Google Summer of Code 2023

PyElastica - Extending contact module capabilities

About Me

Name: Rahul Joon

Github: Rahul-JOON

Email: rahuljoon16@gmail.com

Linkedin: rahul-joon

University: USIC&T, Guru Gobind Singh Indraprastha University

Program: Bachelor in Technology in Electronics and Communication Engineering

Graduation Year: 2025

Location: India

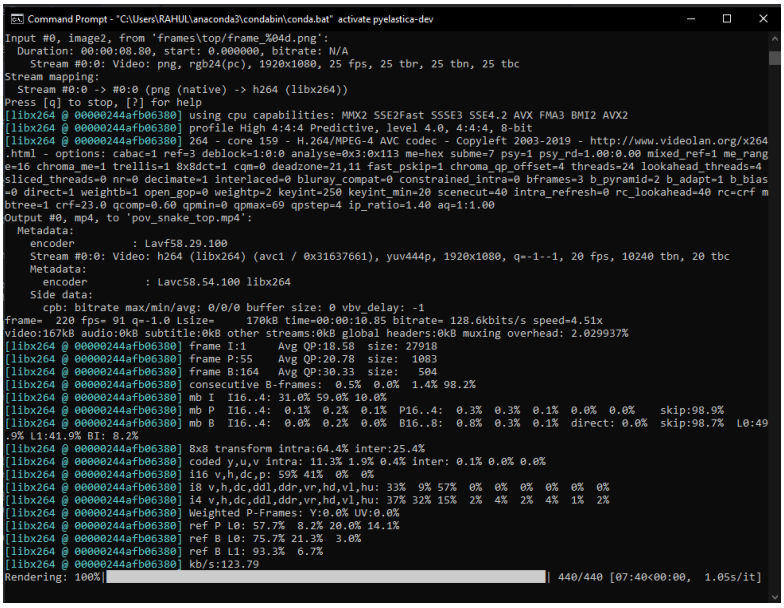
Timezone: Indian Standard Time (GMT +5:30)

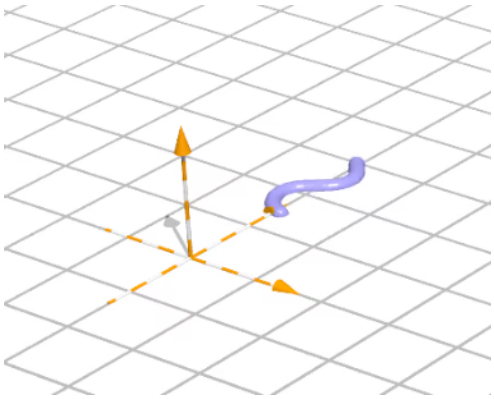
Resume: resume

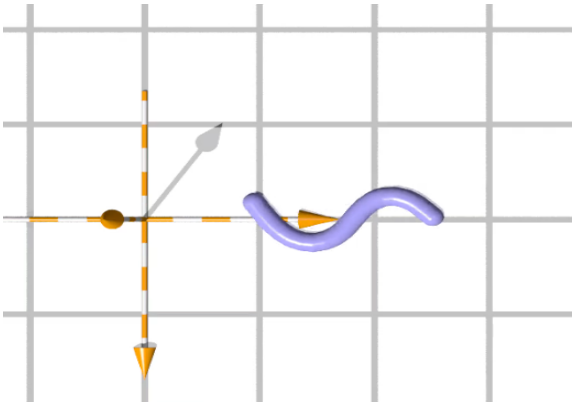
I have been studying computer science from 11th grade of school and my first/core programming language is python**(**>10k lines of code experience**)**. I have significant interest in applied physics and had been contributing to pyElastica since this year only. I am thrilled to work on this project and learn along with the community.

Code Contribution

- **Solved Issue - #169; PR - #232.**
  - **Problem - The continuum snake visualization example was previously using moviepy for plotting the final result, but 'moviepy' was relatively slow and non-uniform among other examples.**
  - **Solution - Moviepy was replaced with 'ffmpeg' for video generation which is faster and uniform among other examples.**





