yt Volume Rendering

Suoqing Ji

University of California Santa Barbara

Requirement: yt version ≥ 3.3

Install yt

- $ wget https://bitbucket.org/yt_analysis/yt/raw/yt/doc/install_script.sh
  $ bash install_script.sh

OR

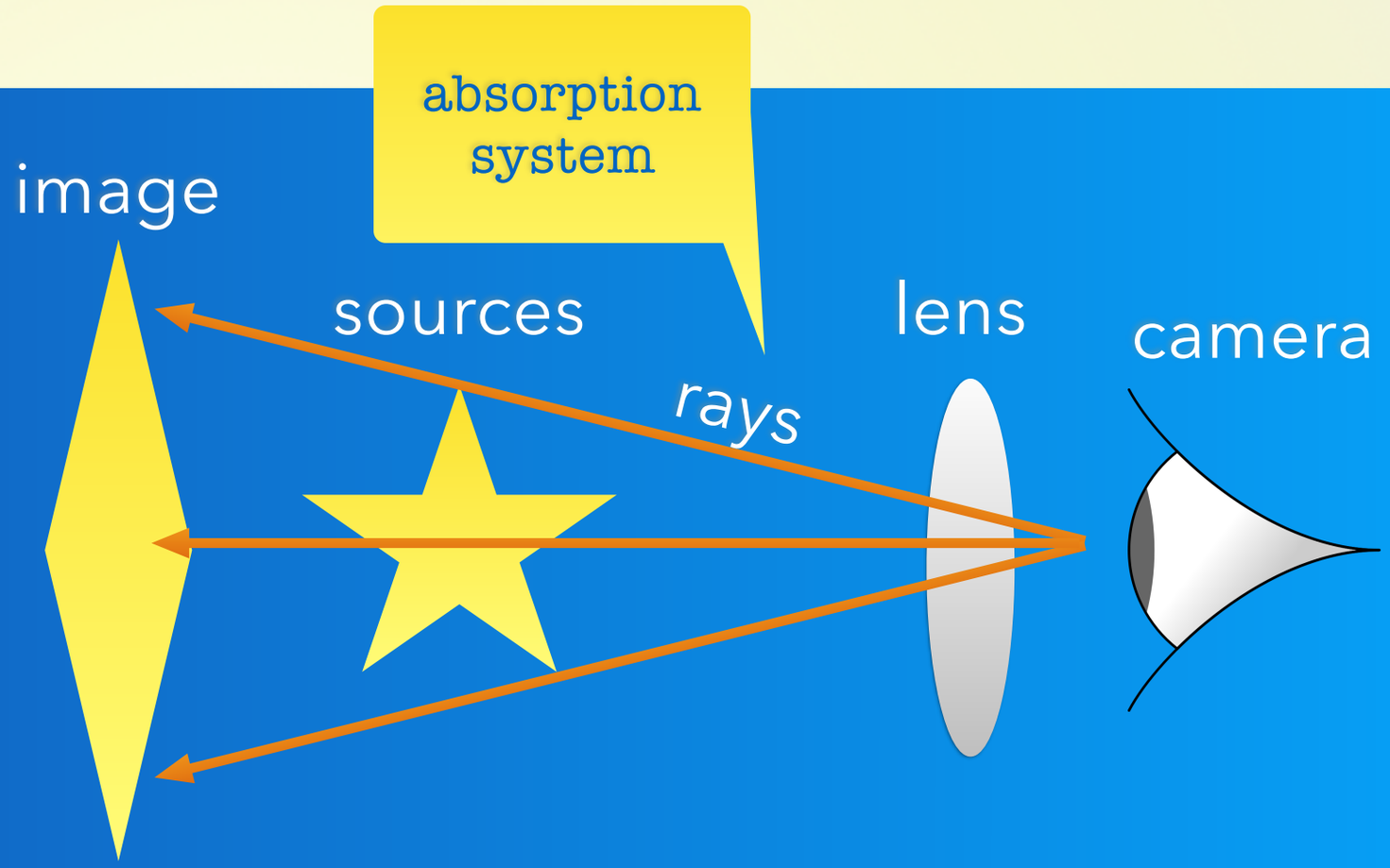
- $ conda install -c conda-forge yt

Start jupyter notebook (iPython notebook)

- $ jupyter notebook

Ray-casting volume rendering

- **camera:** starting point of rays
- **lens:** organize rays
- **source:** provide physical data to rays
- **rays:** accumulate data and convert to colors (via **transfer function**)
- **image:** store final color pixels

