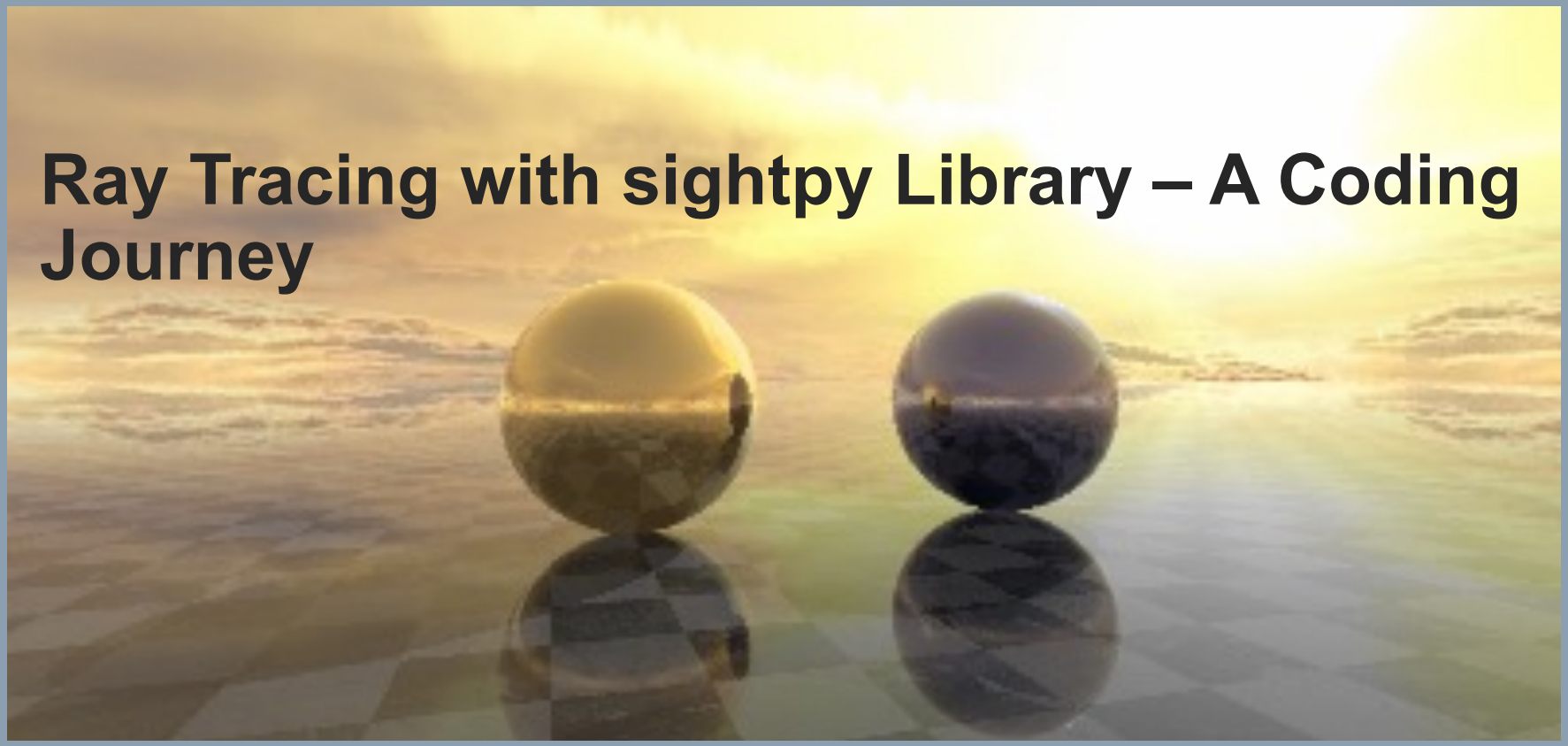
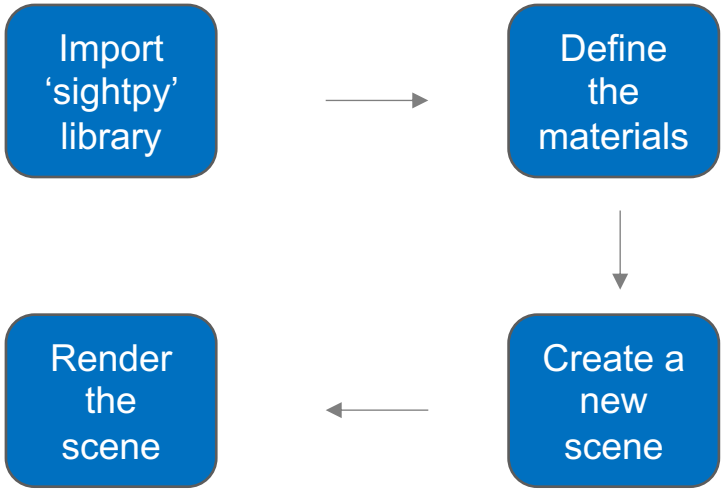
Ray Tracing with sightpy Library – A Coding Journey



