Google Summer of Code 2025 Proposal

Project Title: Enhancing the **2D** Regularized Boolean Operation Demo **(Planar Geometry)**

"This project focuses on improving CGAL’s demo for 2D Boolean set operations in the plane, extending support for non-linear curves and adding a scriptable command-line interface."

1 Personal Information

1.1 Contact Information

- **University -** Motilal Nehru National Institute of Technology, Allahabad
- **Email IDs - samimshoaib01@gmail.com**,
- **GitHub/Portfolio -** samimshoaib01
- **Timezone -** IST (UTC + 5:30)

1.2 Personal Background

I am a second year undergraduate in the department of Computer Science and Engineering at Motilal Nehru National Institute of Technology, Allahabad.

I have completed the following relevant courses in my academic curriculum in the past years :Computer Programming, Engineering Mathematics, Computer Organisation and Architecture, Theory of Computation, Compiler Design, Object Oriented Analysis and Design.

1.3 Programming Background

I currently use Ubuntu 24.04.2 LTS as my primary development system, with Visual Studio Code as my main editor, supplemented by experience with Eclipse IDE and Atom. While I have programming experience across multiple languages including Python 3.x, Java, and web technologies (HTML/CSS/JavaScript) from my undergraduate studies, my recent focus has been on C++ development, particularly using Qt framework for GUI applications.My multi-language background helps me approach problems from different perspectives

while maintaining strong C++ fundamentals for this CGAL-related work.

A list of my ongoing and completed projects is as follows :

- **Campus Rush -** I participated in a university hackathon and secured the position of second runner-up. I developed a 2D RPG game using Phaser.js for the game engine and the MERN stack for backend functionalities. The project required efficient optimization and integration between frontend and backend. Competing under strict time constraints, I gained valuable insights into rapid prototyping and problem-solving.

- **Digital Vernier Caliper Simulator (Physics/Engineering Lab) -** I, along with four team members, developed a Digital Vernier Caliper Simulator to assist students in learning precision measurements in physics and engineering labs. The application, built using Qt, provides an interactive interface where users can adjust the caliper jaws using sliders to measure virtual objects with high accuracy. It displays the main scale reading, vernier scale reading, and the final measurement with least count correction. This tool helps students understand the working of a vernier caliper without needing physical instruments, making it useful for remote learning and virtual labs.

- **AI Powered Coding Assistant** - Developed an AI-powered Chrome extension using OpenAI API to assist with coding challenges, featuring a seamless SPA-integrated UI. Optimized AI prompts for accuracy and added chat history storage, enhancing productivity for developers. Designed for scalability

and performance, aligning with MAANG's high standards for user experience

2 The Project

This project focuses on **2D Boolean set operations in the plane**, enhancing the existing CGAL demo to support non-linear curves and a command-line interface.

2.1 Important terms :

- **2D Arrangements :** Geometric arrangements, or arrangements for short, are subdivisions of some space induced by geometric objects. Figure 2a shows an arrangement of two curves c1 and c2 in the plane.

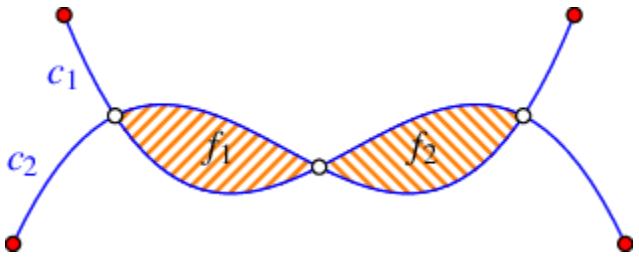


Fig:2a

  - Faces:
    - f_1 : A bounded face (filled with diagonal stripes).
    - f_2 : Another bounded face (filled with diagonal stripes).
  - Vertices:
    - 4 endpoints of $c1$ and $c2$ (drawn as small pieces)
    - 3 intersection points of $c1$ and $c2$ (drawn as small rings)

    (Total: 7 vertices, but only those incident to f_1 and f_2 matter for the faces.)
  - Edges:
    - 8 edges (maximal non-intersecting curve portions)

- **Regularized Boolean Operations :** Regularized Boolean Operations (RBOs) are geometric operations used in computational geometry, typically involving the combination of two or more geometric shapes (often polygons or polyhedra) to

produce new shapes. These operations are "regularized" because they ensure that the result is always a well-defined, valid geometric object, free of ambiguity or non-manifold structures.
The full scope of Regularized Boolean Operations includes the following:

- **Union**: Combines two shapes into a single shape that includes all the points from both.