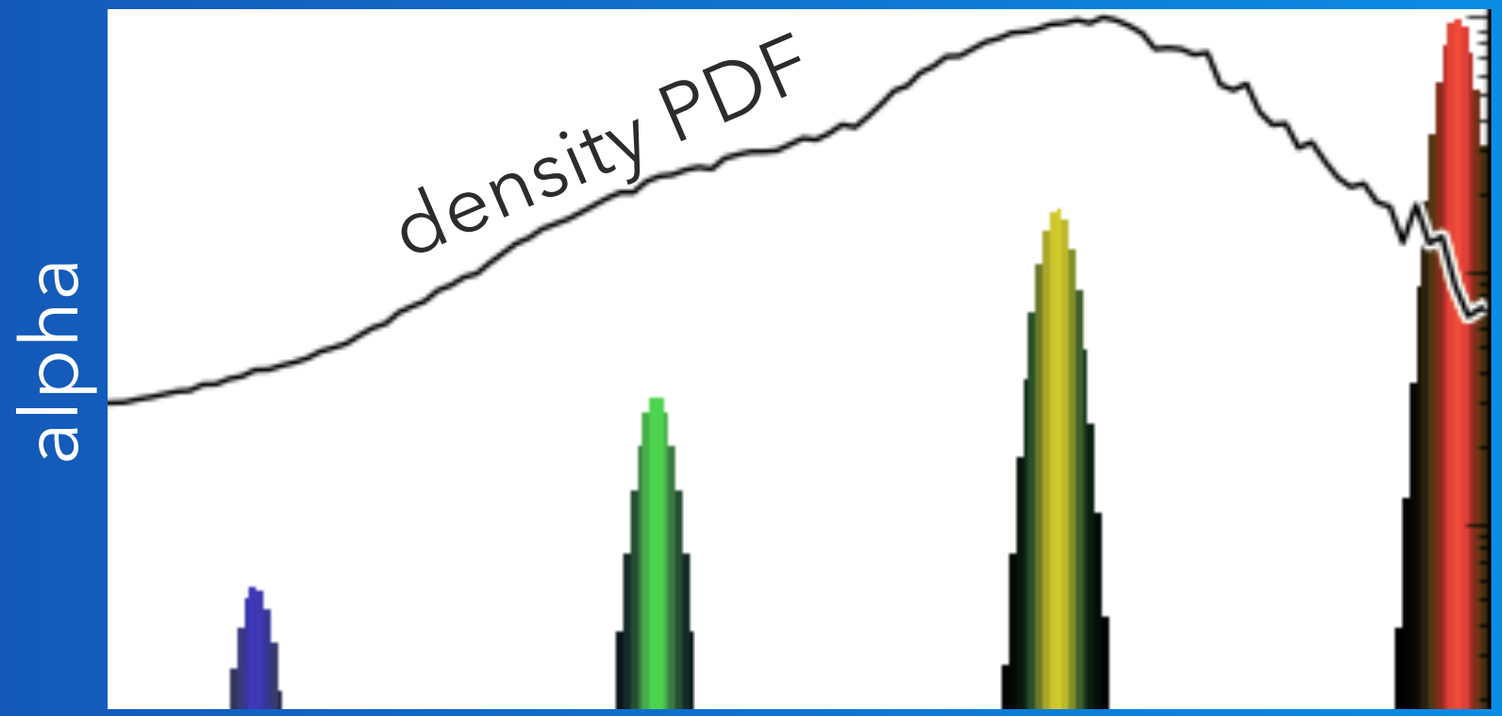


file: 1_simple_render_and_tf.ipynb

```
from yt.testing import \
    fake_vr_orientation_test_ds

ds = fake_vr_orientation_test_ds()
# now we do volume rendering
sc = yt.create_scene(ds, field='density')
sc.render()
sc.show(sigma_clip=6.0)
```

