Machine Learning for Rendering

Framework Presentation

Python Ray Tracer - PyRT

2025-2026

Master in Intelligent Interactive Systems (MIIS)

Ricardo Marques (ricardo.marques@upf.edu)

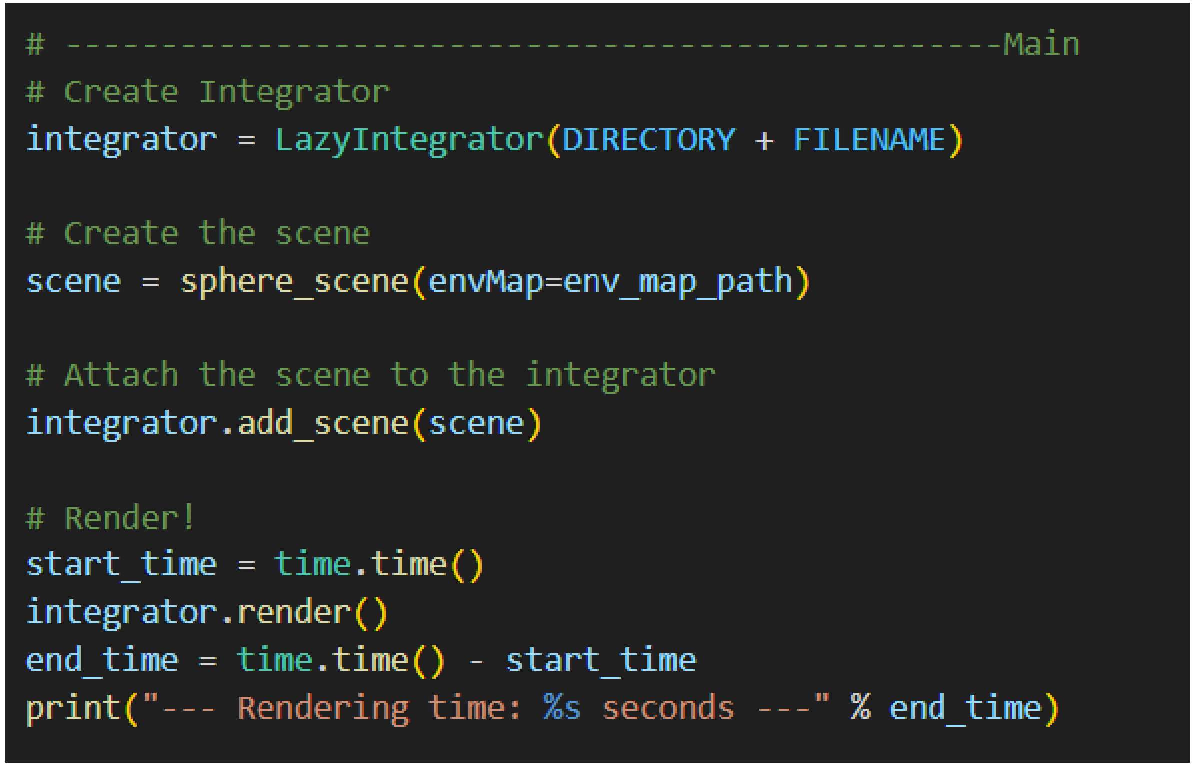
Python Ray Tracer - PyRT

A Ray Tracer for Easily Prototyping Monte Carlo-based Rendering

The Framework (4 files)

- *PyRT_Common.py*
  - Basic classes such as *Ray*, *Vector3D*, *RGBColor*, etc.
  - Useful set of functions mainly related to spherical projections and transforms
- *PyRT_Integrators.py*
  - Relies on *PyRT_Common.py*
  - Provides a template *Integrator* class (i.e., an Abstract Base Class, ABC)
  - Different concrete *Integrator* classes implement different strategies for computing a pixel's color
- *PyRT_Core.py*
  - Relies on *PyRT_Common.py*
  - Contains the core of the Ray Tracer
  - Most important classes: *Scene*, *Primitive* (i.e., shapes), *BRDF* (i.e., material), etc.
- *AppRenderer.py*
  - Script which sets-up the
  - Create scene, create integrator, launch rendering,…
  - Store result to file, show it to the user

AppRender.py



```
# -------------------------------------------------Main
# Create Integrator
integrator = LazyIntegrator(DIRECTORY + FILENAME)

# Create the scene
scene = sphere_scene(envMap=env_map_path)

# Attach the scene to the integrator
integrator.add_scene(scene)

# Render!
start_time = time.time()
integrator.render()
end_time = time.time() - start_time
print("--- Rendering time: %s seconds ---" % end_time)
```

PyRT_Core.py – Scene

class Scene

Camera	Rendered Image	Environment Map
List of Objects	List of Point Lights	Scene’s Ambient Light

- Setters and Getters
- any_hit(Ray r)
  - Given a ray r, determine whether or not there exists any intersection between r and the scene’s geometry
  - Returns a Boolean
- closest_hit(Ray r)
  - Given a ray r, determine the closest intersection (if it exists) between r and the scene’s geometry
  - Returns a structure (of type HitData) with the details of the closest intersection

PyRT_Core.py - Primitive

class Primitive (Abstract)

Emission (RGB Color)	BRDF (material)

