1. **Raytracing – Theoretical Part**

   a. Explain general raytracing algorithm in steps. You can use diagrams and sketches to visualise your answer.

   b. Write equations for: ray – sphere intersection, ray – plane intersection

   c. Using raytracing requires a lot of CPU power. How can we take advantage of modern GPUs and pipeline architectures to optimize it.

   d. Write pseudocode for ray tracer such that:

      i. The camera can be moved freely in the scene

      ii. It is possible to have more than one sphere in the scene (sphere tracing code is encapsulated in a function similar to the one below)

      ```cpp
      bool TraceSphere(Sphere sph, Material mat, Ray ray, inout vec3 col)
      Where, Ray, Sphere and Material structures are defined as shown below.
      ```

      ```cpp
      struct Ray
      {
        vec3 O;
        vec3 D;
      };

      struct Sphere
      {
        vec3 Pos;
        float Rad;
      };

      struct Material
      {
        vec3 AmbientCol;
        vec3 DiffuseCol;
        vec3 SpecularCol;
      };
      ```
2. **Phong Model—Theoretical Part**
   a. Explain what *ambient*, *diffuse* and *specular* coefficients represent and how to calculate them.
   b. (Optional) What is the difference between *Phong Shading* and *Gouraud Shading*.

3. **Shadertoy setup – Practical part, Task 0**
   a. Go to [shadertoy.com](http://shadertoy.com) website and create new account by clicking “new” on the upper right corner and following the instructions. If you do not know what shadertoy is, you can find out more here: [https://www.patreon.com/shadertoy](https://www.patreon.com/shadertoy)
   b. Once you completed the process, you should be able to log in. Once you logged in, click “new” on the upper right corner. You have now created new shader. Name it “Intro to gfx – homework” and add some description. Spend some time learning shadertoy interface - we will be using it during our next supervision extensively.
   c. Open new browser window and navigate to shader at: [https://www.shadertoy.com/view/ltVfRz](https://www.shadertoy.com/view/ltVfRz).
      
      On the left side you should see the following image

![Shadertoy Image](https://www.shadertoy.com/view/ltVfRz)

Copy the entire content of the code window, and paste it in your newly created shader replacing the auto-generated code.
Run your code by pressing button located on the left side, below the code window. You should see the shapes and background, but the floor plane is black. This is because it is texture mapped, but the texture is not specified. You can fix it by clicking on iChannel0 and selecting the texture from the bottom right.

Now you should see the correct image. We can finally start the coding!

4. **Phong Model – Practical part, Task 1**

In your shader code find the following function:

```cpp
void Phong(vec3 L, vec3 N, vec3 V, float sh, out float d, out float s)
{
    // IMPLEMENT PHONG MODEL HERE:
    d = 0.0;
    s = 0.0;
}
```

This function is already called in appropriate places in the code (feel free to explore where), the only missing part is the Phong illumination model. As you can see, ambient part is already there, so your task is to write the code which will calculate diffuse and specular coefficients.
If your result looks similar to the one shown on the image below – Congratulations!

Do not forget to save your progress!

5. **Moving the camera – Practical part, Task 2**

   In this task your goal is to move the virtual camera by 4 units up and 1 unit back, then rotate it by 35 degrees down. You may be able to find the code for creating rotation matrices in the shader if you look for it.

The final result should look like this:
6.  *Procedurally generated pattern, Practical part, Task 3*

a. Add black wave colour pattern to the floor.

The results should be similar to this:

![Image of a black wave pattern](image1)

b. To make it a bit more interesting add a colourful horizontal wave pattern, let’s say purple (any other colour of your liking will also do).

You should see something similar to this:

![Image of a colourful wave pattern](image2)
If you in a particularly experimental mood, you can try to add some colour variation to make it even more interesting.

c. Instead of having our floor uniformly lit, it would be nice to simulate a simple spotlight effect, where the pixels on the plane become darker as their distance from the center of the scene increases. This can be easily achieved by calculating the brightness coefficient using the `length()` function.

The final result should look like this:
d. Now try to make your spotlight move (HINT: use \sin() function and uniform \texttt{iTime} provided by shadertoy to achieve this simple animation).

7. \textit{Screen Space techniques – Practical part, Task 3}

a. Implement simple grayscale filter.

The results should be similar to these shown here:
b. Implement left to right transition from grayscale to fully saturated image. The transition should not be linear, but modelled with the power function with exponent $= 2.8$.

The following figure shows the shape of the transition function:

![Transition Function Graph](image)

The result should look like this:

![Result Image](image)

You have made it! Well done!
Copy the link to your shader and send it to km722@cam.ac.uk along with your written work, no later than 24 hours before your supervision. Remember to set the shadertoy privacy setting to “public” or “unlisted”. It is set to “private” by default. If you do not do this, I will not be able to access your work.