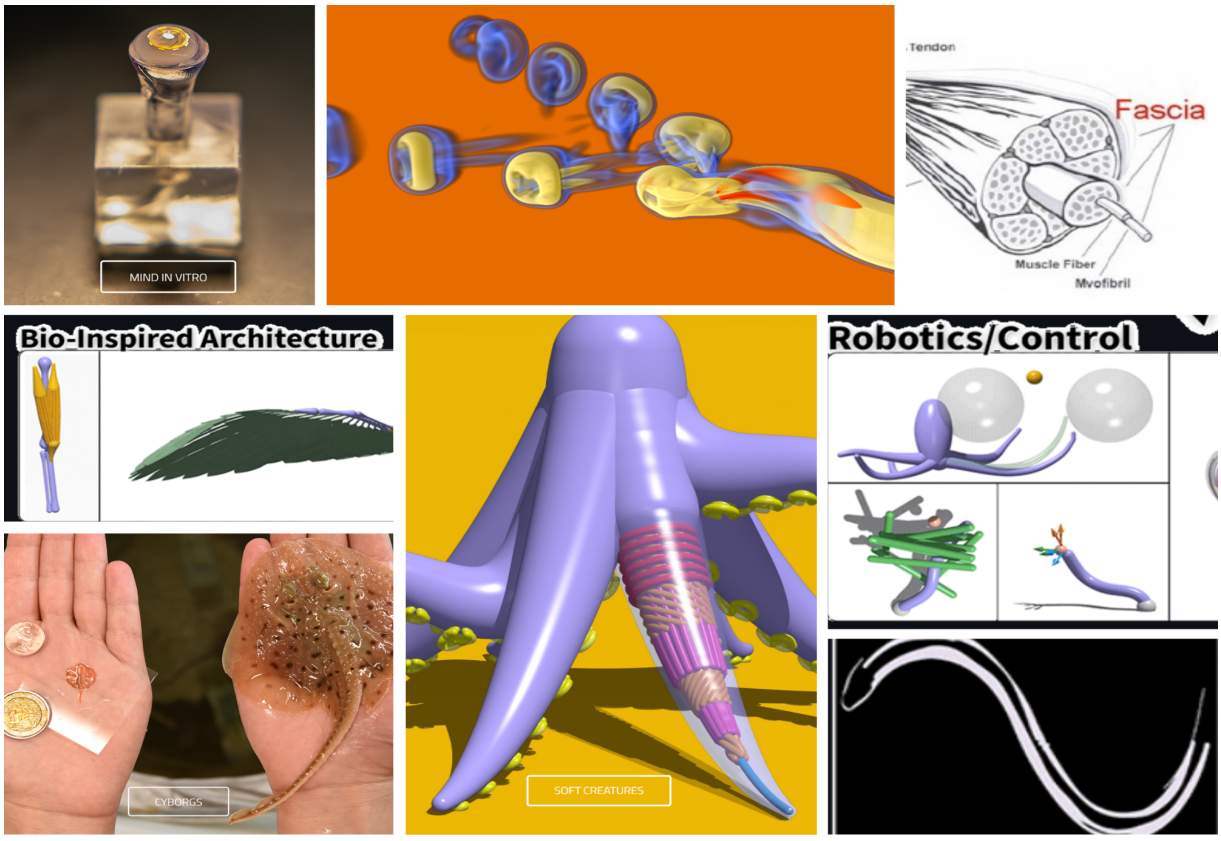


About the Organisation

Sub-org Name: PyElastica

PyElastica is the Python version of **Elastica**, a project that uses Cosserat Rod theory to simulate assemblies of thin one-dimensional bodies. Elastica is free and open-source software. It is simple to use, extensible, and modular in design. It enables the user to specify a set of Cosserat rods that are susceptible to both internal and external (such as muscle torque, gravity, friction, etc…) forces. Rods can be combined to form assemblies of rods, which can then be used to model progressively more complex systems. Rods also take into consideration self-contact.

The Gazzola Lab at the University of Illinois at Urbana-Champaign created and maintains Elastica. Visit https://mattia-lab.com for more details about the projects



.

Project Information

Abstract

Currently, in PyElastica we can model the contact of rods with a frictional plane or rigid cylinder and themselves.

Compatibility

Forcing	Rod	Rigid Body
NoForces	✅	✅
EndpointForces	✅	❌
GravityForces	✅	✅
UniformForces	✅	✅
UniformTorques	✅	✅
MuscleTorques	✅	❌
EndpointForcesSinusoidal	✅	❌
Interaction	**Rod**	**Rigid Body**
AnisotropicFrictionalPlane	✅	❌
InteractionPlane	✅	❌
SlenderBodyTheory	✅	❌

This project aims to extend the contact module for the rod and any arbitrary shape. Implementation of the contact module that can input any STL mesh and create its geometry in PyElastica and also apply contact force between the rod and imported shape. This will improve the capabilities of PyElastica as a whole.

