**Star Field**

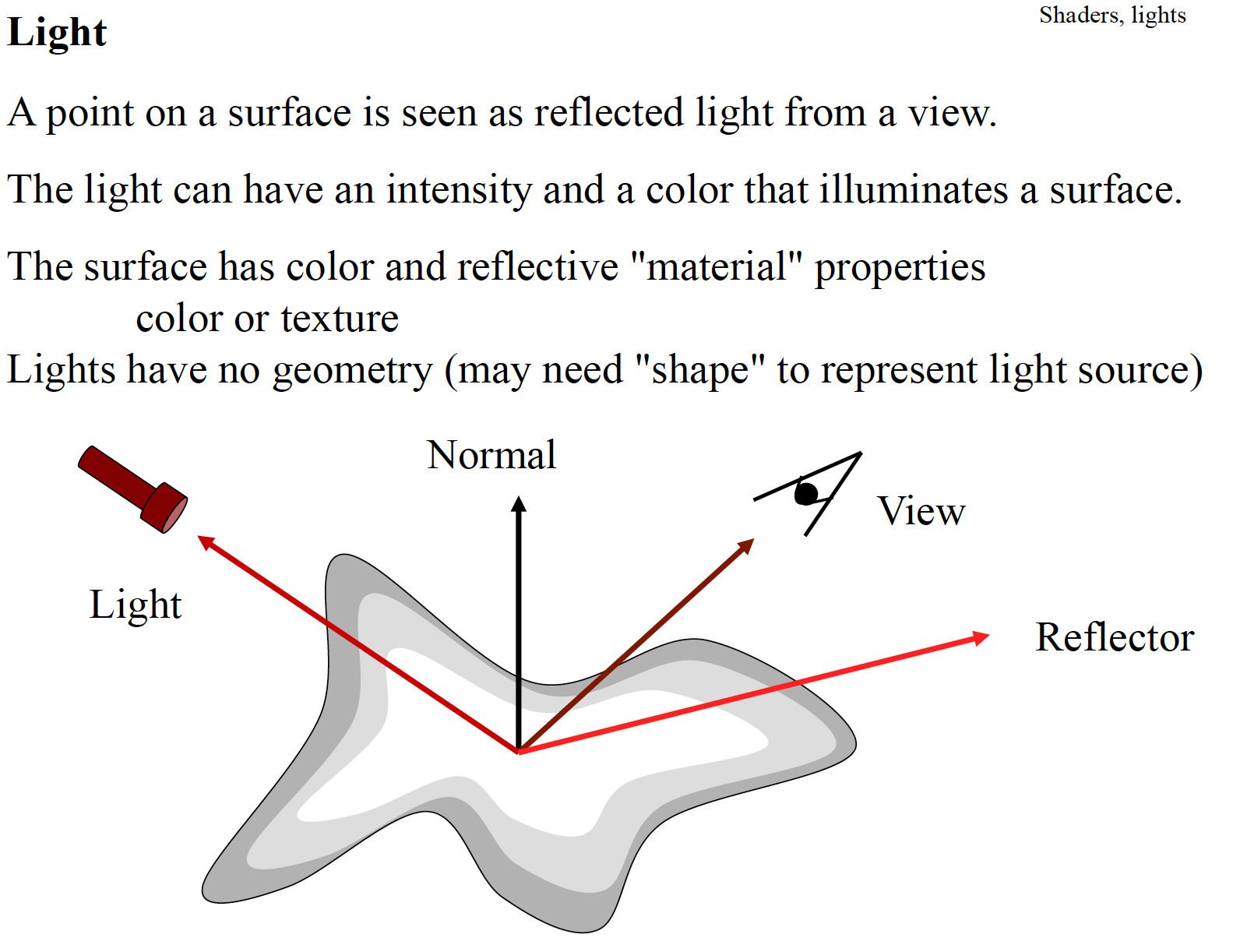
The star field is implemented as 6 planes that form a cube around the playable area.Vertex and fragment shaders have been edited to render the texture properly. The texture is a 640x480 .raw file edited via irfanview. The placement / rotation / order of each plane (or square) are stored in arrays.

