# Spectral Simulation of Electrochromic Glass Using Python

Tammie Yu, LBNL

# Electrochomic Tinting + Daylight Dimming

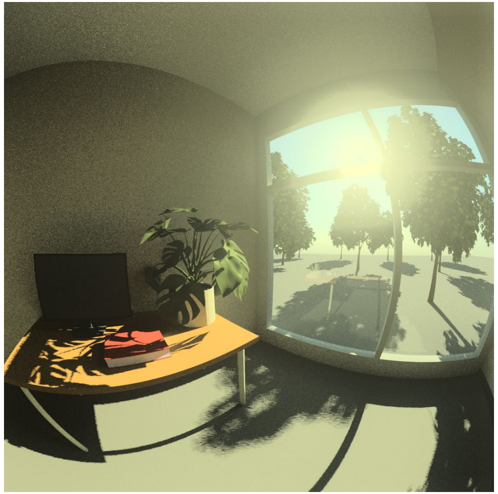
Goal: To spectrally simulate electrochromic tinting with daylight dimming and to see the impacts on spectral irradiance, ev, melanopic illuminance, and thermal loads.

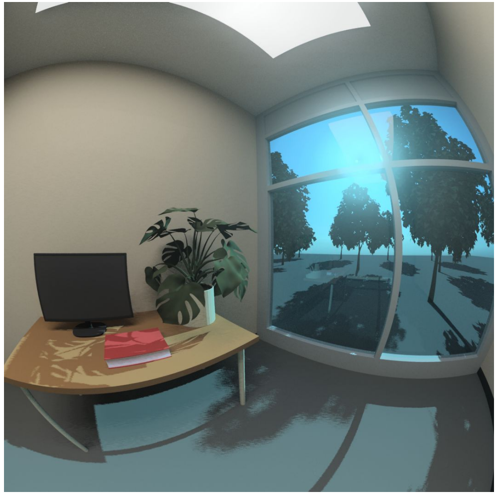
1. Setup scenes

2. Interpolate intermediate tinted and dimming state

3. Results

4. Animation





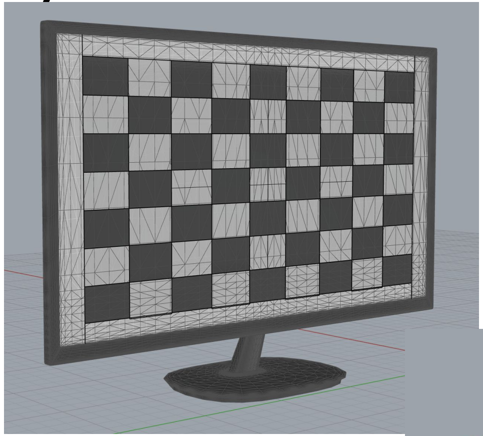
# Radiance - 1. Scene geometry

1. Build geometry in Rhino

2. Categorize material types into own layers

3. Assign a material to each layer

(only the name matters)

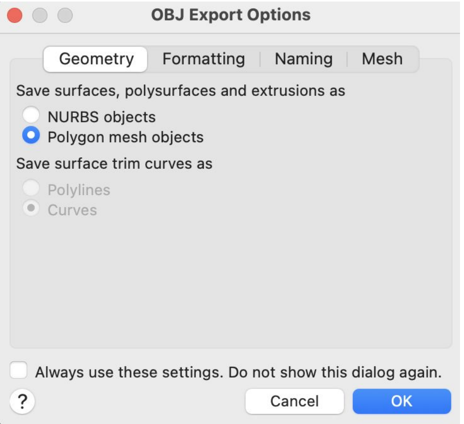


# Radiance - 1. Scene geometry

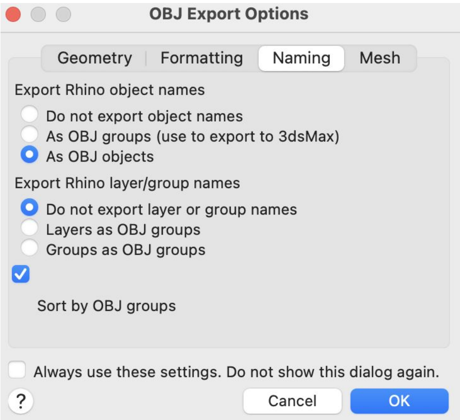
4. Export each layer separately as an **.obj** file

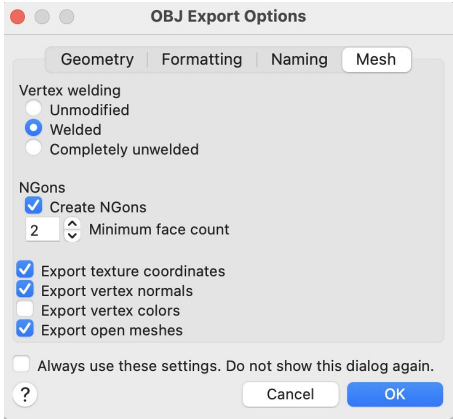
5. Use **obj2rad** to convert obj to rad geometry

```
Obj2rad -f leaves.obj > leaves.rad
```





