CSC D18 – Fall 2017
Assignment 1 – How to become a Light Bender

Assignment due date: Sep. 28th 8:00am

Hand-in to be submitted at the start of lecture,
code to be submitted on the mathlab server by the above due date
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 2017
Assignment 1 – How to become a Light Bender

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 2017
Assignment 1 – How to become a Light Bender

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 – Parametric forms and the geometry of light transport

1) A light ray $\vec{l_1}(\lambda_1)$ hits an object at a point $\vec{p_i}$ with surface normal $\vec{n_i}$

$$\vec{l_1}(\lambda_1) = \vec{p_1} + \lambda_1 \vec{d_1}, \text{ with } \vec{p_1} = [p_{1x}\ ,\ p_{1y}], \vec{d_1} = [d_{1x}\ ,\ d_{1y}]$$

$$\vec{p_i} = [p_{ix}\ ,\ p_{iy}] \qquad \vec{n_i} = [n_{ix}\ ,\ n_{iy}]$$

a) [3 marks] – If the object is a 2D circle, $x^2 + y^2 = r^2$ (centered at the origin, with radius *r*)
Develop an equation in terms of $\vec{p_1},\ \vec{d_1},\ \text{and}\ r$ that yields the value of λ_1 at the intersection point between the ray and the circle.

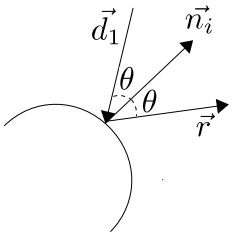
Specifically, your derivation should provide the values for *A*, *B*, and *C* in the familiar solution to 2nd degree polynomial equations:

$$\lambda_1^* = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

b) [2 marks] – Given the value of λ_1 at the intersection, give the components of $\vec{p_i},\ \text{and}\ \vec{n_i}$ at the intersection point. The normal should be unit length.

c) [2 marks] – If the circle is ***not*** centered at the origin, but instead has a center at $\vec{c} = [c_x \quad c_y]$ show how to use a suitable 2D transform so that we can use the intersection test in a) for this circle.

d) [3 marks] - Consider the diagram below, and show how to use a 2D rotation to obtain vector $\vec{r}$ in the reflection direction from $\vec{d_1},\ \text{and}\ \vec{n_i}$ (you should provide an equation to obtain the rotation angle, as well as the rotation matrix and the point or vector it is applied to In order to obtain $\vec{r}$)



e) [1 marks] The direction of rotation depends on whether the normal is to the right, or to the left of the incoming ray. Show how to determine the *sign* of the angle found in d) so that the rotation works for every case.

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

CSC D18 – Fall 2017
Assignment 1 – How to become a Light Bender

B) Parametric curves and 3D transforms

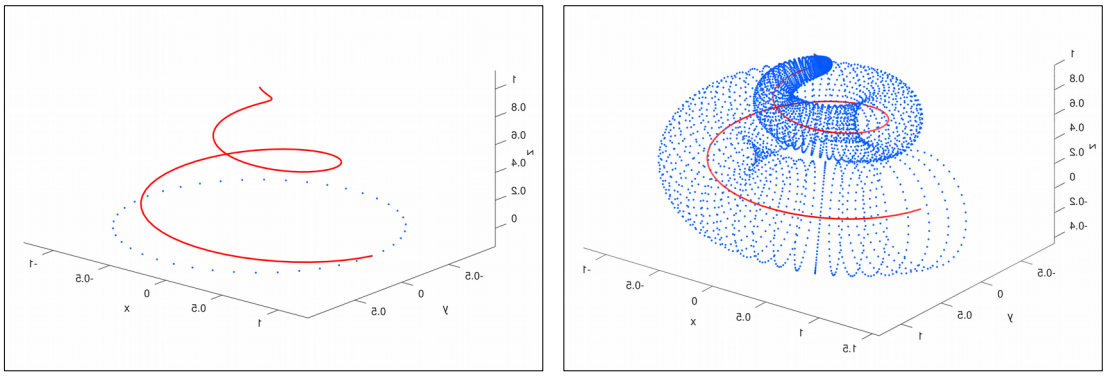
Here, you will use parametric curves in 3D, a bit of calculus, and 3D transformations to develop a simple procedure for rendering the snail-surface shown below.

Given a parametric curve $\vec{S}(t)$ in 3D given by

$$Sx(t) = (1-t)cos(4\pi t) \qquad Sy(t) = (1-t)sin(4\pi t) \qquad Sz(t) = t \qquad 0 \leq t \leq 1$$

And a circle on the XY plane given by

$$C_x(\lambda) = cos(2\pi\lambda) \qquad C_y(\lambda) = sin(2\pi\lambda) \qquad C_z(\lambda) = 1 \qquad 0 \leq \lambda \leq 1$$



Original curve and circle, and resulting snail-shell surface

a) [3 marks] Find the equations for each of the components of the ***tangent*** vector to $\vec{S}(t)$ for any value of *t*.