- **Intersection**: Produces the overlapping region between two shapes.

- **Difference**: Subtracts one shape from another, resulting in the part of the first shape that doesn't intersect with the second.
- **Symmetric Difference**: Includes the areas that are part of either of the two shapes but not both. Essentially, this is the union of the differences between the shapes.
- **Complement**: This operation finds the region outside of a given shape, essentially inverting it. It’s like taking everything in space except the shape itself.

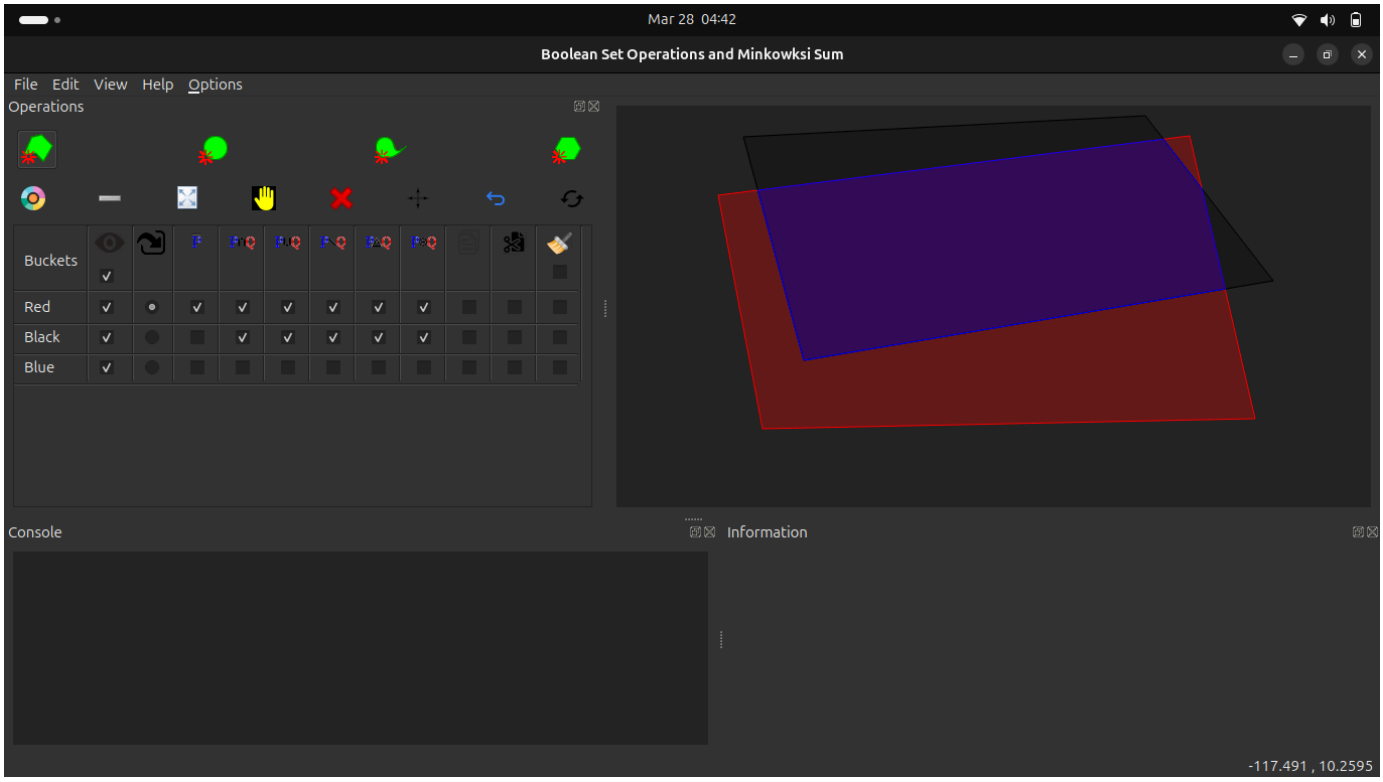
- **Minkowski Sum**: A more advanced operation, the Minkowski sum combines two shapes by adding every point of one shape to every point of the other. For example, adding a polygon with a point, or adding two polygons together, results in a shape that represents the sum of their areas.