Changing transfer function

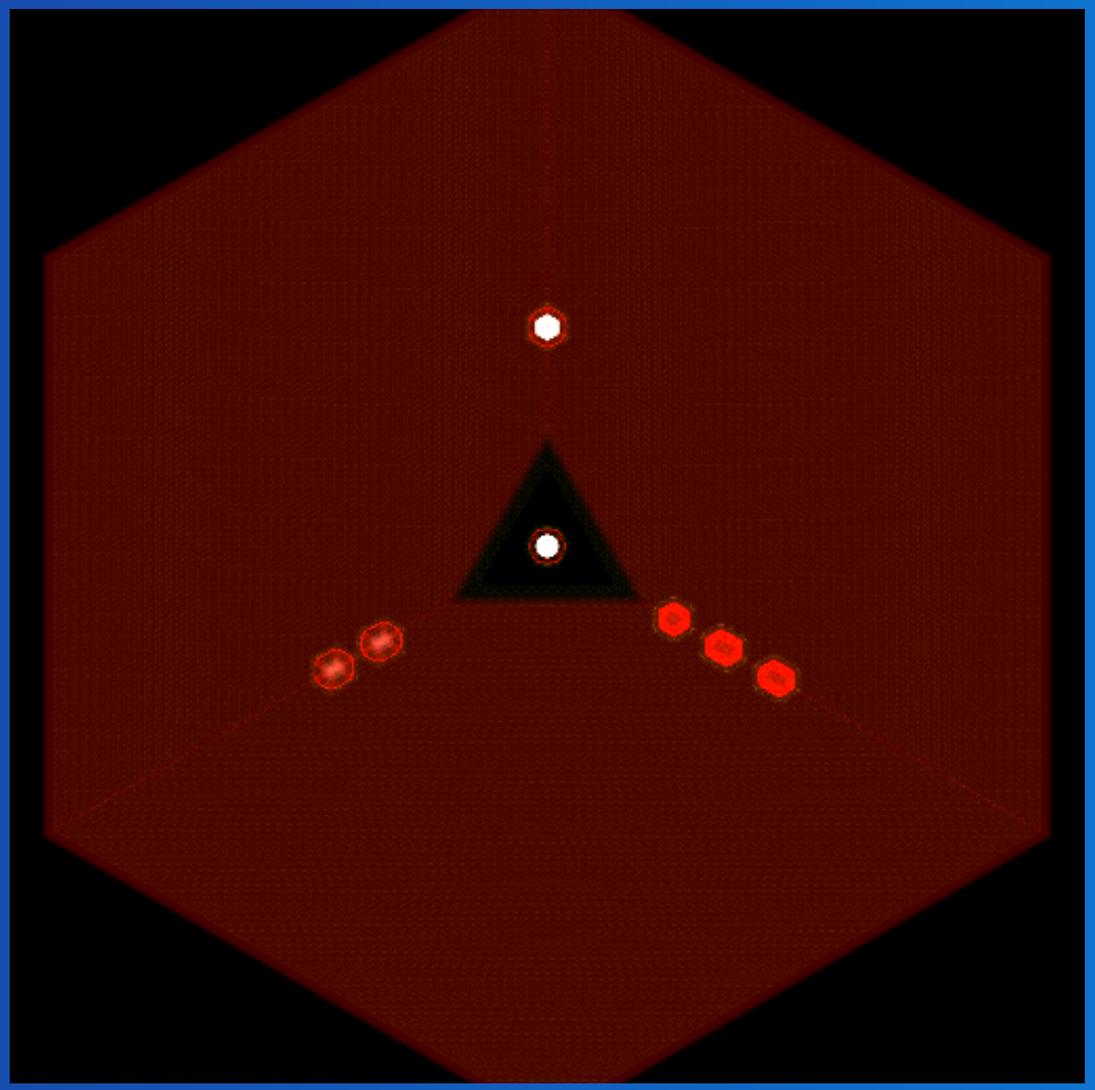


file: 1_simple_render_and_tf.ipynb

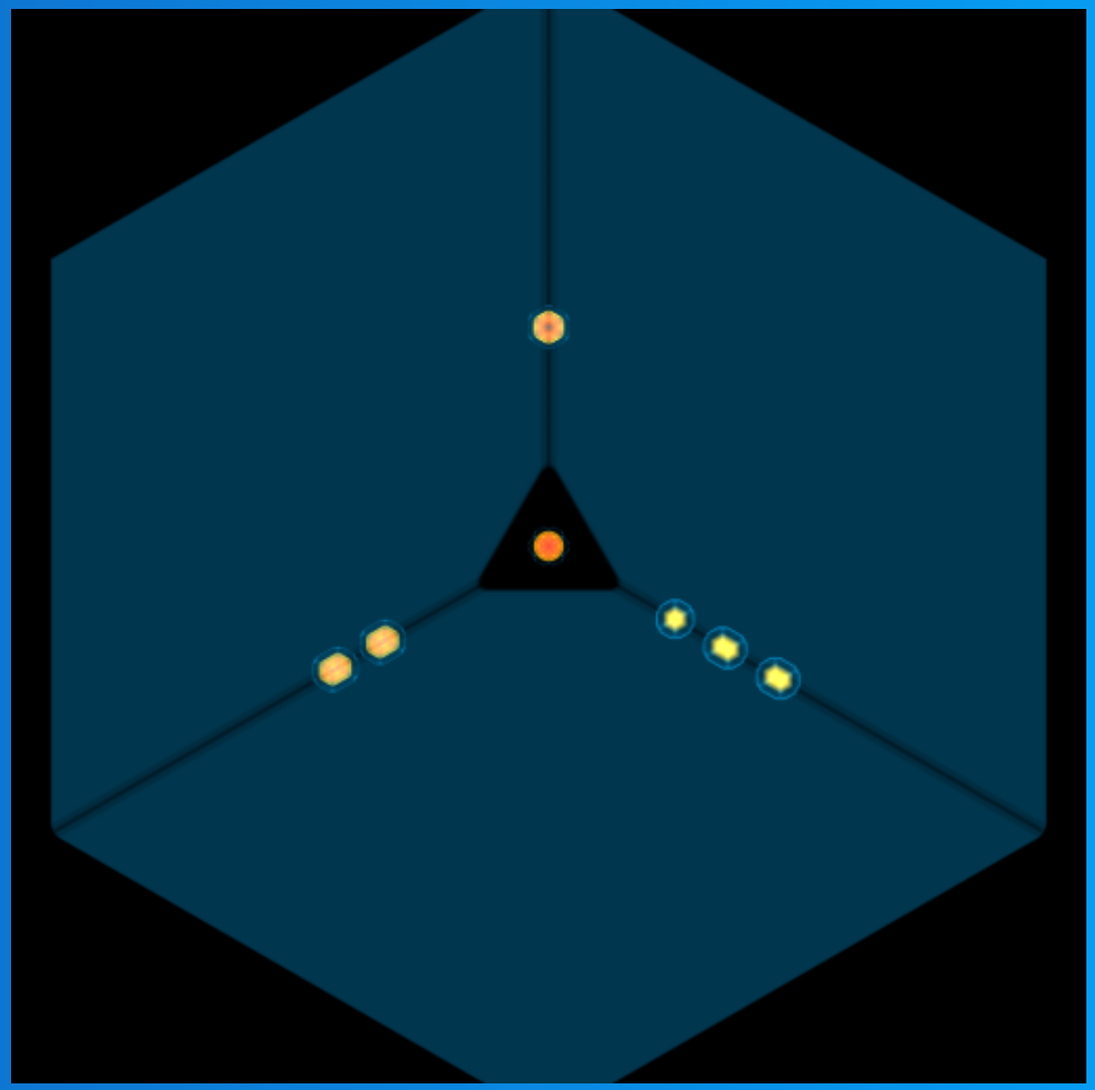
```
# Change transfer function
tf = yt.ColorTransferFunction((np.log10(0.1), np.log10(1.)))
tf.sample_colormap(np.log10(0.9), 0.01, colormap="spectral")
tf.sample_colormap(np.log10(0.8), 0.01, colormap="spectral")
tf.sample_colormap(np.log10(0.6), 0.01, colormap="spectral")
tf.sample_colormap(np.log10(0.2), 0.01, colormap="spectral")
render_source = sc.get_source(0)
render_source.transfer_function = tf
```

