# Monday

## 7:30-7:45, 8:45-9:15 Fill out last week’s work log

## 8:45-9:40 Help Bryan with the problem



This was the error. It used around 3,5 Gb of space on the stack

## 9:40-9:45 Fill out last week’s work log

## 9:45-10:30 Being distracted

I should start talking less honestly

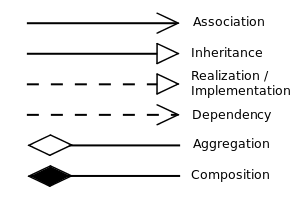
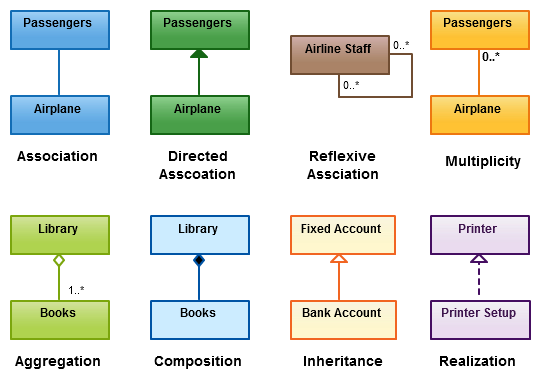
## 10:30-11:25, 12:00-12:25 Fill out last week’s learning log

## 11:25-12:15 Fill out today’s work log

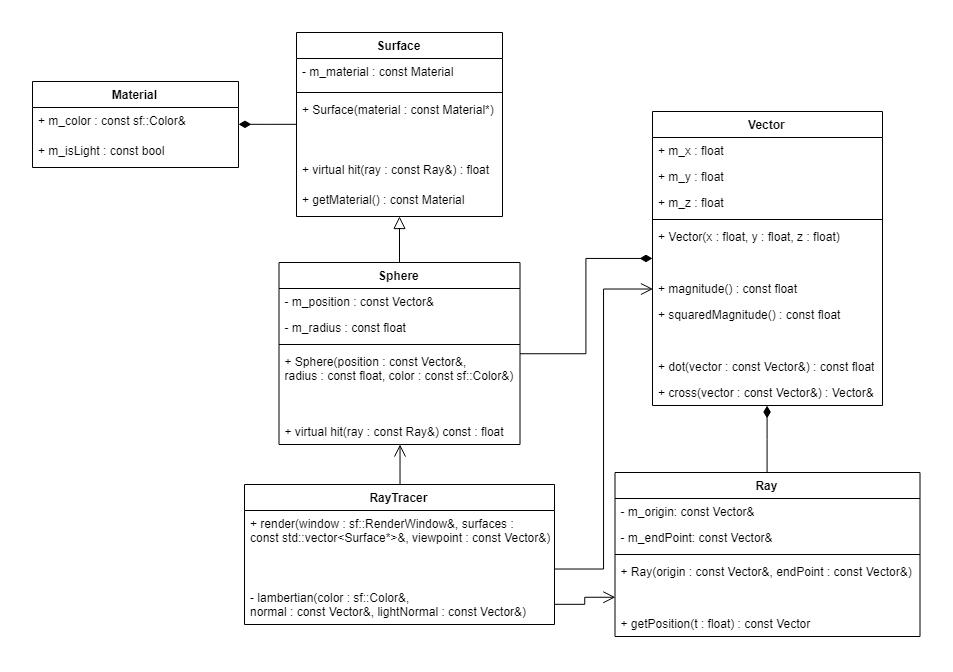
## 12:50-13:20 Help Mathijs with maths

He should research into linear algebra

## 13:20-16:00 Research on UML and class diagrams

* UML is a modeling language used to visualize the design of a system. There are 2 different views of a system model.
* Static (structural) design represents the static structure of a program. Class diagrams are a static design of UML.
* Dynamic (behavioral) design shows how a system behaves and how the internal states of objects changes. State machine and sequence diagrams are an example of this.
* There are three types of relationships:
* **Instance-level**
  + Association
    - One class use the service another class has to offer
    - Aggregation and Composition are a form of association
  + Aggregation
    - Association but the class uses the other class. The class can also only belong to one object
  + Composition
    - Association but the class owns the other class
* **Class-level**
  + Implement
    - Inherit from interface/pure virtual class
  + Inheritance
    - Inherit from anything but an interface
* **General** 
  + Dependency
    - Weakest relationship of all, used if the class is placed in function argument or as variable in the function body
  + Multiplicity
    - Shows the amount of references

