ZC - University of Science and Technology
Communications & Information Engineering Program
CSCI 101: Introduction to Computer Science

Spring 2023
Team: **ReflectoRay**

# Project Report

Team: **ReflectoRay**
Team Members:

| | | |
|---|---|---|
| **202201079** | **SalahDin Ahmed Salh Rezk** | **s-salahdin.rezk@zewailcity.edu.eg** |
| **202201293** | **Ahmed Muhammad Abdullah** | **s-ahmed.abdullah@zewailcity.edu.eg** |
| **202201517** | **Salah Mahmoud Gamal** | **s-salah.gamal@zewailcity.edu.eg** |

Team Contact: **s-salahdin.rezk@zewailcity.edu.eg**

## Contents

## List of Figures

**Abstract**

The Ray Reflection Simulation addresses challenges in comprehending the principles of ray reflection in geometric optics. Developed in Python using the Turtle graphics library, ReflectoRay offers an interactive and configurable platform for users to visualize and experiment with real-time reflections off mirrors. Through a JSON file, users can specify initial conditions, including mirror positions, source locations, and ray angles. The project's significance lies in its potential to enhance the learning experience, engage users in dynamic simulations, and contribute practically to the understanding of geometric optics. ReflectoRay allows users to save simulations as images or videos, fostering documentation and collaborative learning, making it a valuable tool for educational purposes.

# 1 Problem Description

## 1.1 Background

Understanding the principles of ray reflection is crucial in the field of geometric optics. Students and enthusiasts often face challenges in visualizing and comprehending the complex interactions between rays of light and reflective surfaces. Traditional teaching methods, such as diagrams on paper, may not effectively convey these concepts in an engaging and interactive manner.

## 1.2 Challenges

The challenges include:

1. **Lack of Interactive Tools:** Existing educational resources may lack interactive simulations that allow users to actively engage with the principles of ray reflection.
2. **Difficulty in Visualization:** Visualizing the reflection of rays off mirrors can be challenging without dynamic and interactive simulations. Static diagrams may not adequately represent the real-time behavior of light rays.
3. **Configurability Limitations:** Many available simulations may not offer sufficient configurability options, limiting users' ability to explore different scenarios and initial conditions.

## 1.3 Project Rationale

The Ray Reflection Simulation project aims to address these challenges by providing an interactive and configurable tool for simulating the reflection of rays off mirrors. The simulation will utilize the Turtle graphics library to create a dynamic and visually appealing environment, allowing users to experiment with various configurations and observe the outcomes in real-time.

## 1.4 Significance

This project is significant for the following reasons:

- **Educational Impact:** Enhancing the understanding of ray reflection principles through interactive simulations can improve learning outcomes in the field of geometric optics.
- **User Engagement:** The project's focus on user interaction and configurability aims to make the learning experience more engaging and enjoyable for students and enthusiasts.
- **Practical Application:** The simulation provides a practical tool for users to experiment with different scenarios, fostering a deeper understanding of geometric optics concepts.

# 2 Solution Description

## 2.1 Overview

The Ray Reflection Simulation project proposes a comprehensive solution to the challenges identified in the problem description. By leveraging the capabilities of the Turtle graphics library in Python, the project aims to provide an interactive and configurable platform for simulating the reflection of rays off mirrors.

## 2.2 Key Features of the Solution

The proposed solution includes the following key features:

- **Interactive Visualization:** The use of the Turtle graphics library enables the creation of an interactive and dynamic simulation environment, allowing users to observe the real-time reflection of rays.

- **Configurability:** Users have the flexibility to define the initial conditions of the simulation through a JSON file. This includes specifying mirror positions, source locations, initial angles of rays, and other simulation parameters.
- **Visualization Enhancements:** The simulation incorporates features such as different colors for rays, graphical representation of mirrors, and dynamic updates to enhance the visual representation and user experience.
- **Image and Video Output:** Users can choose to save the simulation as a static image (PNG) or as a video (MP4) using OpenCV, allowing for documentation and further analysis.
- **Progress Visualization:** The Rich library is employed to display a progress bar during the simulation, providing feedback on the simulation's progress and duration.