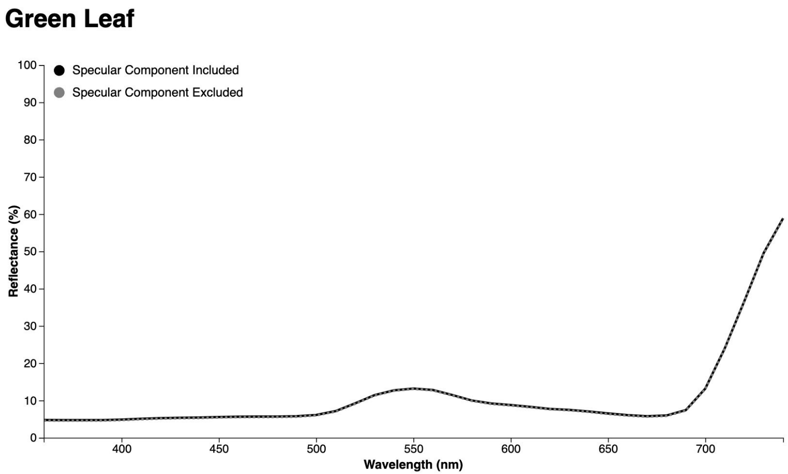


# 1. Spectral material - Opaque

1. Get spectral material data

    (ex. https://spectraldb.com/)




## Radiance Definition

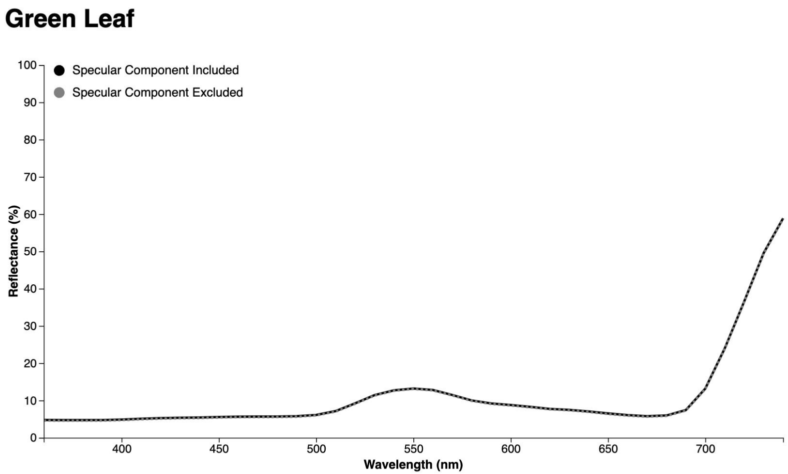
```
void plastic green_leaf
0
0
5 0.0830 0.1108 0.0461 0.0012 0.2000
```

# 1. Spectral material - Opaque

1. Get spectral material data

   (ex. https://spectraldb.com/)

2. Make a spectrum material modifier




```
void spectrum green_leaf_mat
0
0
41 360 740 4.6 4.58 4.57 4.59 ...
36.39 49.36 58.69
```

extrema

Spectral reflectance

## Radiance Definition

green_leaf_mat

~~void~~ plastic green_leaf

0

0

5 ~~0.0830~~ ~~0.1108~~ ~~0.0461~~ 0.0012 0.2000

0.01 0.01 0.01

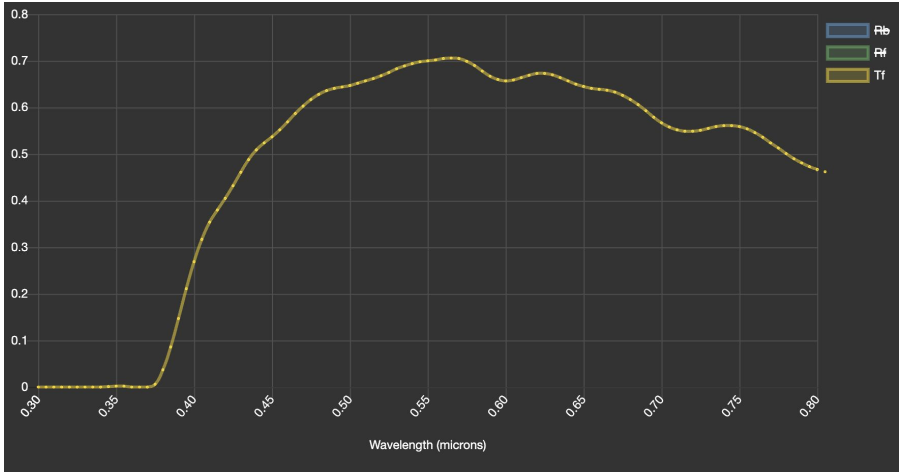
Because data is in percentage

# 2. Spectral material - window

1. Get spectral material data

   - https://igsdb.lbl.gov/

2. Make a spectrum material modifier




```
void spectrum ec60
0
0
38 380 780 0.037 0.147 0.269 0.354 ... 0.546 0.524 0.501
```

