CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

Assignment due date: October 3 at 11:59pm

Electronic submission on Quercus
This assignment can be completed individually, or by a team of 2 students

Student Names (Last, First)

Student #1:

Student #2:

Student numbers

Student #1:

Student #2:

Student UtorIDs

Student #1:

Student #2:

We hereby affirm that all the solutions we provide, both in writing and in code, for this assignment are our own. We have properly cited and noted any reference material we used to arrive at this solution, and have not shared our work with anyone else.

_____ _____

Student 1 signature ***Student 2 signature***

(note: 3 marks penalty if any of the above information is missing)

CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

This assignment is intended to help you master the basic geometric principles we will use for the rest of the term to build our advanced rendering engine. To that end, you will practice 2D geometry, develop an intuition of how 2D parametric lines can be used to represent light rays, understand how we can represent simple objects and how these interact with our light rays, and use simple transformations to build a simple, but accurate renderer for light transport in 2D.

After completing this assigment, you will have gained the ability to simulate and visualize light and its interactions with scene components!

Learning Objectives – after completing this assignment you should be able to:

Manage points and vectors using simple operations such as additions, dot, and cross products.

Represent light rays using 2D parametric equations. Represent objects using 2D implicit and parametric forms.

Apply affine transformations to points and vectors, and use them to simulate the propagation of light through a simple scene.

Determine points of intersection between light rays and simple scene objects (this is the basis of ray tracing, which we will be fully developing soon).

Simulate the basic behaviour of light as it interacts with object surfaces: Reflection, scattering, and refraction.

Explain how simple light sources behave, and the difference between primary and secondary illumination.

Skills Developed:

Thinking in terms of geometry and vectors.

Manipulating simple geometric entities algebraically: rays, circles, boxes.

Using 2D transformations to implement code that bounces, reflects, and refracts light rays according to fundamental principles of optics.

Understanding and extending code that deals with images, light rays, and objects.

Reference material:

The Lecture notes up to this point, found on the course website.

This handout (be sure to read everything ***carefully***).

The comments in the starter code.

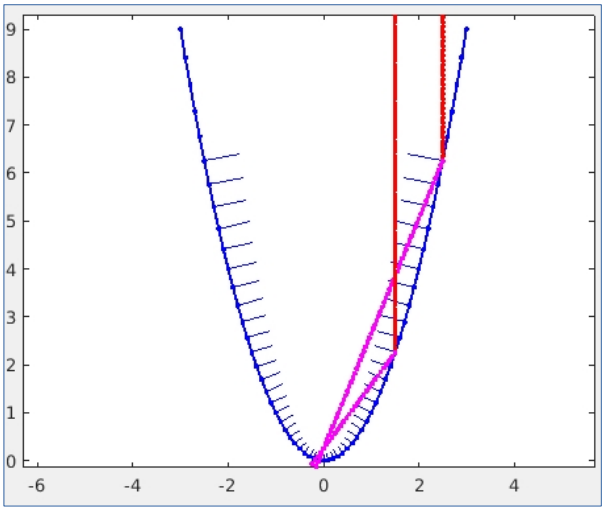
CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

Part 1 – Written work. Be sure to provide clean, legible derivations and do not omit any steps your TA may need to understand your work. Use diagrams wherever appropriate.

A) 2D – Focusing Light With Mirrors

Parabolic mirrors are used extensively in Astronomy for large telescopes. They have the property that parallel rays of light arriving at the mirror are all focused at a single point. Let's see how they work. The image below shows the parabola corresponding to

$$y = x^2$$



Short blue lines show the normal direction at various points on the parabola. Red lines show Two incoming rays of light, and the magenta lines show how these rays are reflected. Notice they converge at a point on the y axis.

[1 mark] Provide the equation for the normal vector for points along the parabola.

[1 mark] Given the ray $x = c$, where *c* is a constant, find the coordinates [x y] of the intersection point between the ray and the parabola.

[2 marks] Give the normal vector at the intersection point – please note that the normal vector has to be unit length!

[3 marks] Give the ***parametric equation*** for a ray from the intersection point, in the direction of the ***reflected ray***. You can determine the direction of perfect reflection using this very handy vector identity:

$$\vec{r} = -2(\vec{d} \cdot \vec{n})\vec{n} + \vec{d}$$

where $\vec{d}$ is a vector in the direction of the ***arriving*** light ray.

[3 marks] Find the ***focal point*** for the parabola, this will be the intersection of the reflected ray with the y-axis.