5. sightpy Library

5.1 Code Flow

- Following is the code flow:



5. sightpy Library

5.2 Features

- A powerful Python tool.
- Simplifies the implementation of ray tracing projects.
- The collection of classes and functions.
- Handles ray generation, intersection testing, and shading computations.

5. sightpy Library

5.3 Materials

- gold_metal and bluish_metal.
- Type "Glossy" materials.
- Shiny, reflective surfaces.
- The floor material is also of type "Glossy" but uses an image texture ("checkered_floor.png").

```
gold_metal = Glossy(diff_color = rgb(1., .572, .184), n = vec3(0.15+3.58j, 0.4+2.37j,
1.54+1.91j), roughness = 0.0, spec_coeff = 0.2, diff_coeff= 0.8) # n = index of refraction
bluish_metal = Glossy(diff_color = rgb(0.0, 0, 0.1), n = vec3(1.3+1.91j, 1.3+1.91j, 1.4+2.91j),
roughness = 0.2,spec_coeff = 0.5, diff_coeff= 0.3)

floor =  Glossy(diff_color = image("checkered_floor.png", repeat = 80.),
            n = vec3(1.2+ 0.3j, 1.2+ 0.3j, 1.1+ 0.3j), roughness = 0.2, spec_coeff = 0.3,
diff_coeff= 0.9 )
```

5. sightpy Library

5.4 Scene Creation Pipeline

5. sightpy Library

5.5 Camera Setup

- A camera is added to the scene with the specified look_from, look_at positions
- The dimensions of the camera's image plane (400x300 pixels

```
angle = -np.pi/2 * 0.3
Sc.add_Camera(look_from = vec3(2.5*np.sin(angle), 0.25, 2.5*np.cos(angle)  -1.5 ),
        look_at = vec3(0., 0.25, -3.),
        screen_width = 400 ,
        screen_height = 300)
```

5. sightpy Library

5.6 Light Setup

- A directional light source is added to the scene with direction and colour.
- A directional light emits parallel rays of light, simulating sunlight.

```
Sc.add_DirectionalLight(Ldir = vec3(0.52,0.45, -0.5),  color = rgb(0.15, 0.15, 0.15))
```

5. sightpy Library

5.7 Objects

- Two spheres and a plane.
- Each object is associated with a material, positions, sizes.

