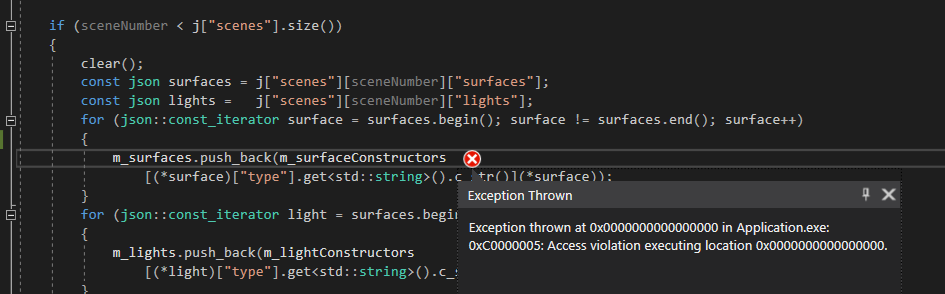
# Monday

## 6:50-7:30 Implement data loaded scene



I’m not too sure what this error means or why it happens.

In the end, the error happened because I used const char pointers. If we index that into the map and the key is not found, the program crashes. The index operator probably has no OOB checks and assumes the user is always correct. Changing the type of the key to a string solved the problem. **Apparently, you can’t compare char pointers because you would just be comparing if the pointers are the same, not the value.**

For some reason, the ray tracer detects no collisions anymore.

## 7:50-8:15 Research about shadows

Does the color of a surface dimm when multiple shadows overlap over it? I don't think so because light only adds up. Shadows are a form of light but only darker?



But here it seems that multiple shadows increase the darker color.

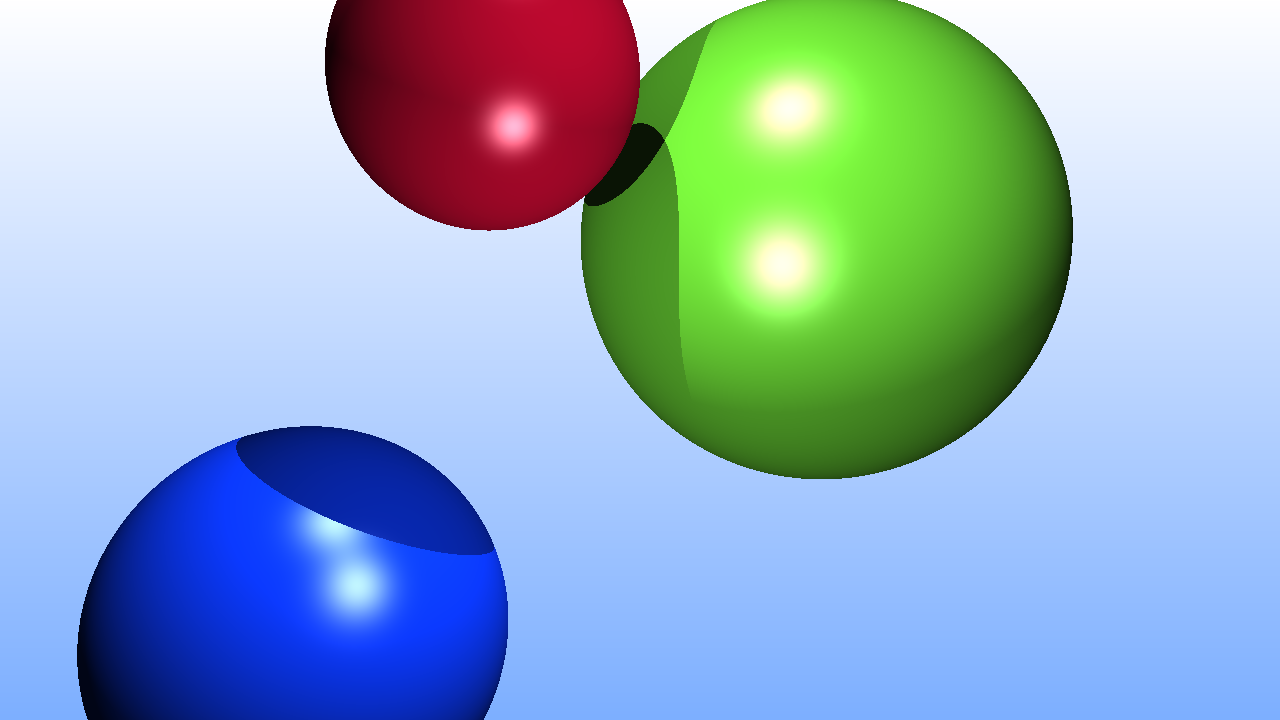
The process of getting a shadow in the book is simple. When the view ray is cast and an object has been hit, add that object's ambient color to the pixel color. Then you have to cast a shadow ray to the light source. If that ray doesn't intersect, add other shading models to final pixel color.

I actually always thought that light dims the longer it travels. Whenever you turn on the light, it gets darker the further you get. But this actually happens because light has to spread over a larger surface the further it goes. I can't imagine how bright the sun must be because I can't look into it at all.

## 9:15-9:45 Fill out today’s work log

* Added groups to the work log so that you can collapse every week

## 9:45-10:15 Implement shadows

[](https://drive.google.com/open?id=1dQ2EQQFPpSHdXv0YFIUTmJm1jOW0smBu)

My shadows seem a bit harsh. I’m not sure if they are correct, but they are implemented.

## 10:15-11:45 Help Mathijs

## 12:10-13:00 Fill out previous week’s sprint

## 13:00-13:15 Fill out this week’s sprint

## 13:30-14:20 Add perforce support to visual code

I am going to try and use visual code to continue my code editing. This is because I prefer this editor. It also drains less battery so I should be able to program more in the train.

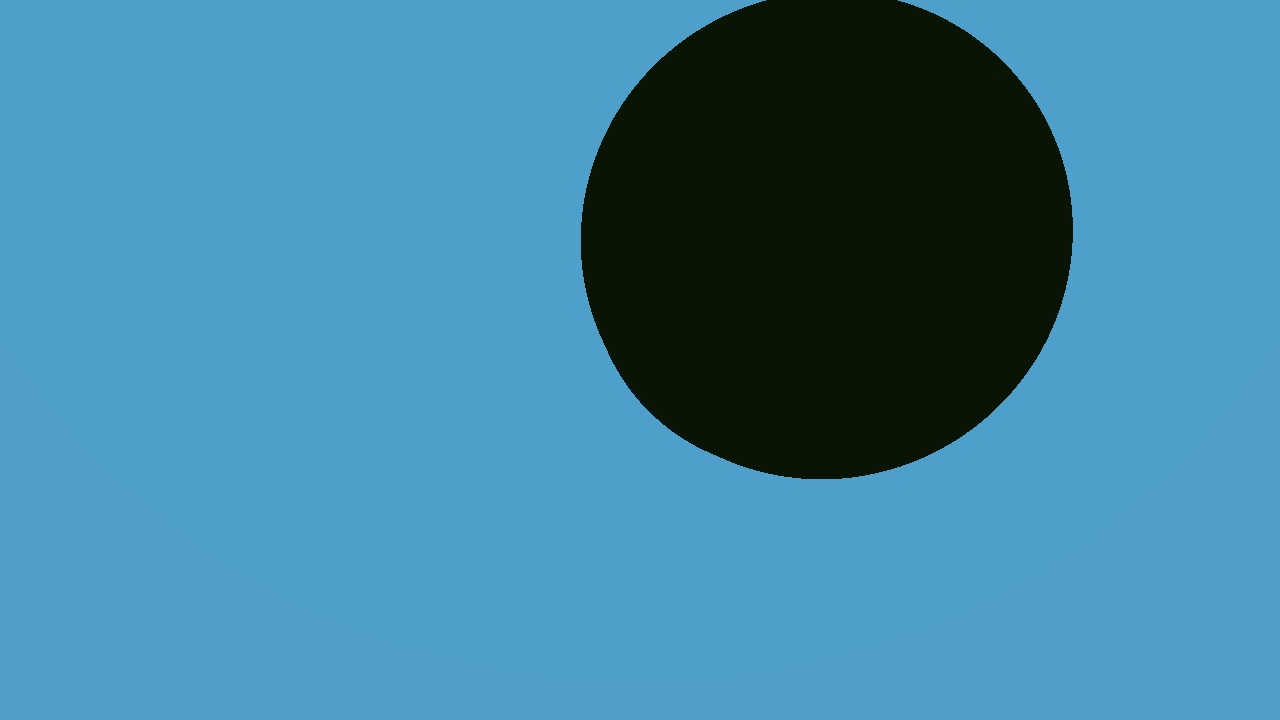
## 14:20-15:00 Research about specular reflection

I find this quite an odd paragraph name because of the word specular meaning ‘highlight’, ‘shininess’ or ‘reflection’. So this paragraph is called ‘reflection’ reflection.

On a second note, this is incorrect. Because specular means mirror-like.

* Basically, keep calculating intersection until max bounces are bounced.

I should actually have planes in the project but I will use huge spheres instead

Never mind, I can’t do it. All these weird things started happening. For example

Let's just add planes. It’s one of my goals so it had to be done either way.

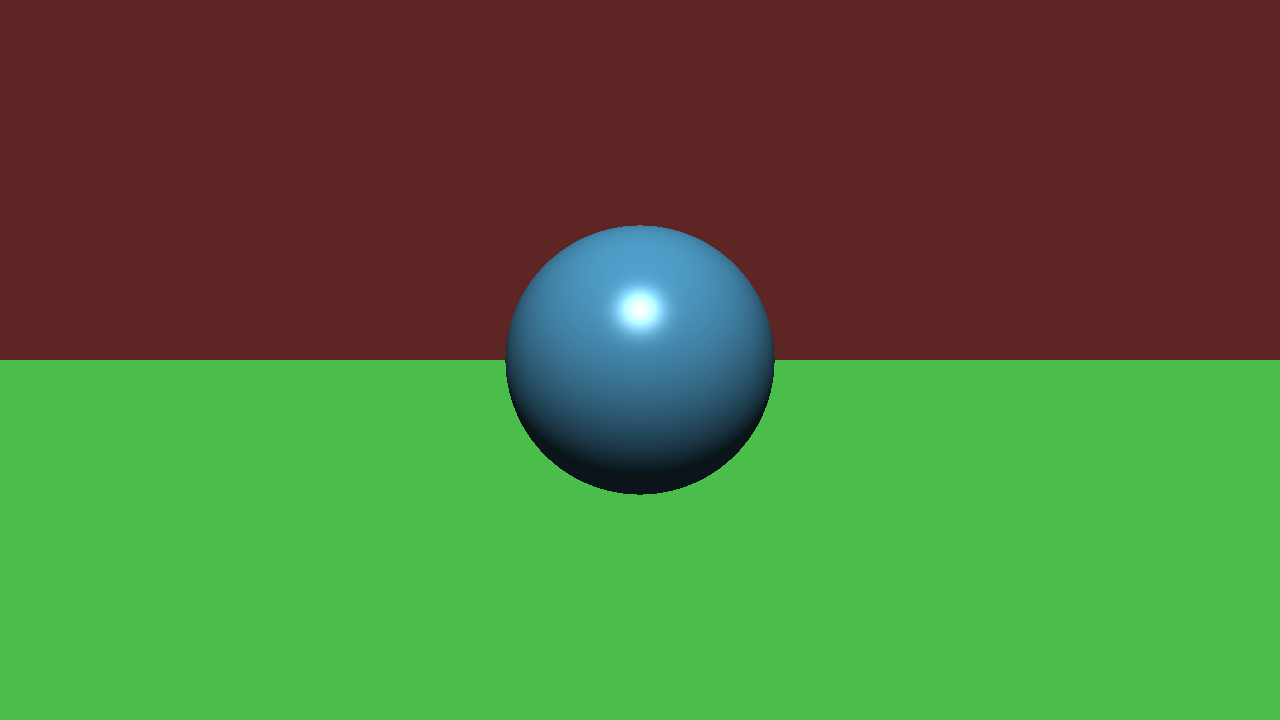
## 15:30-16:00 Research about a plane intersection

Every point of an plane can be calculated by dotting the normal with a vector from a point - center of the plane. If the result is zero, we know it’s on the plane, because it’s perpendicular with it. To check if it's in the size, we just have a width and height value. We get the point position minus the center position and we get a new offset, We can only check this if the plane is parallel to two axes though.