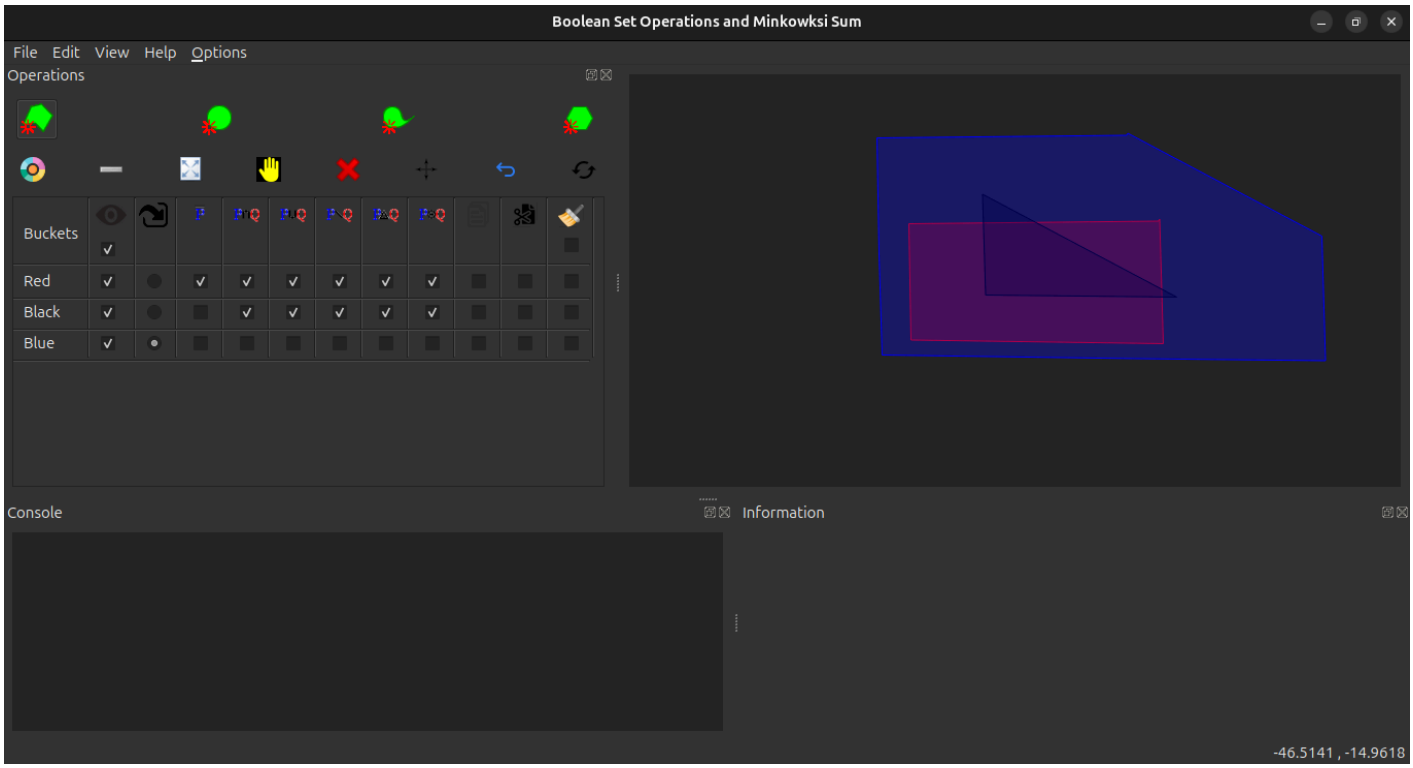
Note: Minkowski sums are only supported for standard polygons (**not non-linear curves**) in CGAL.