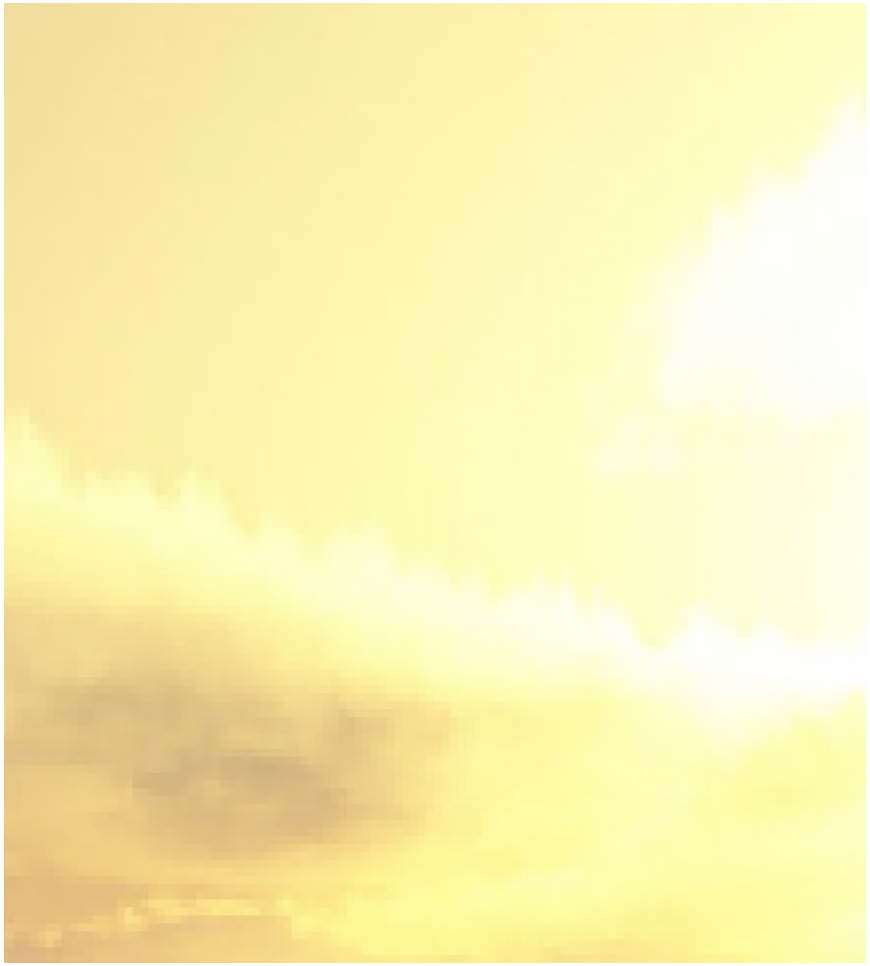
```
Sc.add(Sphere(material = gold_metal, center = vec3(-.75, .1, -3.),radius =  .6, max_ray_depth = 3))
Sc.add(Sphere(material = bluish_metal, center = vec3(1.25, .1, -3.), radius = .6, max_ray_depth = 3))

Sc.add(Plane(material = floor,  center = vec3(0, -0.5, -3.0), width = 120.0,height = 120.0, u_axis = vec3(1.0, 0, 0), v_axis = vec3(0, 0, -1.0),  max_ray_depth = 3))
```

5. sightpy Library

5.8 Background

- ("stormydays.png")
- Provides the environment for the ray tracing.

```
Sc.add_Background("stormydays.png")
```



- The render() method is called on the Scene object to generate the image.
- The parameter samples_per_pixel=6 controls the number of samples per pixel
- Used for antialiasing and improving image quality.

```
img = Sc.render(samples_per_pixel = 6)
```

Classes in sightpy Library

A Brief Explanation

5. sightpy Library

5.10 Glossy

Here's a brief explanation of the code for the Glossy class:

- **Class Definition**: Used to define materials for objects in the ray tracing scene.
- **Constructor**: The __init__ method initializes the Glossy material with specific attributes like diff_color, roughness, spec_coeff, diff_coeff, and n (index of refraction).
- **Color Calculation**: The get_color method calculates the color of the intersected object based on its material properties and the lighting conditions in the scene.
- **Diffuse Reflection**: Calculates the amount of light reflected by the surface based on the angle between the surface normal and the light direction.
- **Shadow Handling**: The code checks if the intersected point is in shadow or not.
- **Reflection Handling**: The code handles reflections by tracing rays from the intersected point in the reflected direction and calculating the color contribution from the reflected ray.
- **Fresnel Factor**: Controls the amount of light reflected and transmitted at the intersection point.
- **Surface Nudging**: A small offset is added to the intersection point to avoid self-intersections when calculating reflections.

