Order of Implementation *and progress*

(Done, In progress, Abandoned)

1. interpreter
2. empty space + background
3. simple rays without reflections
4. output into any format: simple pixel data
5. Sphere
6. Materials
7. Reflections
8. Refractions
9. Further objects:
    1. Plane
    2. Transformation
    3. Complex Objects
10. light sources
11. color

OpenCL interaction

- Interpretation of input data is done on CPU
- Setup of transformation matrices is also done on CPU
- Raytracing:
    - Ray calculations (i.e. intersections and matrix multiplications) are done on OpenCL device → exact setup for kernel number per ray to be done later
    - Base object reflections and color determination is done on OpenCL kernels again
- The resulting image is created on the GPU side by combining the colors returned by OpenCL kernels and sent to the CPU to present using OpenCV

File formats

- Self-written interpreter (with no syntax tree), all values are stored within it's class to be retrieved by the process that sorts rays to OpenCL-kernels and puts the values into a more local memory:
    - Comments are begun with a '/' and are one-line only
    - var assignment: x := y, where x is the name and y the value
        - necessary variables:
            - width, height
            - lookat_x, lookat_y, lookat_z
            - eyepos_x, eyepos_y, eyepos_z
            - ambient_r, ambient_g, ambient_b; ambient_int_r,

        ambient_int_g, ambient_int_b
        - max_reflections
- object creation: !x := y, where x is the name and y the type (sphere, hp)
- object mutation:
    - set material: !x ?= y, where x is an existing object and y is an existing material
    - scale: !x *= a b c, where x is an existing object and a, b, c are x/y/z scales respectively
    - rotate: !x #a= b, where x is an exisiting object, a is a coordinate to rotate around (x, y or z) and b is the degree in radian
    - transform: !x += a b c, where x is an existing object and a, b, c are x/y/z transforms
    - These are necessary to build the tree, but will ultimately not be respected due to the tree algorithm not working:
        - union: !z := x | y, where x and y are two existing objects
        - intersection: !z := x & y, where x and y are two existing objects
        - subtraction: !z := x - y, where x and y are two existing objects
- submit object: !x <=, where x is an existing object
    - submitting means that this object is the ‘primary’ object, meaning that all of its subobjects in the form of unions or intersections include all relevant scene objects. Once an object is submitted, any following submission will be ignored; similarly, changes to the object following submission will be ignored. The submitted object will be the basis for rendering.
- light sources:
    - *x := a b c r g b, where a, b, c are coordinates and r, g, b are color information
- materials:
    - ?x := ambref diffref specref rflec rfrac rfracind shiny r g b, where the values represent the properties of x (ambient reflection component, diffuse reflection component, specular reflection component, reflected component, refracted component, refracted index, shininess constant, color value)

If any line doesn’t have the correct format, the program throws an error and aborts.
If a variable/object is referenced that doesn’t exist, it is simply ignored.
If interpretation ends without an object being submitted or with base variables missing, the program shows an error and aborts.

Notes on the object tree: There is a hierarchy Complex=>Transformed=>Base Object, which needs to be followed through, otherwise an error is thrown at interpretation time. The specific complex operations are ignored due to the algorithm not working and me running out of time.

So, for example, a scene with simply a sphere and a plane might look like this:

```
/ Base vars
width := 600.0
height := 400.0
lookat_x := 0.0
lookat_y := 0.0
lookat_z := 0.0
eyepos_x := 4.0
eyepos_y := 1.0
eyepos_z := 0.5
ambient_r := 0.2
ambient_g := 0.2
ambient_b := 1.0
ambient_int_r := 1.0
ambient_int_g := 1.0
ambient_int_b := 1.0
raydepth := 4.0
/ Base material
?base := 0.0 0.5 0.3 0.3 0.0 1.125 4.0 0.1 0.5 0.1
?red := 0.6 0.2 0.1 0.3 0.0 1.2 1.2 1.0 1.0 0.0
/ Object tree
!sph := sphere
!sph ?= base
!sph += 0.0 1.75 0.0
!plane := hp
!plane ?= red
/!plane #x= 1
!u := sph | plane
!u <=
/ Let there be light!
*x := 0.0 4.0 0.0 1.0 1.0 1.0
*y := 1.0 0.0 3.0 1.0 0.0 0.0
```