Contents

1 Refraction *3*

2 Lenses *5*

2.1 Focal Length *6*

2.2 Refractive Index *6*

3 Images in Python *7*

3.1 Adding color *8*

3.2 Using an existing image *11*

4 Introduction to Polynomials *13*

A Answers to Exercises *17*

Index 19

Chapter 1

Refraction

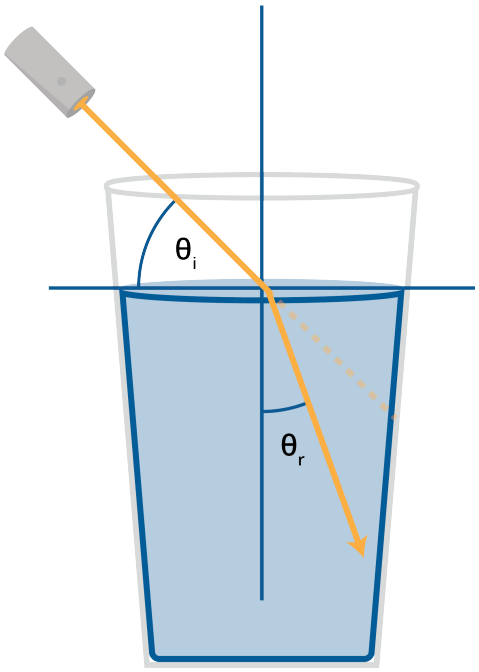
The refraction of light is a phenomenon where light changes its direction when it passes from one medium to another. The change in direction is due to a change in the speed of light as it moves from one medium to another.

This phenomenon is explained by Snell's law, which states:

$$n_1 \cdot \sin(\theta_1) = n_2 \cdot \sin(\theta_2) \tag{1.1}$$

where:

- n_1 and n_2 are the indices of refraction for the first and second media, respectively. The index of refraction is the ratio of the speed of light in a vacuum to the speed of light in the medium. It is a dimensionless quantity.
- θ_1 and θ_2 are the angles of incidence and refraction, respectively. These angles are measured from the normal (perpendicular line) to the surface at the point where light hits the boundary.



The angle of incidence (θ_1) is the angle between the incident ray and the normal to the interface at the point of incidence. Similarly, the angle of refraction (θ_2) is the angle between the refracted ray and the normal.

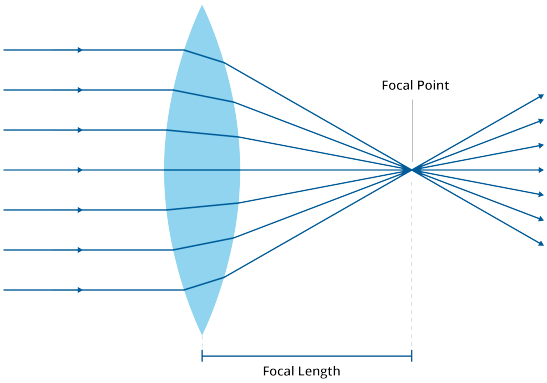
When light travels from a medium with a lower refractive index to a medium with a higher refractive index, it bends towards the normal. Conversely, when light travels from a medium with a higher refractive index to one with a lower refractive index, it bends away from the normal.

CHAPTER 2

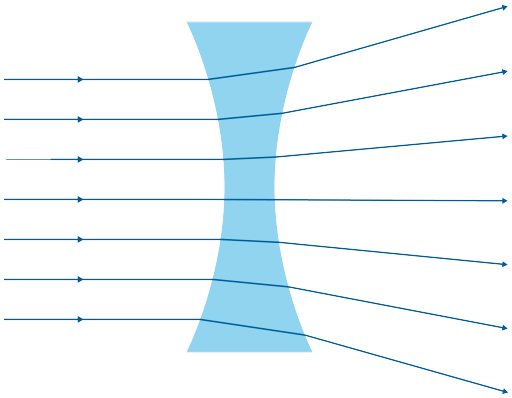
Lenses

Lenses are optical devices with perfect or approximate axial symmetry that transmit and refract light, converging or diverging the beam. There are two main types of lenses, distinguished by their shape and the way they refract light:

- **Converging (or Convex) Lenses:** These are thicker at the center than at the edges. When parallel light rays enter a convex lens, they converge to a point called the focal point. Examples of converging lenses include magnifying glasses and camera lenses.