5. sightpy Library

5.11 Camera

Represents the virtual camera in the ray tracing scene.

- Key aspects of the Camera class include:
  - **Attributes:** screen_width, screen_height, look_from, look_at, field_of_view, aperture, and focal_distance.
  - **Camera Reference Basis:** The camera's reference basis in world coordinates is defined using the cameraFwd, cameraRight, and cameraUp vectors.
  - **Pixel Coordinates:** Casts rays from the camera to the scene.
  - **get_ray:** This method generates a ray from the camera's position and direction.

5. sightpy Library

5.12 Lights

The lights.py file contains the Light class and its subclasses, PointLight and DirectionalLight, representing light sources in the ray tracing scene.

- **Light Class:** The Light class is an abstract base class that defines common attributes and abstract methods for light sources.
- **PointLight Class:** The PointLight subclass represents a point light source and implements methods like get_L, get_distance, and get_irradiance.
- **DirectionalLight Class:** The DirectionalLight subclass represents a directional light source, which emits light in a specific direction. It also implements methods like get_L, get_distance, and get_irradiance.

Represents a sphere object in the ray tracing scene.

- **Constructor**: The __init__ method initializes the Sphere object with attributes such as center, material, radius, max_ray_depth, and shadow.
- **Collider Initialization**: Creates a Sphere_Collider object and adds it to the collider_list for intersection testing.
- **UV Mapping**: The method get_uv is used to get the texture coordinates (u, v) of the hit point on the sphere's surface. It calculates the phi and theta angles based on the hit point's position.
- **Sphere_Collider Class**: Represents the collider for the sphere object.
- **Intersection Test**: Tests the intersection between a ray (O, D) and the sphere collider. It calculates the discriminant and determines if the ray intersects with the sphere. If it intersects, it returns the hit distance h and the hit orientation.
- **Normal Calculation**: Calculates the surface normal at the intersection point (hit point) on the sphere. It returns the normalized vector from the sphere's center to the hit point.
- **UV Calculation**: Calculates the texture coordinates (u, v) for the hit point on the sphere's surface using phi and theta angles.

5. sightpy Library

5.14 Plane

Represents a plane object in the ray tracing scene.

- **Constructor**: Initializes the Plane object with attributes such as center, material, width, height, u_axis, v_axis, max_ray_depth, and shadow.
- **Collider Initialization**: Creates a Plane_Collider object and adds it to the collider_list for intersection testing.
- **UV Mapping**: The method get_uv is used to get the texture coordinates (u, v) of the hit point on the plane's surface. It calculates the u and v coordinates based on the hit point's position and the u and v axes.
- **Plane_Collider Class**: Represents the collider for the plane object.
- **Intersection Test**: The intersect method tests the intersection between a ray (O, D) and the plane collider. It calculates the intersection point M and the distance dis from the ray origin to the intersection point. Checks if the intersection point is inside the plane's boundaries and whether the ray direction is upwards or downwards.
- **Rotation**: The rotate method allows rotating the plane collider around a specified center point using a transformation matrix M.
- **Normal Calculation**: Returns the normal vector of the plane's surface, which is precalculated during initialization.

5. sightpy Library

5.16 Render

Contains functions for rendering images and implementing path tracing for the ray tracing project:

- **trace_ray Function:** Calculates the intersection between a ray and objects in the scene and returns the position and normal of the hit.
- **trace_ray_material Function:** Similar to trace_ray but also returns emission.
- **sample_sphere and sample_hemisphere Functions:** Used for sampling points on a sphere and hemisphere, respectively, which are useful for path tracing.
- **to_image Function:** Converts a numpy array or vec3 buffer to a PIL Image, allowing visualization of the rendered scene.
- **render_path_tracing and render_path_tracing_MIP_lambert Functions:** These functions implement path tracing for the ray tracing scene, where samples are traced through the scene to compute the final image color.