### 2.3 Advantages of the Solution

The proposed solution offers several advantages:

- **Enhanced Learning Experience:** The interactive nature of the simulation provides users with a hands-on and engaging learning experience, promoting a deeper understanding of ray reflection principles.
- **Versatility:** The configurability of the simulation allows users to explore a wide range of scenarios, facilitating experimentation and exploration of different aspects of geometric optics.
- **Documentation and Sharing:** The ability to save simulations as images or videos enables users to document their experiments and share their findings with others, contributing to collaborative learning.
- **Real-time Feedback:** The dynamic visualization and progress bar provide real-time feedback, allowing users to monitor the simulation's progress and make observations as the rays interact with mirrors.

### 2.4 Expected Outcomes

The successful implementation of the proposed solution is expected to result in an educational tool that not only addresses the challenges associated with understanding ray reflection but also provides an enjoyable and informative platform for users to explore the fascinating world of geometric optics.

## 3 Work Distribution

| Name | Input | Return | Description | Member |
|---|---|---|---|---|
| `parse_arguments` | None | `argparse.Namespace` | Parse command line arguments. | Ahmed |
| `load_initial_conditions` | `file_path: str` | `dict` | Load initial conditions from a JSON file. | Ahmed |
| `setup_screen` | None | `turtle.Screen` | Set up the turtle screen for the simulation. | Salah |
| `draw_mirrors` | `mirrors: list` | None | Draw mirrors on the turtle screen. | Salah |
| `create_ray` | `angle: int,` `start: tuple,` `color: str` | `turtle.Turtle` | Create a turtle object representing a ray. | Salah |
| `create_rays_from_sources` | `angles: list,` `sources: list` | `list` | Create rays from the sources. | Salah |

| | | | | |
|---|---|---|---|---|
| `distance` | `point: tuple, line_start: tuple, line_end: tuple` | `float` | Calculate the distance between a point and a line. | Salah |
| `reflect_ray` | `incident_angle: float, line_start: tuple, line_end: tuple` | `float` | Calculate the reflection angle based on incident angle and mirror orientation. | SalahDin |
| `extend_ray` | `ray: turtle.Turtle` | None | Extend the ray to simulate reflection. | SalahDin |
| `setup_simulation` | `mirrors: list, sources: list, angles: list` | `turtle.Screen, list` | Set up the simulation with mirrors, sources, and angles. | SalahDin |
| `simulate_rays` | `rays: list, mirrors: list` | None | Simulate the reflection of rays off mirrors. | SalahDin |
| `run_simulation` | `screen: turtle.Screen, rays: list, mirrors: list, iterations: int, video: str, tmp: TemporaryDirectory` | None | Run the simulation with progress tracking and optional video recording. | SalahDin |
| `save_image` | `screen: turtle.Screen, output: str` | None | Save the screen as a PNG image. | Ahmed |
| `convert_eps_to_png` | `folder: str` | None | Convert EPS images in the input folder to PNG. | Ahmed |
| `save_video` | `folder: str, output: str, fps: int, codec: str` | None | Save images in the input folder as a video. | Ahmed |
| `save_output` | `screen: turtle.Screen, image: str, video: str, tmp: TemporaryDirectory` | None | Save the output as an image or video. | Ahmed |
| `main` | None | None | Main function to run the ray reflection simulation. | Salah |