Or not, we can rotate the point around if needed.

## 16:20-17:00,18:20-19:20 Implement axis-aligned plane intersection

The plane is only detected from one side. This is because the normal only points in one direction. I wanted to swap this during the intersection test by flipping the multiply value if the normal were negative, but this results into problems when I attempt lightning. With lightning, the normal of the plane is on the other side so the plane is always dark.



There is a problem. To make the rays finite, I need to rotate the plane. I don’t know how to do that. I could also move the camera and make the plane a axis aligned plane. I don’t know how to move the camera.

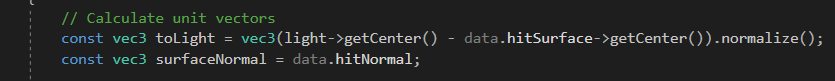
I must implement the camera or at least implement rotation. Thus that will be my next subject to research.

# Tuesday

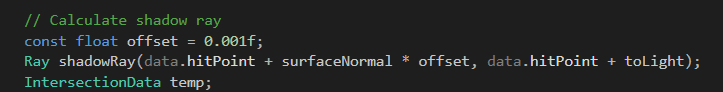
## 10:35-10:45 Fill out today’s work log

## 10:45-12:35 Finish up line intersection

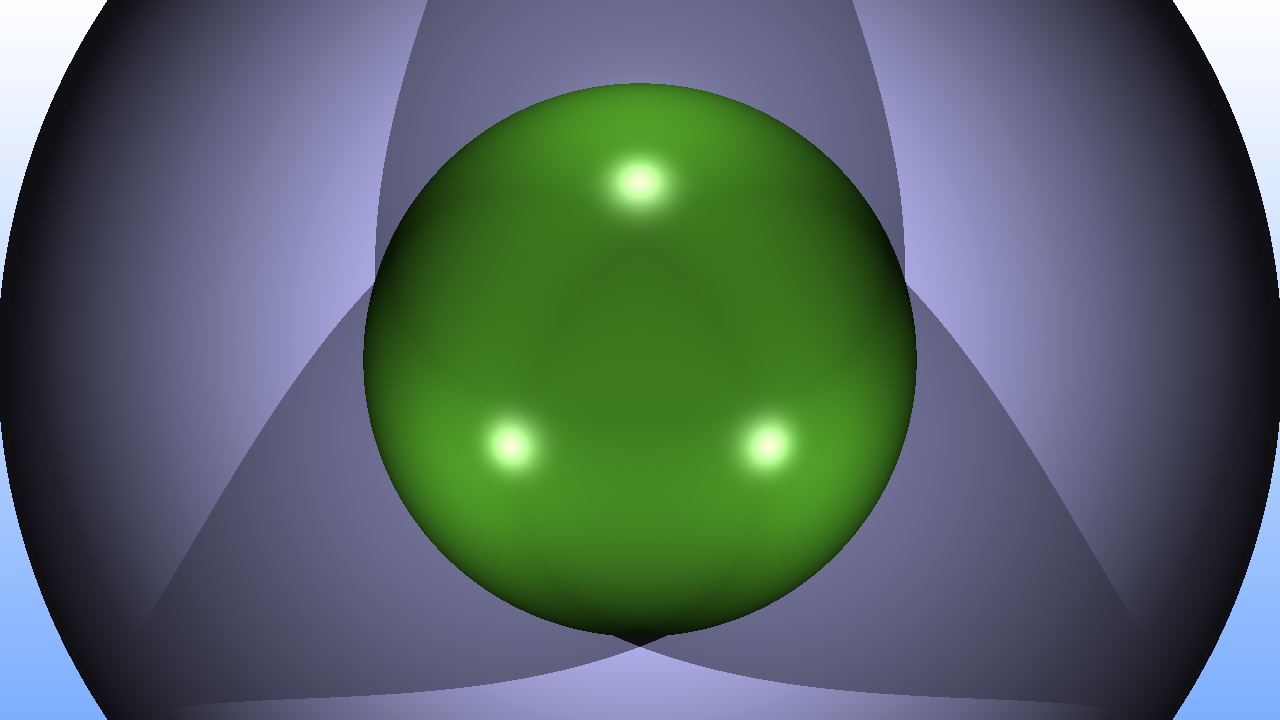
When generating the shadow ray for the plane, for some reason the shadow ray has no direction. Because of that normalizing, it crashes the program.

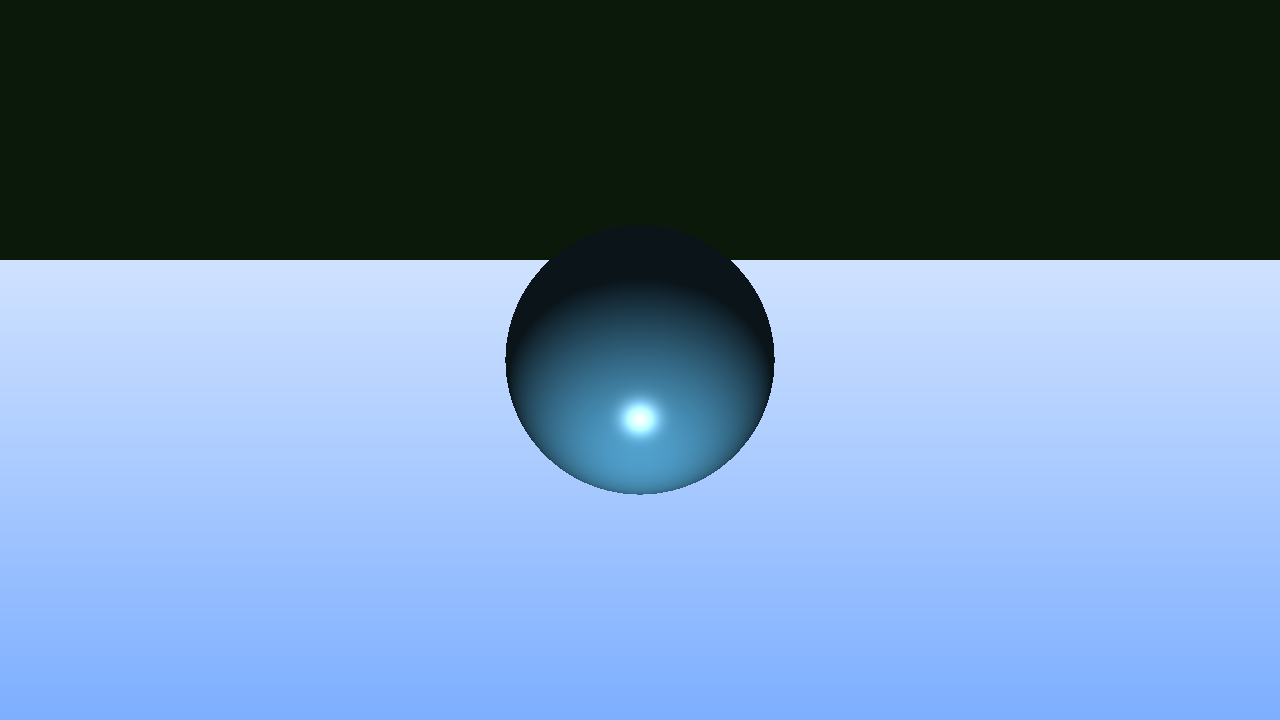
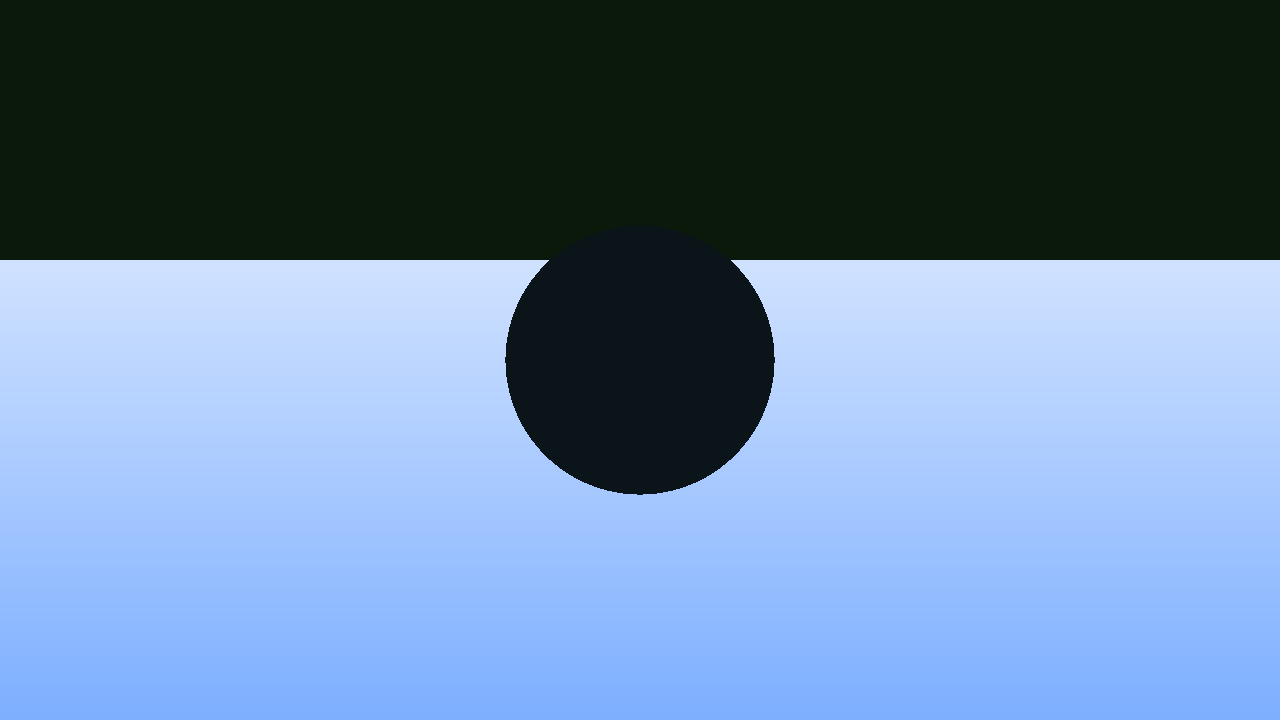