5. sightpy Library

5.17 Ray

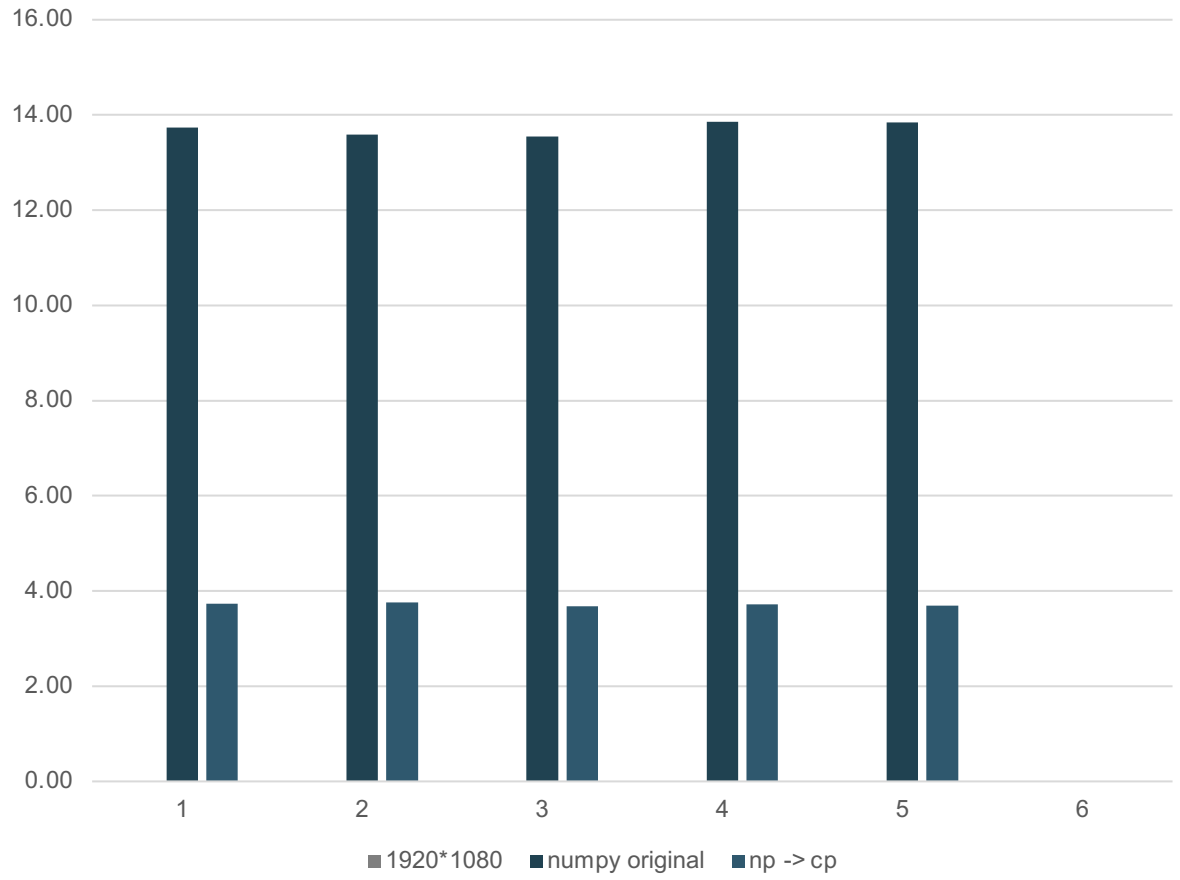
Contains the Ray and Hit classes, essential for ray-object intersection and tracing. Let's explore these classes:

- **Ray Class:** Represents a ray in the scene and includes attributes like origin, direction, depth, n (index of refraction), reflections, transmissions, and diffuse_reflections.
- **Hit Class:** Stores information about the intersection between a ray and a surface, including distance, orientation, material, collider, and surface properties.
- **get_raycolor Function:** Traces rays in the scene and computes the color contribution from each intersected object based on material properties.