Spectral transmittance

```
ec_60 glass glass_mat
0
0
4 1 1 1 1.52
```

# 3. Spectral material - luminaire

1. Download luminaire specification in **.ies file**

2. Use **ies2rad** to convert to rad geometry and material.

```
ies2rad -o troffer -di troffer.ies
```

```
void brightdata 'troffer_dist'
5 boxcorr 'roffer.dat' source.cal src_phi src_theta
0
4 0.786028 1.18262 0.582168 0.12192

'troffer_dist' light 'troffer_light'
0
0
3 1 1 1
```

# 3. Spectral material - luminaire

1. Download luminaire specification in **.ies file**

2. Use **ies2rad** to convert to rad geometry and material.

```
ies2rad -o troffer -di troffer.ies
```

3. Download spectral material data for luminaire (https://spectralcalculator.pnnl.gov/example-library)

```
void spectrum light_mat
0
0
41 360 740 0.015063 0.008414 0.020084
... 0.00285 0.002307 0.001221
```

light_mat

```
void brightdata 'troffer_dist'
5 boxcorr 'roffer.dat' source.cal src_phi src_theta
0
4 0.786028 1.18262 0.582168 0.12192

'troffer_dist' light 'troffer_light'
0
0
3 1 1 1
```

# Build a scene

Import pyradiance as pr

room = pr.Scene(

sid="room",

surfaces=["./Groups/exterior.rad", "./Groups/room.rad"],

materials=["./Materials/material.rad", "./Materials/ec60.rad"]

)

# 4. Spectral Sky

1. Create a spectral sky using **genssky**

```
ssky = pr.genssky(

    dt=datetime.datetime(2024,12,12,12),

    latitude=37,

    longitude=122,

    timezone=120,

    year=2024

)

for prim in pr.parse_primitive(ssky.decode()):
    room.add_source(prim)
```

```
void spectrum sunrad
0
0
22 380 780 0.342 0.642 0.701 0.922
1.026 1.051 1.068 1.111 1.131 1.139
1.149 1.140 1.148 1.124 1.130 1.098
1.063 1.039 1.005 0.969

sunrad light solar
0
0
3 6128380.0 6128380.0 6128380.0

solar source sun
0
0
4 0.021784 -0.876704 0.480537 0.533

void specpict skyfunc
5 noop ./out_sky.hsr .
'Atan2(Dy,Dx)/PI+1' '1-Acos(Dz)/PI'
0
0
```

# 4. Spectral Sky

```
# add skydome and ground plane

room.add_source(
    pr.Primitive("skyfunc", "glow", "skyglow",
[], [1,1,1,0])
    )

room.add_source(
    pr.Primitive("skyglow", "source",
"skydome", [], [0,0,1,180])
    )

room.add_source(
    pr.Primitive("skyglow", "source",
"groundplane", [], [0,0,-1,180])
    )
```

```
void spectrum sunrad
0
0
22 380 780 0.342 0.642 0.701 0.922
1.026 1.051 1.068 1.111 1.131 1.139
1.149 1.140 1.148 1.124 1.130 1.098
1.063 1.039 1.005 0.969

sunrad light solar
0
0
3 6128380.0 6128380.0 6128380.0

solar source sun
0
0
4 0.021784 -0.876704 0.480537 0.533

void specpict skyfunc
5 noop ./out_sky.hsr .
'Atan2(Dy,Dx)/PI+1' '1-Acos(Dz)/PI'
0
0
```

# Generate a .hsr - hyperspectral rendering

```
pr.SamplingParameters(ab=2, aa=0, ad=16, lw=1e-5, dj=0.8,
                      cs=20, co=True)
```

# of channels Spectral output

```
img = pr.rtpict(rays=view, octree=room.octree, nproc=8, xres=3200,
yres=3200, params=render_params.args())

with open("./tinted.hsr", "wb") as wtr:
    wtr.write(img)
```