## 16:20-16:30 Convert old UML to correct UML

I also updated this later today [here](#_pjzywcy3d5j)

## 16:30-16:55 Discussed Graphics terminology with Bas

## 16:55-17:10 Fill out today’s work log

## 18:20-18:55 Update existing class diagram

## 19:05-19:15 Research about Blinn-Phong shading

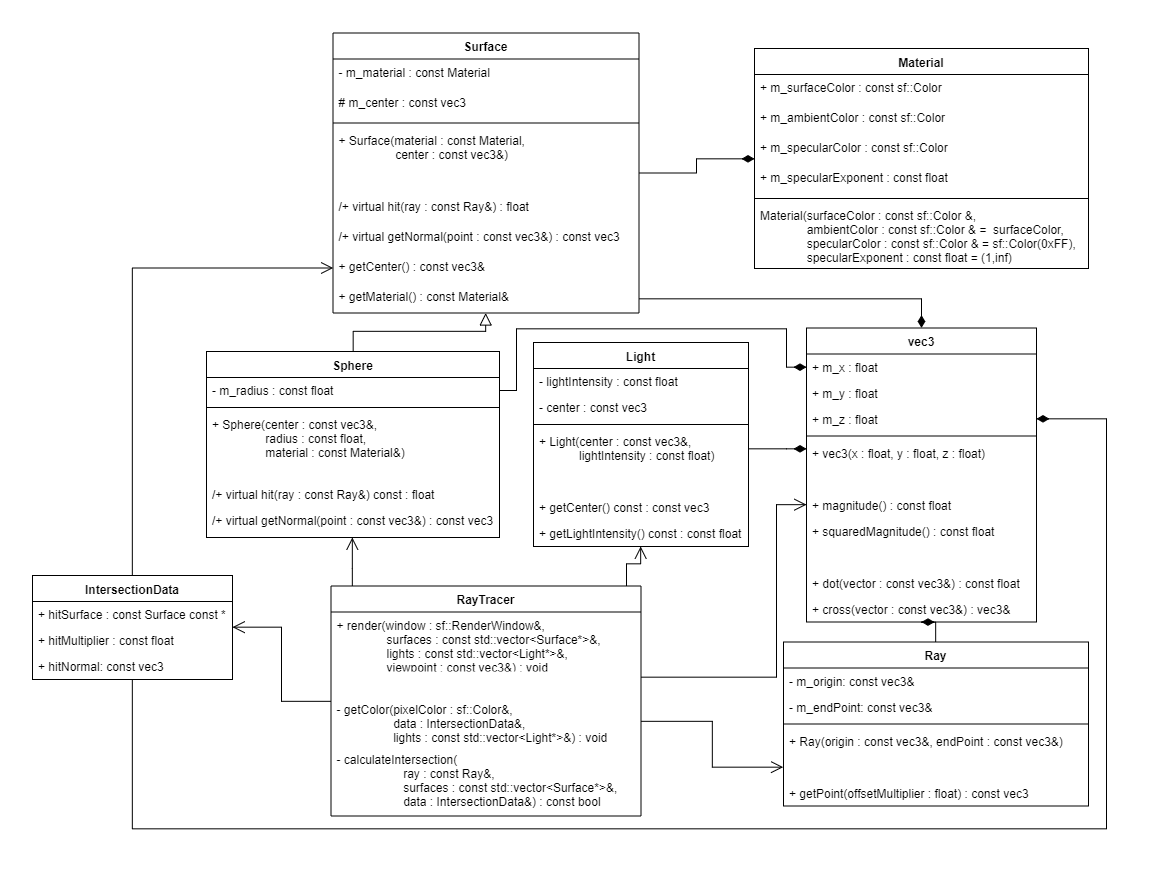
* Lambertian shading makes matte looking surfaces but it doesn't take the view angle in the calculation. It's view independent. Because of this, many people use Lambertian shading for the diffuse component and another algorithm for the specular component. Specular meaning highlights, shininess or reflection.
* Blinn (some guy from 1976) updated Phong's (some guy from 1975) model. I do not know who contributed what, but Blinn-Phong shading is one of the most commonly used algorithms from today.
* The algorithm is based the angle the view ray and the light source makes. If the angle is the same, we have a mirror-like reflection, It diminishes exponentially when the angle becomes less equal to each other
* We have two unit vectors. **v** is the unit vector to the viewpoint, **l** is the unit vector to the light and **n** is the unit surface normal. If we calculate the half unit vector **h** from **v** and **l,** then we dot the vector with **n**, we get a ratio that specifies how bright the highlight is