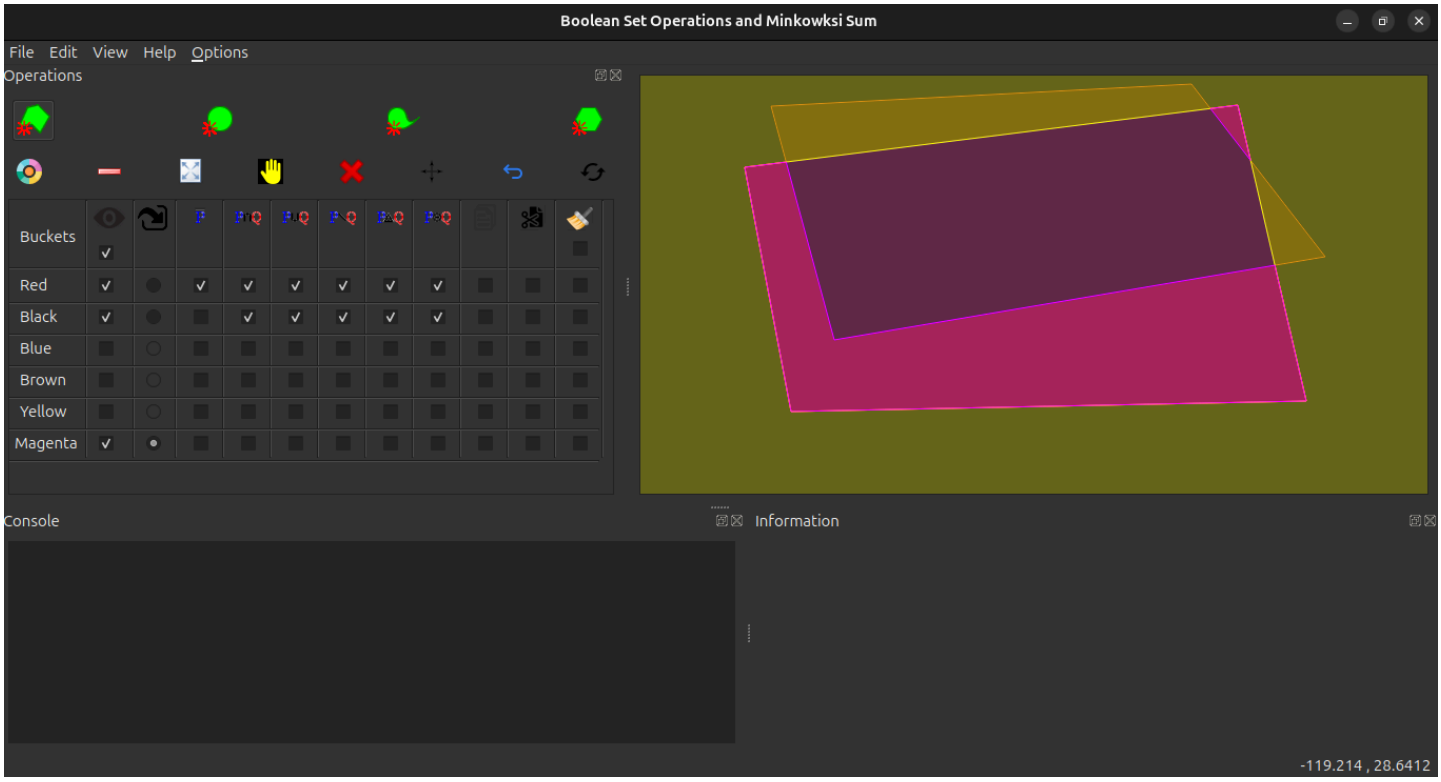
2.2 Achieved Milestones

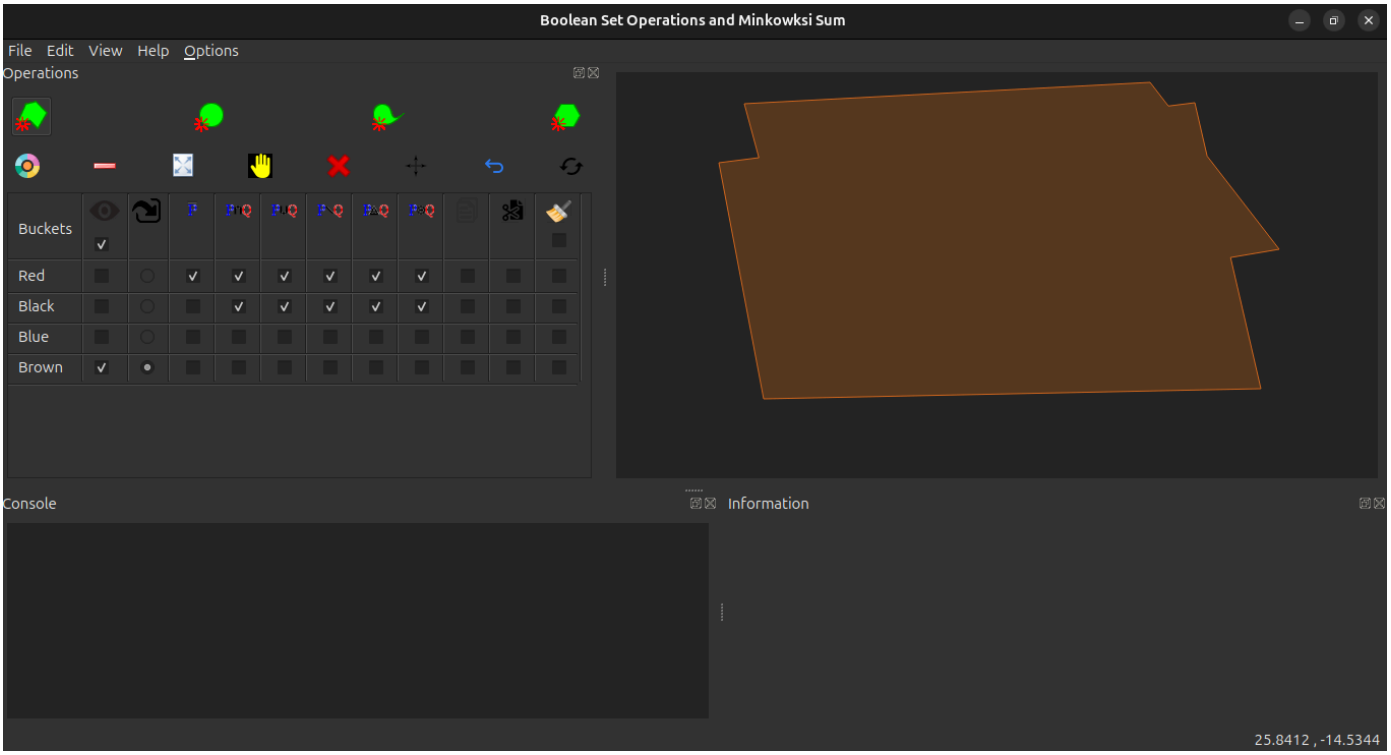
- **Core Boolean Operations**
  - Standard operations (union, intersection, difference) work on **simple polygons** (straight-line edges).
  - **Minkowski sum** is implemented (at least for basic shapes).
- **Pipeline Processing**
  - Supports chaining operations.
- **Demo Framework**

Following are the some of the examples :