- Setters and Getters
- intersect(Ray r)
  - Abstract method
  - Given a ray r, determine the closest intersection (if it exists) between r and the primitive
  - Returns a structure (of type HitData) with the details of the closest intersection

class Sphere(Primitive)

class InfinitePlane(Primitive)

class Parallelogram(Primitive)

PyRT_Core.py – BRDFs, Point Lights, and Camera

class BRDF (Abstract)

- getValue(in, out, normal)
  - Abstract method
  - Given a pair of directions (in, out) and a normal vector, return the value of the material reflection

class Lambertian(BRDF)

class PointLight

Position	Intensity

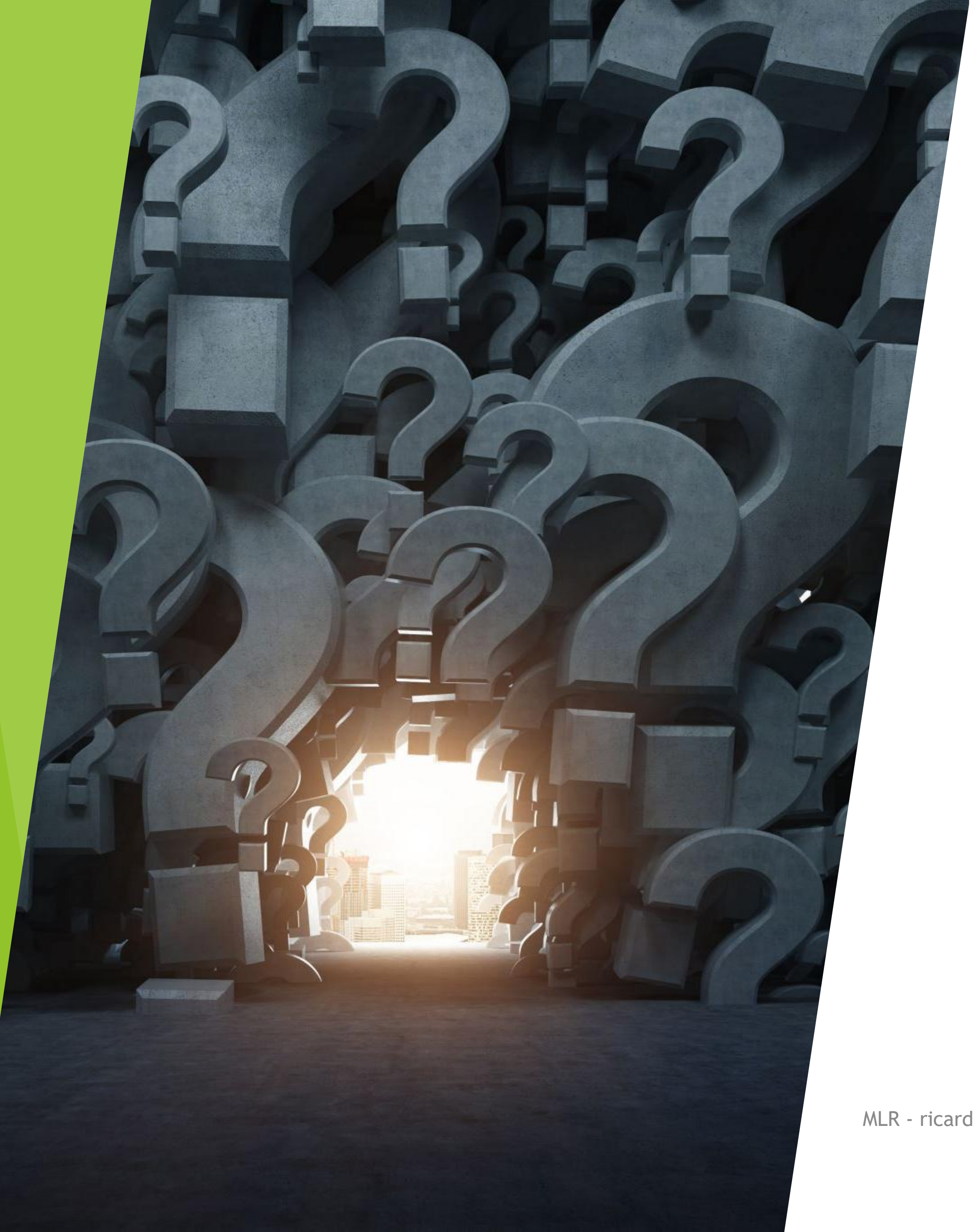
class Camera

Width	Height	Vertical fov	Aspect Ratio

- getDirection(x, y)
  - Given a pixel’s coordinates (x, y) return a ray with origin at the camera position, passing through pixel (x, y)

PyRT_Core.py – Scene

- Useful classes
  - Vector3D (operator overloading)
  - RGBColor (operator overloading and other functions)
  - Ray (origin, direction, t_max, t_min)
  - HitData (has_hit, hit_point, normal, hit_dist, primitive_index)
- Other classes
  - Function, PDF and Environment Map (for later)
- Useful Functions
  - OrientNormal, EuclideanToDisk, SampleSetHemishpere, VisualizeSampleSet
  - OrientHemiDir, RotateAroundY, CenterAroundNormal



The End

Questions?