output of file: 1_simple_render_and_tf.ipynb

before changing tf

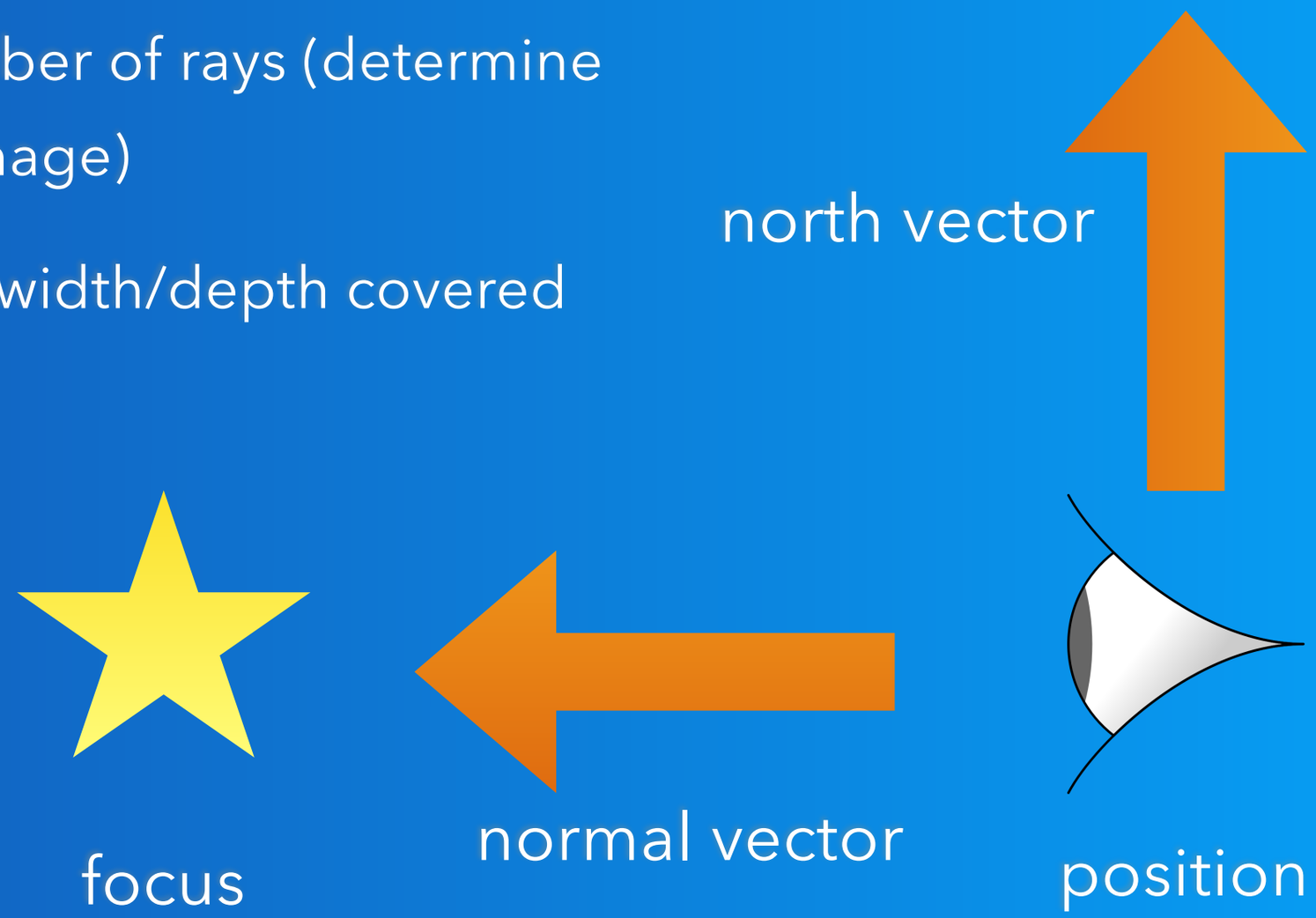


after changing tf



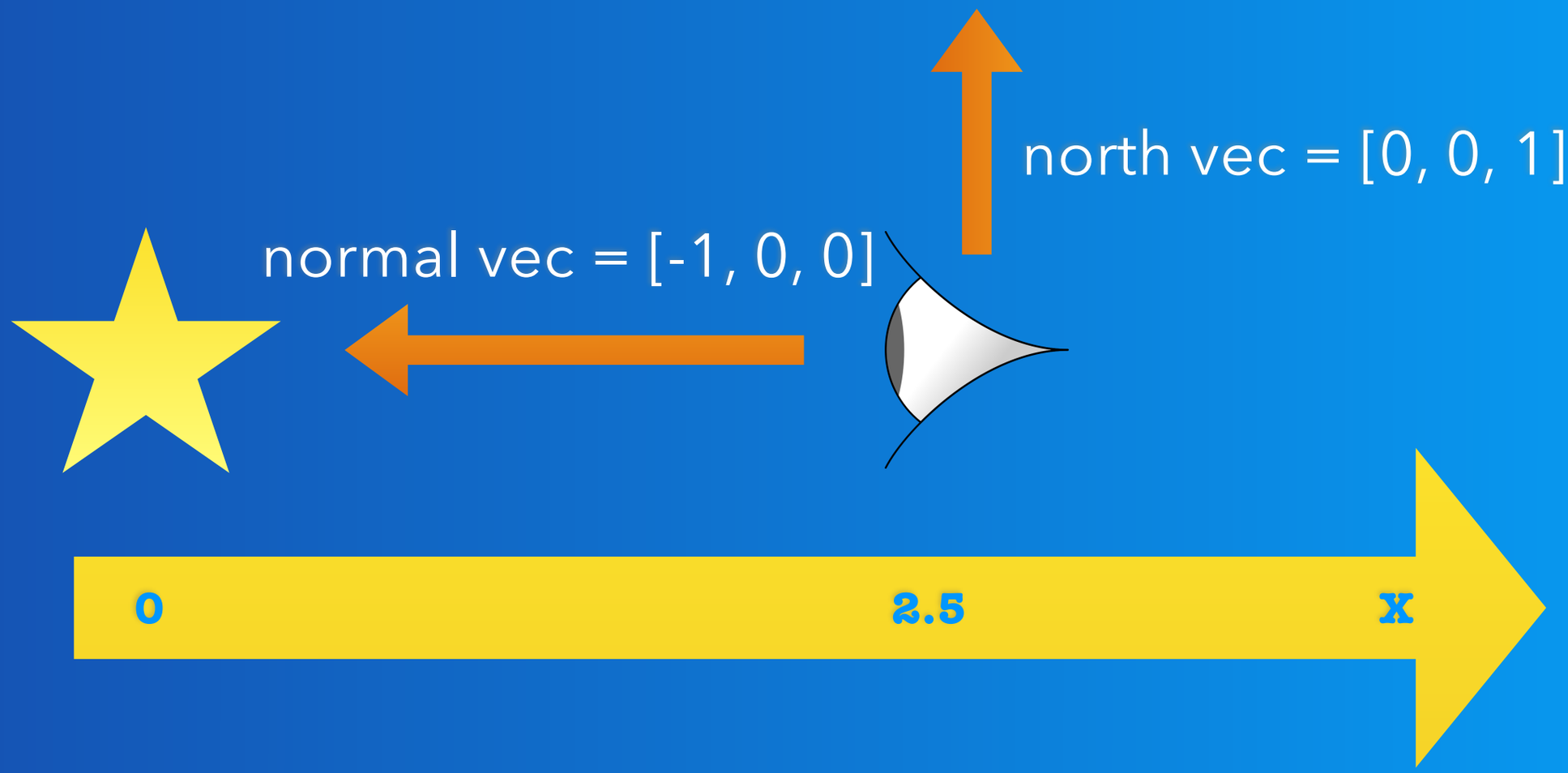
Camera

- **normal vector:** from camera position to focus
- **north vector:** define the “up” direction
- **camera.resolution:** number of rays (determine the resolution of final image)
- **camera.width:** physical width/depth covered by camera

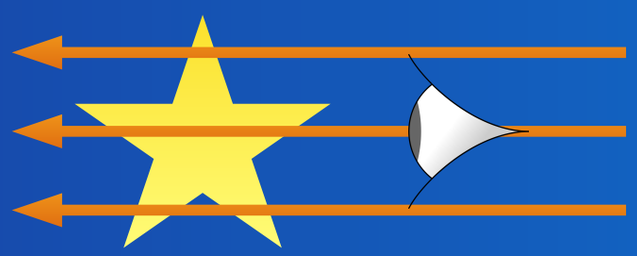


file: 2_camera_and_lens.ipynb

```
# plane-parallel lens
cam = sc.add_camera(ds, lens_type='plane-parallel')