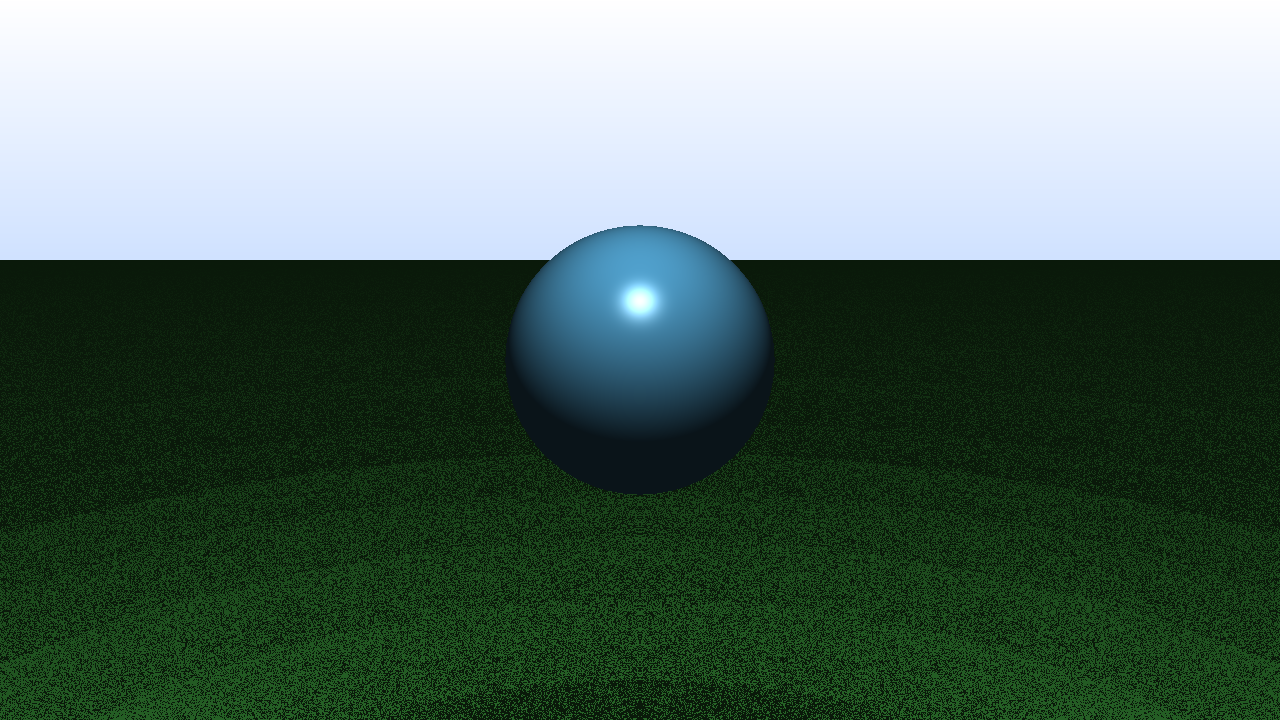
Found it, this here is incorrect. It should be the hitpoint but we take the center. That works for spheres, but not everything is a sphere.



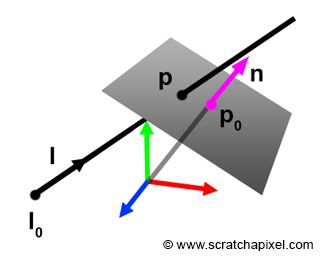
Here is another mistake. The parameters for the ray are begin and end point. Right now we pass in a direction instead of an end point.

This fixed the shadow ray generation. To test it out, see here a before and after.

For some reason, my plane is being rendered upside down.

It’s quite dark because the light source is being blocked. Moving the light source down lits up the sphere but not the plane.

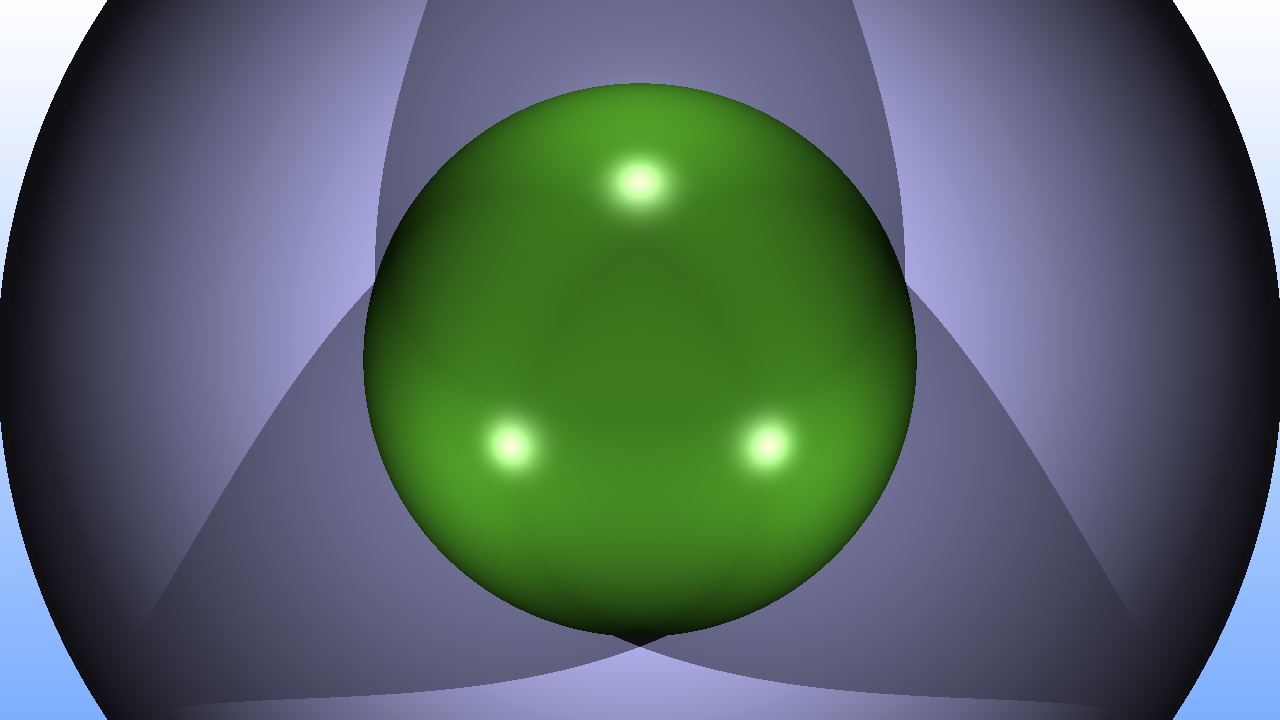
This problem got fixed by flipping around the nominator. Apparently, my formula was incorrect. You have to subtract the **surface origin from the ray origin**. I’m not sure why though because it should result in the same vector.



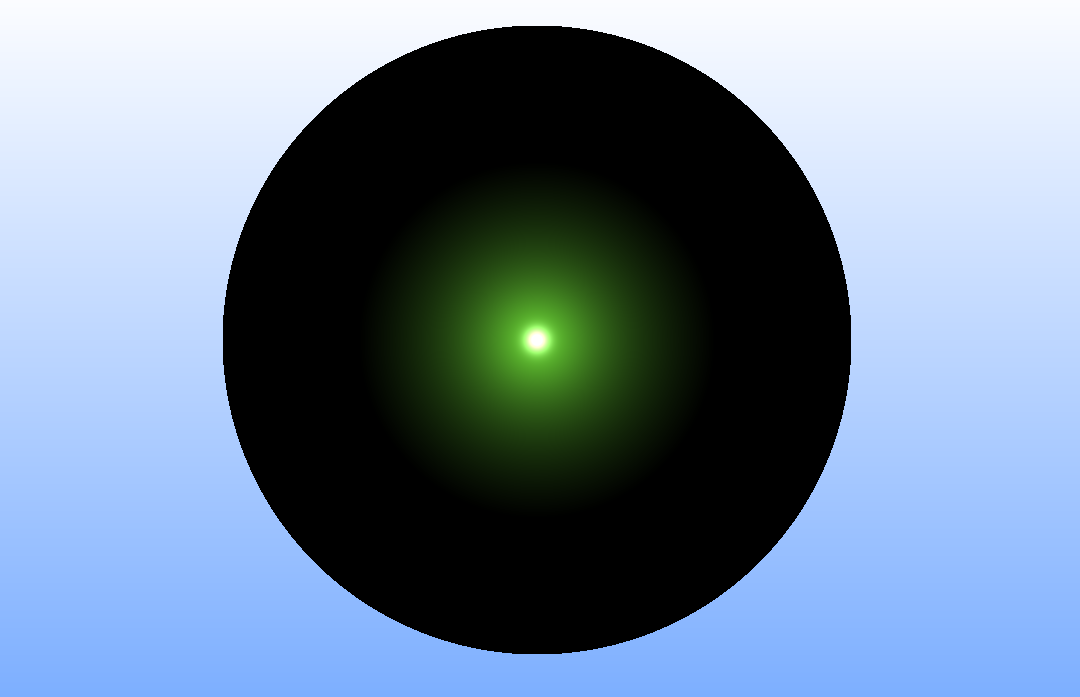
I used a vector from P to Po but should be from Po to P, why is that? I’m not sure myself.

[Here are the notes that I made during this task.](https://drive.google.com/open?id=1B2GOeBCiKNS2V-a94fZ64HkGE0M0I8eH) Most of them are drawings to make sure that what I’m doing was correct.