- **Diverging (or Concave) Lenses:** These are thinner at the center than at the edges. When parallel light rays enter a concave lens, they diverge or spread out. These lenses are often used in glasses to correct nearsightedness.



2.1 Focal Length

The focal length of a lens is the distance between the center of the lens and the focal point. It is determined by the lens shape and the refractive index of the lens material. For a converging lens, the focal length is positive; for a diverging lens, the focal length is negative.

2.2 Refractive Index

The refractive index of a material is a measure of how much the speed of light is reduced inside the material. The refractive index n of a material is given by the ratio of the speed of light in a vacuum c to the speed of light v in the material:

$$n = \frac{c}{v}$$

The refractive index affects how much a light ray changes direction, or refracts, when entering the material at an angle. A higher refractive index indicates that light travels slower in that medium, and the light ray will bend more towards the normal.

Lenses work by refracting light at their two surfaces. By choosing the right lens shape and material, lenses can be designed to bring light to a focus, spread it out, or perform more complex transformations.

Chapter 3

Images in Python

An image is usually represented as a three-dimensional array of 8-bit integers. NumPy arrays are the most commonly used library for this sort of data structure.

If you have an RGB image that is 480 pixels tall and 640 pixels wide, you will need a $480 \times 640 \times 3$ NumPy array.

There is a separate library (imageio) that:

- Reads an image file (like JPEG files) and creates a NumPy array.
- Writes a NumPy array to a file in standard image formats

Let's create a simply python program that creates a file containing an all-black image that is 640 pixels wide and 480 pixels tall. Create a file called `create_image.py`:

```
import NumPy as np
import imageio
import sys

# Check command-line arguments
if len(sys.argv) < 2:
    print(f"Usage {sys.argv[0]} <outfile>")
    sys.exit(1)

# Constants
IMAGE_WIDTH = 640
IMAGE_HEIGHT = 480

# Create an array of zeros
image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)

# Write the array to the file
imageio.imwrite(sys.argv[1], image)
```

To run this, you will need to supply the name of the file you are trying to create. The extension (like .png or .jpeg) will tell imageio what format you want written. Run it now:

```
python3 create_image.py blackness.png
```

Open the image to confirm that it is 640 pixels wide, 480 pixels tall, and completely black.

3.1 Adding color

Now, let's walk through through the image, pixel-by-pixel, adding some red. We will gradually increase the red from 0 on the left to 255 on the right.

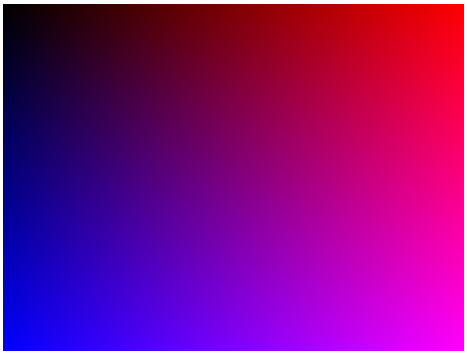
```
import NumPy as np
import imageio
import sys

# Check command-line arguments
if len(sys.argv) < 2:
    print(f"Usage sys.argv[0] <outfile>")
    sys.exit(1)

# Constants
IMAGE_WIDTH = 640
IMAGE_HEIGHT = 480

# Create an array of zeros
image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)

for col in range(IMAGE_WIDTH):

    # Red goes from 0 to 255 (left to right)
    r = int(col * 255.0 / IMAGE_WIDTH)

    # Update all the pixels in that column
    for row in range(IMAGE_HEIGHT):
        # Set the red pixel
        image[row, col, 0] = r

# Write the array to the file
imageio.imwrite(sys.argv[1], image)
```

When you run the function to create a new image, it will be a fade from black to red as you move from left to right:

Now, inside the inner loop, update the blue channel so that it goes from zero at the top to 255 at the bottom:

```
    # Update all the pixels in that column
    for row in range(IMAGE_HEIGHT):

        # Update the red channel
        image[row,col,0] = r

        # Blue goes from 0 to 255 (top to bottom)
        b = int(row * 255.0 / IMAGE_HEIGHT)
        image[row,col,2] = b

imageio.imwrite(sys.argv[1], image)
```

When you run the program again, you will see the color fades from black to blue as you go down the left side. As you go down the right side, the color fades from red to magenta.



