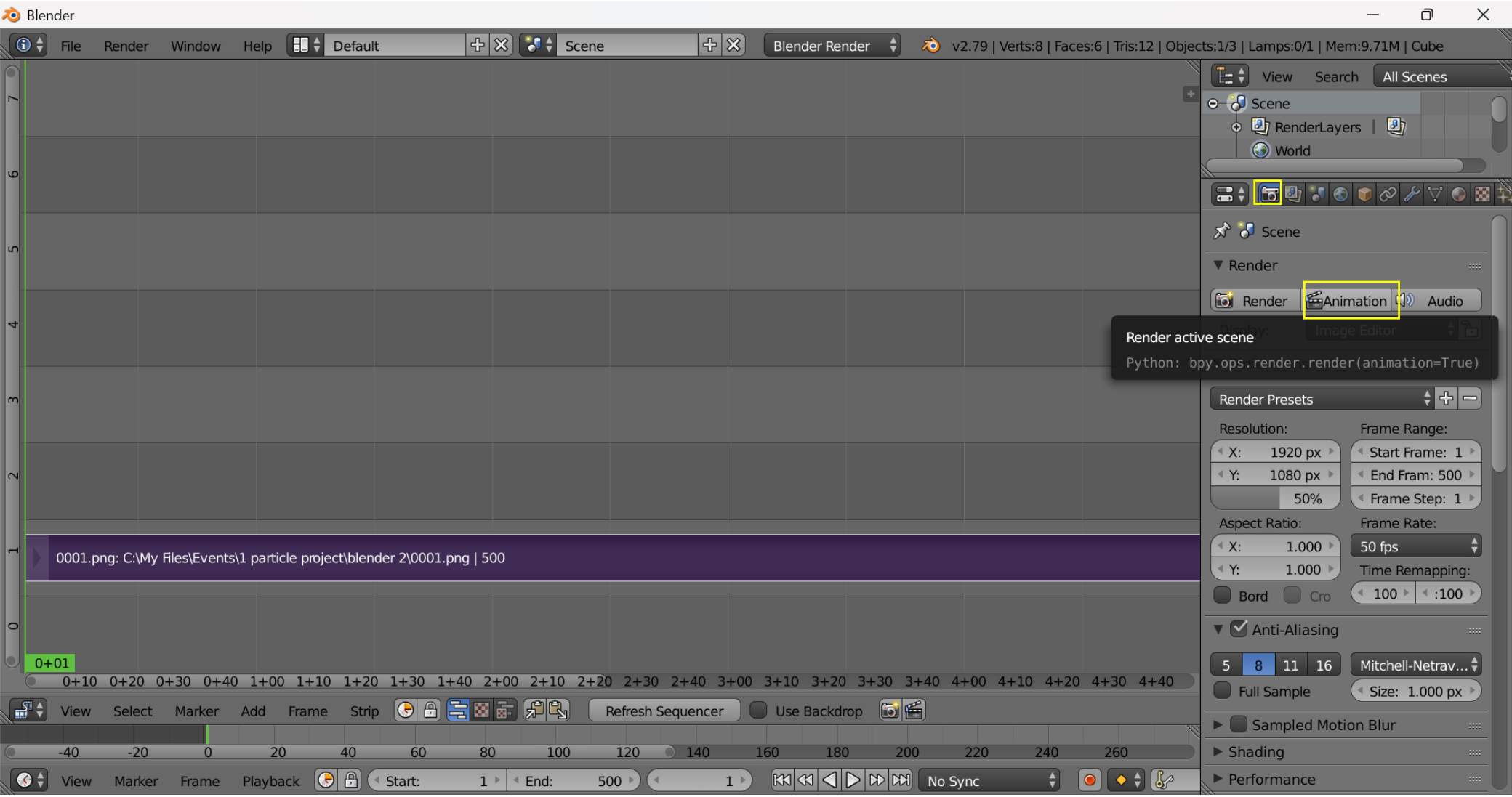
1. Set the ‘end’ frame according to the number of images.
2. Set the Resolution according to the image resolution.
3. Frame rate: **50 fps**