# Tuesday

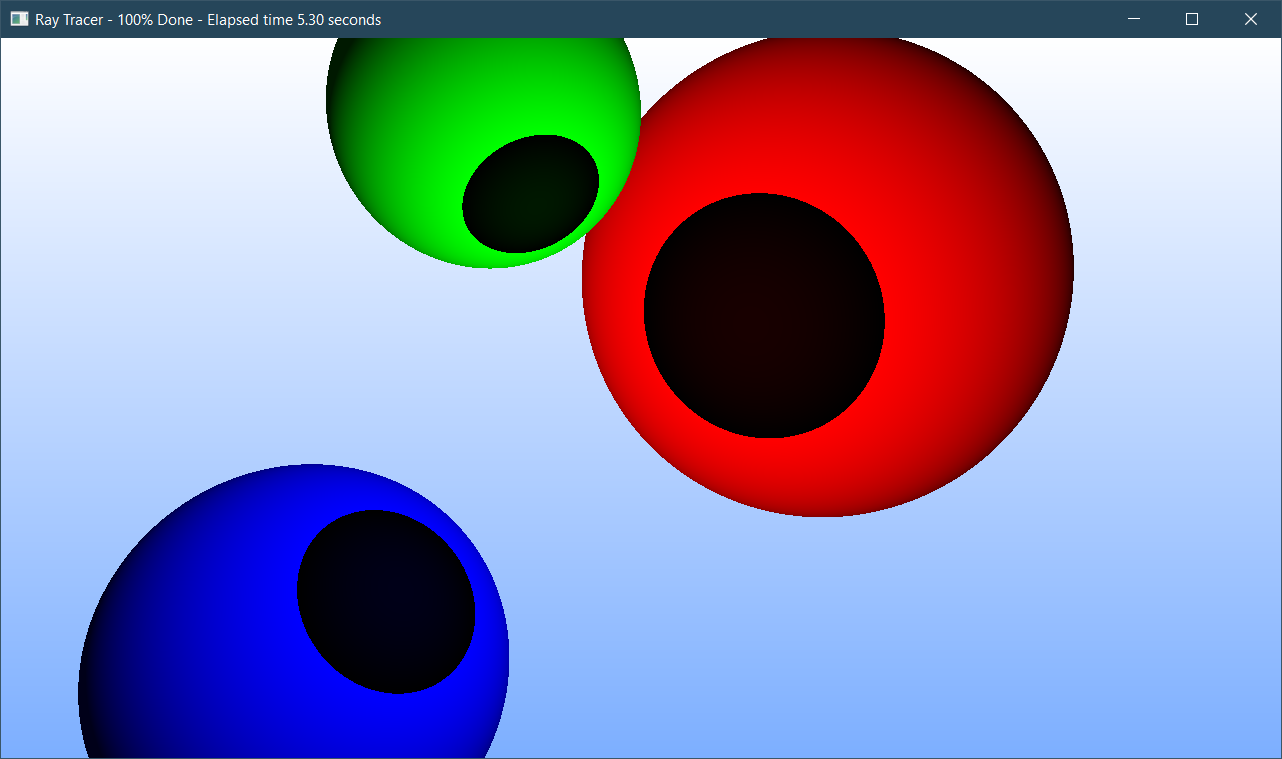
## 10:10-10:20 Fill out work log

## 11:30-13:20 Set up new iPad for little brother

## 10:20-11:30, 13:20-15:30 Implement point light sources

I will change up the ray tracer a bit to support light sources. At first, I wanted to make Point inherit from the surface class. But surfaces are supposed to be classes that can be hit by a ray. It wouldn’t make sense to make point inherit from the surface class because the ray tracer can’t collide with a