

Vikram Singh & Christopher Son

Professor Shaffer

CS 396

4 December 2025

Ballistics Trajectory Solutions

Introduction

This project implements a multi-language ballistics trajectory system that calculates firing angles, generates projectile motion data, validates the solution through logical rules, and produces visualizations of the resulting trajectory. The software models simplified projectile motion assuming no drag and constant gravity. Users select a cartridge and then input a target distance. The program then determines the firing angle (low-angle solution), calculates the trajectory points, and writes the results to two files:

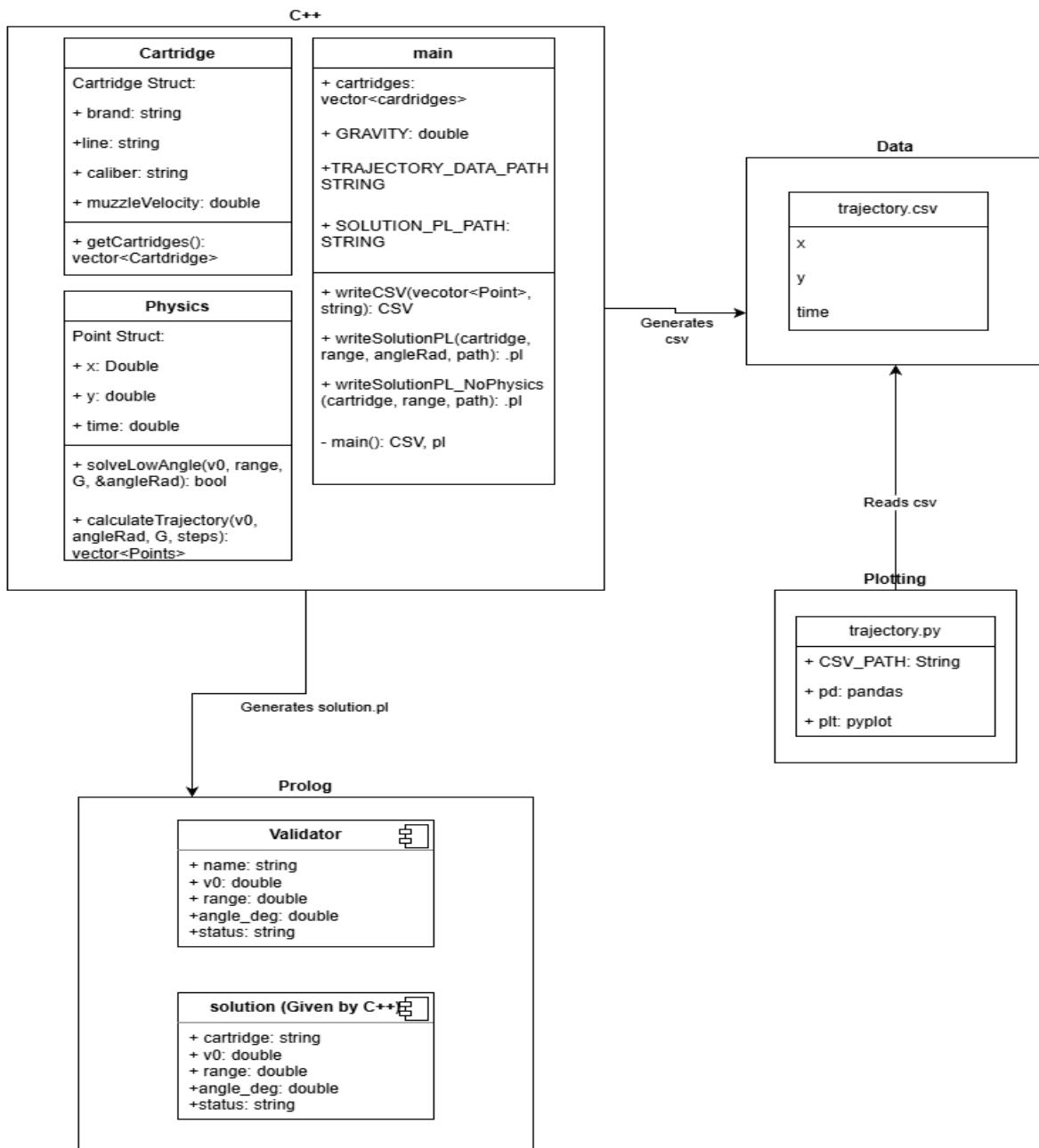
1. **trajectory.csv** – a dataset of sampled projectile coordinates over time

2. **solution.pl** – a Prolog fact representing the firing solution, or a placeholder structure if no valid angle exists.

C++ performs the numerical computation, Prolog validates the solution according to defined logical rules, and Python visualizes the projectile path using matplotlib.

Architecture

Our program follows a basic pipeline architecture with modular components and file-based integration. Five different modules (main, physics, cartridges, plotting, validator) are implemented that each own a certain responsibility. From main.cpp, the program takes user input, an angle is found, and the data is placed into a CSV and a prolog file, which is then (optionally) visualized by Matplotlib and Python ([trajectorysolution.py](#)).



Paradigm Integration

Different languages are used for different paradigms. Here, we use an OOP language (C++) to handle computations, physics, and data & file generation. The computations are then given to a logical programming language (Prolog), where they are validated and an output is produced. Finally, another OOP language (Python) is used to handle the visualization component.

Furthermore, the modules do not communicate directly, instead they communicate from the use of files. (`trajectory.csv` and [solution.pl](#)). These components are loosely coupled, meaning they come from different computations; no shared runtime and no linking between languages.

Motivation

This project comes from the need for sports shooting and hunting. Shooters from many fields and occupations oftentimes need to quickly calculate the firing angle to hit their target. To determine the angle of launch requires that the shooter know the range of the target and the velocity of their cartridge. Utilizing these two known variables, we can easily determine the angle needed for the shooter to take to confidently hit the target. This solution will make it easier for shooters to have a successful hunt or score the most points in a competition.

Conclusion

In conclusion, this project helped demonstrate how multiple programming paradigms can be used together to solve a unified computational problem. Through our

implementations of C++, Prolog, and Python, we were able to show how data can be exchanged across paradigms through file based communication instead of relying on a shared language. Each language helped to contribute to the overall program, each with their own defined responsibilities. This project helped us understand how the principles of programming languages we've studied this semester can interoperate effectively and efficiently to create a unique software system to solve real world needs like calculating the correct firing angle from known values like range and cartridge velocity.