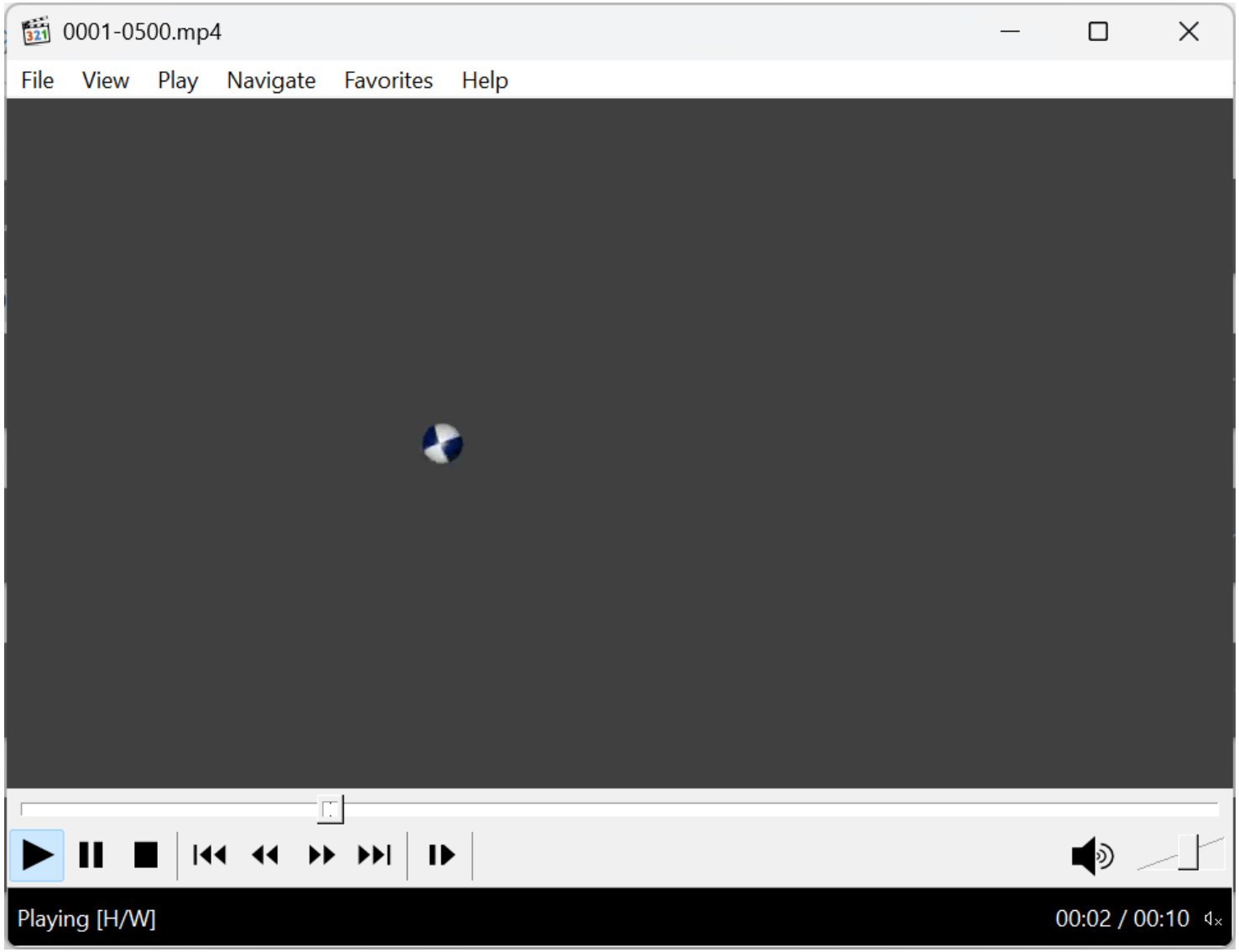
1. Click **Open** → Select a folder to save the animation → **Accept**
2. File format → **FFmpeg video**
3. Container → **MPEG-4** (output file format: mp4)



Scroll up
Click **Animation**



Animation successfully created — ready to play.



All questions are accepted

putri.widartiningsih@gmail.com