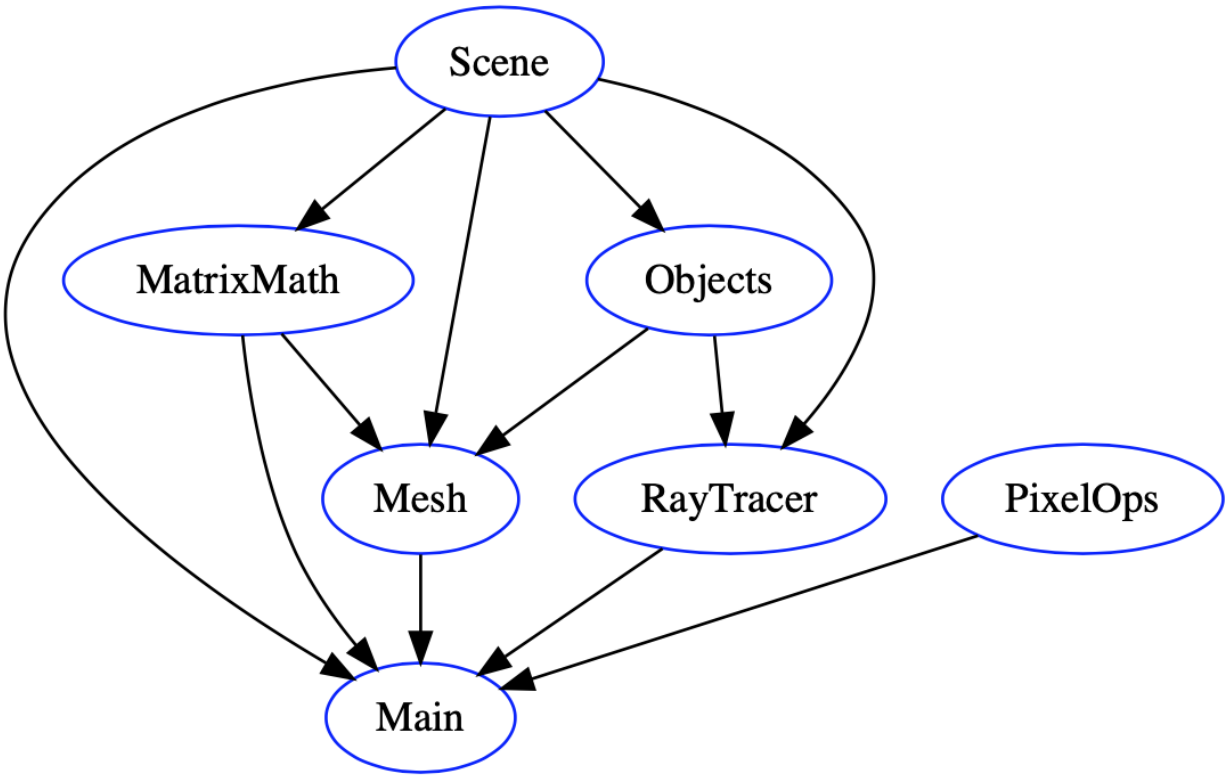
Project Name:

Trace-Ray

Structure of the Code:



- **meshes** : Contains `ply` files of different objects that can be rendered
- **renders** : Contains images that have been rendered by Trace-Ray
- **scenes** : Contains `yaml` files of scene descriptions. Each scene description contains the placement of the camera, the view plane, background color, lighting, and objects with their color, material and transformation properties.
- **src** : This contains the code files
  - **Main:** This is the entry point of the project which takes in a `yaml` file name and makes use of other modules to render an image for it.

- **MatrixMath:** This module contains functions which perform matrix operations to build world and view matrices, and to transform points to world and camera space.
- **Mesh:** This module defines a type to represent triangle meshes. Additionally, it takes in `ply` files and scene descriptions and creates meshes for each object in the scene.
- **Objects:** This module has type descriptions for vertices, rays and triangles. It also defines a ray-triangle intersection routine.
- **PixelOps:** This module contains helper functions to aid in transforming RGB Color data to ByteStrings which are used to render the image.
- **RayTracer:** This traces rays through the scene and collects resulting color values for all pixels.
- **Scene:** This contains types for the elements of our scene description `yaml` file. It contains definitions to help the YAML parser parse our scene.

Additional Haskell Libraries:

1. bmp
2. bytestring
3. colour
4. linear
5. ply-loader
6. text
7. vector
8. yaml

Running Trace-Ray:

Trace-Ray uses stack for project management.
To install all dependencies, just run

```
stack install
```

To run Trace-Ray on a specific `yaml` file, run

```
stack run <path-to-scene-file>
```

If you do not specify the filename, it uses `scenes/cube.yaml`

Questions:

1. Is it better to declare types inside their relevant modules or should we make a file containing all the type definitions?
2. Besides the absence of multi threading, is there any bottleneck in our code which is hindering performance?
3. Are there any Haskell idioms we could use to improve the readability of the code?