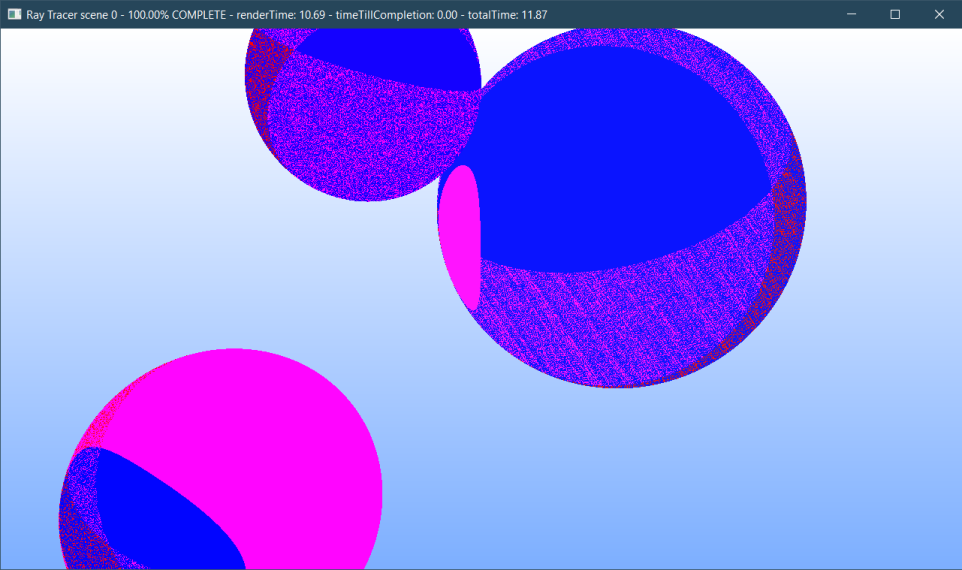
## 12:35-13:00,16:45-18:10 Fix shadows



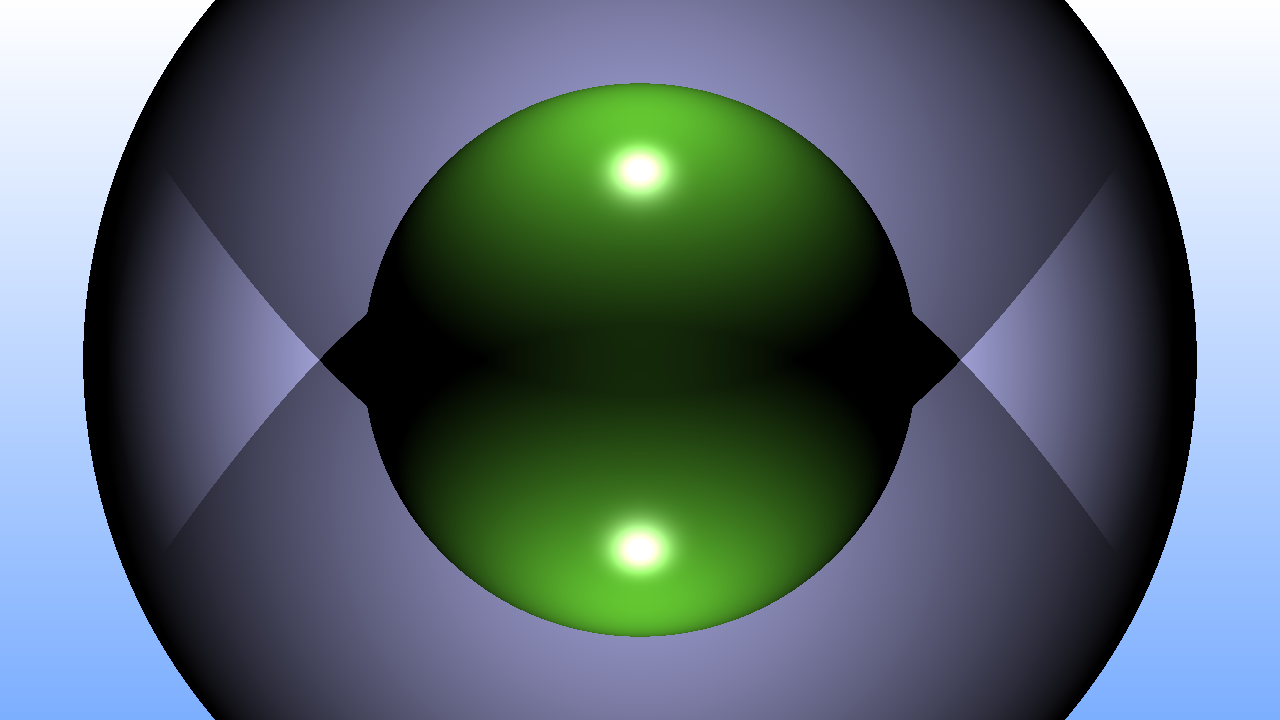
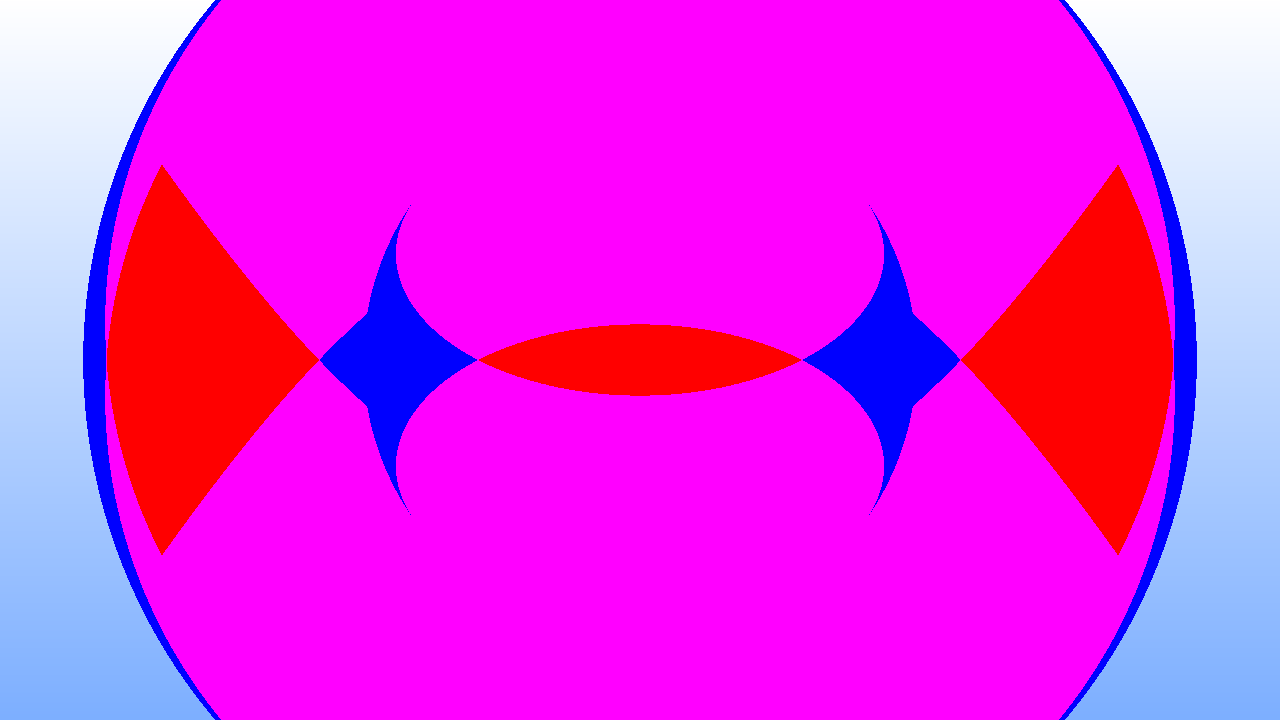
I’m not too sure if this is incorrect, but I think that the shadow on the right should be lit up. I don’t understand why the shading is dark.



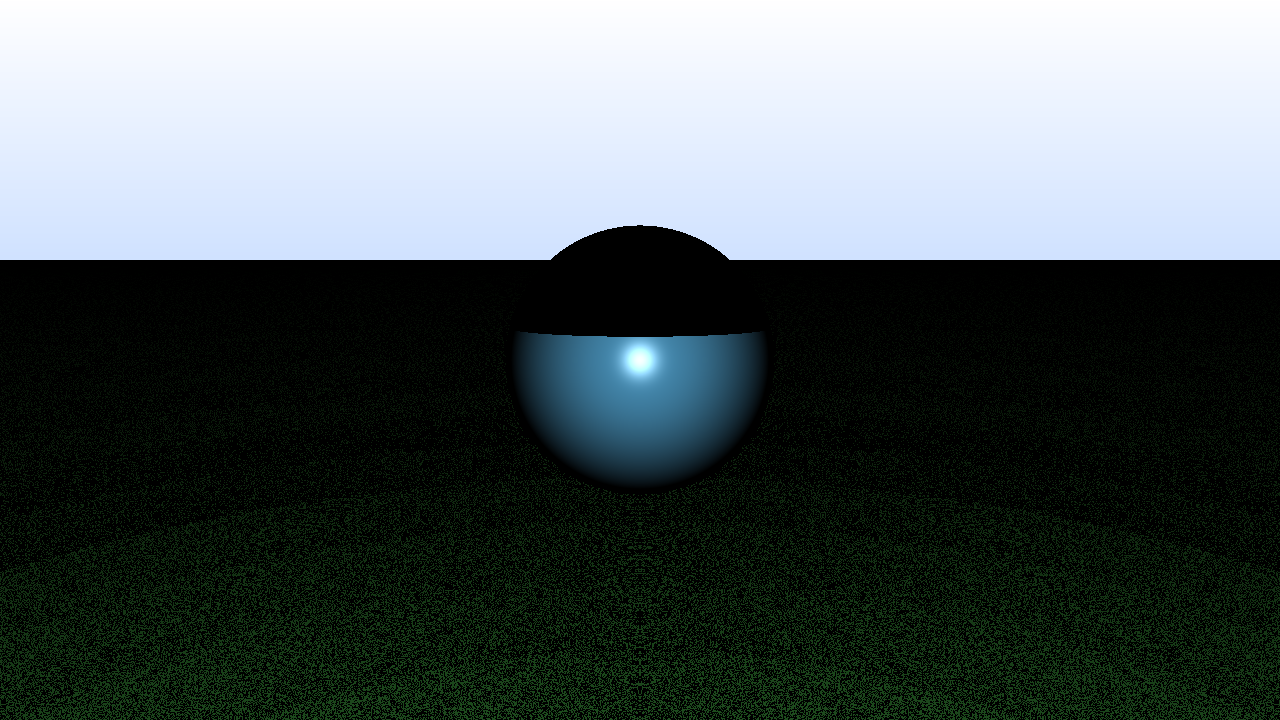
I have found quite a useful [site](https://www.geogebra.org/3d?lang=en) to debug a scene. I’m having a tough time translating scenes into 3D in my head and I’m a bit lazy to draw a whole 3D scene. So I’m going to use this as a reference.



Apparently, my shadow offset was a bit too low. Blue parts receive light. Red parts are in shadow. Purple parts are a combination (3 light sources). My offset was 1e-6, now it’s 1e-3.

I don’t think that there is really something wrong with my shadows right now.

Looks right to me. So let’s move on to planes cus that’s where the problem lies.



The sphere is incorrect, mapping this into GeoGebra results into this.