- If you did everything right, this should be independent of c -

(Attach any work for this part ***immediately*** after this page)

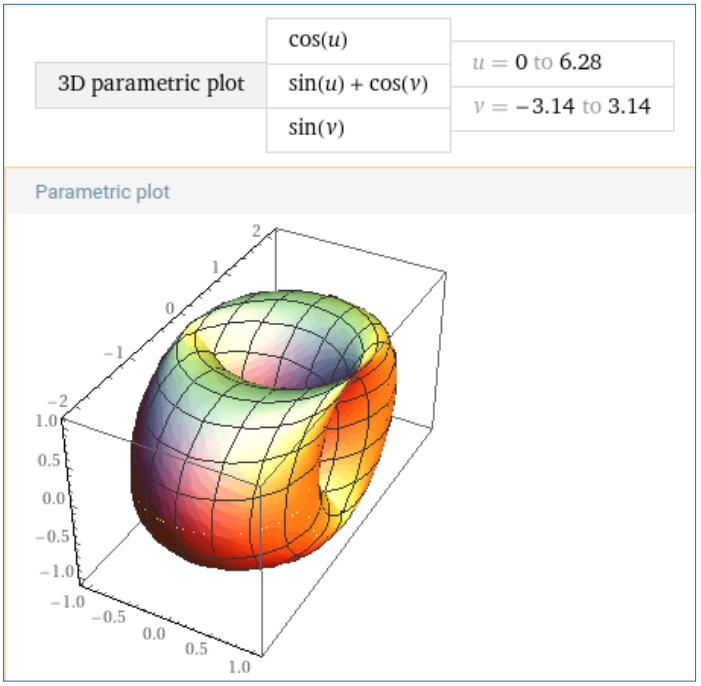
CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

B) Parametric surfaces, surface geometry, and transforms

Here, you will analyze the surface shape of a parametric 3D surface. The goal is to identify tangent planes and normal directions at arbitrary points on the surface – this would be required to render the surface in any 3D software.

The image below (plot courtesy of Wolfram Alpha) is the result of the parametric equations

$$x(u,v) = Cos(u)$$
$$y(u,v) = Sin(u) + Cos(v)$$
$$z(u,v) = Sin(v)$$



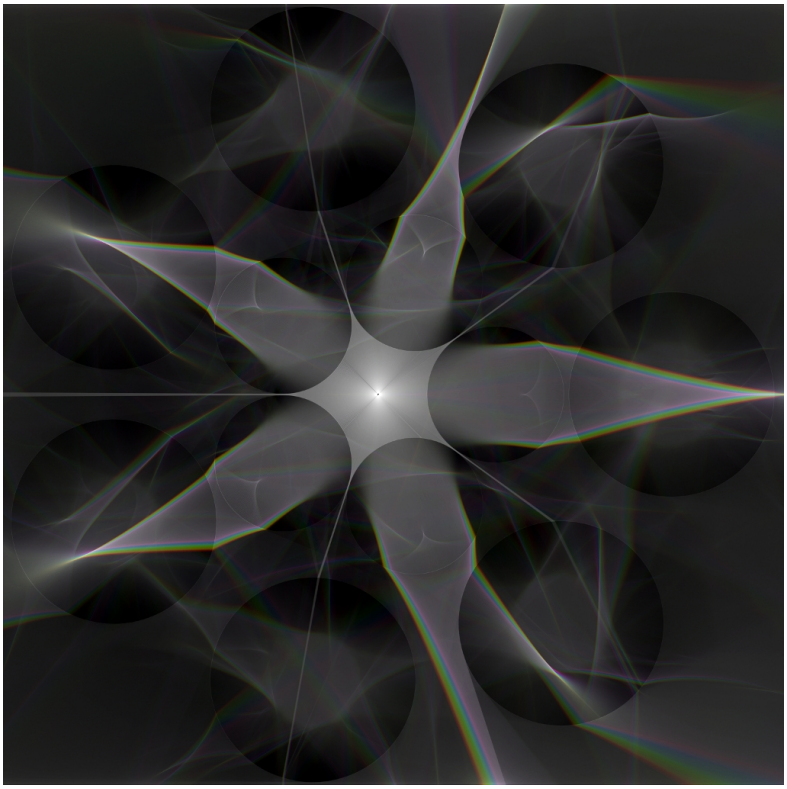
a) [3 marks] Derive the equations for the ***tangent plane*** at any point (s,t) on the surface. This is a parametric surface so we expect the tangent plane as a parametric plane defined by two vectors spanning the plane.