Skills Required: Python

Mentors: Arman and Noel

Expected size of Project: 350 hours

Following is the overview of the tasks of this project:

- Develop an algorithm to import STL mesh and convert it into PyElastica Geometry.
- Develop a library that implements contact forces between rod and imported shape.
- Validate implemented features/craft examples.
- Document implemented work.

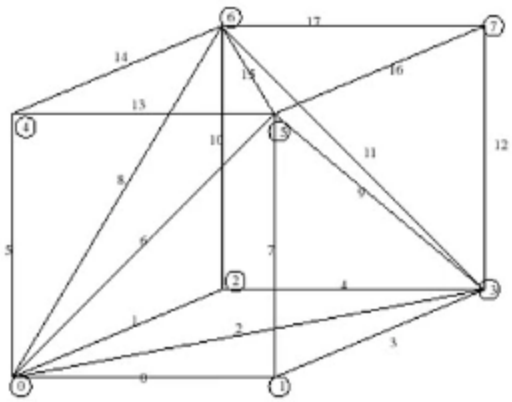
Description

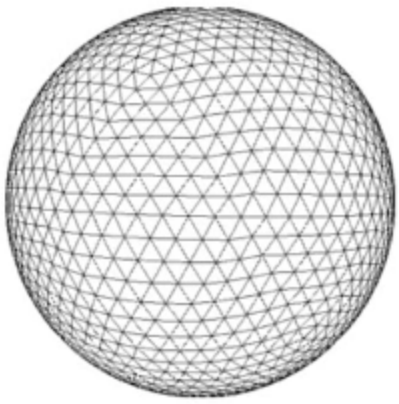
Algorithm to convert STL Mesh to PyElastica Geometry

An **STL** file is a file format used for 3D printing and computer aided design(CAD). The name STL is an acronym that stands for stereolithography - a popular 3D printing technology; it also stands for **"**Standard Triangle Language**"**. This file format is widely use for rapid prototyping, 3D printing and computer manufacturing.

The main purpose of the STL file format is to encode the surface geometry of a 3D object. It encodes this information using a simple concept called "tessellation" that is the process of tiling a surfacewith on or more geometric shapes such that there are no overlaps or gaps.

For example, below is a STL mesh of a 3D model of a cube and sphere.





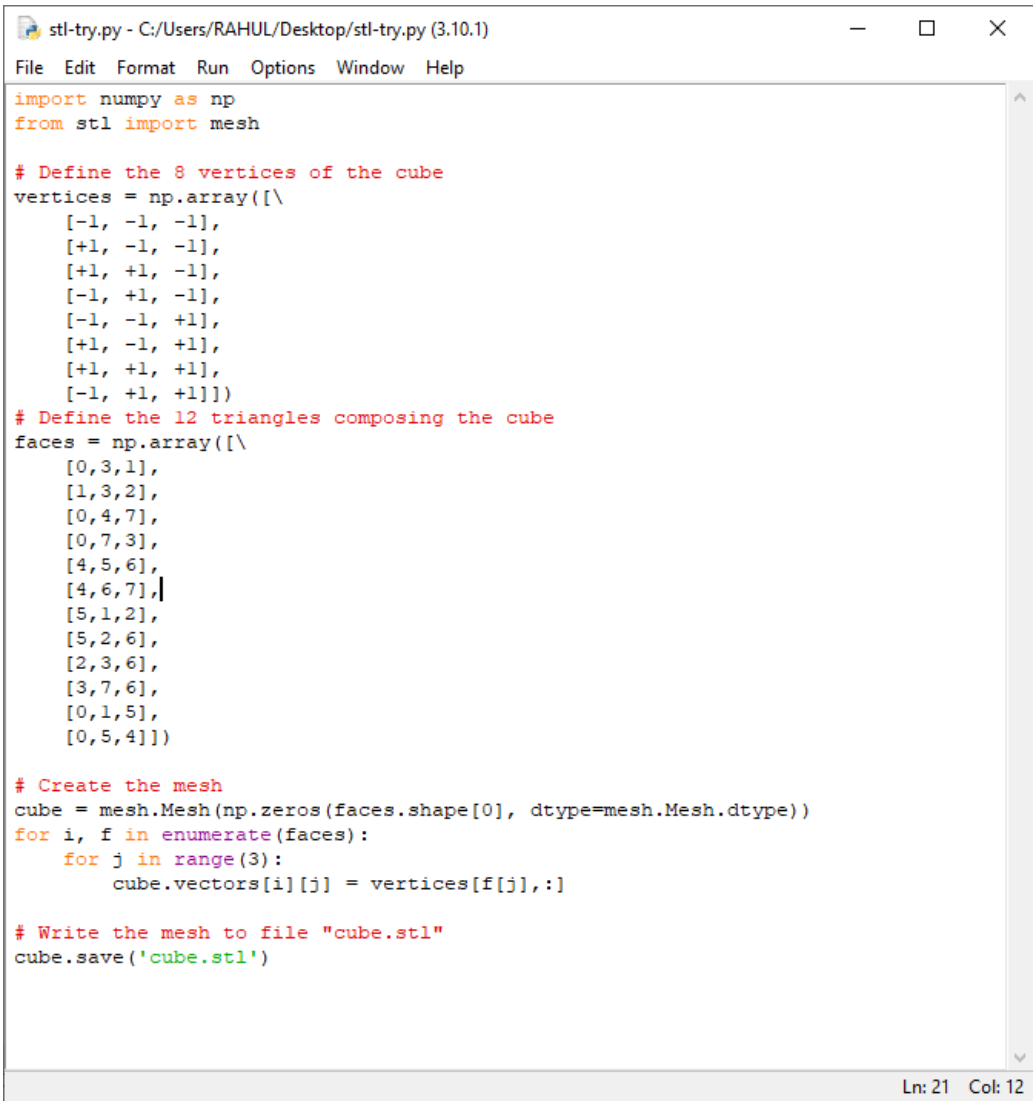
More info about STL formatting can be found here.