cam.resolution = (500, 500)
cam.width = ds.quan(4, 'code_length')
cam.position = ds.arr([2.5, 0, 0], 'code_length')
cam.switch_orientation(normal_vector=[-1, 0, 0],
                       north_vector=[0, 0, 1])
sc.render()
sc.show(sigma_clip=2.0)
```



Different types of lens

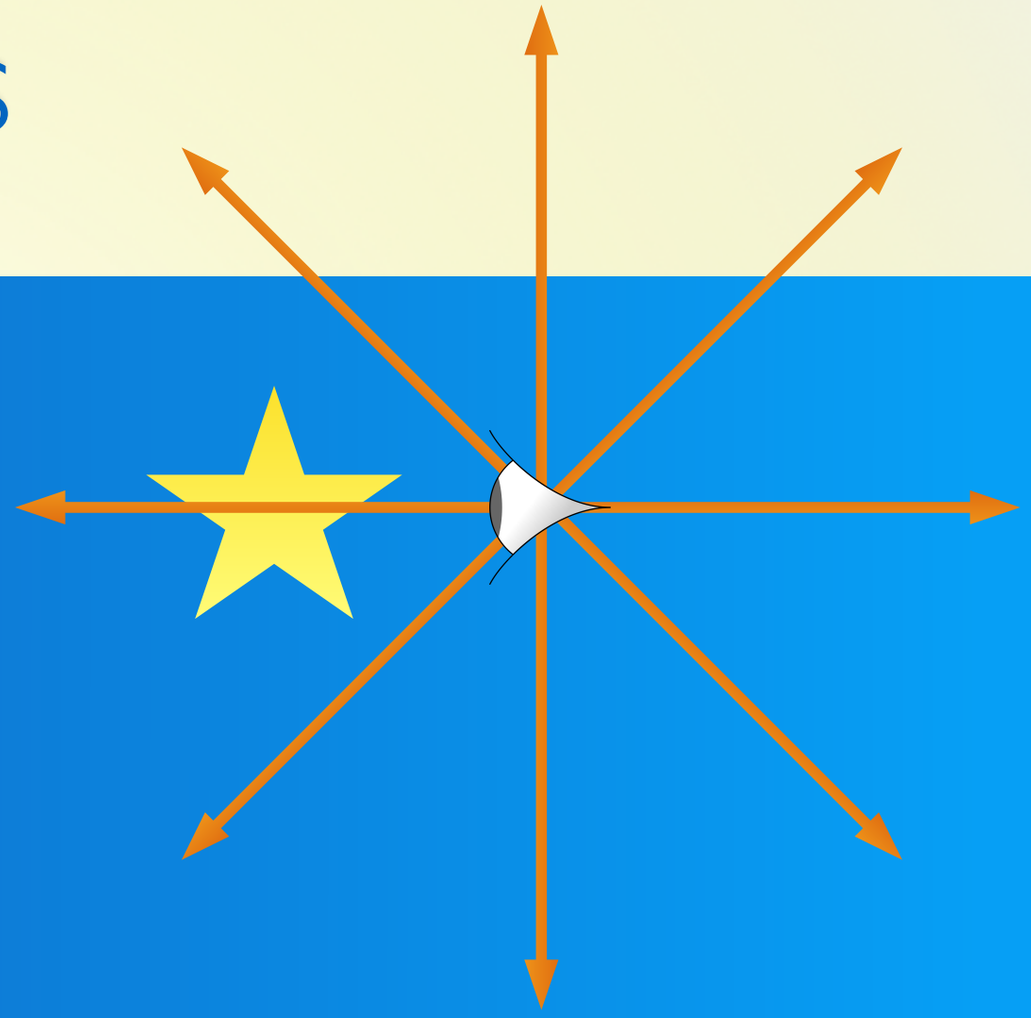


plane-parallel

- rays extend to infinity on both sides
- no depth information (camera can not “go inside”)

perspective

- more natural and intuitive
- rays extend to infinity only on one side
- camera **can** “go inside”