b) [3 marks] Determine the ***implicit*** equation for the tangent plane. This will requite you to figure out the ***normal to the plane***.

c) [3 marks] The shape dome above is at the origin, and it is axis-aligned. suppose we need to render this shape so that the center of the shape is at $\bar{p_c} = [p_x, p_y, p_z]$, and the vertical axis of the bowl is aligned with the vector $\vec{v_d} = [v_x, v_y, v_z]$

Determine the sequence of transformations that takes the points in the original shape, and produces the points in the transformed one. For rotations, show the rotation matrix and ***show how the angle of rotation is computed***. Your transforms should involve the components of the point and vector given above, and you may want to verify your work by plotting things on Matlab.

Part 2 – Understanding how light works, 2D light transport



Pattern of light resulting from a point light source (at the center of the image) and a set of refracting spheres

Your task for this assignment is to implement the core components of a light-propagation Algorithm. The task of the algorithm is fairly simple:

Given a scene consisting of
a) 1 light source (which has a known position, colour, and type)
b) A set of objects (which in this case are circles, with known position and material type)

The program will:
1) Emit a light ray from the light source
2) Propagate the ray through the scene until it hits an object (or one of the 4 walls that make up the image boundary)
3) It then will bounce the ray in a physically consistent way, depending on the material the object the ray hit is made of

The starter code provides most of the nuts and bolts you need to implement the interesting bits, so you will focus on the geometric issues involved with the steps described above, and write a little (but not a lot!) of code. Then you can sit back and watch your program trace light around a scene you created!

Part 2 – Understanding how light works, 2D light transport (cont.)

Step 1) Download and uncompress the starter code into a suitable directory. Take time now to compile and run it – learn what the command line arguments do, and how to use the program. Of course, at this point it won't be able to trace light, but it will show you a box of the size you specified, and the outlines of the objects defined for the scene.

Step 2) Read all the ***header*** files included with the starter code – I am providing you with a lot of functionality, so you should know what is already there for you to use, and not waste time implementing functions I'm providing. You will also get a picture of how the code is structured, and what parts you need to implement.

Step 3) Read ***CAREFULLY*** the comments in rays2D.c – this is the file you will be working on, and it has a couple of functions you must implement. The code has comments that will help you understand what you need to do.

Once you're done reading these comments, take a short break, then come back and read them again to make sure you didn't miss anything.

Step 4) Implement your solution. There are of course many ways to go about doing this, but I would suggest starting with casting rays from simple 'laser' light sources, and intersections between rays and walls.

That gives you a basic framework on which to build the rest. The starter code will tell you what needs to be implemented. ***Test everything thoroughly***, and for more than one case – your code will be auto-tested, so make sure it does the right thing on different scenes.

For testing:

- *1 or 2 samples, and recursion depth of 1 or 2 so you can check the basic geometry is correct.*
- *Once you have the fundamental components working, increase sampling and recursion depth to check the process works to check the scene is rendered as expected.*

Of course, you should have a good idea what the scene should look like given the light source and the objects in it

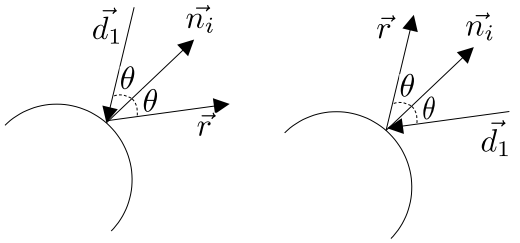
Marking:

[2 marks] – Working directional light source (rays propagate from origin in the right direction)
[2 marks] – Working point light source (rays propagating in all directions from location)
[2 marks] – Intersection with circles working properly and no visible artifacts/numerical issues
[2 marks] – Intersection with walls working properly and no visible artifacts/numerical issues
[2 marks] – Reflection working for circles and walls – correct directions and no artifacts
[3 marks] – Refraction working for circles – correct direction entering or leaving circle
[3 marks] – Recursive light propagation working – multiple bounces/transmission visible
[4 marks] – Render a cool scene (requires thoughtful use of refraction and scene setup)

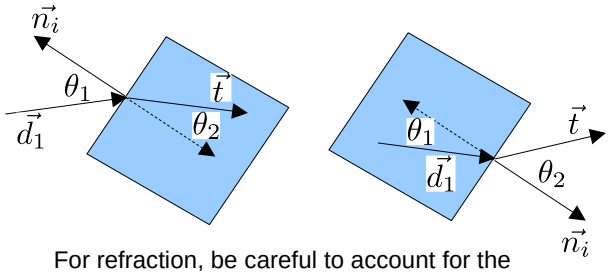
CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

Part 2 – Understanding how light works, 2D light transport (cont.)