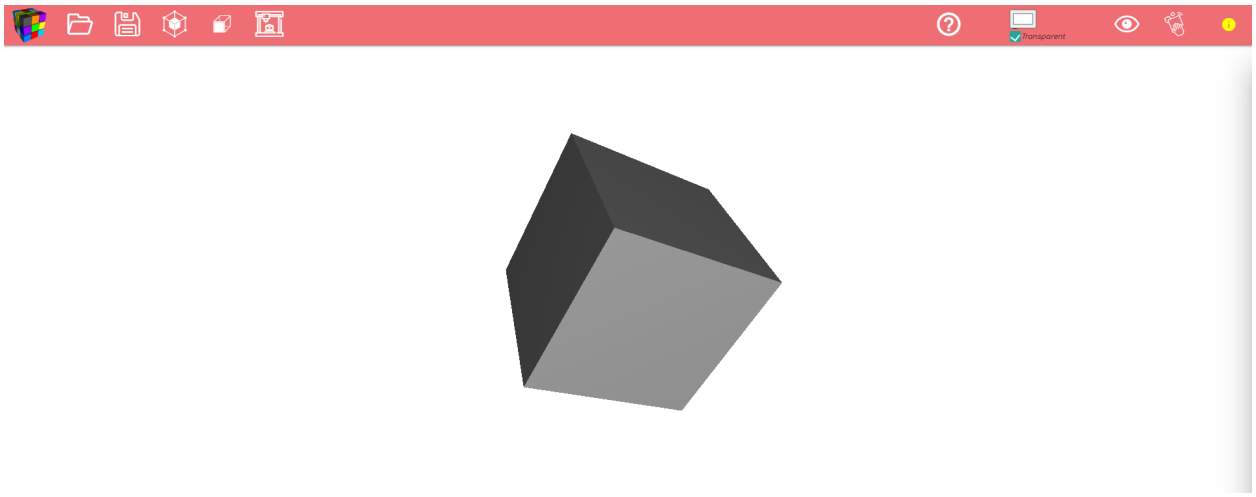
STL Mesh Data Manipulation tool

We can extract the data stored in an STL file using the 'numpy-stl' python package. *'Numpy-stl'* is a simple library to make working with STL files fast and easy. Due to all operation heavily relying on numpy this is one of the fastest STL editing libraries for Python available.

With *'numpy-stl'*, we can create, plot, edit existing STL meshes.

For example, code snippet provided below creates a simple cube (displayed *below the snippet*),





More examples of code snippets modifying STL meshes are given at [numpy-documentation].

Projection

Numpy-stl also provides support with '**Matpotlib**' for plotting/visualization; but our aim here is to develop the said object in PyElastica Geometry. The required parameters for defining a rod in PyElastica are:

Cosserat Rod

	On Nodes (+1)	On Elements (n_elements)	On Voronoi (-1)
Geometry	position	director, tangents length, rest_length radius volume dilatation	rest voronoi length voronoi dilatation

While STL meshes store the data about solids as surfaces, it doesn't store data about the texture or other properties of solid like dilation etc.