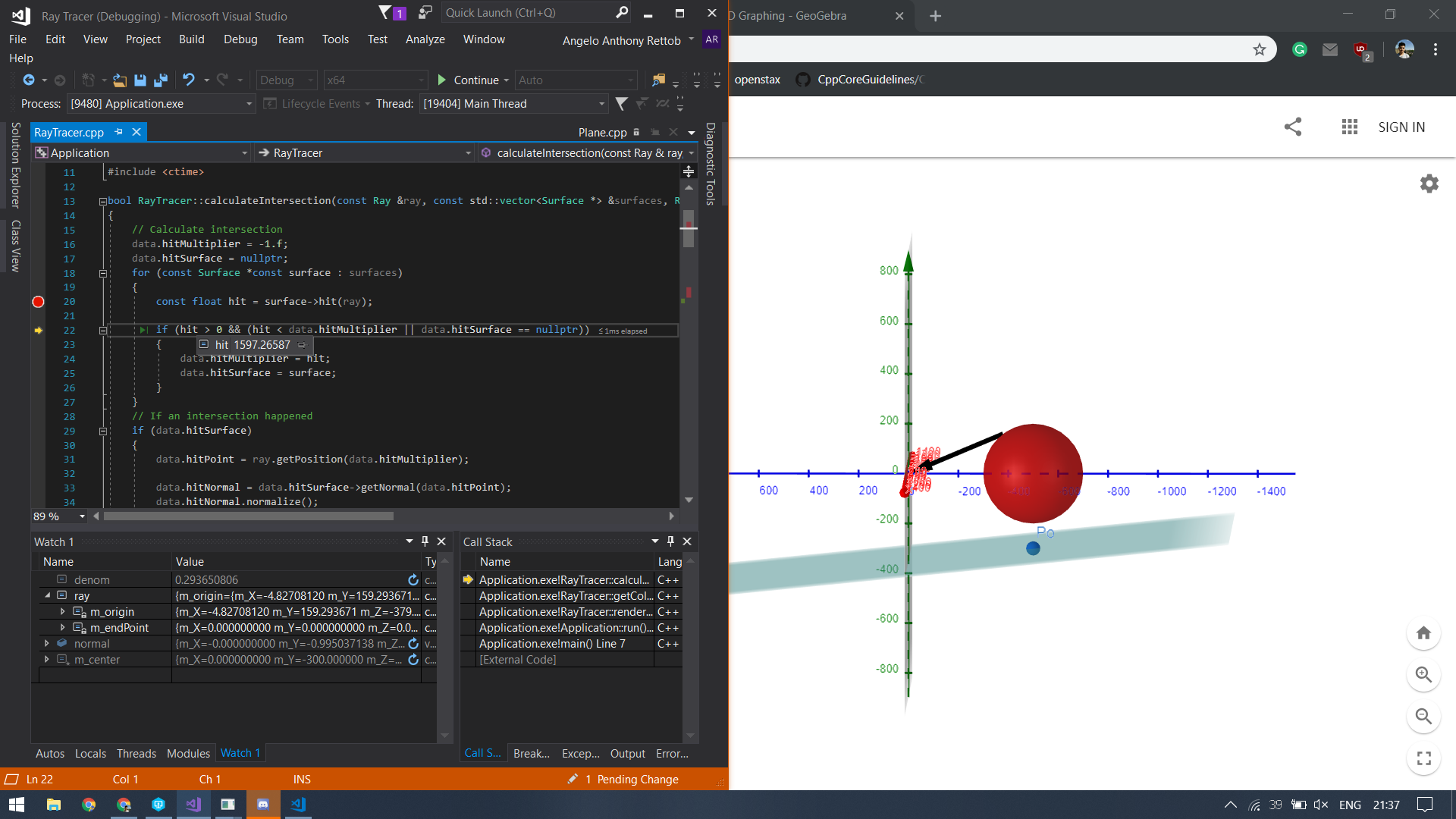
Long story short, I hate GeoGebra, I hate it so much that I decided to use blender instead.

## 18:10-19:00 Learn basics of Blender

Blender uses a different coordinate system than me, that should be fun.

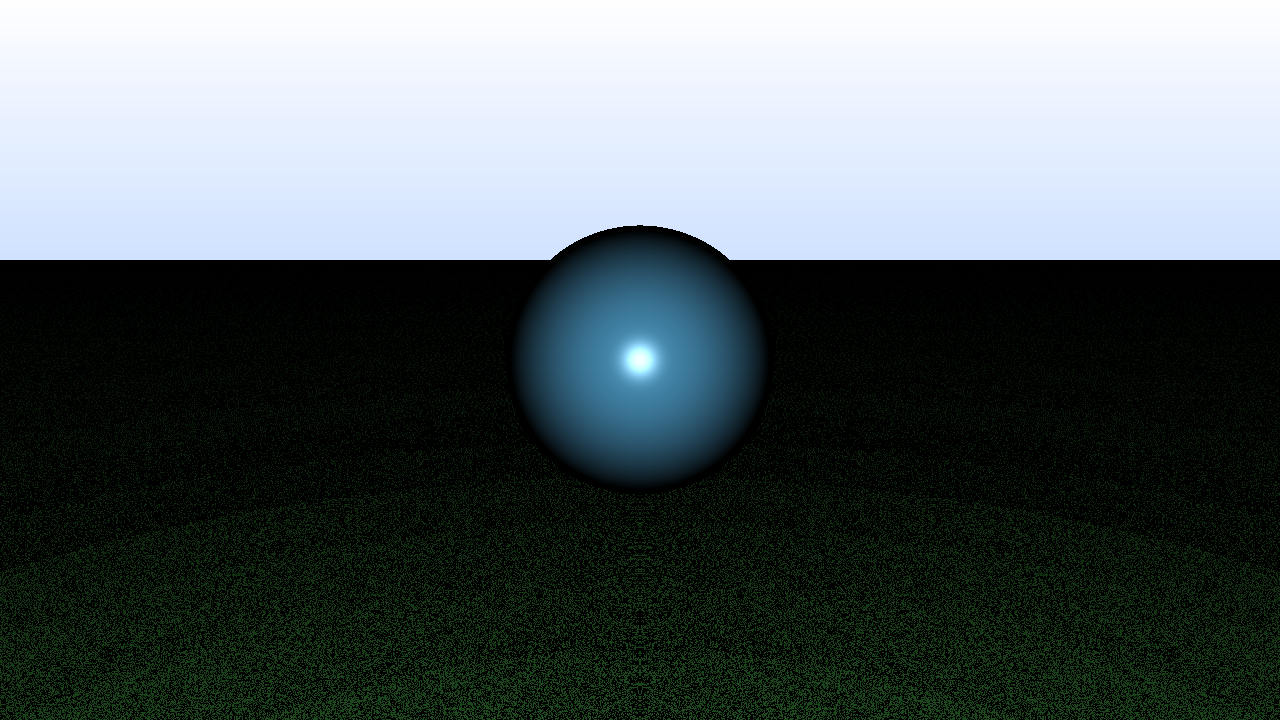
Blender is not the solution. It’s way too advanced. I’m just looking for some simple 3D plotter. I think that I’ll just have to get used to GeoGebra.

## 21:20-22:00 Debug weird sphere shadow when a plane is in the scene

[](https://drive.google.com/open?id=1QF9Dx9RwcrzLG7jKxLQp720oFKgk0pB-)

Sorry for the picture, click on it to expand. Because of the plane being infinite, the vector that is supposed to go to the light continues to towards the plane. While the lines are not parallel, the plane will be hit. Because of this, the hit value is very very large but it’s actually behind the light.

This means that if we were inside of a sphere, a shadow would always have been cast. To fix this, we need to check whether the potential hit is past or before the point of interest (the light in this case). We can do that with a simple dot product.

[](https://drive.google.com/open?id=1OQnRKHk-qvUVVWQo1FDTdHGdJMhWdSub)

I finally solved a problem and not made a new one. Unfortunately, this didn’t fix the plane. So that will be my next task.

# Wednesday

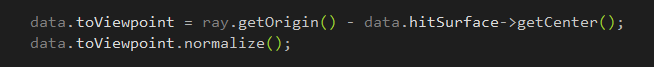
## 8:40-8:50 Fill out today’s work log

## 8:50-10:25,11:10-11:50 Solve shading error with infinite planes

When there is only a plane in the scene, the plane shades accordingly. Though I’m a bit confused about one thing. That is the shading. It doesn’t matter where I move my light, the shading stays the same. I’ll first have to solve that.

[****](https://drive.google.com/open?id=1Gp7nqMXUPDEtnBzxMxqdyI0GIqRM9w-B)

I did actually found an error in the calculation of the toViewpoint vector.

[****](https://drive.google.com/open?id=1mOdEKcxmbGrxbSWz6adbAAqTllCXKG9o)

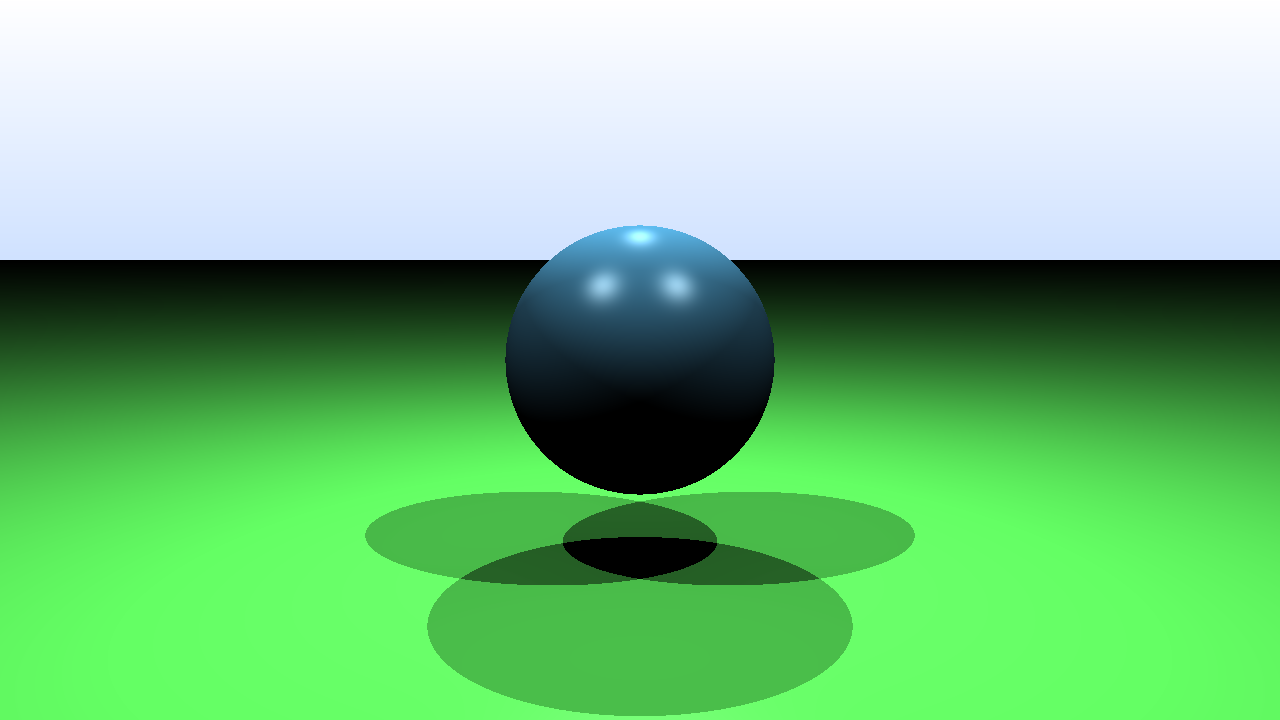
It should be viewpoint - hitpoint, not viewpoint - surface center. This should make Lambertian more accurate.

