Light Waves Project

Importing libraries:

Explanation:

Our program runs on python. It uses the NumPy library to calculate the angle of refraction and Tkinter to render the results. Here are is the code we used to import the library into the code:

Code:

```
from tkinter import *
import numpy as np
```

`from tkinter import *`. The * means that instead of writing for example: `tkinter.Label()`, you just have to write `Label()`. For the NumPy library. I used the `import numpy as np` . The `as np` means that instead of righting `numpy.sin()` you just have to write `np.sin()` .

Snell's Law:

Explanation:

The relationship between the path taken by a ray of light in crossing the boundary or surface of separation between two contacting substances and the refractive index of each.

$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

https://www.dropbox.com/s/aun7aozf1n3yq3r/Video%202022-05-05%2C%2016%2034%2051.mov?dl=0

Code:

```
theta_2 = np.degrees(np.arcsin(n1 / n2 * np.sin(theta_1))
```

This line of code is a representation of Snell's Law in python. The `np.degrees()` is used because NumPy is defaulted to using radians, so that line of code tells the script to output the result in degrees and not radians.

User input:

Explanation:

For calculating theta 2, the program takes 2 user inputs. the value of n2, and theta 1.

Code:

```
mat = ['air','diamond', 'ice','glass', 'water']
reflective_index = {
  'air': 1.00,
  'water': 1.33,
  'ice': 1.30,
  'glass': 1.49,
  'diamond': 2.42}
dropdown = OptionMenu(ws,n2_in,*mat,command=calc_theta_2)
theta_1_in = Scale(ws, from_=0.00, to=90.00, length = widthx/2, orient
= 'horizontal')
n2 = (reflective_index[n2_in.get()])
theta_1 = theta_1_in.get()
```

the `mat` variable is a list of items in the dropdown menu. When the item is selected in the drop down menu, it is stored as the variable `n2_in` . Then the n2 variable equals the assigned value of the key from the `reflective_index` dictionary. the angle of incidence is set by the `tkinter .Scale()` .

Displaying the graph:

Explanation:

To Display the graph, first the x and y axis is drawn on setup. Then the program draws the incident ray, then the reference incident ray, then the reflected ray, then the refracted ray. To graph the lines at the correct angle, the program runs a script to convert a polar coordinate pair to a cartesian coordinate pair.

Code:

```
angle_in_radians_re = (theta_1_fl+90) * np.pi/180
line_length = 250
center_x = 0
center_y = 0
end_x_re = center_x + line_length * np.cos(angle_in_radians_re)
end_y_re = center_y + line_length * np.sin(angle_in_radians_re)

c.create_line(0,0,-end_x_re,-end_y_re, fill='red',width=3, tags='ref')
c.create_line(0,0,end_x_de,end_y_de, fill='red',width=3, tags='ref')
c.create_line(0,0,end_x_re,-end_y_re, fill='green',width=3, tags='ref')
c.create_line(0,0,-end_x_re,end_y_re, dash=(10,5), fill=('green'),
width=3, tags='ref')
```

The first line of code converts the `theta_1` variable from degrees to radians. 5th and 6th line calculates the x and y coordinates. Then script script draws lines.

Code: https://github.com/MuteKnowLESS/Light-Sim-V4.git