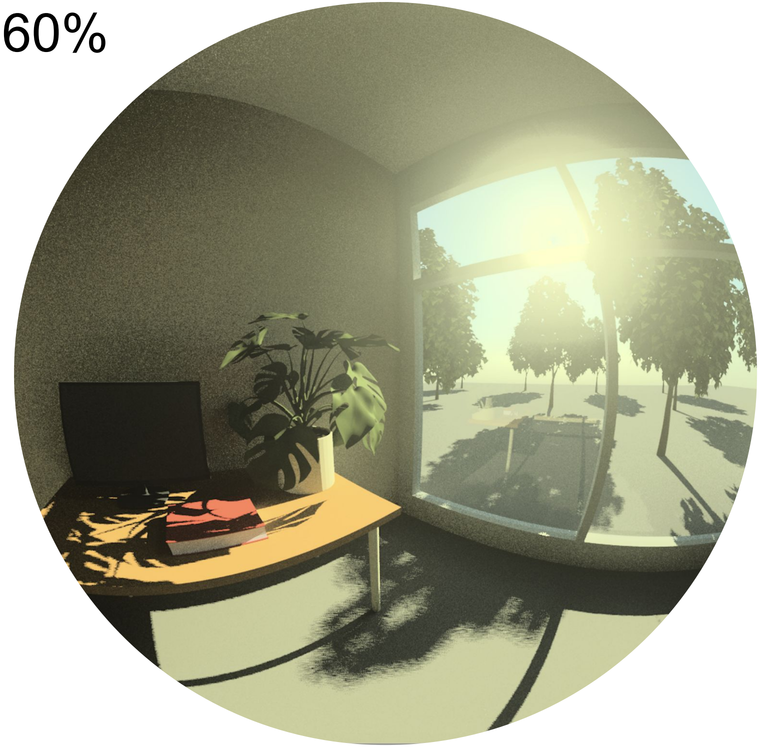
# Convert hsr to hdr

```
input = pr.RcombInput("./tinted.hsr")

hdr = pr.rcomb([input], transform="RGB")

ds_hdr = pr.pfilt(combined_hdr, xres='/3', yres='/3',
gaussian_filter_radius=.6, one_pass=True)

with open("./tinted.hdr", "wb") as file:

    file.write(ds_hdr)
```

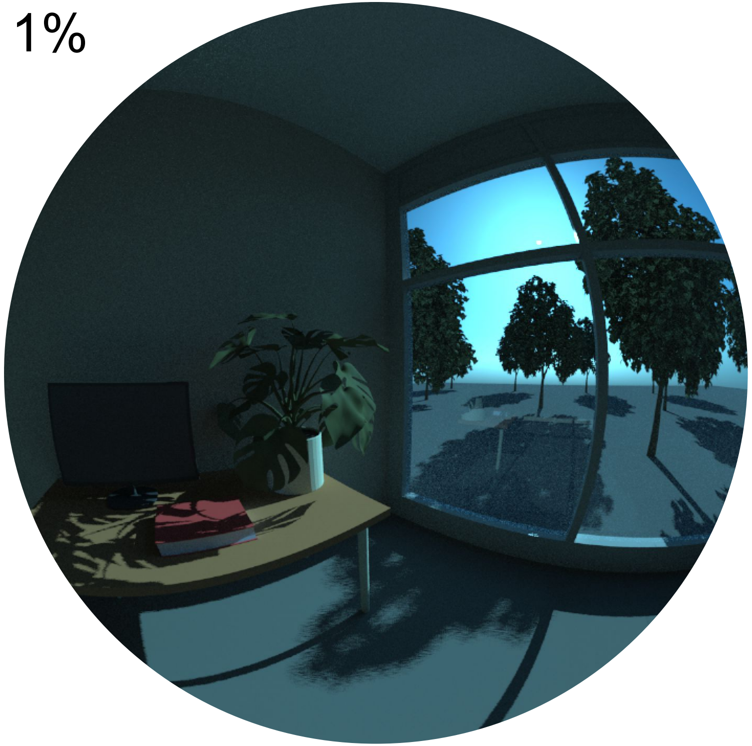
Convert hsr to hdr

# Electrochromic

Tvis 60%



Tvis 1%



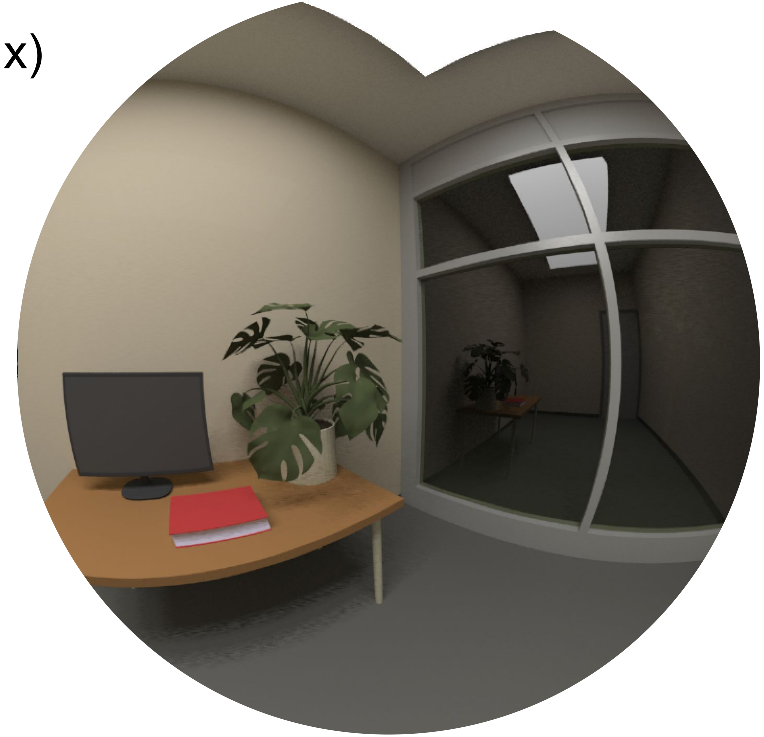
Runtime: ~5 min with -n 8

# Daylighting dimming

To meet target workplane illuminance (500 lx)