point class. So I made it its own class. I recommend opening this image because I have lots of comments about this UML. [](https://drive.google.com/open?id=1QSAx-4ujgVgtgYvcwloaI90jG2o3MuuQ)

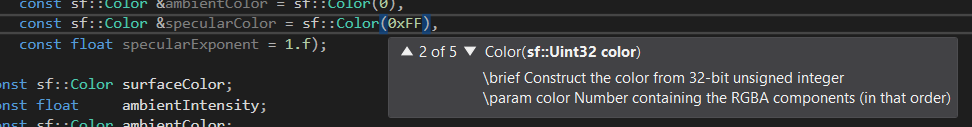
## 19:30-20:10 Fixing Blinn-Phong shading highlights

[](https://drive.google.com/open?id=14etrzaFBNoYoArZ74d4u3DMbSXp3mHwu)

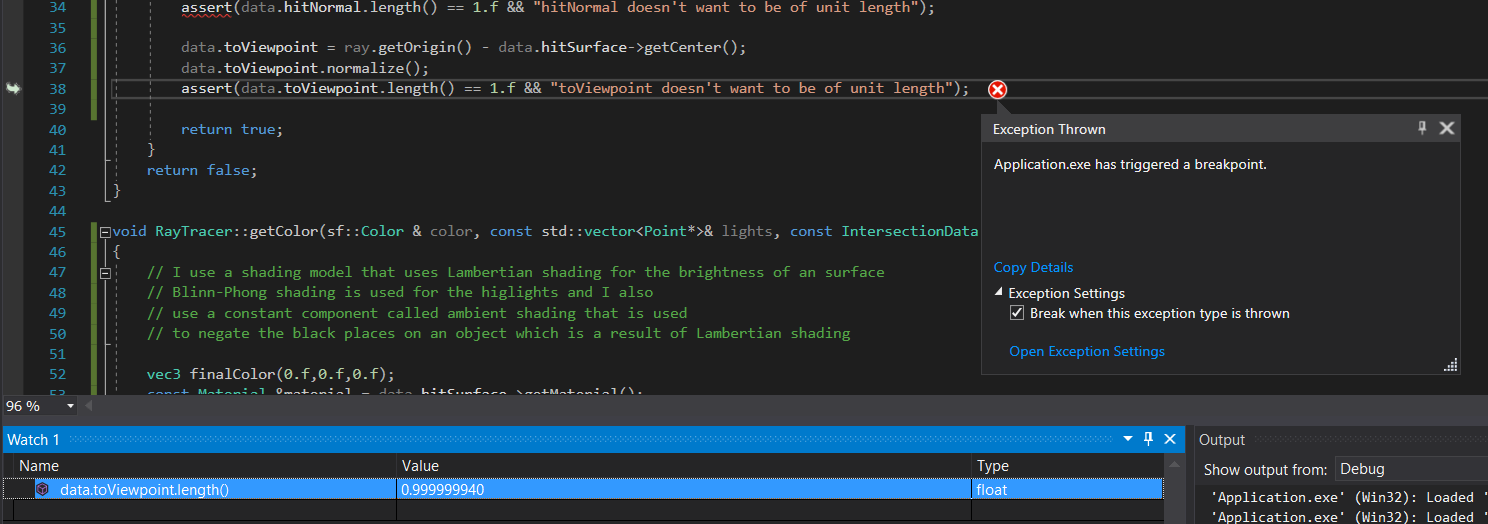
There are two things that confuse me:

* Why is my specular color black?
* Why is my Lambertian factor 0?

