New BRL-CAD GUI

GSoC 2022 Proposal BRL-CAD

Personal Information

- **Name:** Himanshu Sekhar Nayak
- **E-Mail:** himanshuwindows8.1@gmail.com, h1manshu@proton.me
- **GitHub:** https://github.com/Himanshu40
- **LinkedIn:** https://www.linkedin.com/in/himanshu-sekhar-nayak
- **Timezone:** IST (UTC+5:30)
- **Location:** Balasore, Odisha, INDIA

Background

Education

I am currently in the final year of the undergraduate program in Computer Applications at Utkal University which includes courses like Computer Programming, Computer Networking, Operating Systems, Data Mining, Discrete Mathematics, Database Management System, etc. I joined this undergraduate course back in 2020. I am primarily interested in Computer Programming and Operating systems and mainly worked with C, C++, and Java.

Work Experience

The following project that I have done during my undergraduate program:

S No.	Name	Description
1.	**Image Filter**	• Modified each pixel in such a way that a particular effect is apparent in the resulting image for **BMPs** images. • Implemented **sepia**, **blur**, and **reflection** effects. • Programming Languages Used: C, Make.

Open-Source Experience

The following are some contributions that I have done during GCI:

S No.	Organization	Description
1.	**RTEMS**	Implemented **POSIX API Signature Compliance Tests** for .h file and secured **Finalist** in **GCI 2018**.
2.	**Apertium**	Fixed segmentation faults and memory leaks in **lsx-comp** which is a module for reordering separable/discontiguous multi-words and processing them in the pipeline.
3.	**Sugar Labs**	Created a web project which is a math quiz game on basic calculations.

Experience with BRL-CAD

- Currently I am reviewing upon codebase and exploring the Arbalest GUI to find bugs and thinking about what features can be added.
- Previously I worked on adding a feature Annotation Text Segment Extending which is already contributed by ***Ali Haydar*** during GSoC 2019.
- During GCI 2019, I contributed to Lua bindings for the BRL-CAD C++ core interface and secured **Grand Prize Winner**.

Project Information

New BRL-CAD GUI

- The project aims to add more CAD features and improvements to arbalest.
- Arbalest will be an improvement over existing editors **MGED** and **Archer** since they are old GUIs developed during the 80s and 90s which lack modern GUI features and are not so user-friendly compared to Arbalest which is a modern **Qt-based UI**.
- This project will borrow geometry management services from BRL-CAD's **core C++ interface** which is a self-contained object-oriented interface and also from **LIBGED** which is basically all commands available to both MGED and Archer.

Project Size: Large (350 hrs)

Approach

Current Scenario

Currently, Arbalest has some features implemented during GSoC 2020 which include:-

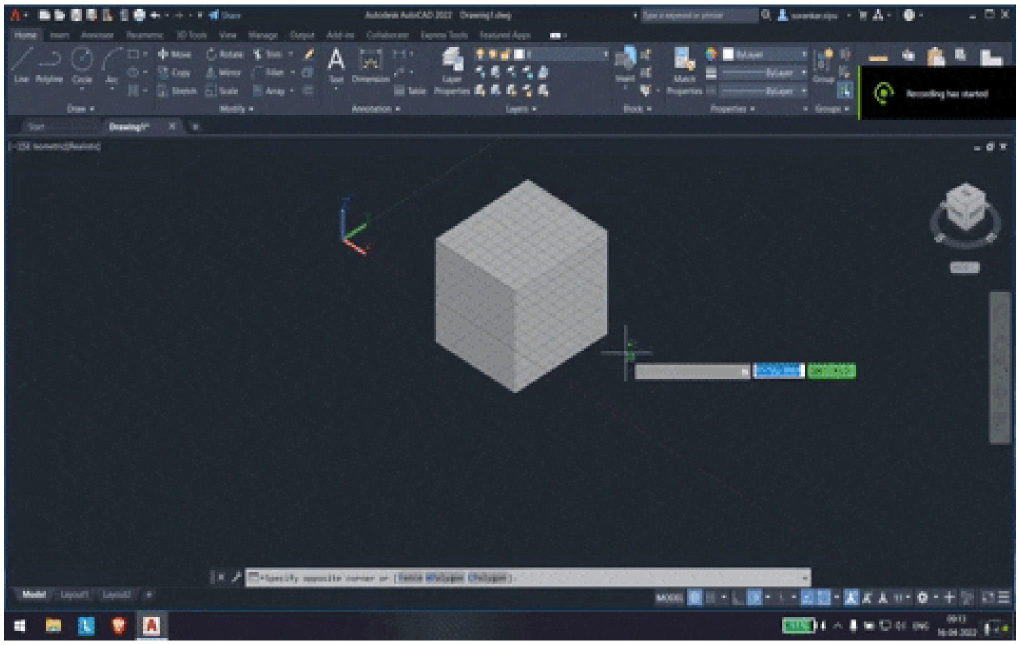
- Opening multiple .g files which are displayed as tabs.
- Top objects are displayed in the right pane.
- Use the mouse wheel to zoom in and zoom out.
- Move the camera by dragging with the mouse right button pressed.