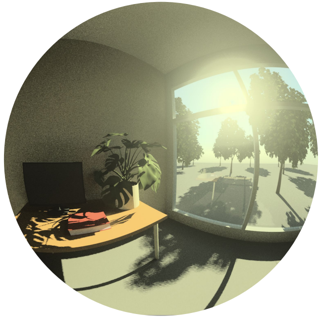
Light source:
- ONLY electric lighting
- no sky

Runtime: ~13 min with -n 8



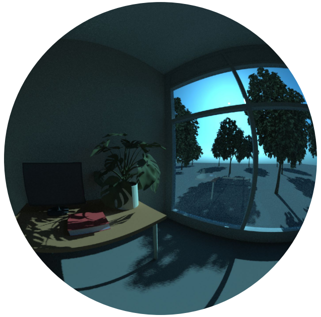
# Combine electrochromic tinting and daylight dimming

Clear state



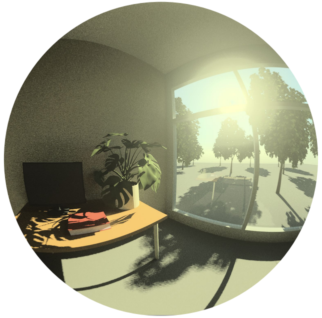
**x tinted factor**

Tinted state



# Combine electrochromic tinting and daylight dimming

Clear state

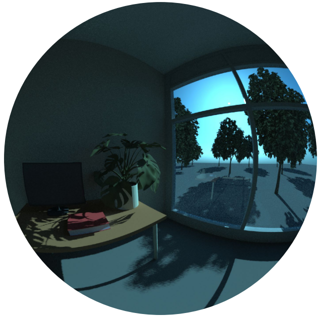


**x tinted factor +**

Fully powered electric light

**x dimming factor**

Tinted state

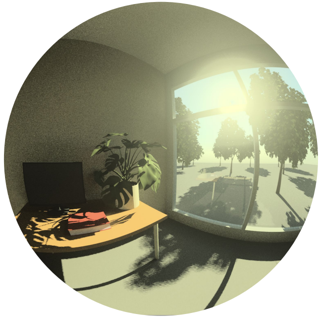


# Combine electrochromic tinting and daylight dimming

Clear state

Fully powered electric light

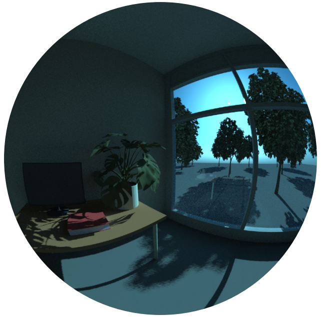
=



x tinted factor +

x dimming factor

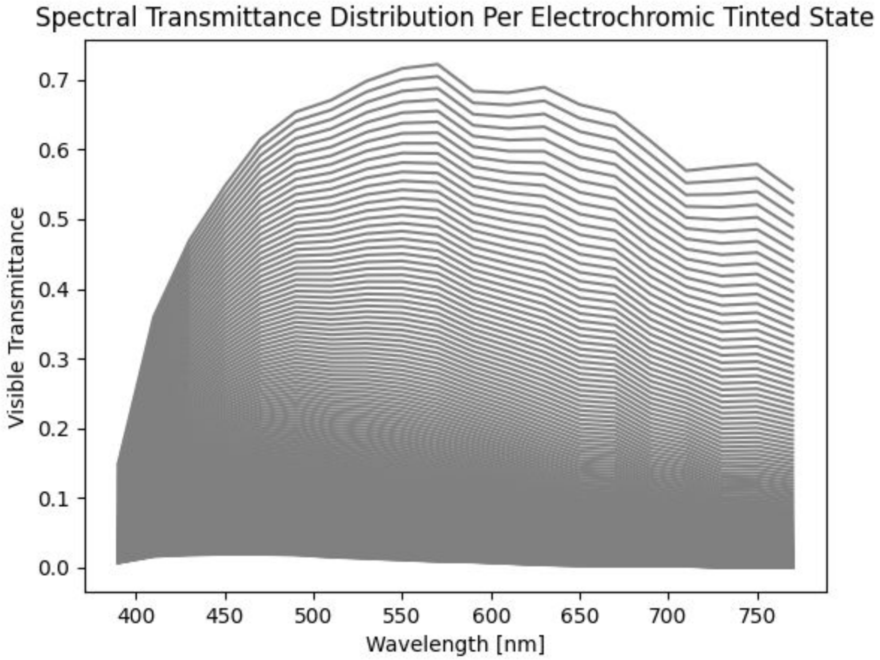
Tinted state

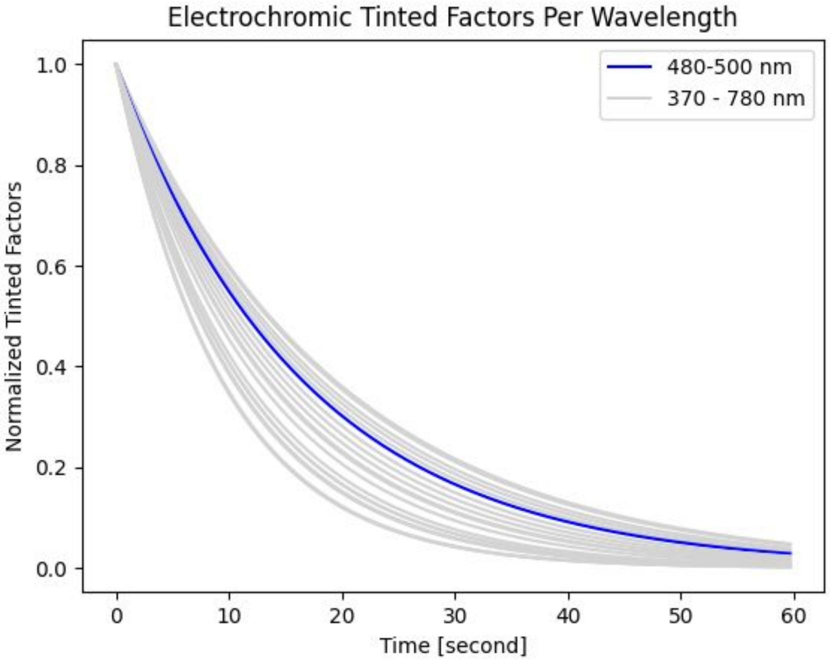


