1 Implementation

1.1 Blender Exporter (Python)

Implemented `Export.py` to export scene data from Blender to ASCII format. The script exports:

- Camera: location, gaze direction, up direction, focal length, sensor dimensions, resolution
- Lights: location and intensity
- Primitives: spheres, cubes, and planes

Output files are saved to `../ASCII/` relative to the `.blend` file location.

1.2 Camera Class (C++)

Implemented a `Camera` class in `camera.h/cpp` with:

- `readFromFile()`: Parses exported ASCII file and stores camera parameters
- `calculateBasis()`: Computes orthonormal basis vectors (right, up, forward) from gaze direction using cross products
- `pixelToRay()`: Converts 2D pixel coordinates to 3D rays using the pinhole camera model

Supporting `Vector3` struct includes operators for addition, subtraction, scalar multiplication/division, dot product, cross product, and normalization.

1.3 Image Class (C++)

Implemented an `Image` class in `image.h/cpp` for PPM file I/O:

- `readPPM()`: Parses P3 ASCII PPM format
- `writePPM()`: Writes images in P3 format
- `getPixel()/setPixel()`: Access individual pixels

2 Testing

2.1 Camera Testing

Created `test_camera.cpp` to verify:

- File reading: Loaded camera from `test.txt`, confirmed parameters matched Blender export

- Ray generation: Saved generated rays to `rays.txt` to display in Blender

Results: Camera data loaded correctly. Rays aligned with camera when loaded in Blender.

2.2 Image Testing

Created `test_image.cpp` to verify:

- Write: Generated 100×100 gradient and 300×100 RGB stripe test images
- Read/Write: Loaded gradient.ppm and re-saved as modified gradient_copy.ppm

Results: All images read and written successfully.