

# Particle Collider Simulation

## Introduction

You will be working in groups of **6 students** to develop a **Particle Collider Simulation** using Python. The project simulates collisions similar to those conducted at CERN, where particles collide at high speeds to explore fundamental physics.

The objective is to apply your **programming skills, OOP, data analysis, and visualization** techniques to create a scientifically-inspired simulation.

## Project Overview

Each group will independently develop the same simulation project. You will simulate particles colliding, transforming into new particles, and analyze the resulting data to extract insights.

## Keywords

**particle collisions**, momentum conservation, and energy transformations, **OOP**, **NumPy**, **Matplotlib**, and **SciPy** for simulations, data analysis, visualizations, **data logging and analysis**.

- You must use **Python** and **Jupyter Notebook**.
- Apply **Object-Oriented Programming (OOP)** principles.
- Include **3D visualization** using Matplotlib.
- Use **GitHub** for version control and collaboration.
- Submit a final **report and presentation** along with your code.

## Project Breakdown

Each group has **6 members**. Below are the detailed tasks, divided among team members.

### Task 1: Particle Class Development

**Assigned To:** Student 1

- Create a Particle class with:
  - Attributes: position, velocity, mass, energy, type (e.g., proton, meson, lepton, electron, muon, etc.).
  - Methods: Update position, calculate kinetic energy, detect decay, and display particle information.

- Implement **particle decay**, where high-energy particles may spontaneously transform into other particles over time.

## Task 2: Collider Class & Particle Transformations

**Assigned To:** Student 2

- Develop a Collider class to:
  - Detect collisions based on proximity.
  - Apply momentum conservation during collisions.
  - Implement **stochastic particle transformations (Monte Carlo Simulations)**. For example, after a collision, particles may split into multiple new particles with specific probabilities.
- Introduce **fusion events**: if two particles collide with enough energy, they can merge into a new, heavier particle.

## Task 3: Simulation Controller

**Assigned To:** Student 3

- Develop a Simulation class to manage the entire simulation process:
  - Initialize particles with random positions, velocities, and types.
  - Control the simulation loop with adjustable time steps.
  - Allow users to start, pause, and reset the simulation.
  - Implement a **GUI interface** using libraries like tkinter to provide user control over the simulation parameters (e.g., particle count, speed).

## Task 4: 3D Visualization & Real-Time Animation

**Assigned To:** Student 4

- Create a Visualizer class to:
  - Use Matplotlib or Plotly to display particle movements in 3D.
  - Animate collisions, particle transformations, and decay in real-time.
  - Implement color coding for different particle types and transformations.
  - Add a **heatmap** overlay to show regions with the highest collision frequencies.

## Task 5: Data Logging, Analysis, and Reporting

**Assigned To:** Student 5

- Develop a DataLogger class to:
  - Record collision events, particle transformations, decay events, and fusion outcomes.
  - Save data in a structured format (CSV or JSON).

- Write scripts to analyze the results and generate summary reports, including charts and graphs.
- Implement machine learning algorithms to predict the types of particles that may form based on collision data.

## Task 6: Performance Optimization and Parallel Computing

**Assigned To:** Student 6

- Optimize the code for better performance, focusing on **speed and memory usage**.
- Implement **parallel processing** using Python's multiprocessing library to handle large simulations with thousands of particles.
- Write unit tests for each module to ensure correctness and stability.
- Use tools like cProfile to profile the code and identify bottlenecks.

## Project Milestones

- Initial research, team setup, and GitHub repository creation.
- Complete class structures for Particle, Collider, and Simulation.
- Finalize visualization, data logging, and analysis.
- Complete GUI interface, optimization, and performance testing.
- Polish documentation and prepare the final presentation.
- Submit the project and deliver your group presentation.

## Assessment Criteria

Your project will be evaluated based on the following criteria:

1. **Correctness & Complexity** (30%): Does your simulation correctly model particle collisions, transformations, and decay?
2. **Visualization & User Interface** (20%): Are your visualizations clear and engaging? Does your GUI enhance usability?
3. **Data Analysis & Reporting** (20%): How well have you analyzed the results, and are your reports insightful?
4. **Teamwork & Documentation** (30%): Is your project well-documented, and was the work evenly distributed among team members?

## Submission

- Submit your final project on **GitHub** and share the repository link.
- Include a **Jupyter Notebook** with all your code, visualizations, and explanations.
- Prepare a **10-minute presentation** to showcase your project, focusing on your approach, challenges, and results.