# Electrochromic tinting curve

exponential function: y = ab^x

Total 180 tinted states



# Combine electrochromic tinting and daylight dimming

```
tinted = pr.RcombInput("./tinted.hsr", scale=tinted_factor)
```

20 tinted/dimming scalars map to 20 channels

```
dimming = pr.RcombInput("./dimming.hsr", scale=[dimming_factor]*20)
```

```
combined_hdr = pr.rcomb([tinted, dimming], transform='RGB')
```

Convert hsr to hdr

```
combined_downsampled_hdr = pr.pfilt(combined_hdr, xres='/3', yres='/3',
gaussian_filter_radius=.6, one_pass=True)
```

downsample

```
with open("./combined_downsampled_hdr", "wb") as file:
        file.write(combined_downsampled_hdr)
```

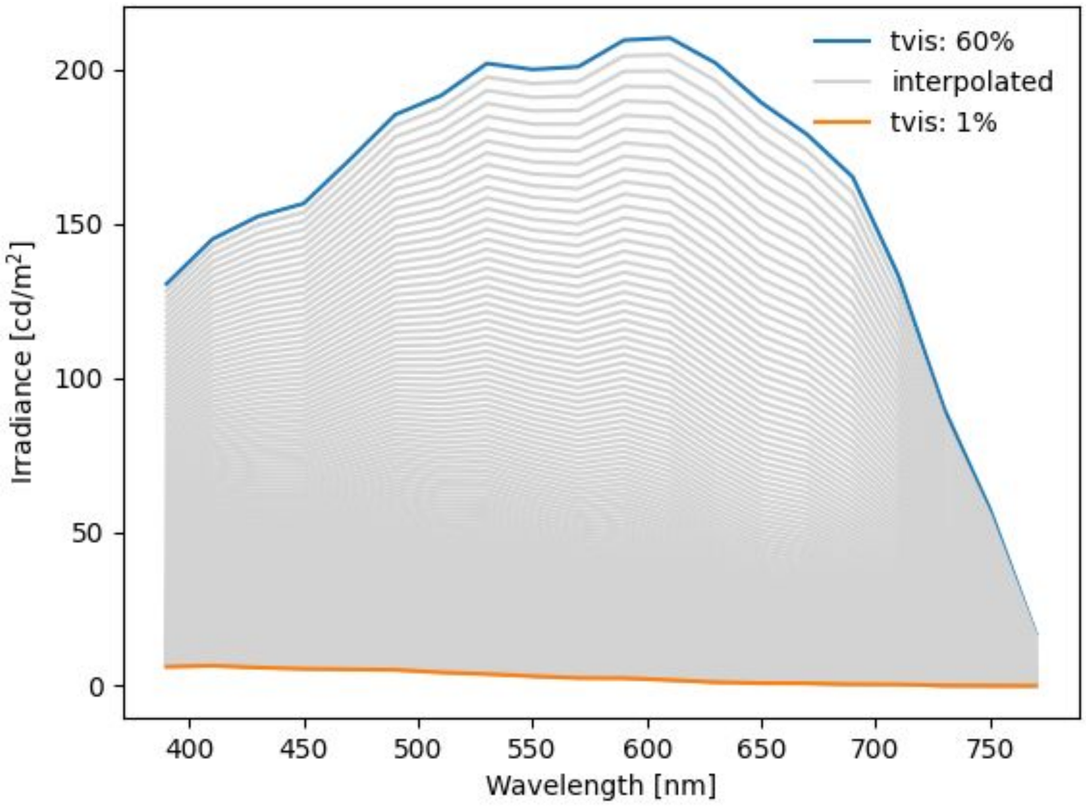
# Add tone mapping

```
combined_downsampled_tonemapped_hdr = pr.pcond(
        "./combined_downsampled.hdr",
        human=True
    )
```