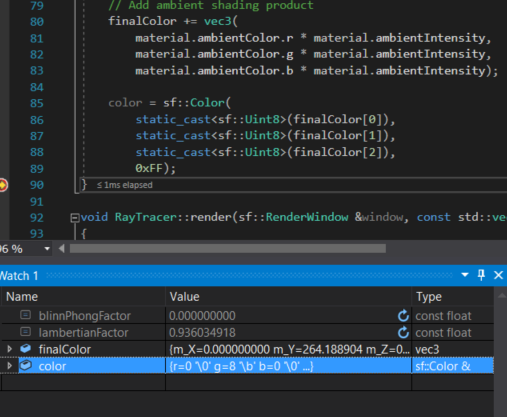
For the first issue, this was because of a misassumption in the way SFML handles colors. I thought that when you enter one color code, SFML uses it for every component, but you have to use a 32-bit unsigned integer. That is why my value was incorrect.

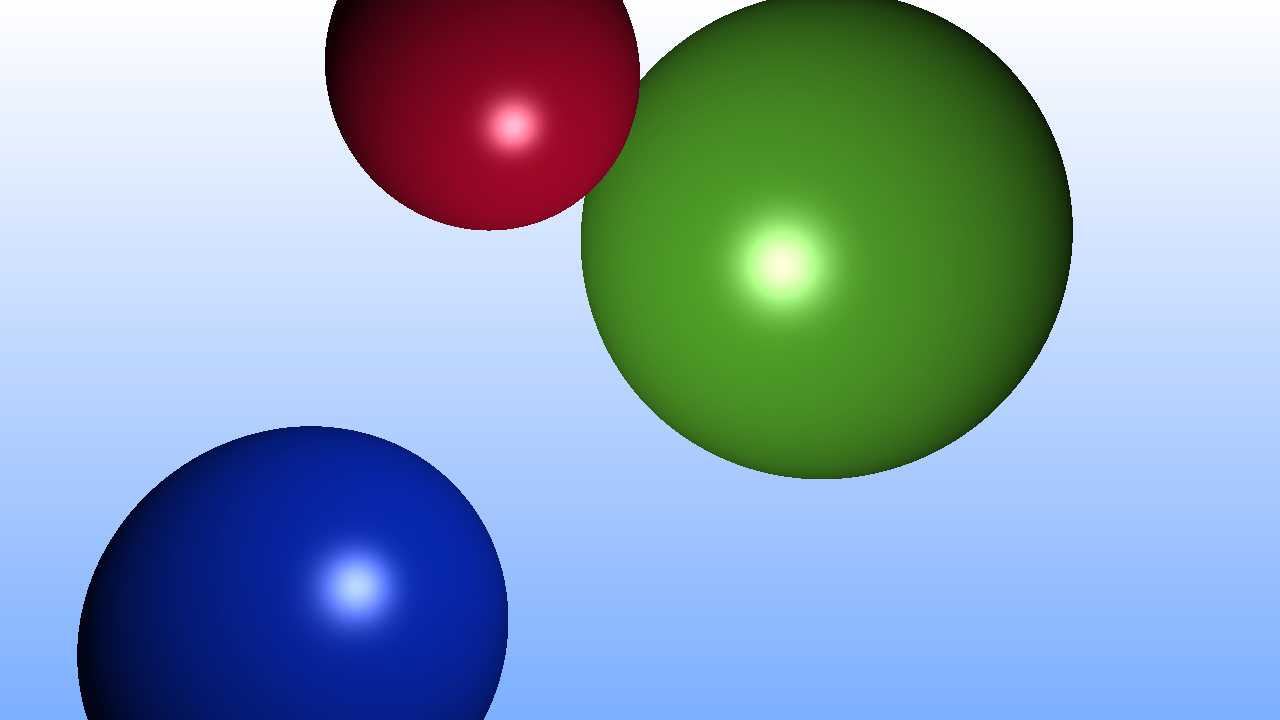
[](https://drive.google.com/open?id=1dZRdc2s7jBs2L7wgJpfCVlf93dOpZRt4)

Also, for some reason, my normalize function didn’t normalize the vector precisely.