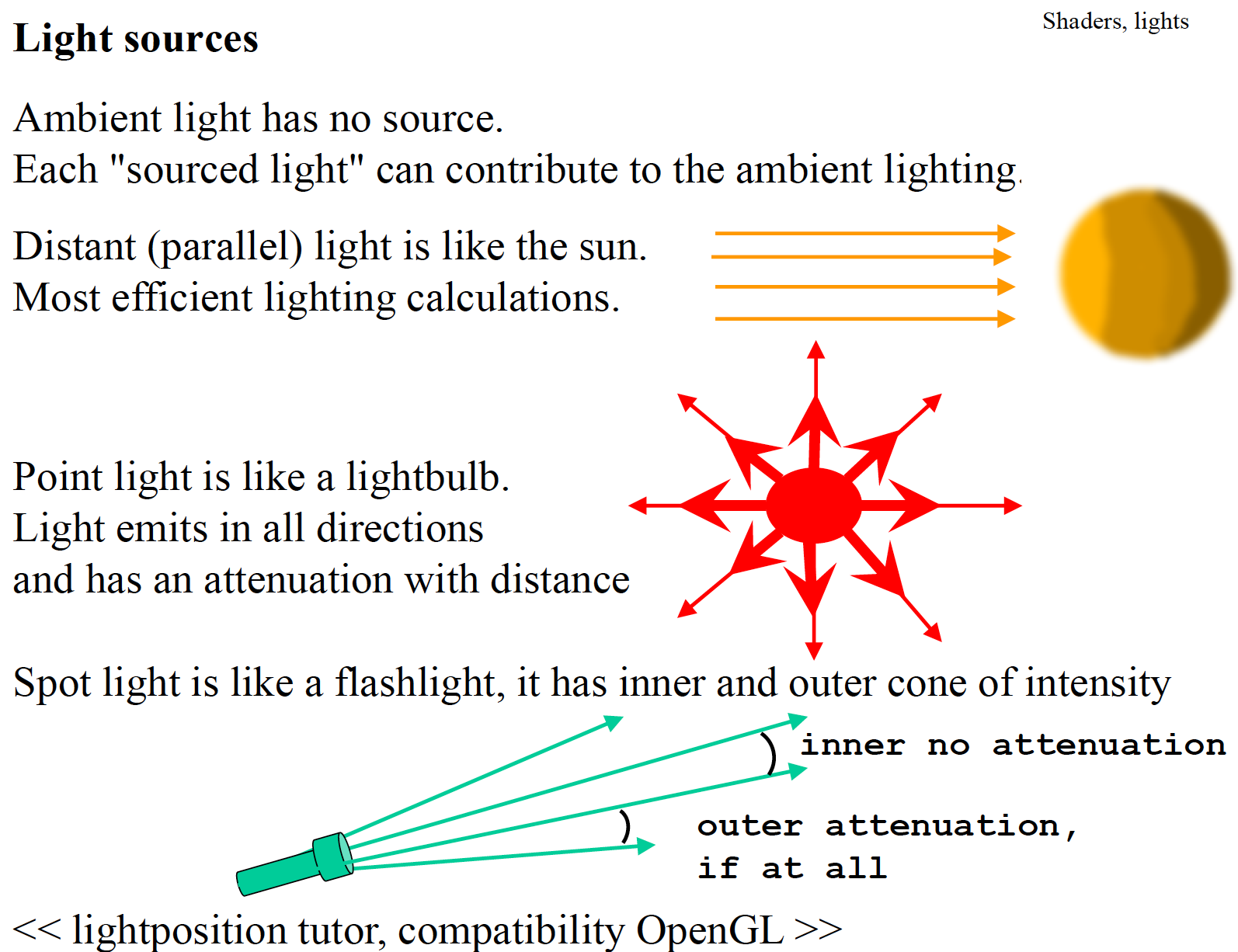
Vertex Shader

Fragment Shader

We were unable to get the fragment shader working correctly with lighting. Lighting is either way too bright or too dark.

Examples of how lighting within GLSL should work.





Point light from Ruber

Ruber is supposed to be set up as a point light that is diffuse when facing away from the light. Meaning that when you look at a planet and Ruber’s light is hitting it you will see the planet light up. In the pointLightPosition method in the simpleFragment.glsl file we attempted to calculate the diffuse by taking the dot product of the point’s normal surface by the direction of the light ( basically where the camera was in place in our space box ). Following the example form the shaders, lights slides and the example given in chapter 7 of the OpenGL programming textbook we attempted to add attenuation to the point light off Ruber to make it seem more like a real system.

Directional Light ( Headlamp for ship ) looking same direction as camera down z-axis

On this section we began by creating a new vec3 method for the directional light that would be given down the camera view. Our main goals were to check if there was any ambient light that we would have to contend with, to see if any surfaces would not light up by the light, and to see how much light should be reflected.

Extra Things we Added:

# Sounds and Music (MIND YOUR EARS)

**Warning: Make sure your sound is on low initially since it varies from system to system.**

Sounds and music were added using [irrklang](http://www.ambiera.com/irrklang/) a cross platform sound library using C++, C#, and all .NET languages.

We Simply had to download and copy the files to the project, link the library and create an object from that library and we were able to utilize its functionality.

## Sounds:

* Added various sounds from Star Trek: The Original Series, Star Trek: The Next Generation television series, and Star Trek: Armada video game.
  + When the user restarts the game all the sounds reset, the music track replays and a uncloaking sound plays with a new warbird appears since a new student is trying out for the war college.
  + When the user’s ship is collides with an object it plays an explosion sound from Star Trek TOS.
  + When the user warps it plays a warping sound from Star Trek: The Next Generation.
  + When the user fire a missile it plays a torpedo sound from Star Trek TNG.
  + We were planning on adding a sound when a missile is locked onto the warbird but didn’t get around it. The sound is still in our project folder its called: MRom300.wav
  + We were also planning on using the Romulan cloaking sound when the warbird is hit but it didn’t make sense for a Star Trek fan!
* Voices
  + Various voices were added from Star Trek Armada 1.
    - When the program is started there are three voice lines that are randomly selected to play. Keep restarting the program to hear all the different voice lines!
    - When already fired the missile there are two different voice lines that may play to give feedback.

## Music:

* When the user restarts the game the music resets.
* There are two different music tracks that can play.
  + Music is the Romulan Faction themes from Star Trek Armada 1 and Star Trek: Armada 2 Video Games.
* Added special “surprise” music if you win the game! When you win the game all the other sounds and music are stopped so you can listen to the wonderful victory music.

## Warbird Model Credits:

The model we used for our Warbird is modeled after the Warbird design from Star Trek TOS. We changed it to a tri model and colored it based on how it looked in the original series.

Romulan Bird of Prey, Classic Star Trek by jpcotedsbn is licensed under CC Attribution-NonCommercial-ShareAlike