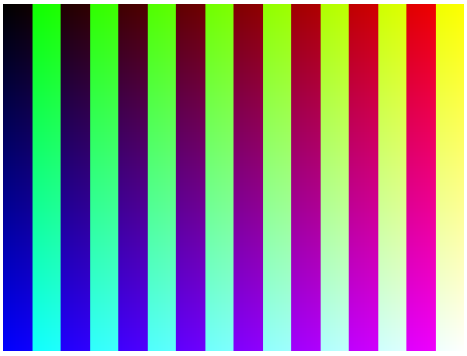
Notice that red and blue with no green looks magenta to your eye.

Next, let's add some stripes of green:

```
import NumPy as np
```

```
import imageio
import sys

# Check command line arguments
if len(sys.argv) < 2:
    print(f"Usage sys.argv[0] <outfile>")
    sys.exit(1)

# Constants
IMAGE_WIDTH = 640
IMAGE_HEIGHT = 480
STRIPE_WIDTH = 40
pattern_width = STRIPE_WIDTH * 2

# Create an image of all zeros
image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np.uint8)

# Step from left to right
for col in range(IMAGE_WIDTH):

    # Red goes from 0 to 255 (left to right)
    r = int(col * 255.0 / IMAGE_WIDTH)

    # Should I add green to this column?
    should_green = col % pattern_width > STRIPE_WIDTH

    # Update all the pixels in that column
    for row in range(IMAGE_HEIGHT):

        # Update the red channel
        image[row,col,0] = r

        # Should I add green to this pixel?
        if should_green:
            image[row,col,1] = 255

        # Blue goes from 0 to 255 (top to bottom)
        b = int(row * 255.0 / IMAGE_HEIGHT)
        image[row,col,2] = b

imageio.imwrite(sys.argv[1], image)
```

When you run this version, you will see the previous image in half the stripes. In the other half, you will see that green fades to cyan down the left side, and yellow fades to white down the right side.



3.2 Using an existing image

imageio can also be used to read in any common image file format. Let's read in an image and save each of the red, green, and blue channels out as its own image.

Create a new file called `separate_image.py`:

```
import imageio
import sys
import os

# Check command line arguments
if len(sys.argv) < 2:
    print(f"Usage {sys.argv[0]} <infile>")
    sys.exit(1)

# Read the image
path = sys.argv[1]
image = imageio.imread(path)

# What is the filename?
filename = os.path.basename(path)

# What is the shape of the array?
original_shape = image.shape

# Log it
print(f"Shape of {filename}:{original_shape}")

# Names of the colors for the filenames
colors = ['red','green','blue']

# Step through each of the colors
```

```
for i in range(3):

    # Create a new image
    newimage = np.zeros(original_shape, dtype=np.uint8)

    # Copy one channel
    newimage[:,:,i] = image[:,:,i]

    # Save to a file
    new_filename = f"{colors[i]}_{filename}"
    print(f"Writing {new_filename}")
    imageio.imwrite(new_filename, newimage)
```

Now, you can run the program with any common RGB image type:

```
python3 separate_image.py dog.jpg
```

This will create three images: `red_dog.jpg`, `green_dog.jpg`, and `blue_dog.jpg`.