After having researched the plane lightning, it actually seems to be alright. Debugging the shading showed that the hitpoint had a z coordinate of -3000000. That is huge. Because of that, it seemed like the light source did not move, but my very small slope (1 pixel up for every pixel going further) resulted in the large gradient. I have yet to solve the shading when another object is in the scene though.

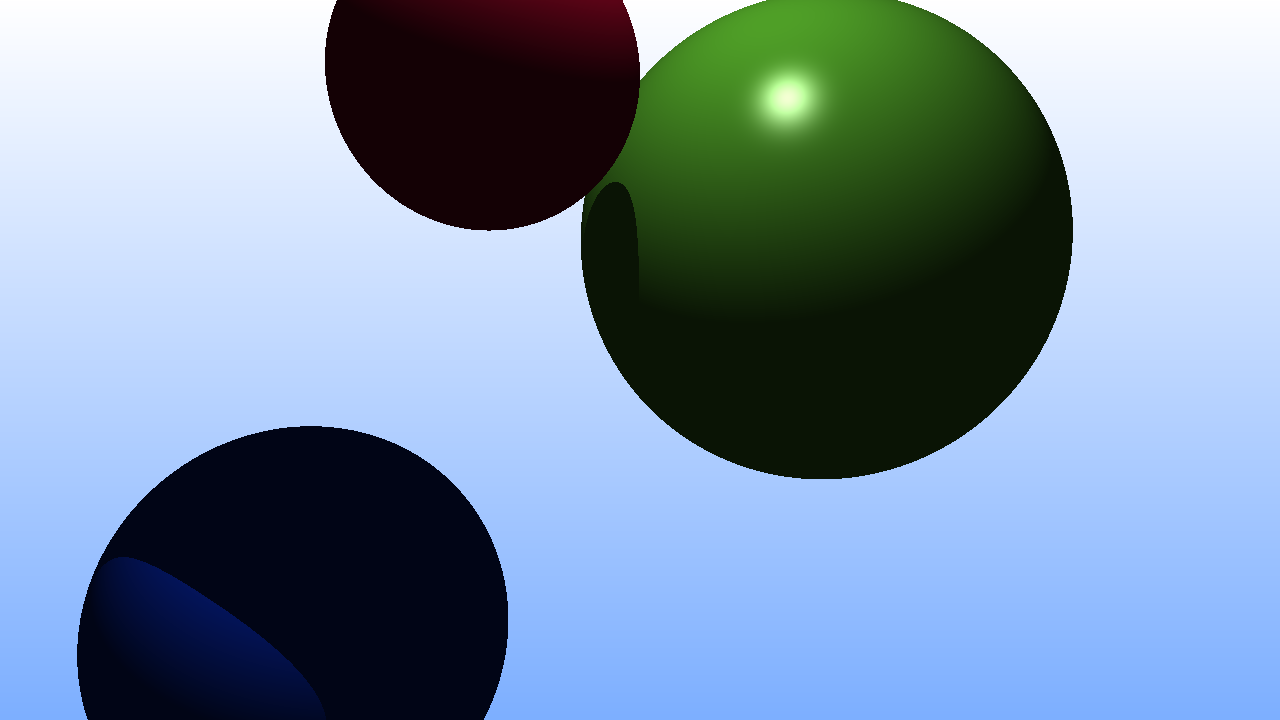
****

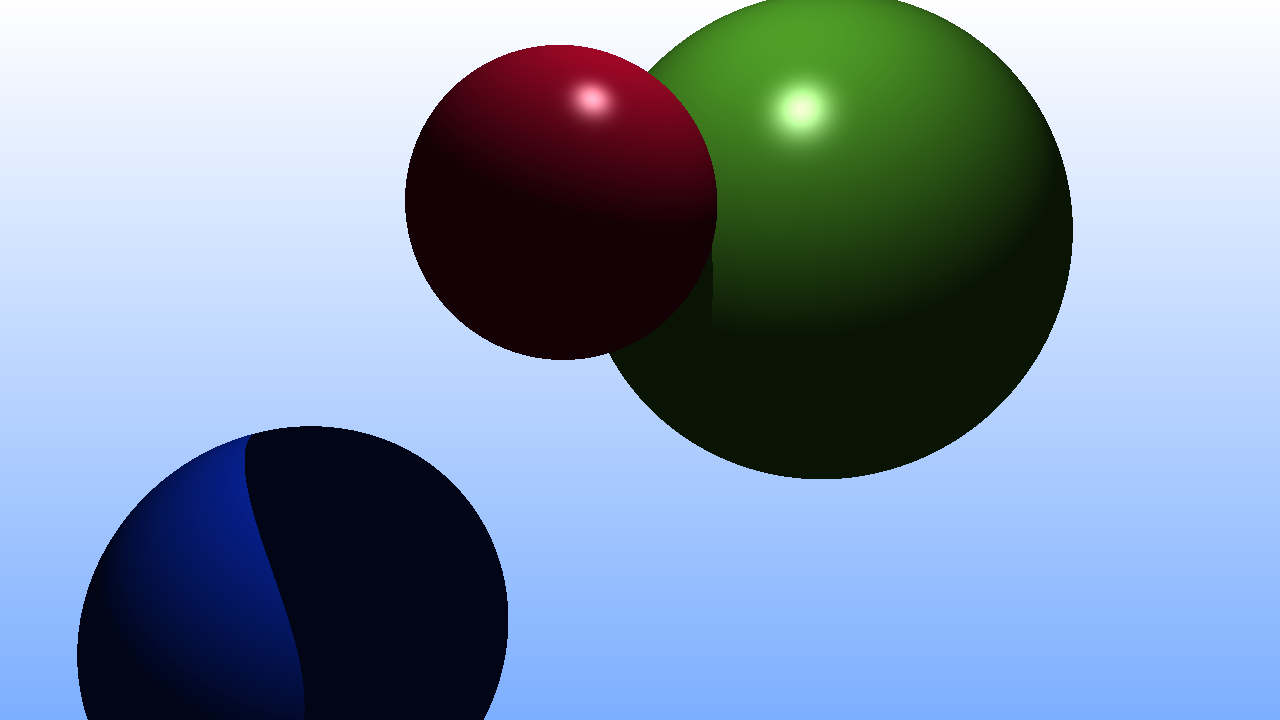
I found out why the shading was weird and it’s a really obvious error. If we call ray.getPosition, the direction that is used is not normalized. When we calculate the point of intersection for the plane, we use a normalized view direction. Because of this, when we calculate the point, it actually gets past the plane. I’m quite surprised that the sphere even works! Here is a [PDF](https://drive.google.com/open?id=19fKt_5Z1K4bN9-pnozGEXkRnbKtfG5rg) for clarification (I accidentally made it a PDF)

After having changed the ray internally and made some modifications inside of the sphere intersection, the problem has been solved

[](https://drive.google.com/open?id=1IviS-3KiCXFYygItw7oQUqUj5NWrImOZ)

## 12:20-12:40 Research if sphere shading is correct

[](https://drive.google.com/open?id=1UHESAEisv_hBLuAAUCsSQrV0bqhHgnMP)

Light comes from above but the bottom is shaded. It’s probably an illusion. This scene looks alright.

## 13:00-13:35 Big O notation

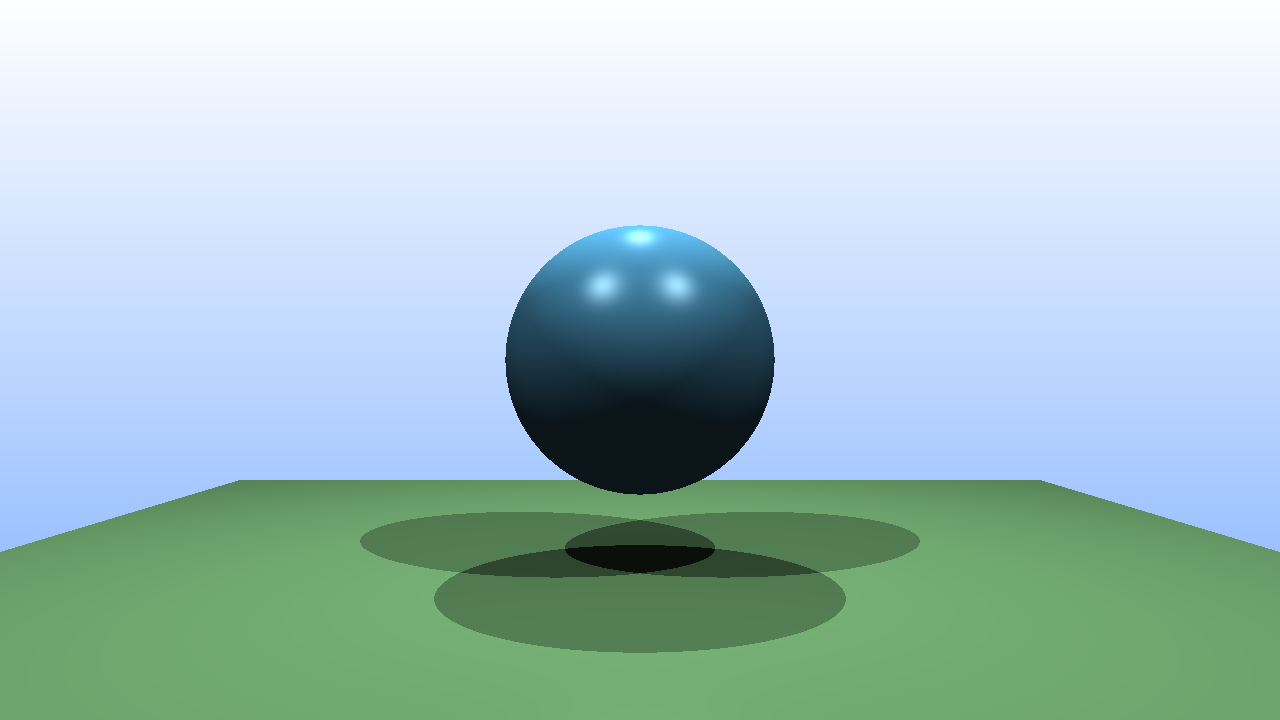
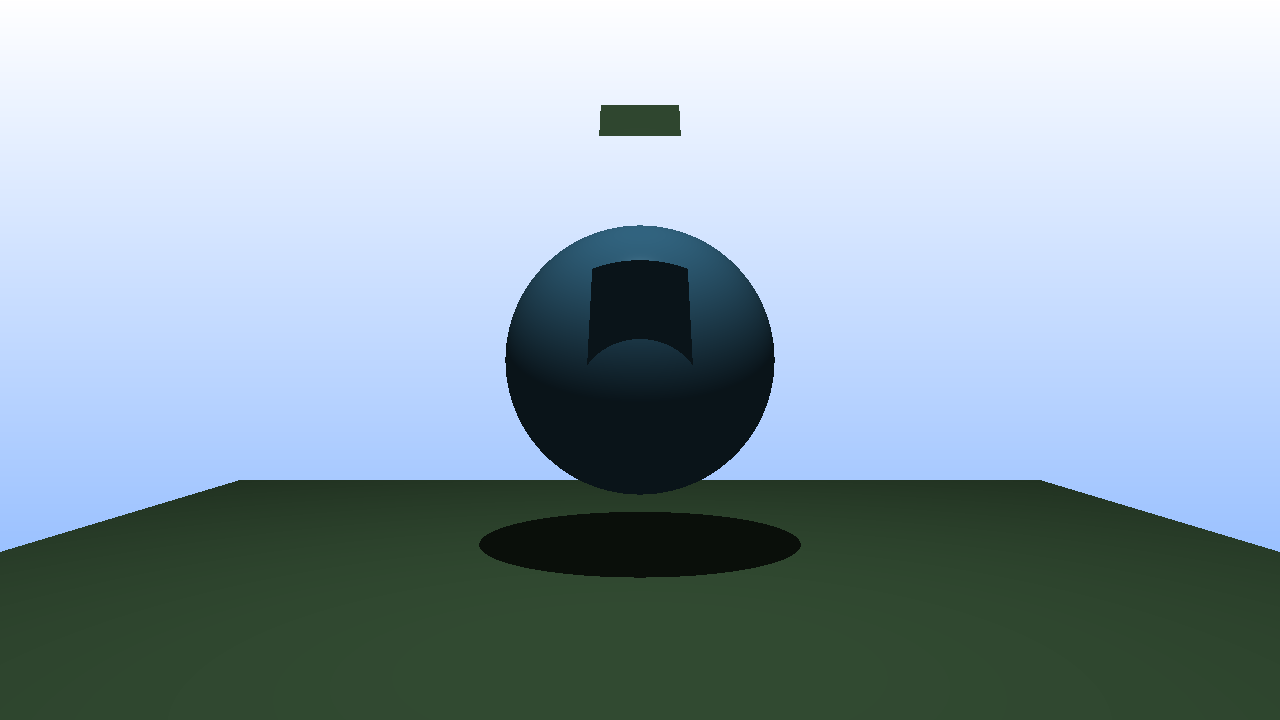
A way to measure the runtime of an algorithm independent of variables like hardware.