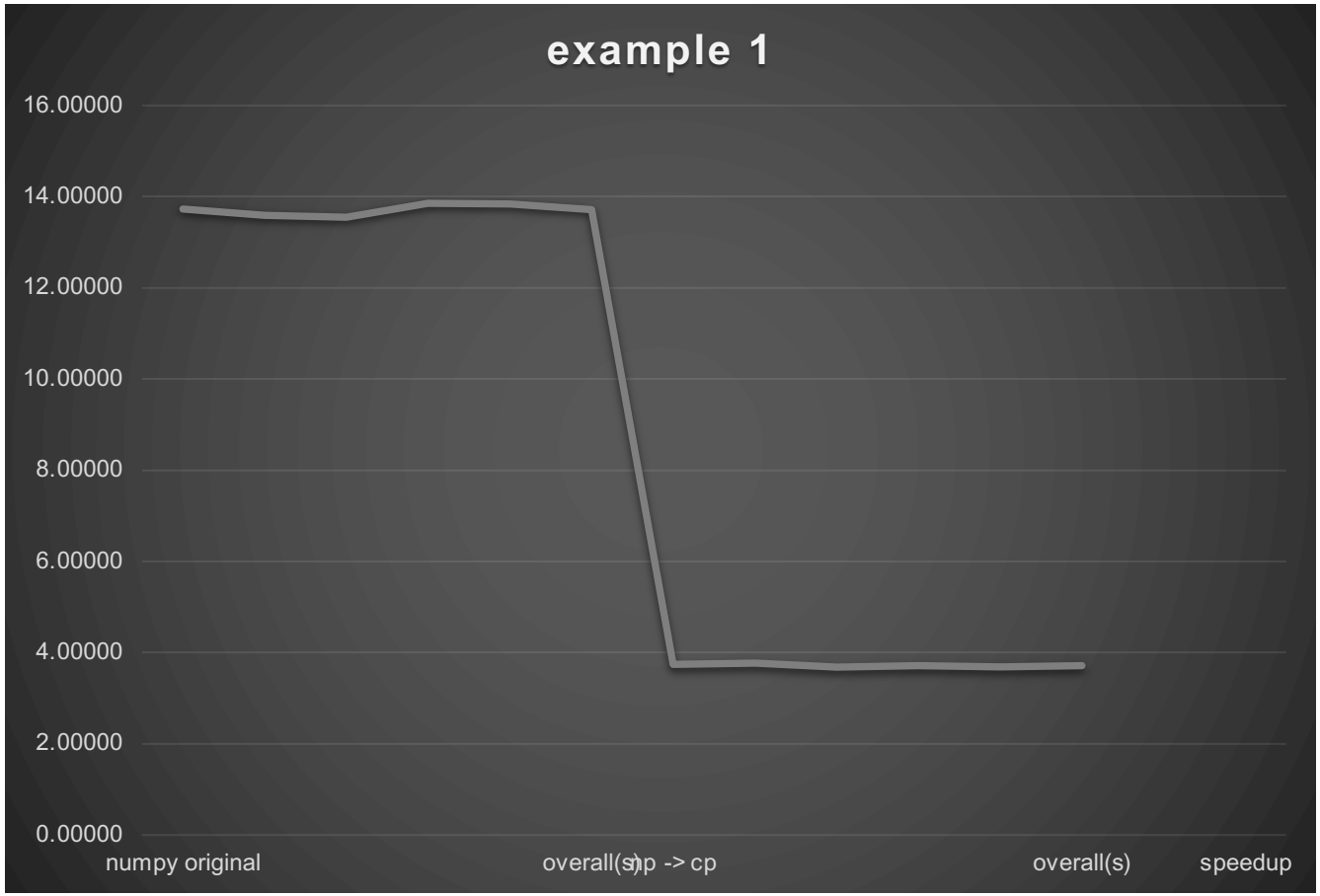
Results Comparison

6 Comparison & Summary

6.1 Cupy vs Numpy



Contd.


example 1
16.00000
14.00000
12.00000
10.00000
8.00000
6.00000
4.00000
2.00000
0.00000
numpy original
overall(s)
np -> cp
overall(s)
speedup

Thank You!