Turn on human visual adjustment

# Spectral irradiance

```
params =
pr.SamplingParameters(
     ab=2,
     aa=0,
     ad=2048,
     lw=1e-5,
     dj=0.8,
     cs=20,  - 20 spectra channels
     co=True - spectral output = True
)
irradiance = pr.rtrace(
     rays=view_pt,
     octree=room.octree,
     params=params.args(),
     irradiance=True,
     header=False
)
```

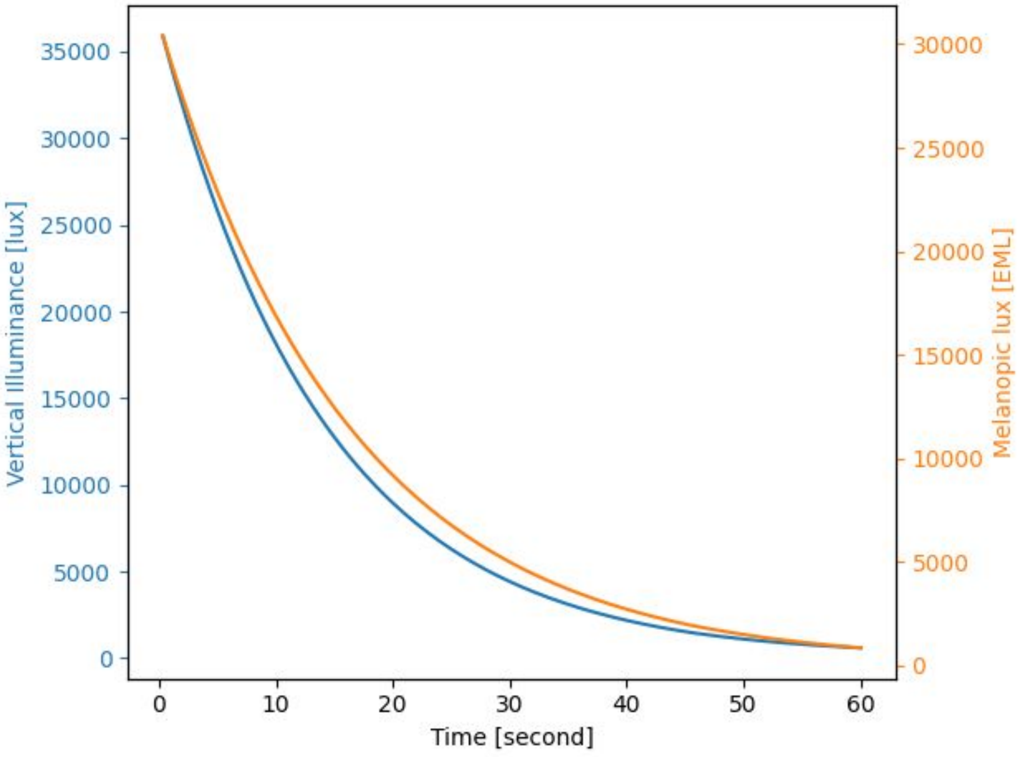


# Illuminance and melanopic illuminance

```
ev = pr.rtrace(
    rays=view_pt,
    octree=room.octree,
    params=params.args()+["-pY"],
    irradiance=True,
    header=False
)

mlux = pr.rtrace(
    rays=view_pt,
    octree=room.octree,
    params=tr_params.args()+["-pM"],
    irradiance=True,
    header=False
)
```

photopic

melanopic



# Thermal load

- Use **pywincalc** to calculate the solar heat gain coefficient of EC tvis 60% and 1%
- thermal load = solar heat gain * incident irradiance

# Incident irradiance

- Build an empty scene with gendaylit sky
- get irradiance at view point facing same direction as window

```
scene1 = pr.Scene("scene1")

sky_prims = pr.gendaylit(datetime.datetime(2024,12,21,12), 37, 122, 120,
solar=True, dirnorm=800, diffhor=80).decode()

for prim in pr.parse_primitive(sky_prims):

    scene1.add_source(prim)

scene1.add_source(pr.Primitive("skyfunc", "glow", "skyglow", [], [1,1,1,0]))

scene1.add_source(pr.Primitive("skyglow", "source", "skydome", [],
[0,0,1,180]))

scene1.add_source(pr.Primitive("skyglow", "source", "groundplane", [],
[0,0,-1,180]))

incident_irradiance = float(pr.rtrace(b"0 -10 10 0 -1 0", scene1.octree,
header=False, params=["-ab", "1", "-I"]).decode().split()[0])
```

