PySpeos cheat-sheet

Version: v0.6.dev0

Connect to Speos Service

Start instance locally:

```
from ansys.speos.core.launcher import
    launch_local_speos_rpc_server

speos_server = launch_local_speos_rpc_server()
# returns a connected speos instance
```

Connect to an existing instance:

```
from ansys.speos.core import Speos
host_ip = '127.0.0.1' # localhost here
speos_server = Speos(host=host_ip, port=50098)
```

Speos Solver files

Basic methods

Load a solver file:

```
import ansys.speos.core as core

project = core.Project(speos=speos_server,
    path='path_to.speos')
print(project)
```

Combine solver file:

```
from ansys.speos.core.workflow.combine_speos
    import SpeosFileInstance, combine_speos

combined_project = combine_speos(
    speos=speos_server,
    speos_to_combine=[
        SpeosFileInstance(
            speos_file='combine_path_to.speos',
            axis_system=[x1, y1, z1, 1, 0, 0, 0,
                1, 0, 0, 0, 1],),
        SpeosFileInstance(
            speos_file='combine_path_to.speos',
            axis_system=[x2, y2, z2, 1, 0, 0, 0,
                1, 0, 0, 0, 1],
        ),
    ],
)
```

Preview solver file:

```
# method to show mesh stored in solver
project.preview(viz_args={'opacity': 0.7})
# method to display Lightpath
lxp = core.LightPathFinder(speos=speos_server,
    path='path_to_lpf')
lxp.preview(project=project)
```

Navigation methods

Find methods and it possible uses:

```
from ansys.speos.core.simulation import
    SimulationDirect

found_items = project.find(name='test')
# any item with name 'test' are returned as a list
found_items = project.find(name='.*',
    name_regex=True,
    feature_type=SimulationDirect)
# any direct simulation item are return as a list
```

Modify/Change Solver files

Sources

Create surface source:

```
from ansys.speos.core.source import SourceSurface

s_source = project.create_source(name='Surface.1',
    feature_type=SourceSurface)
# Choose one way to set source power
# for any conflicting property the last set
    property counts
s_source.set_flux_from_intensity_file()
s_source.set_flux_luminous(683)
s_source.set_flux_radiant(1)
s_source.set_flux_luminous_intensity(5)
s_source.set_exitance_constant(
    [(core.GeoRef.from_native_link(
        'TheBodyB/TheFaceF'), False)])
s_int = s_source.set_intensity()
s_int.set_cos(1,120)
s_source.set_spectrum()
s_source.commit()
```

Create rayfile source:

```
from ansys.speos.core.source import SourceRayFile

r_source = project.create_source(name='Rayfile.1',
    feature_type=SurfaceRayFile)
r_source.set_ray_file_uri('path/ray.ray')
# Choose one way to set source power
# for any conflicting property the last set
    property counts
r_source.set_flux_from_ray_file()
r_source.set_flux_luminous(683)
r_source.set_flux_radiant(1)
r_source.set_spectrum_from_ray_file()
r_source.set_exit_geometries(
    [core.GeoRef.from_native_link(
        'TheBodyB/TheFaceF')])
r_source.set_axis_system([0,0,0,1,0,0,0,1,0,0,0,1])
r_source.commit()
```

Material

Create optical property:

```
mat1 =
    project.create_optical_property(name='Material.1')
# Choose one way of setting Surface Properties
# for any conflicting property the last set
    property counts
mat1.set_surface_mirror(80.0)
mat1.set_surface_opticalpolished()
mat1.set_surface_library('path/surface.brdf')
# Choose on way of setting Volume Properties
mat1.set_volume_none()
mat1.set_volume_opaque()
mat1.set_volume_optic(index=1.7, absorption=0.01,
    constringence=55)
mat1.set_volume_library('path/mat.material')
# Geometry
mat1.set_geometries(
    [core.GeoRef.from_native_link(
        geopath='TheBodyB')])
mat1.commit()
```

Geometries

Create new Geometries:

```
root_part = project.create_root_part()
root_part.commit()
body_1 = root_part.create_body(name='TheBodyB1')
body_1.commit()
body_1_face_1 =
    body_1.create_face(name='TheFaceF1')
body_1_face_1.set_vertices([])
body_1_face_1.set_facets([])
body_1_face_1.set_normals([])
body_1_face_1.commit()
```

Sensor

Create radiance sensor:

```
from ansys.speos.core.sensor import SensorRadiance

r_sensor =
    project.create_sensor(name='Radiance.1',
    feature_type=SensorRadiance)
# define size
dim = r_sensor.set_dimensions()
dim.set_x_start(-5).set_x_end(5).set_x_sampling(10)
dim.set_y_start(-5).set_y_end(5).set_y_sampling(10)
# define type and Wavelength range
col = r_sensor.set_type_colorimetric()
wl = col.set_wavelengths_range()
wl.set_start(380).set_end(780)
wl.set_sampling(50)
# define Layer separation
```

```
r_sensor.set_layer_type_source()
r_sensor.commit()
```

Create irradiance Sensor:

```
from ansys.speos.core.sensor import
    SensorIrradiance

i_sensor =
    project.create_sensor(name='Irradiance.1',
    feature_type=SensorIrradiance)
# methods are similar to Radiance Sensor
i_sensor.commit()
```

Create camera Sensor:

```
from ansys.speos.core.sensor import SensorCamera

c_sensor = project.create_sensor(name='Camera.1',
    feature_type=SensorCamera)
c_sensor.set_distortion_file_uri(
    'distortion_file_path')
# Choose photometric mode
photo_cam = c_sensor.set_mode_photometric()
photo_cam.set_transmittance_file_uri(
    'transmittance_file_path')
# Choose color mode
c_mode = photo_cam.set_mode_color()
c_mode.set_blue_spectrum_file_uri(
    'blue_spectrum_path')
# same method for red and green spectrum
c_sensor.commit()
```

Simulation definition

Create direct simulation:

```
from ansys.speos.core.simulation import
    SimulationDirect

direct_sim =
    project.create_simulation(name='Direct.1',
    feature_type=SimulationDirect)
# All project geometry will be taken into account
# General Simulation API
direct_sim.set_sensor_paths(['Irradiance.1'])
direct_sim.set_source_paths(['Surface.1'])
# the stop condition which is hit first will stop
    the simulation
direct_sim.set_stop_condition_rays_number(5000000)
direct_sim.set_stop_condition_duration(3600)
direct_sim.commit()
```

Create inverse simulation:

```
from ansys.speos.core.simulation import
    SimulationInverse

inverse_sim=
    project.create_simulation(name='Inverse.1',
    feature_type=SimulationInverse)
# methods similar to direct simulation
```

Create interactive simulation:

```
from ansys.speos.core.simulation import
    SimulationInteractive

interactive_sim =
    project.create_simulation(name='Interactive.1',
    feature_type=SimulationInteractive)
# methods similar to direct simulation
```

Run Simulation on CPU or GPU:

```
simulation_feature.compute_GPU()
simulation_feature.compute_CPU()
```

Open results (Windows only):

```
from ansys.speos.core.workflow.open_result import
    open_result_image, open_result_in_viewer
# Display result image
open_result_image(simulation_feature,
    'result_name')
# opens result in VirtualPhotometricLab
open_result_in_viewer(simulation_feature,
    'result_name')
```