[](https://drive.google.com/open?id=19oJPxTGQt_Hr1__SJA3-wj-CPm1EAW7-)

I deemed this as no serious issue because the precision issue is minimal. You won’t really see the difference. The important thing is that both vectors are of the same size.

I found the next problem, which was another misassumption about SFML.

This actually is a stupid error on my part. I thought that SFML would clip values that are too high to 255. Maybe it does, in my case, I cast the value to an 8-bit uint. This means that our value would wrap. Something that I forgot about. Because of that, the value started from zero again which resulted in the black spots. After having fixed all of the issues above, I got a nice raytracer capable of producing this image:

## 20:10-20:20 Update work log and plan the next tasks

In the end, no next tasks have been planned, but the work log has been completed.

# Wednesday

## 12:15-12:25 Fill out today’s work log

## 12:25-13:25 Create new pseudocode

[See the pseudo code here](https://docs.google.com/document/d/1xK5j2FXUPyqTJDMy2LKmxP2JGxQorz7h-sGrxG_5n6o/edit?usp=sharing)

## 13:25-14:10 Refactor and update class diagram

I am not sure how to represent the math library in the UML. This is because math library is used in almost every class and having an arrow from every class seems kind of counter-intuitive.

[See the class diagram here.](https://drive.google.com/open?id=1-pwIv7sfrdmrULFkMje11YWM__Om3nbJ)

## 14:25-14:35 Update work log

Updated tasks class diagram and pseudocode refactoring

## 14:35-15:30,16:00-17:15 Refactor project and update it to reflect new class diagram

I got a little bit caught up in updating the title bar of my window. I really like displaying data there. Although I have to admit that the information is quite useless because rendering in release takes about 1 second and you don’t need an estimation if it takes that long. Or maybe you do, either way, the application is so much nicer to interact with now. I have decoupled the ray tracer from the window. It minds its own business now.

I have one little problem though. I kinda forgot the syntax JSON uses. I will research that now because I will be needing that in my project.

## 21:20-21:30 Update work log

# Thursday

## 6:45-7:00 Fill out today’s work log

## 7:00-7:05 Update work log format

Added a function that sums up the time of a daily planning, This way. I don’t have to count it manually.

## 7:30-8:05 Research JSON format

I don’t get the difference between a JSON array and object.

* Arrays should be used to group objects.
* Objects variables can be referenced by variable name, arrays only by index
* Arrays seem to be contiguous in memory, objects not.

"config" :  
{  
 "renderLimitRate" : 60  
}

This doesn’t seem to work though.

I forgot to put everything in brackets

After making an example JSON and validating it I deem myself worthy of understanding JSON again.

## 9:25-9:45 Research about different JSON parsers

There are 2 libraries that I mainly look at. [RapidJSON](https://github.com/Tencent/rapidjson) and [JSON for modern c++](https://github.com/nlohmann/json) (I’ll call it jfmc). The hugest difference is that RapidJSON is faster (around 8 times as fast according to this [benchmark](https://rawgit.com/miloyip/nativejson-benchmark/master/sample/conformance.html).) but jfmc is easier to use. Jmfc tries to make a JSON object feel like a first data class, making it support many operations that python for example supplies for JSON files.

I feel like jfmc is the better solution here. My JSON files aren’t that performance critical and I’d rather use a library with a nice interface. That should also help with development speed. Jfmc claims that if you know std::vector, you already know enough to use this library. Both libraries are header-file only which makes it very easy to implement.

## 9:45-10:00 Help peers with maths

## 10:00-12:00 Study usage of lib JSON for modern c++

I can’t loop over the JSON array like I wanted to.

## 13:00-13:35 Meeting with Bojan

## 14:00-15:00, 15:45-16:10 Add JSON parser to project