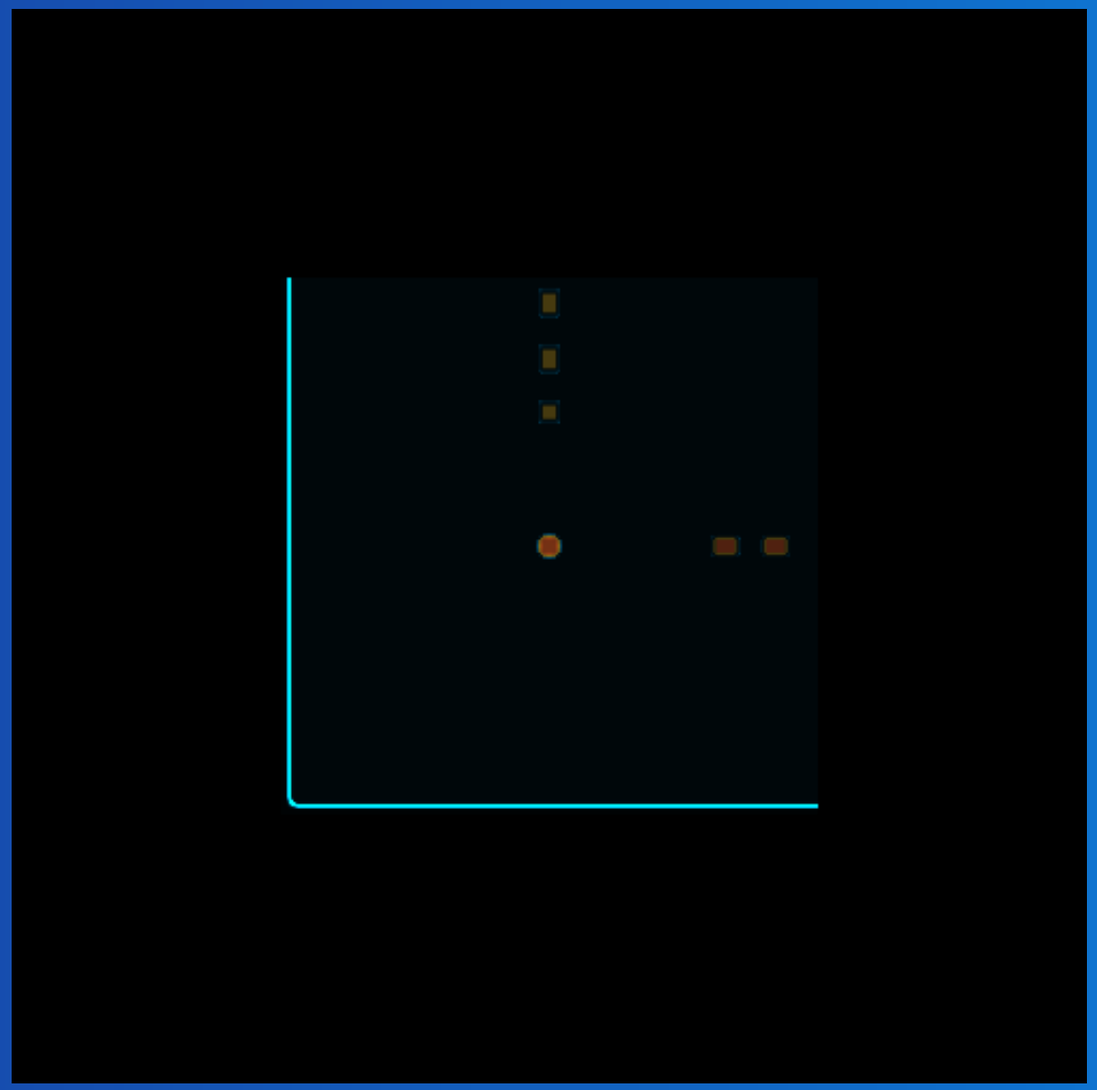
spherical

- image can be pasted to the inter surface of a sphere
- 360-degree movie (https://support.google.com/youtube/answer/6178631)