~f(N) means that something divided by f(N) approached 1. I don’t understand why this is used.

The workshop is more examples for certain orders of growth. I understand the concept. I don’t feel like I need to know more about this at the moment. Implementing reflections and finite planes feel more important.

## 13:35-14:00 Implement finite planes

The concept is simple, all planes will have a width, height, and depth. These values represent the max distance a plane can take in the axis. Because of this, when planes are rotated, these values are not really true. The planes are still finite though. I’m not too sure about the shadows but I don’t think that they’re that bad. I’ll look into them when I’ve finished the moving camera.

[](https://drive.google.com/open?id=1CN-rurCcOhjka3Hhb5hNhstte0HJ2khY)[](https://drive.google.com/open?id=1drXfzLH76tVG6WC5ltMX9XBZp94Gw6L3)

## 14:50-16:20 Think about the design of the ray tracer

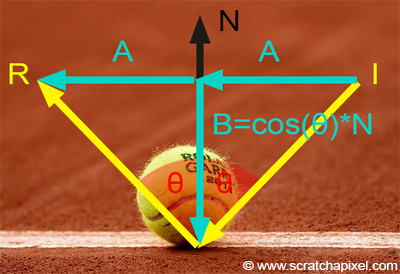
also, I feel like my design is a bit incorrect. Right now, the specular and ambient color is decided upon by the surface itself. This is not realistic. I’m going to refactor my project a bit and change certain responsibilities to other parts. A surface is only responsible for its surface color and Phong exponent. The scene is responsible for the ambient intensity and the light sources are responsible for the specular component.

This also means that I have to change a bit of the JSON. I will take this opportunity to change the format a bit.

## 16:20-16:30 Convert other scenes to the new format

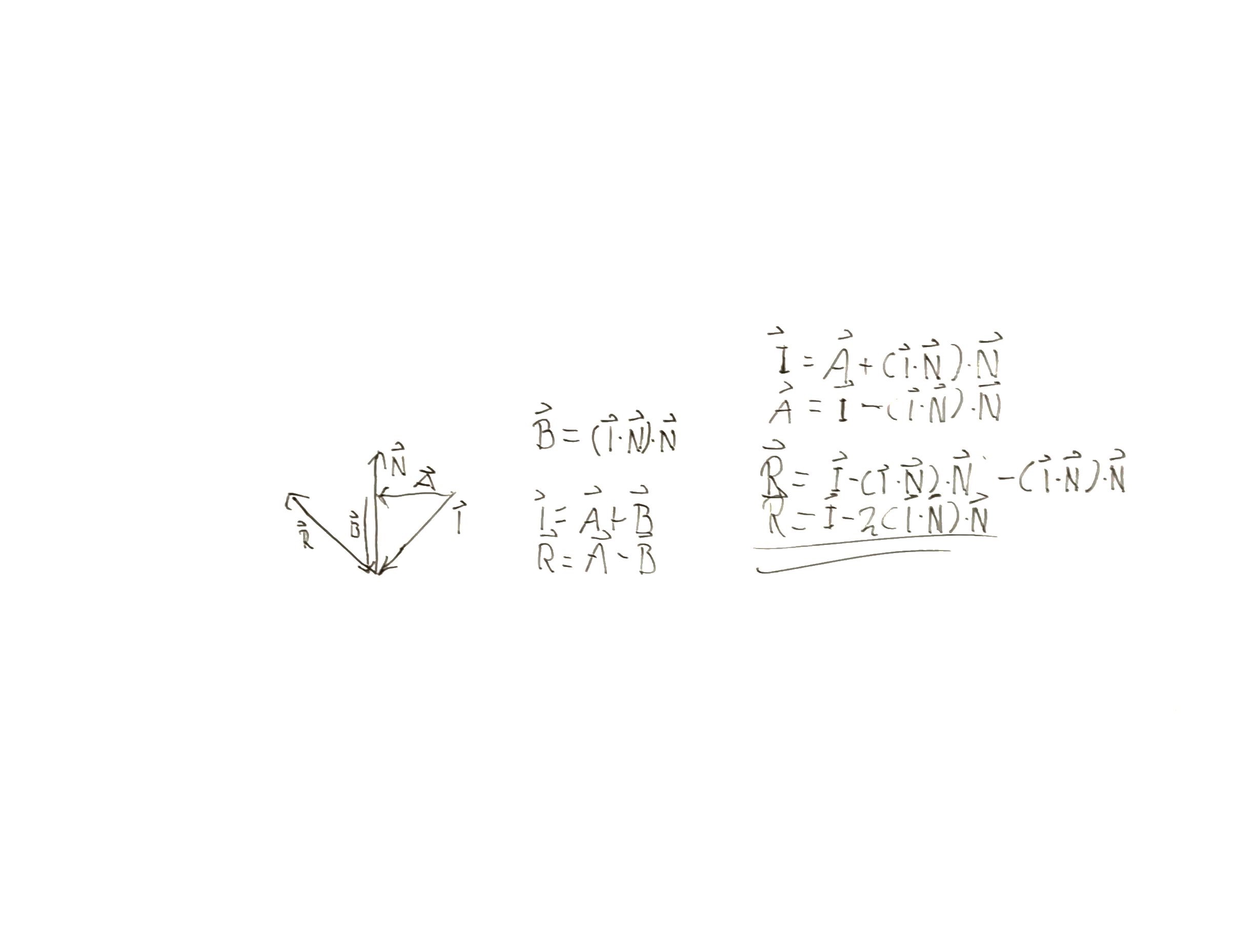
## 18:40-19:50 Research about reflection and refraction

* Water or glass are everyday things that have reflection and refraction. **Reflection**, meaning to reflect light and **refraction**, a phenomenon called transmission. Both things have both properties, to know how much of the light is reflected and how much is retraced, we use the **Fresnel** effect.



This image makes no sense to me. To calculate B, you should do: dot(I,N)\*N.

If I and N are of unit length, the result is the same as cos(o).

[](https://drive.google.com/open?id=1Quvwrd-0jPMJ_POzS5pXrs2P6_zT_InX)

## 19:50-20:30 Implement reflection

# Thursday

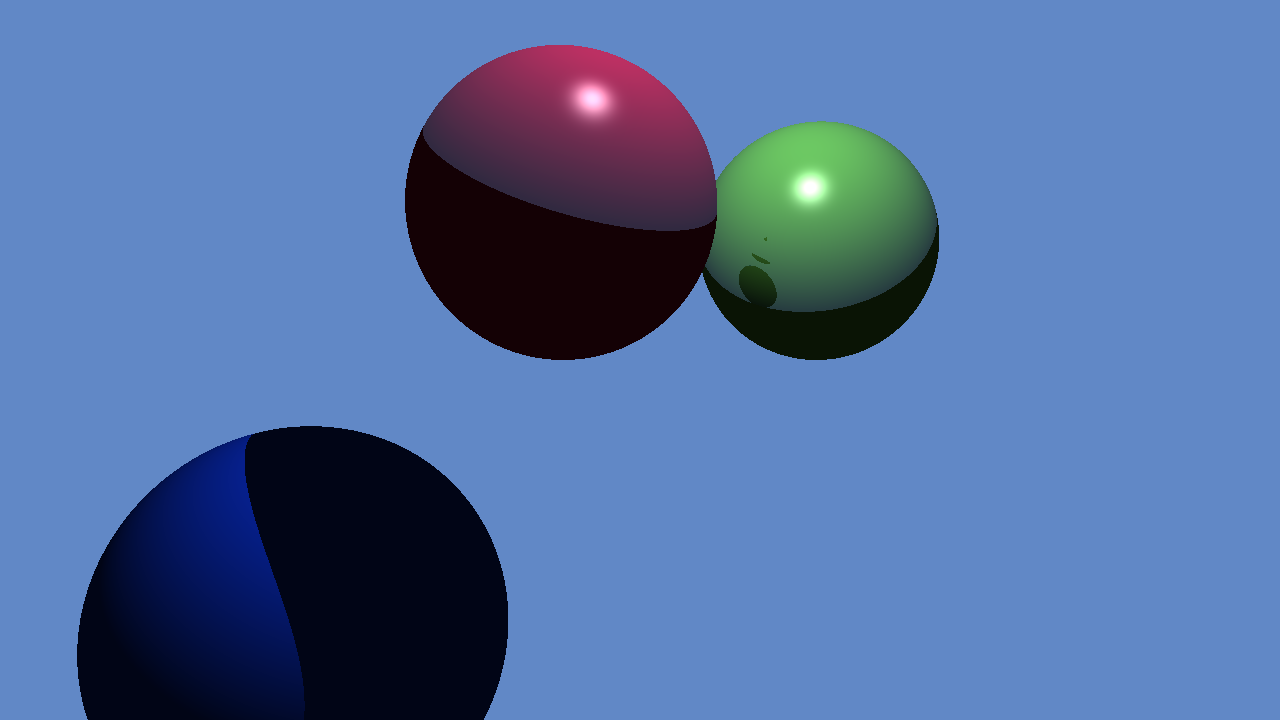
## 7:25-7:35 Fill out today’s work log

## 7:35-8:15,10:00-14:00 Fix reflection color

To add support for this, I want to make the ray tracer be able to trace when inside of an object. This is because I’ll be putting a texture on it. A sky box texture.