Boolean Set Operations and Minkowksi Sum
File Edit View Help Options
Operations
Buckets
Red
Black
Blue
Console
Information
-46.5141 , -14.9618


Boolean Set Operations and Minkowksi Sum
File Edit View Help Options
Operations
Buckets
Red
Black
Blue
Brown
Yellow
Magenta
Console
Information
-119.214 , 28.6412



2.3 Deliverables

- Refactored and well-documented codebase following CGAL coding standards.
- Support for general polygons bounded by circular, conic, Bezier, and algebraic curves, basically **non-linear curves**.
- A command-line console for easier interaction with the demo.
- Comprehensive test cases to ensure the correctness of implemented features.
- Updated documentation and a polished version of the demo ready for CGAL distribution.
- Scriptable CLI .

2.4 Benefits to the Community

- Improving the maintainability and readability of the demo's source code.
- Extending the Boolean operations to support additional types of curves.
- Introducing a command-line interface for better usability.
- Enhancing testing and documentation to facilitate future contributions.

2.5 Technical Details

The project requires expertise in:

- **C++ Generic Programming** (Templates, STL, Boost)
- **CGAL's 2D Boolean Set Operations (planar geometry only)**
- **Qt6** (for GUI components)
- **Computational Geometry & Linear Algebra**

I have successfully **installed CGAL** from source with Qt6 and CORE support and have **explored** the 2D Arrangements and 2D Regularized Boolean Operations packages.

2.6 Timeline

Community Bonding Period

- Engage with the CGAL community, discuss the project plan with the mentor.
- Get familiar with the existing codebase of the demo.
- Experiment with examples from the 2D Arrangements package.

Coding Period

Weeks 1-2: Code Refactoring & Cleanup

- Split large files (e.g., Boolean_set_operations_2.cpp) into modular components.
- Replace `typedef` with `using` declarations.
- Follow CGAL's coding standards (indentation, line length, naming conventions).
- Remove unnecessary typedefs and redundant code.
- Add detailed comments and documentation.
- Identify and rectify immediate and obvious bugs.

Weeks 3-4: Support for Circular Arc Polygons

- Modify data structures to support general polygons bounded by circular arcs.
- Implement Boolean operations for these polygons.
- Create test cases for validation.

Weeks 5-6: Extend to Other Curve Types

- Implement support for additional arc types (conic, Bezier, algebraic).
- Adapt Boolean operations to handle these curves.
- Extend test cases to cover all new curve types.

Weeks 7-8:Command-Line Console Development and I/O

Design a minimal language to:

- Record operations (e.g., `union(A, B)`).
- Execute commands via console input or script files.

Implement the following features:

- An interpreter to parse and execute commands.
- Automatic recording of GUI actions as command scripts.
- An interactive console prompt with:

    - Real-time feedback (e.g., `> union(A, B)` triggers the operation).
    - Toggleable verbosity (to show or hide command echoing).

- File I/O support (e.g., load("script.csg2") to load and execute a script).

Weeks 9-10: Testing and Optimization

- Perform extensive testing on different datasets.
- Optimize performance for large inputs.
- Fix bugs and refine the implementation.

Weeks 11-12: Final Integration and Submission

- Finalize documentation and testing.
- Prepare the demo for official inclusion in CGAL.
- Submit final project report.

Note: Scope will be adjusted iteratively with mentor guidance

2.7 Expected Outcome

At the end of GSoC, the demo will have:

- A more modular and maintainable codebase.
- Support for general polygons with circular and algebraic arcs.
- A command-line interface for easier interaction.
- Comprehensive testing and documentation.
- Ready-to-publish status in CGAL.

3 Future Involvement

After GSoC, I plan to continue contributing to CGAL by maintaining and improving the Boolean Operation Demo . I am

interested in exploring more computational geometry problems and contributing to CGAL's future enhancements.

4 Commitment & Availability

- I confirm that I will be available full-time during GSoC 2025.
- No academic or professional commitments will interfere with my work.

5 References

- CGAL 6.0.1 - Manual
- Github