Table 1: Functions Implementation

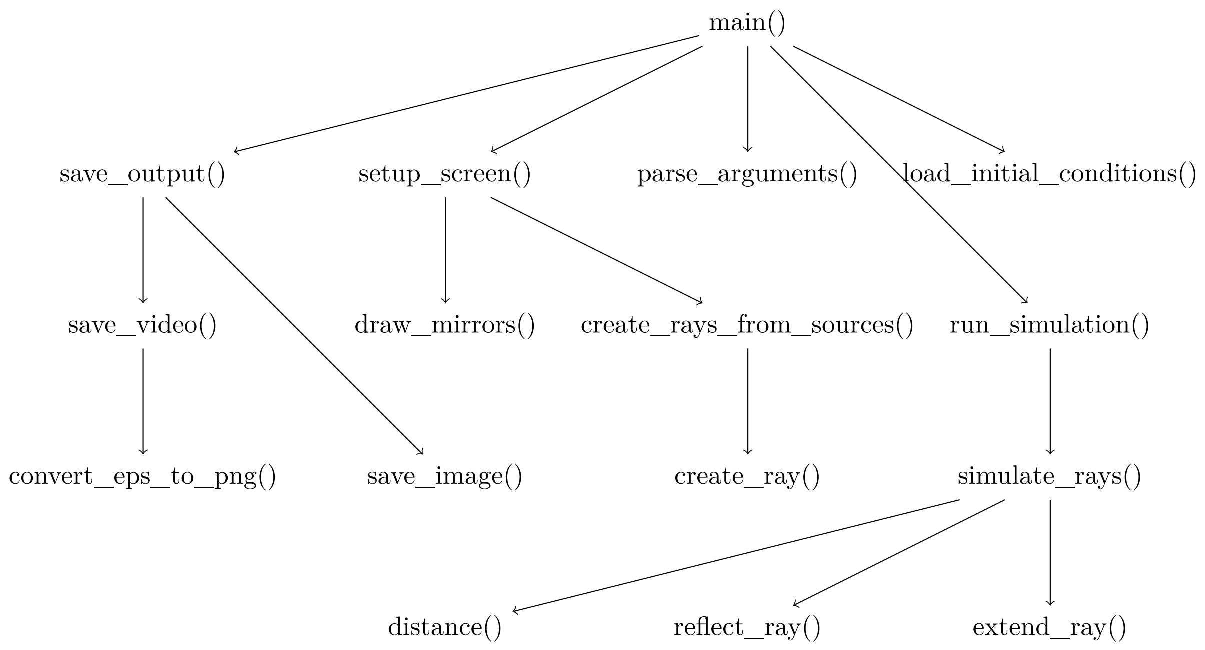

Figure 1: Flow Chart of Functions

## 4 Usage Manual

### 4.1 System Requirements

Before using the Ray Reflection Simulation, ensure that your system meets the following requirements:

- Python 3.x installed
- Required Python libraries: Turtle, OpenCV, PIL, Rich

### 4.2 Installation

#### 4.2.1 Installing Python

If Python is not already installed on your system, you can download it from the official Python website (`https://www.python.org/downloads/`).

#### 4.2.2 Installing Required Libraries

Install the necessary Python libraries using the following command:

```
pip install turtle opencv-python pillow rich
```

### 4.3 Running the Simulation

#### 4.3.1 Clone the Repository

Clone the Ray Reflection Simulation repository from the project's GitHub page:

```
git clone https://github.com/salastro/reflectoray
```

#### 4.3.2 Navigate to the Project Directory

Enter the project directory:

```
cd ReflectoRay
```

### 4.3.3 Configuring the Simulation

Edit the configuration JSON file (‘initial_conditions.json‘) to define the initial conditions of the simulation. Specify mirror positions, source locations, initial angles of rays, and other parameters as needed.

The configuration file is structured as follows:

```
{
  "mirrors": [
    [[100, -100], [100, 100]],
    [[100, 100], [-100, 100]]
  ],
  "sources": [
    {"start": [0, 0], "color": "black"},
    {"start": [-50, 50], "color": "red"}
  ],
  "angles": {
    "start": 0,
    "end": 360,
    "step": 10
  },
  "iterations": 100
}
```

The ‘mirrors‘ array contains the coordinates of the endpoints of each mirror. Each mirror is represented as a list of two points, where each point is a list of two numbers representing the $x$ and $y$ coordinates of the point.

The ‘sources‘ array contains the coordinates of the starting points of each ray. Each ray is represented as a dictionary with two keys: ‘start‘ and ‘color‘. The ‘start‘ key contains a list of two numbers representing the $x$ and $y$ coordinates of the starting point of the ray. The ‘color‘ key contains a string representing the color of the ray, which can be any well-known color name or a hexadecimal color code.

The ‘angles‘ dictionary contains the starting angle, ending angle, and step size for the rays. The ‘start‘ key contains a number representing the starting angle of the rays. The ‘end‘ key contains a number representing the ending angle of the rays. The ‘step‘ key contains a number representing the step size between each ray.

The ‘iterations‘ key contains a number representing the number of iterations to run the simulation for.

