File - /Users/sqc0815/Not_Backed_Up/hwRepo/ASTR5760/proj2/project_inputs.py

```python
import sys
sys.path.append('../modules/')

from modules import rayTracing as rt
import numpy as np
from Basilisk.utilities import RigidBodyKinematics as rbk

# instrument design inputs
cass_inputs = rt.CassegrainDefinition()  # a struct for my inputs
cass_inputs.f_num_1 = 3.  # primary f number
cass_inputs.d_1 = 1.  # primary diameter [m]
cass_inputs.f_num_total = 15.  # instrument total fnumber
cass_inputs.e = 0.15  # distance secondary focus behind primary vertex
cass_inputs.primary_x = 3.  # position of the primary in the lab frame [m]
cass_inputs.orientation_212 = [-np.pi / 2., 0, 0]  # orientation of the instrument wrt lab frame
cass_inputs.focal_plane_offset = 0.  # put the detector this far off of the secondary focus

# instrument for this project
cass = rt.Instrument()
cass.set_surfaces(rt.cassegrain_set_up(cass_inputs))
surfaces = cass.surfaces[1:]  # get rid of the detector

basic_paraxial_ray_inputs = rt.AngledCircleRayDef()
basic_paraxial_ray_inputs.rad = cass_inputs.d_1 / 2.
basic_paraxial_ray_inputs.num_circ = 15
basic_paraxial_ray_inputs.per_circ = 150
basic_paraxial_ray_inputs.angles = [0.]
starts, dirs = rt.make_angled_circle_rays(basic_paraxial_ray_inputs)
basical_paraxial_rays = rt.Ray()
basical_paraxial_rays.set_dir(dirs[0])
basical_paraxial_rays.set_pos(starts[0])


# basic paraxial rays
basic_paraxial_ray_inputs = rt.AngledCircleRayDef()
basic_paraxial_ray_inputs.rad = cass_inputs.d_1 / 2.
basic_paraxial_ray_inputs.num_circ = 15
basic_paraxial_ray_inputs.per_circ = 150
basic_paraxial_ray_inputs.angles = [0.]
starts, dirs = rt.make_angled_circle_rays(basic_paraxial_ray_inputs)
basical_paraxial_rays = rt.Ray()
basical_paraxial_rays.set_dir(dirs[0])
basical_paraxial_rays.set_pos(starts[0])

# grating stuff
line_density = 3600.  # per mm
mode = 1
lam = 1600E-10
d = 1. / (line_density * 1000.)
alpha = np.arcsin(mode * lam / d)
rotation_for_beta_zero = rbk.euler2122C([alpha, 0., 0.])
DCM_basic = rbk.euler2122C([-np.pi/2., 0., 0.])
DCM_SL = np.dot(rotation_for_beta_zero, DCM_basic)
grating = rt.RowlandCircle()
grating.set_radius(1.0)
grating.set_line_density(line_density)
f_num_2 = cass_inputs.f_num_total - cass_inputs.f_num_1
offset = grating.r * np.cos(alpha)
```

```
59 focus = surfaces[-1].L_r_L
60 pos = focus + np.array([offset, 0, 0]).reshape([3, 1])
61 grating.set_position(pos)
62 grating.set_DCM(DCM_SL)
63 grating.set_width(0.24)
64 grating.set_order(0)
65 grating.set_wavelength(1600.)
66
67 cylindrical_detector = rt.CylindricalDetector()
68 cylindrical_detector.set_radius(1.0)
69 cylindrical_detector.set_height(1.0)
70 cylindrical_detector.set_sweep_angle(np.pi)
71 dcm_cyl = np.dot(rbk.euler2122C([np.pi, 0., 0.]), DCM_SL)
72 dcm_rot = np.dot(rbk.euler1232C([0., 0., -np.pi/2.]), dcm_cyl)
73 cylindrical_detector.set_DCM(dcm_rot)
74 offset = np.dot(DCM_SL.transpose() , np.array([0., 0., 1.0]))
75 cylindrical_detector.set_position(pos + offset.reshape([3, 1]))
76 cylindrical_detector.set_y_limits()
77
78 # make an edge ray to determine secondary diameter
79 angle_to_capture = 0.
80 edge_ray_X, edge_ray_d = rt.make_one_edge_ray(cass_inputs.d_1 / 2.,
   angle_to_capture)
81 edge_ray_X = np.zeros(3).reshape([3, 1])
82 edge_ray = rt.Ray()
83 edge_ray.set_pos(edge_ray_X)
84 edge_ray.set_dir(edge_ray_d)
```