We can let this data be set to some default values and an option to change as per need can be provided to the end user.

For **Rigid bodies** though, the parameters required are,

Rigid Body

type
Cylinder
Sphere

class elastica.rigidbody.rigid_body.RigidBodyBase [source]

Base class for rigid body classes.

Notes

All rigid body class should inherit this base class.

compute_position_center_of_mass() [source]

Return positional center of mass

compute_translational_energy() [source]

Return translational energy

compute_rotational_energy() [source]

Return rotational energy

class elastica.rigidbody.cylinder.Cylinder(*start, direction, normal, base_length, base_radius, density*) [source]

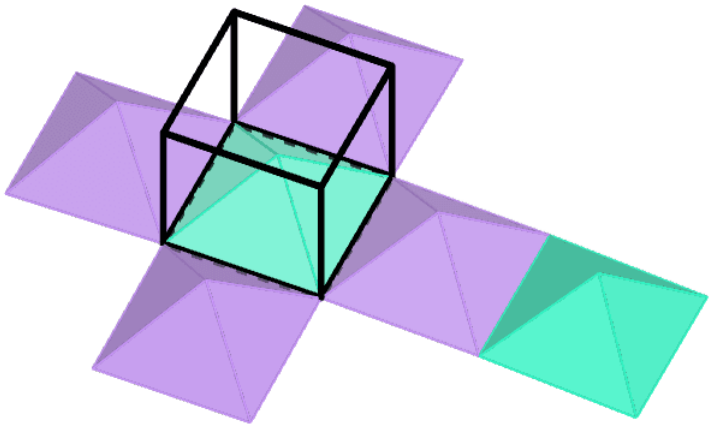
class elastica.rigidbody.sphere.Sphere(*center, base_radius, density*) [source]

We can calculate the *base_radius, base_length* etc from the STL mesh file by using numpy-stl.

Complex shapes from STL Mesh

Developing arbitrary or complex shapes in PyElastica is a different task on its own. A Starting approach could be to build geometric complex uniform obhjects from existing basic shapes like sphere, surface, rods etc.

For example, a ‘hollow’ cube or cuboid can be constructed by joining 6 planes from edge to edge as shown below.



The contact forces can then be applied at all joint faces and a hollow cube/cuboid and more uniform hollow geometries can be achieved likewise.

Though this is just a starting point, more research and brainstorming needs to done to achieve the ultimate goal.

Contact forces between rod and imported shape

The contact forces include frictional, gravity, torques etc; which have already been gracefully implemented in PyElastica among known shapes like rods, rigid cylinders etc.

Compatibility

Forcing	Rod	Rigid Body
NoForces	✅	✅
EndpointForces	✅	❌
GravityForces	✅	✅
UniformForces	✅	✅
UniformTorques	✅	✅
MuscleTorques	✅	❌
EndpointForcesSinusoidal	✅	❌

Interaction	Rod	Rigid Body
AnisotropicFrictionalPlane	✅	❌
InteractionPlane	✅	❌
SlenderBodyTheory	✅	❌

If we are to import known shapes, then applying forces will not be a difficult task.

But for any arbitrary shape, the collision forces have not yet been defined. This is also a task to brainstorm about with the help of Gazzola RSOS 2018.

Validation/Example Cases for New Users

This is relatively easy part; we simply can make a rod and temporary('temp') STL mesh file that contains a model of a uniform cube that collides with relevant contact forces and the result can be plotted in a video with different planes as reference frames.

Video generation will be achieved through using ffmpeg and POVray like in other examples.

Timeline

I am familiar with the codebase, so I'll be looking to start early so that things go smoothly during the coding period.