### 4.3.4 Run the Simulation

Execute the following command to run the simulation:

```
python reflectoray.py
```

Help information can be displayed by passing the ‘–help‘ flag to the command:

```
python reflectoray.py --help
```

which will display the following output:

```
usage: reflectoray.py [-h] [-i [IMAGE]] [-v [VIDEO]] [initial_conditions]

Ray Reflection Simulation

positional arguments:
  initial_conditions    Path to the initial conditions file (default:
                        initial_conditions.json)

options:
  -h, --help            show this help message and exit
  -i [IMAGE], --image [IMAGE]
                        Save the simulation as a png image
  -v [VIDEO], --video [VIDEO]
                        Record the simulation as a video
```

### 4.3.5 Save the Simulation

The simulation can be saved as a static image (PNG) or as a video (MP4). To save the simulation as an image, pass the '–image' flag to the command:

```
python reflectoray.py --image
```

To save the simulation as a video, pass the '–video' flag to the command:

```
python reflectoray.py --video
```

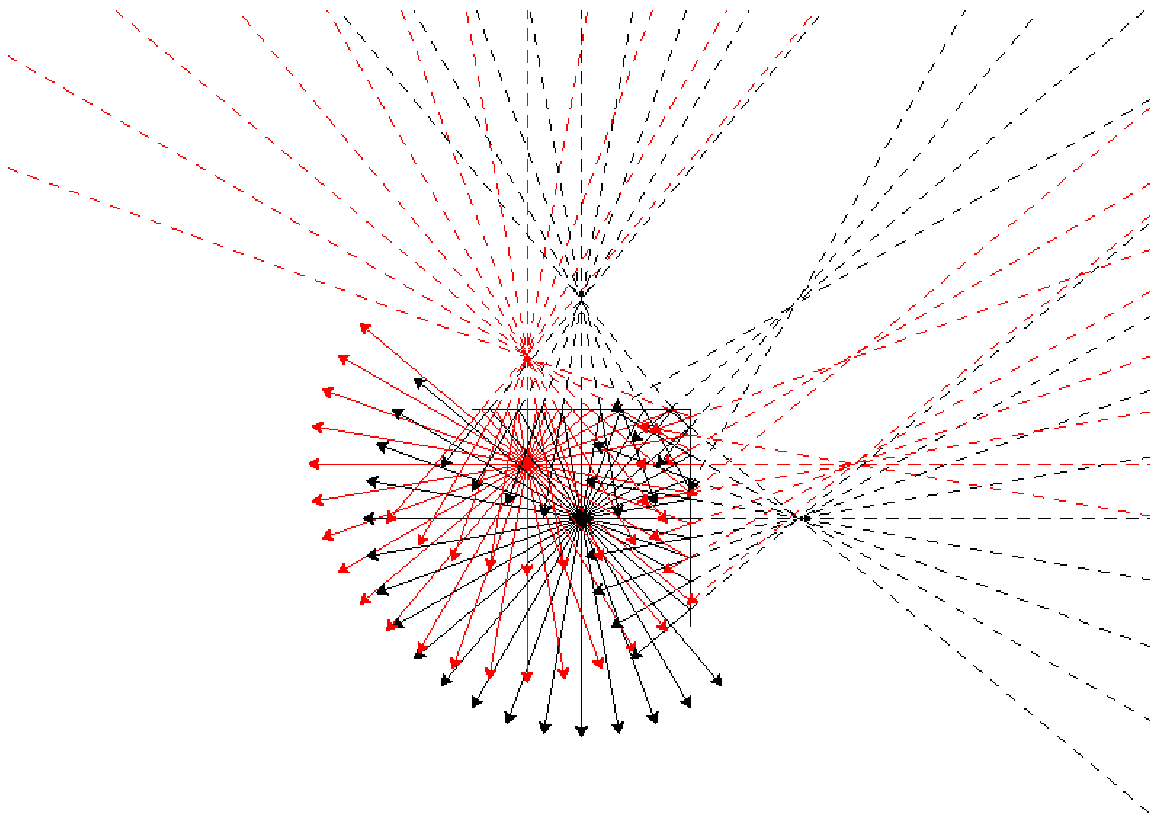

Figure 2: Simulation Image Output (of the default initial conditions)