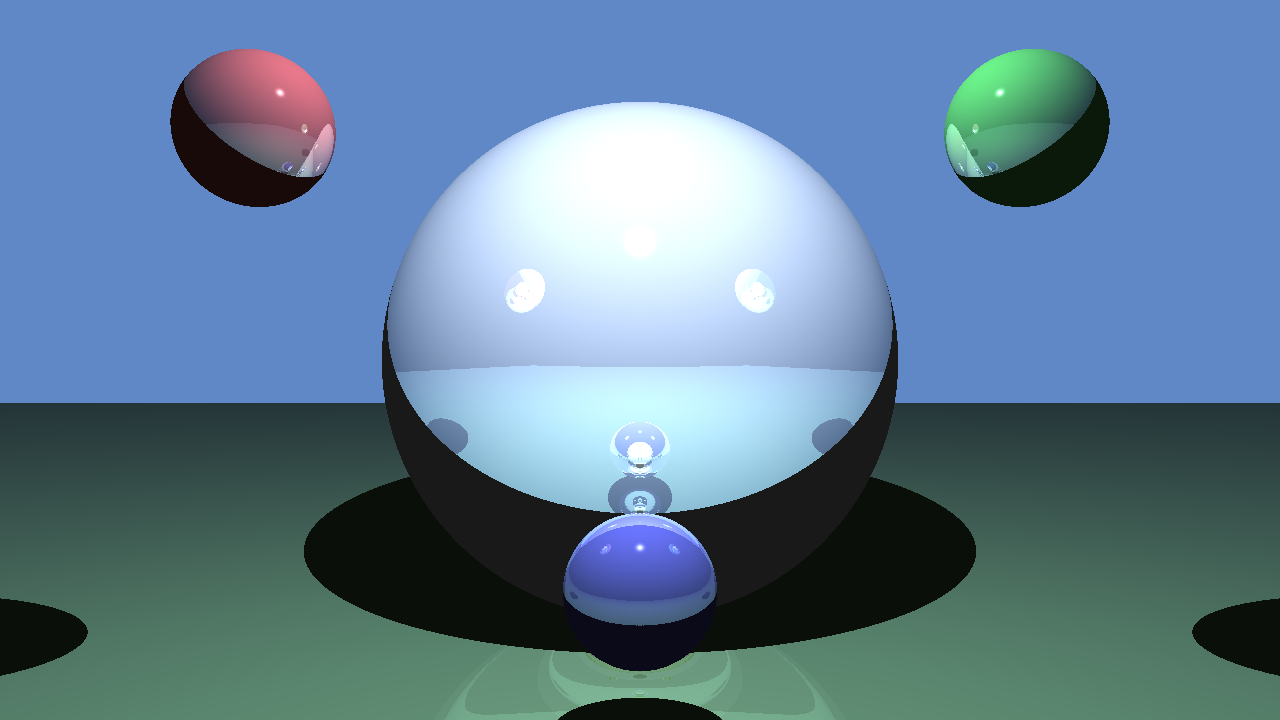
After having observed the picture it appears that the picture does not have a gradient background. This is a good thing for us because it means that I can easily implement the reflection when no collision is happening.

I decided to hard code the background color because it is not something that I’m planning to change.



When multiple surfaces are reflective, the rays get send to the background. Because of this blank spots will appear.

I realized that I was rewriting code. I already have a function that can take care of all the coloring details. By just calling the getColor function this became my result.



The only thing that I’m not sure about is if I have to also reflect the shaded areas. Ambient lighting is lightning so the rays should still be reflected. I’ll probably implement that now.

I still feel like the shadows are too rough. I think that my Lambertian shading gets ignored or something.

The reflection color could be called the surface color right. So if that is the case, instead of using diffuse color, we use the reflection color.

Surfaces started to lose their surface color when we use the reflection color. This resulted into weird bits of code when I made all the surfaces reflective.

I feel like my reflection is quite a mess. The code is becoming inflexible. I will probably roll back the changelists in perforce or remove the reflection after I have researched reflection and refraction to its fullest.

## 13:15-14:00 Help David with material class

## 14:00-15:45 Help Bryan with ???

## 15:45-16:15 Help Mathijs with finite planes

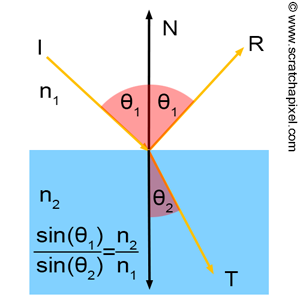
## 16:15-16:30 Help David with reflection

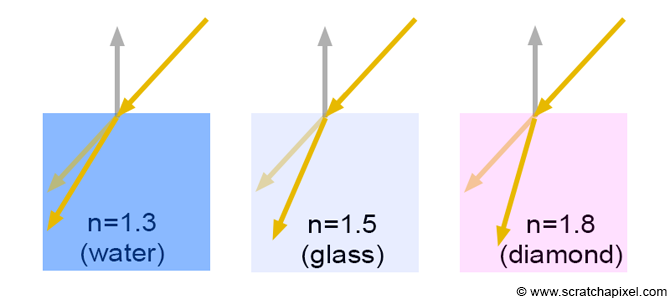
## 17:30-18:55 Research about reflection,refraction and the fresnel effect

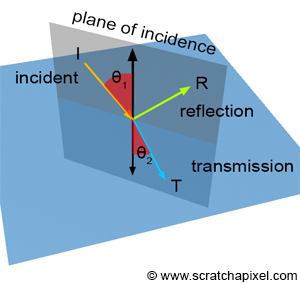
**Reflection 17:30**

* On scratchapixel. The programmer sums all the reflected colors with each other. They don’t use an index, they use a perfect mirror like surface.
* Cast a reflection ray, if the reflection ray hits a surface cast a new ray with the new depth. Get the hitcolor \* 0.8 (some magic number) and keep adding all the hit color values. In the end, the hit color is probably only one value.
* The book however gets the color recursively and keeps adding it up to the final value. Then it gets multiplied by the specular color, or just a value like 0.8 to make it distinguishable from the background. This of course removes the whole concept of surface color from the surface, as it reflects everything.

**Refraction 17:55**

* When light passes from transparent medium to another, it changes direction as illustrated to the right. 
* The direction depends on two factors. The angle of incidence and the refractive index ( index of refraction/ior) of the new medium. The ior of water and glass are 1.3 and 1.5 respectively.

*n denotes ior*

* Light travels at the speed of light in a vacuum. It travels slower in other mediums such as water. We can denote the refractive index as where n is the refractive index, c is the speed of light and v is the speed of light in a medium. N is simply the ratio of that. This means that light travels faster in water than in glass, but slower in air.
* Refraction is described by **Snell’s law** which states where 1 represents the first medium and 2 represents the second one. 
* Using the image on the right, the formula would become Thetha2 = acos(sin1 \* n1 / n2)
* If all vectors are of unit length. Sin1 = dot(-I,N) and sin2 = dot(T,-N) if the normal points up.

# Friday

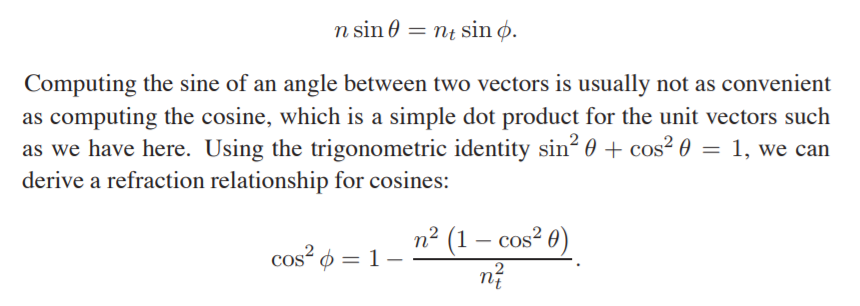
## 13:10-13:30 Fill out today’s work log

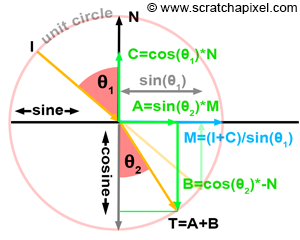
## 13:30-16:00 Research about reflection,refraction and the fresnel effect

**Trigonometry 13:30**

I have to refresh on this for a bit because I can’t understand this [image](https://www.scratchapixel.com/images/upload/shading-intro/shad-refraction7.png?).

**Refraction 13:55**

****

* **14:20** I’m having trouble understanding this formula
* **14:50** I’ve solved the equation. When I replaces sine with the identity, I forget to first square the whole equation.
* **15:00** I’m having very much trouble understanding this picture. First of all, shouldn’t cosine and sine be flipped? Cosine is supposed to represent x and sine is supposed to represent y.
* **15:10** A and B are simple geometry translated to coordinates. I understand that now. B is the x, A is the y
* **15:15** I’m not sure why C = cos(o1) \* N. Shouldn’t it be sin(o1) \* N? Because it represents the y position. Couldn’t we just normalize the normal? What does dividing by sin(o1) do?
* **15:25** I understand how c is calculated. In a unit circle cos = a/h but h = always 1. So cos = a. If we make a triangle of I center and C, it becomes obvious.   
  With that, I + C also becomes obvious. Now I just need to figure out why we divide by sin(o1).
* **15:30** The length of sin(o1) is equal to the length of I + C. That is because only the direction has changed, the length hasn’t. We divide by sin(o1) to normalize the vector. By normalizing it we can multiply it with the sin(o2).
* **16:05:** I have created the formula to create the direction of the refracted ray. See this [PDF](https://drive.google.com/open?id=1HBWgk4oE9sMuzs5ZXPQIfACOP9Jx7cRj). I’ve made it by following [scratchapixel](https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel). Honestly, I understand every step of the formula and how it is created. But it feels like something that I wouldn’t have been able to come up with myself. The guidance of scratchapixel and the step by step notation really helped me to understand how this formula was created and what every variable is supposed to represent.

## 18:00-18:30 Research about reflection,refraction and the fresnel effect

**Refraction**

* **18:00** You have to make sure that the dot product between the incident ray and normal is always positive, if it is **negative**, the ray goed inside the medium and you have to change the sign of the dot product. If it is **positive**, you leave the medium. Be sure that the ior is correct because you should divide the current ior by the new one. Also, be sure that the normal is in the right direction when calculating the ‘transparent’ ray.
* **18:25** The **critical angle** is a certain angle where the ray that is supposed to leave the medium is instead reflected. The phenomenon is called **total internal reflection**. This only happens when the ior of the new medium is lower than the ior of the old medium.   
  You know that this phenomenon happens when c2 is negative. Refer to the [PDF](https://drive.google.com/open?id=1HBWgk4oE9sMuzs5ZXPQIfACOP9Jx7cRj) to see what c2 is.