Chapter 4

Introduction to Polynomials

Watch Khan Academy's **Polynomials intro** video at https://youtu.be/Vm7H0VTlIco

A *monomial* is the product of a number and a variable raised to a non-negative (but possibly zero) integer power. Here are some examples of monomials:

$3x^2$ πx^2 $7x$ $-\frac{2}{3}x^{12}$

$-2x^{15}$ $(3.33)x^{100}$ 3 0

The exponent is called the *degree* of the monomial. For example, $3x^{17}$ has degree 17, $-7x$ has degree 1, and 3.2 has degree 0 (because you can think of it as $(3.2)x^0$).

The number in the product is called the *coefficient*. Example: $3x^{17}$ has a coefficient of 3, $-2x$ has a coefficient of -2, and $(3.4)x^{1000}$ has a coefficient of 3.4.

A *polynomial* is the sum of one or more monomials. Here are some polynomials:

$4x^2 + 9x + 3.9$ $\pi x^2 + \pi x + \pi$ $7x + 2$

$-2x^{10} + (3.4)x - 45x^{900} - 1$ 3.3 $3x^{20}$

We say that each monomial is a *term* of the polynomial.

$x^{-5} + 12$ is *not* a polynomial because the first term has a negative exponent.

$x^2 - 32x^{\frac{1}{2}} + x$ is *not* a polynomial because the second term has a non-integer exponent.

$\frac{x+2}{x^2+x+5}$ is *not* a polynomial because it is not just a sum of monomials.

Exercise 1 Identifying Polynomials

Circle only the polynomials.

Working Space

$-2x^3+\frac{1}{2}x+3.9(4.5)x^2+\pi x$ $\qquad 7$

$x^{\frac{2}{3}}$ $\qquad 3x^{20}+2x^{19}-5x^{18}$

$2x^{-10}+4x-1$

Answer on Page 17

We typically write a polynomial starting at the term with the highest degree and proceed in decreasing order to the term with the lowest degree:

$$2x^9-3x^7+\frac{3}{4}x^3+x^2+\pi x-9.3$$

This is known as *the standard form*. The first term of the standard form is called *the leading term*, and we often call the coefficient of the leading term *the leading coefficient*. We sometimes speak of the degree of the polynomial, which is just the degree of the leading term.

Exercise 2 Standard of a Polynomial

Write $21x^2-x^3+\pi-1000x$ in standard form. What is the degree of this polynomial? What is its leading coefficient?

Working Space

Answer on Page 17

Exercise 3 Evaluate a Polynomial

Let $y = x^3 - 3x^2 + 10x - 12$. What is y when x is 4?

Working Space

Answer on Page 17

We would be remiss in our duties if we didn't mention one more thing about polynomials: Mathematicians have defined a polynomial to be a sum of a *finite* number of monomials.

It is certainly possible to have a sum of an infinite number of monomials like this:

$$1 + \frac{1}{2}x + \frac{1}{4}x^2 + \frac{1}{8}x^3 + \frac{1}{16}x^4 + \dots$$

This is an example of an *infinite series*, which we don't consider polynomials. Infinite series are interesting and useful, but we will not discuss them in detail until later in the course.

Appendix A

Answers to Exercises

Answer to Exercise 1 (on page 14)

$\boxed{-2x^3 + \frac{1}{2}x + 3.9}$

$\boxed{(4.5)x^2 + \pi x}$

$\boxed{7}$

$2x^{-10} + 4x - 1$

$x^{\frac{2}{3}}$

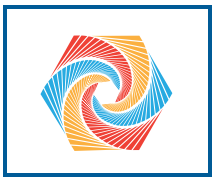
$\boxed{3x^{20} + 2x^{19} - 5x^{18}}$

Answer to Exercise 2 (on page 14)

Standard form would be $-x^3 + 21x^2 - 1000x + \pi$. The degree is 3. The leading coefficient is -1

Answer to Exercise 3 (on page 15)

$4^3 - (3)(4^2) + (10)(4) - 12 = 64 - 48 + 40 - 12$. So $y = 44$



Index

coefficient
    polynomial, 13

degree
    polynomial, 13

focal length, 6

lenses, 5

monomial, 13
    coefficient, 13
    degree, 13

polynomial, 13
    definition of, 13

refractive index, 6

standard form
    polynomial, 14