# Pywincalc - Environment data

See pyWinCalc/examples/environmental_conditions_user_defined.py

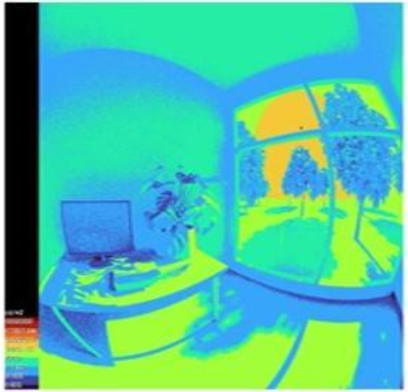
```
inside_environment = pywincalc.Environment(
    air_temperature=294.15,
    pressure=101325.0,
    convection_coefficient = 0.0,
    coefficient_model = pywincalc.BoundaryConditionsCoefficientModelType.CALCULATED_H,
    radiation_temperature = 294.15,
    emissivity = 1.0,
    air_speed = 0.0,
    air_direction = pywincalc.AirHorizontalDirection.NONE,
    direct_solar_radiation = 0
)

outside_environment = pywincalc.Environment(
    air_temperature=285.15,
    pressure=101325.0,
    convection_coefficient = 26.0,
    coefficient_model = pywincalc.BoundaryConditionsCoefficientModelType.CALCULATED_H,
    radiation_temperature = 255.15,
    emissivity = 1.0,
    air_speed = 5.0,
    air_direction = pywincalc.AirHorizontalDirection.WINDWARD,
    direct_solar_radiation = incident_irradiance
)

environmental_conditions = pywincalc.Environments(outside_environment, inside_environment)
```

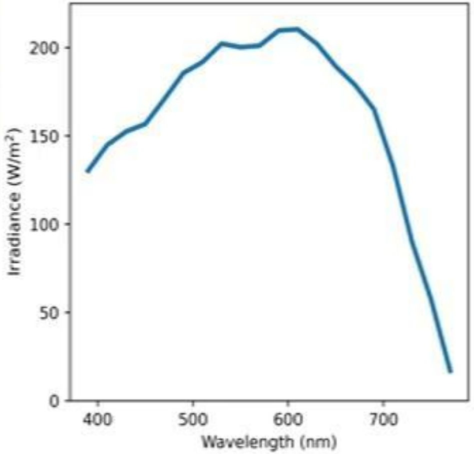
# Pywincalc - glazing system

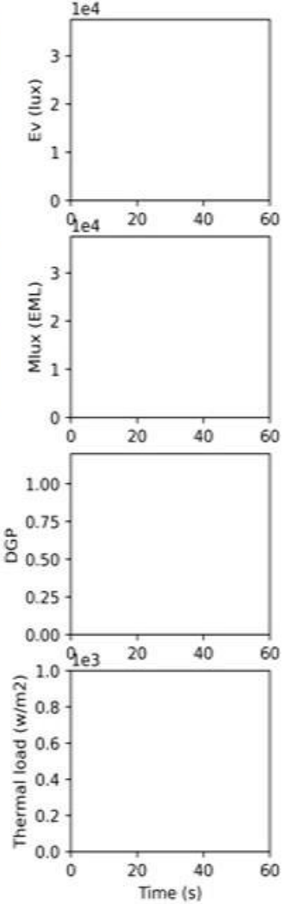
```
ec01 = pywincalc.parse_json_file(
          "./igsdb_product_7405.json"
)

solid_layers = [ec01]

gs01 = pywincalc.GlazingSystem(
       solid_layers=solid_layers,
       environment=environmental_conditions
)

solar_heat_gain = gs01.shgc()
```

https://igsdb.lbl.gov/

thermal load = solar_heat_gain * incident_irradiance






Time: 12:00:00.0
Photopic illuminance: 35909 lux
Melanopic illuminance: 30390 EML
DGP: 1.0
Thermal load: 843 w/m2

# Falsecolor

pyradiance

```
pr.falsecolor("./combined_downsampled_hdr", contour="p", scale='1e6', decades=6)
```

Or

Matplotlib (slightly prettier version, maybe?)

```
 _, yres, _, xres = pr.getinfo(combined_ds_hdr, dimension_only=True).decode().split()

xres, yres = int(xres), int(yres)

img_data = pr.pvalue(combined_ds_hdr, header=False, outform='f', resstr=False)

img_array = np.frombuffer(img_data, dtype=np.single).reshape(xres, yres, 3)

luminance = img_array[:, :, 0] * 47.4 + img_array[:, :, 1] * 119.9 + img_array[:, :,
2] * 11.6

plt.imshow(luminance, cmap='turbo', norm=mpl.colors.LogNorm(vmax=1e6, vmin=5.623))

plt.colorbar()

plt.axis('off')
```

# Convert hdr to ppm, to png

```
combined_downsampled_tonemapped_ppm = pr.ra_ppm(
      combined_downsampled_tonemapped_hdr
)

with open(f"./tinted.ppm", "wb") as file:
       file.write(combined_downsampled_tonemapped_hdr)

subprocess.run(f"magick ./tinted.ppm ./tinted.png".split())
```

# Thank you! Questions?

See workflow at

https://github.com/TammieYu/spectral_simulation_of_electrochromic

Tammie Yu - cuitingyu@lbl.gov

Taoning Wang - taoningwang@lbl.gov