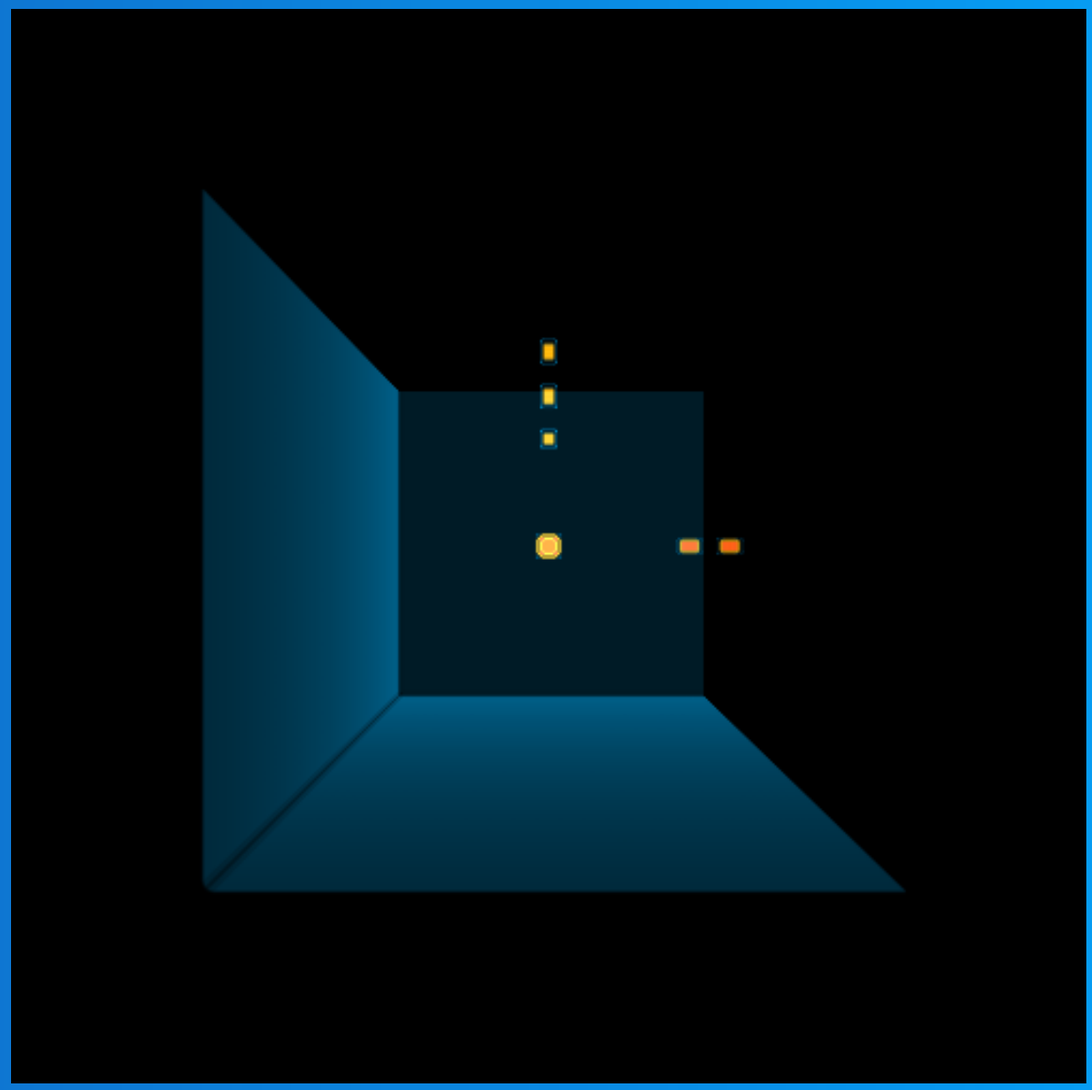
stereo-* lens: two-eye image; a little bit more complicated (see https://developers.google.com/vr/jump/rendering-ods-content.pdf)