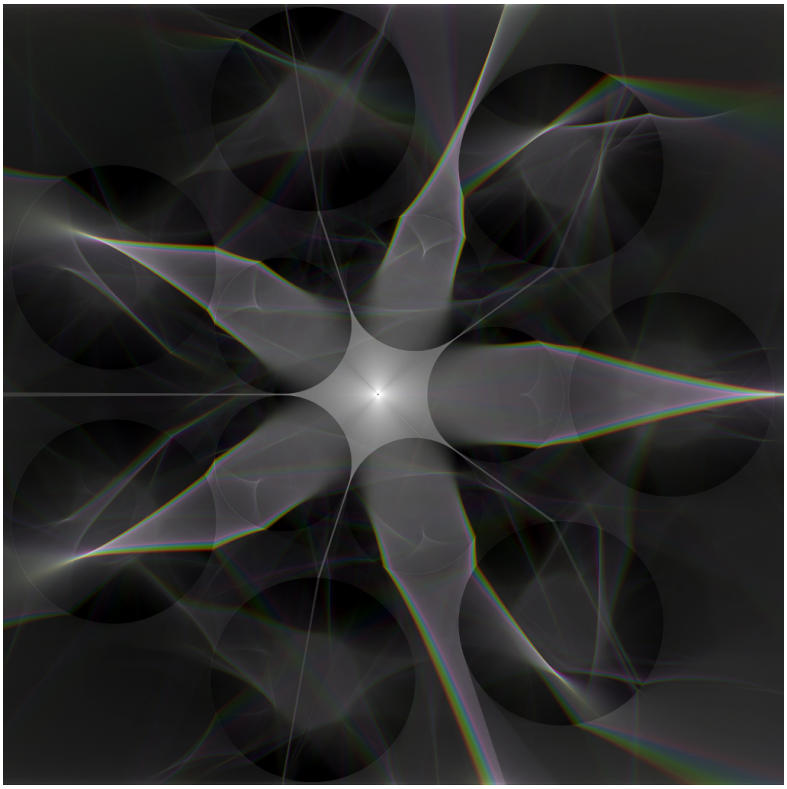
b) [3 marks] Given that the circle has normal [0 0 1], show the sequence of transformations that aligns the circle's normal with the tangent vector from a). This should take the form of a sequence of rotations, so you must specify their order, the axis of rotation, and how to obtain the angle for each rotation.

c) [2 marks] Give the complete sequence of transformations that maps points in the original circle to points on a circle centered at $\vec{S}(t)$ and whose normal is tangent to the curve.

d) [2 marks] Modify the transformation from c) so that the radius of the circle decreases toward the center of the snail.

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

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.*

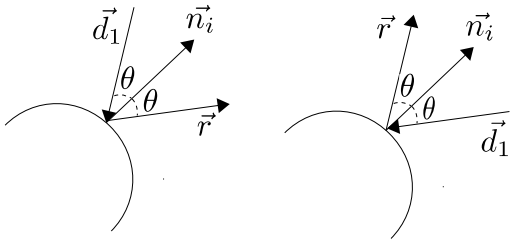
- *Once you have the fundamental components working, increase sampling and recursion depth to check the process works to check the scene is rendered as expected.*

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

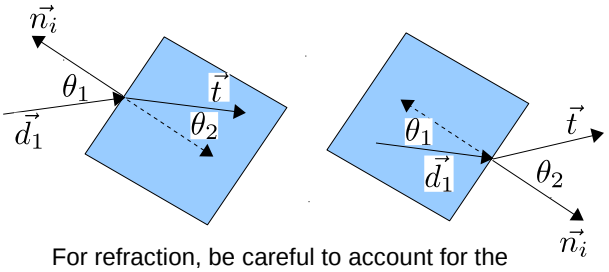
CSC D18 – Fall 2017
Assignment 1 – How to become a Light Bender

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

Notes: *You are expected to use 2D rotations to generate the vector directions you need.* This means your code needs to compute the relevant angles, determine suitable centers of rotation (and points to rotate), and then do the rotation to obtain whatever vector directions you need. Remember:



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 code for submission:

- Complete 'autotester_id.txt' so the auto-tester knows who you are.

Compress your code into a single compressed ***.tgz*** file named

light2D_studentNo1_studentNo2.tgz **(e.g. light2D_11223344_55667788.tgz)**

The *tar* command syntax is:

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

Then submit the compressed file

```
>submit -c cscd18f17 -a A1 -f name_of_your_compressed_tar_file.tgz
```

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

Marking Scheme	
Written problems	20 marks
Working code	30 marks (auto-tested)
In-class quiz	50 marks (individual)

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

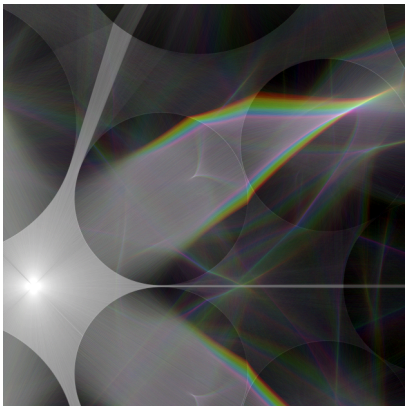
Get Crunchy!

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:

[5 marks] - Render a very cool scene (it should not be a clone of the one in this handout) Spend some time placing objects with suitable materials around, and render the scene at a reasonably good resolution, for lots of samples.

[10 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).

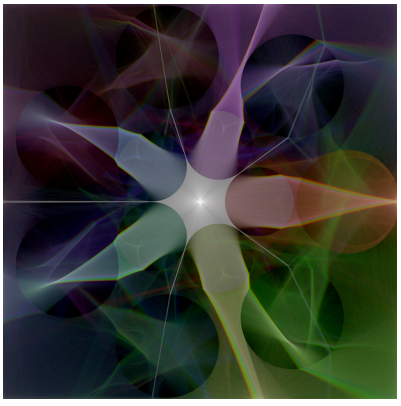
Modify your code to handle this by cleverly using sampling, and by manipulating the index of refraction of the material depending on light wavelength.



Dispersion of light by refracting materials

[5 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.

[5 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!