Notes: *You can use rotations to determine reflection and refraction directions, or you can use an algebraic approact (such as for example what you did in the written part of the assignment. The figures below show the expected geometry, mind the fact that light rays may arrive at a point on either side of the normal!*



For reflection, the *sign of the angle* depends on whether the normal is to the left or to the right of the incoming ray



For refraction, be careful to account for the directions of the different vectors which depend on whether the ray is arriving at, or leaving an object

Step 5) Pack your ***entire assignment for submission:***

- Complete 'autotester_id.txt' so the auto-tester knows who you are.
- Create a .pdf of your written work ***make sure it contains both student names and student numbers if you're in a team.***
- Compress your solution code, .pdf, and autotester_id.txt into a single file

light2D_studentNo1_studentNo2.tgz **(e.g. light2D_11223344_55667788.tgz)**
- or -
light2D_studentNo1.tgz **(if working alone)**

The *tar* command syntax is:

>tar -cvzf name_of_your_compressed_tar_file.tgz *.pdf *.c *.h autotester_id.txt

Then submit the compressed file on Quercus.

Double check that your compressed file uncompresses properly, and that it contains all the code as well as the 'autotester_id.txt' file and your written work.

Marking Scheme	
Written problems	19 marks
Working code	20 marks (matching reference images)
Crunchy bonus	Up to you!

CSC D18 – Fall 2021
Assignment 1 – Basic Geometry and 2D Light Transport

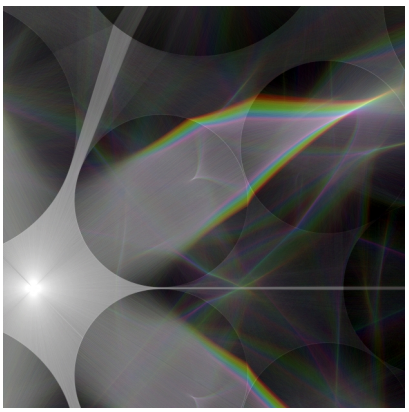
Part 2 – Understanding how light works, 2D light transport (cont.)

Get Crunchy!

Of course, once you have a working 2D light transport engine, you want to use it to render very cool images. For bonus marks, you can extend your renderer to:

[up to 3 marks] - You need to render a cool scene regardless, but if you go above and beyond And we are impressed with your render, you can earn up to 3 extra marks. For this You should make clever use of the light patterns produced by refraction and Reflection, and carefully design your scene setup.

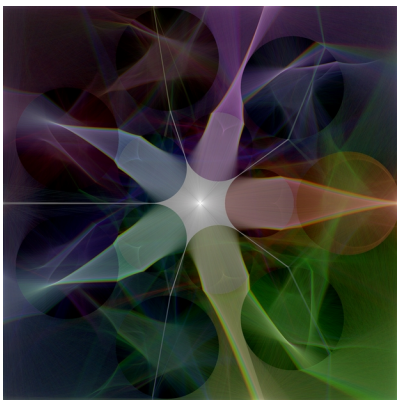
[3 marks] – Implement ***dispersion***. White light is a mixture of light across all visible wavelengths. Refracting objects bend light by different amounts depending on the wavelength, and we expect them to spread out the different colour components in white light (creating rainbows).
Modify your code to handle this by cleverly using sampling, and by manipulating the index of refraction of the material depending on light wavelength.



Dispersion of light by refracting materials

[3 marks] - Implement spectral power distributions for lightsources. Implement proper light sources whose colour comes from a specific mixture of wavelengths at different amounts, and have your light source emit rays that follow this specific mixture's distribution.
For full marks, your solution must use a a proper model for the SPD and random sampling

[2 marks] - Implement coloured objects. Until now, all objects are 'white' in that they do not in any way change the colour of the light bounced by, or transmitted through them. modify your code so that it accounts for coloured objects.



Scene rendered with a white lightsource and colour Refracting spheres.

Drop by and talk with me if you want to work On any of these features but need a hint or two, or if you have other crunchy ideas to Try.

Have Fun!

Submitting crunchy stuff: Include a rendered scene with your compressed submitted file, as well as a 'crunchy.txt' describing what you did!