output of file: 2_camera_and_lens.ipynb

plane-parallel

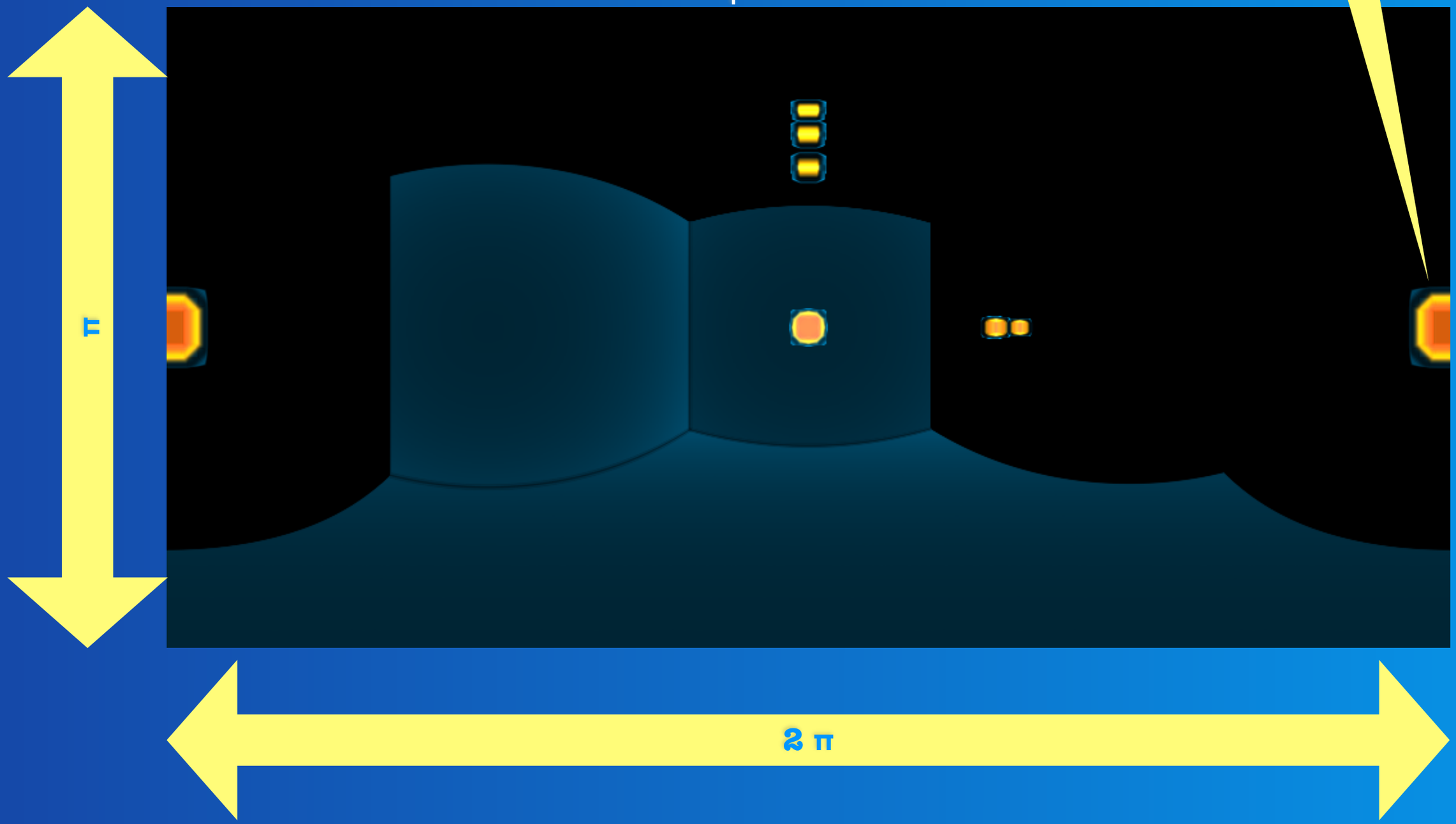


perspective



output of file: 2_camera_and_lens.ipynb

spherical



- stereo-spherical lens: make virtual reality movies (https://support.google.com/youtube/answer/6316263)

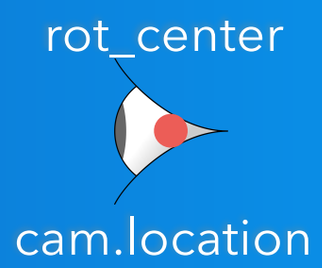
Moving and rotating camera

- Rotation: options rot_vector and rot_center can be provided
- Use iter_move and iter_rotation functions to take a series of actions
- *Run file 3_move_camera.ipynb*

rot_vector: define the plane of rotation

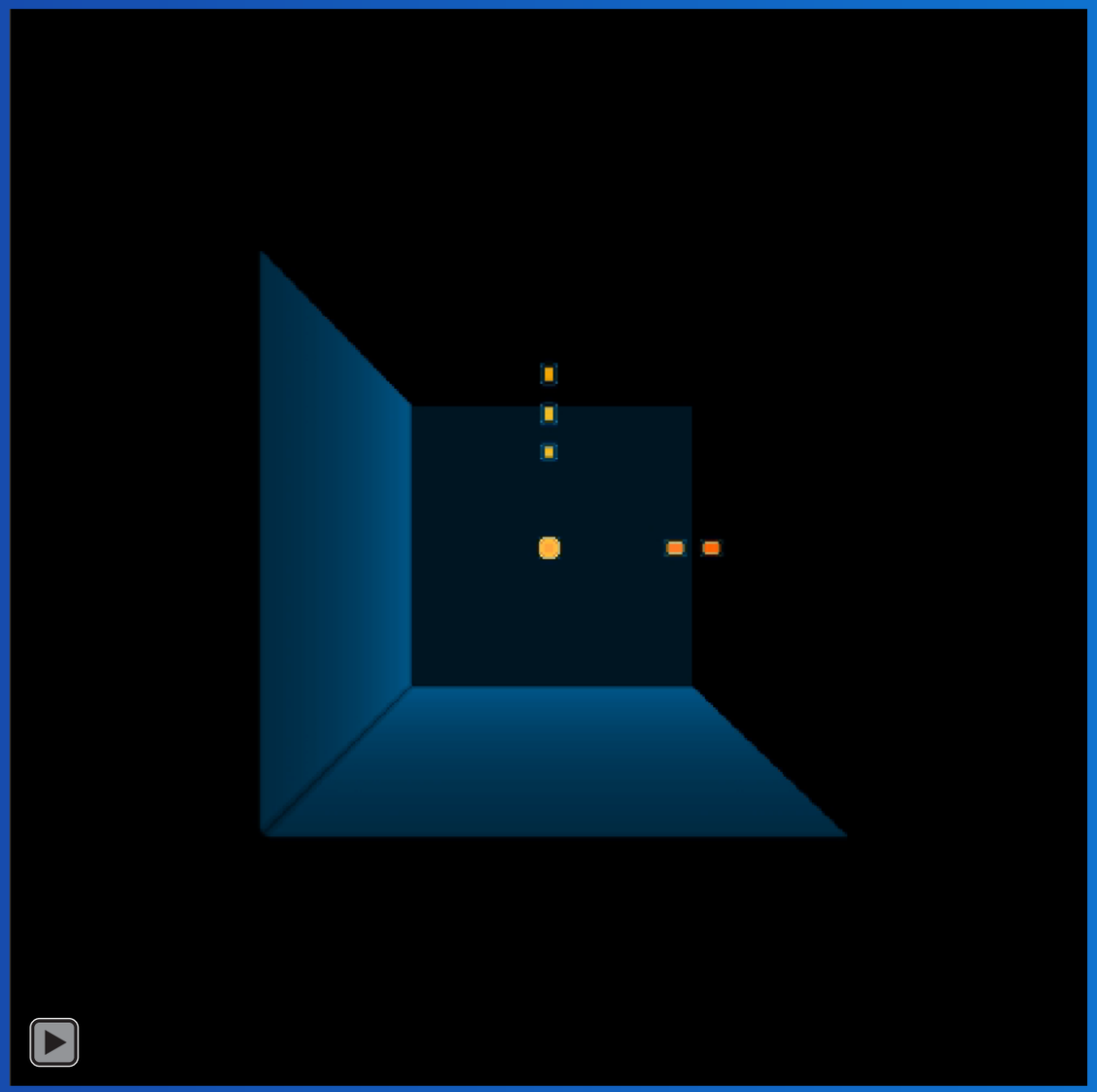
rot_center ≠ cam.location

rot_center = cam.location



some output of file (movies): 3_move_camera.ipynb

move



rotation