The whole project can be broadly categorised into milestones and labels:

Milestone 1: STL Mesh file to PyElastica geometry

- Label 1.1: Develop method to import stl file without raising exceptions
- Label 1.2: Develop algorithm to convert data to PyElastica geometry form
- Label 1.3: Returning shape objects to users
- Label 1.4: Documenting functions definitions

Milestone 2: Complex Shape Genisis

- Label 2.1: Discuss the approach/workplan with mentor
- Label 2.2: Implement via code
- Label 2.3: Document the workflow, features added

Milestone 3: Implement contact forces between imported shapes

- Label 3.1: Research and discuss with mentor
- Label 3.2: Implement the final algorithm
- Label 3.3: Document the features added

Milestone 4: Validation/Examples

- Label 4.1: Discuss the best example approach for new users
- Label 4.2: Basic code for collission
- Label 4.3: Output the video files
- Label 4.4: Document instructions and implemented features

Community Bonding Period (May 4 - May 29)	• Interaction with the community. • Discuss the project design. • Modifying approach/rectifying any misunderstanding regarding the project. • Continue to solve bugs that are within my scope. • Documenting work. • Setup blogs for weekly/fortnightly updates.
Week I (May 29 - June 6)	• Method to import STL mesh without raising exceptions (label 1.1). • Testing and documenting.
Week II (June 6 - June 12) - Week III(June 12 - June 19)	• Converting data to PyElastica Geometry(Basic Shapes). (label 1.2) • Testing and Documenting.
Week IV(June 19 - June 26)	• Returning finished outputs to users. (label 1.3) • Testing and Documenting. (Accomplishment of Milestone 1)

Week V (June 26 - July 3)	• Discussing the solution to build complex shapes. (label 2.1) • Blog and documenting.
Week VI (July 3 - July 10)	• Implementing the outcome of discussion or researching more as per need. (lable 2.2) • Documenting. (Accomplishment of milestone 2)
Midterm Evaluation (July 10 - July 14)	• Debugging. • Working on reviews. • Documenting.
Week VII (July 14 - July 21)	• Researching regarding contact forces among arbitrary shapes in physics models. (label 3.1) • Document findings.
Week VIII (July 21 - July 28) - Week IX (July 28 - August 4)	• Implementing contact forces library. (label 3.2) • Documenting. (Accomplishment of milestone 3)
Week X (August 4 - August 11)	• Discussion about example format. (label 4.1) • Implementing example for first time users. (label 4.2) • Output the video files. (label 4.3) • Documenting with instructions for first time users. (Accomplishment of Milestone 4)
Week XI (August 11 - August 18)	• Blog writing all the work done so far. • Buffer week for any backlog.

Final Week (August 21 - August 28)	• Backlog clearance. • Finalising code for submission.
Final Evaluation (August 28 - September 5)	• Wrap up everything. • Work on review(if any).

The provided timeline is tentative and not rigid; if the features call for it, we will gladly expand it to add more features, including the development of more complicated shapes as previously mentioned.

Post GSoC

- If this project continues, I plan to keep contributing everything I can to PyElastica.

Other Commitments

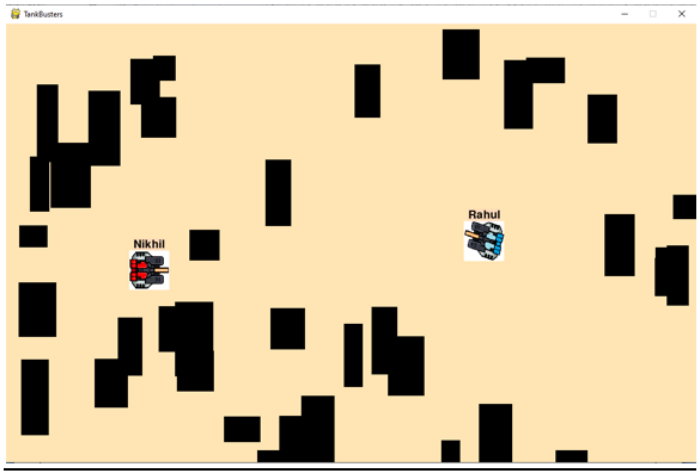
- The GSoc timeline is mostly in sync with my summer break of university thus allowing me ample time to work.
- Though I may have my university end sem exams at the end of July, nevertheless the buffer week can be used to adjust for it.
- I am not applying to any other organizations and have no other obligations during my GSoc.

Why Me

My primary programming language is Python, and long before PyElastica, I worked on projects involving physics models, which are listed below:

- Tank Busters
    - A physics maze simulation game, that implements conservation of momentum for rebounding and rotational mechanics.

Gameplay



- Lost in Space
    - An arcade shooting game in python using the PyGame framework.
    - It forumulates the mechanics of collision.
- Also,
    - I have been studying and implementing computer science since High School and am proficient in Data Structures and Algorithms (qualified ACM-ICPC regionals during high school).
    - I have on-hands project experiences from personal projects to internships at SDC, USICT; ranging from python development to web & app development(GitHub).