Aim

- To add the select feature for the primitive through mouse support.
- To add and resize features for the primitive through mouse support.
- Extend object-oriented C++ Geometry API by adding support for more primitives.
- To find and fix bugs.

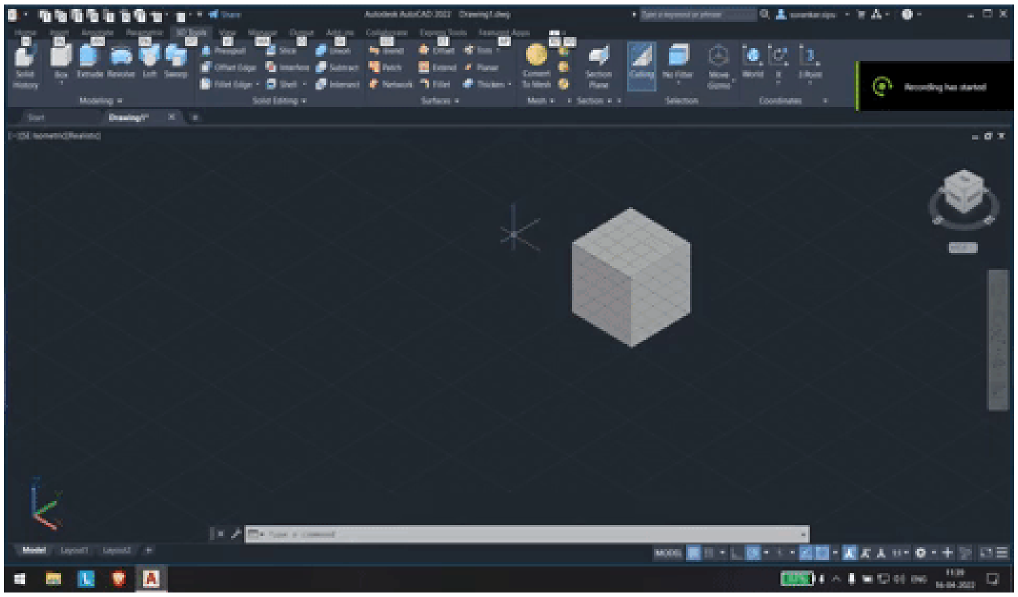
Proposed Method

- I have found a feature and bugs that can be implemented during GSoC 2022 which are the following:
    - **Feature**
        i. To implement the selection of primitive through **mouse drag**. Here is a gif of how to select the primitive in AutoCAD:



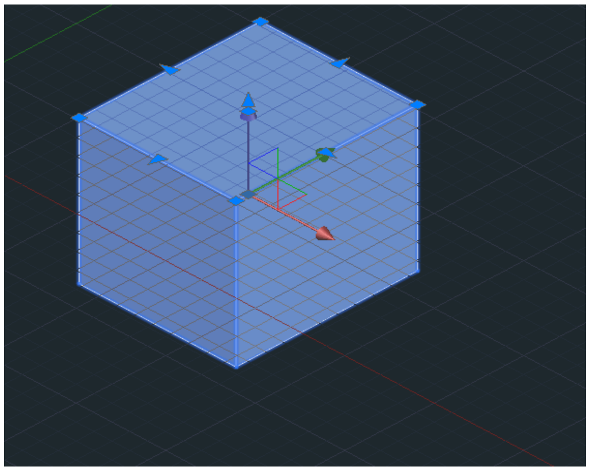
Selecting a primitive using mouse drag

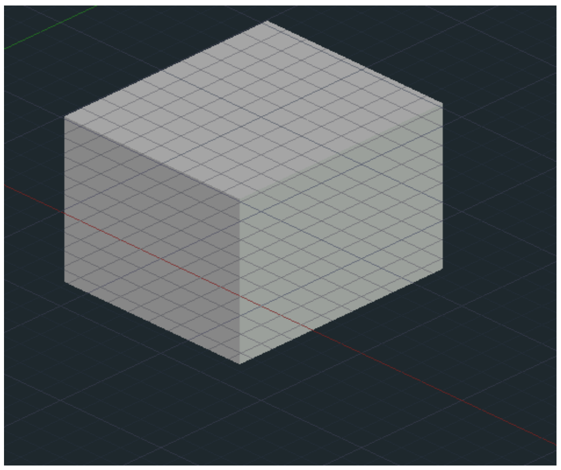
Otherwise there is another way to select a primitive by only left-clicking on the primitive and deselecting the primitive by pressing the **Esc** key or clicking outside of the primitive.



Selecting and Deselecting the primitive using the left mouse click

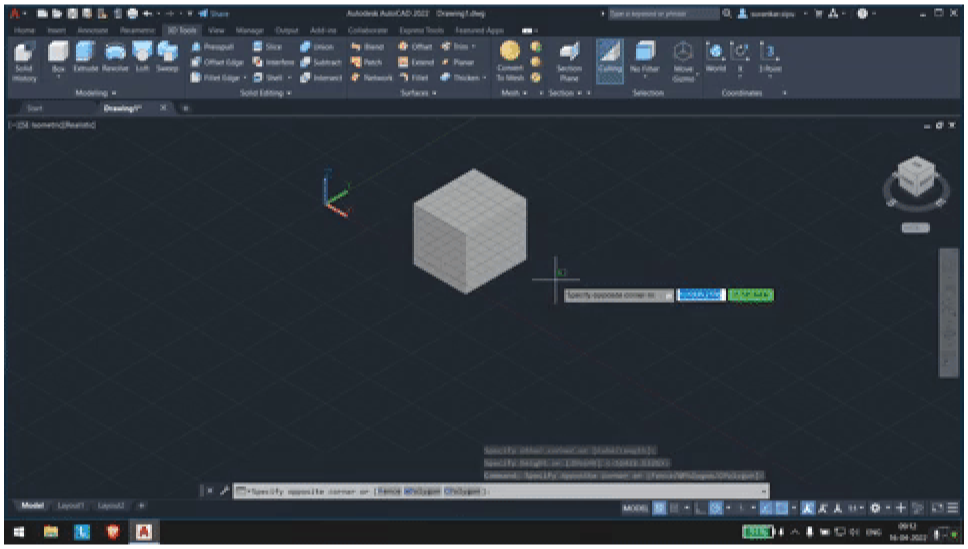
Here is a quick glimpse of how the primitive looks after selection vs deselection:





A primitive is selected ***A primitive is deselected***

ii. To implement resizing of primitive by having the **Scale** option inside the **Edit** option in Arbalest. First the primitive will be selected through the above feature then inside the Edit option click on the Scale option. Then click on any corner of the primitive and drag mouse to enlarge or dwindle the primitive. Here is a gif of how to resize the primitive using mouse support in AutoCAD:

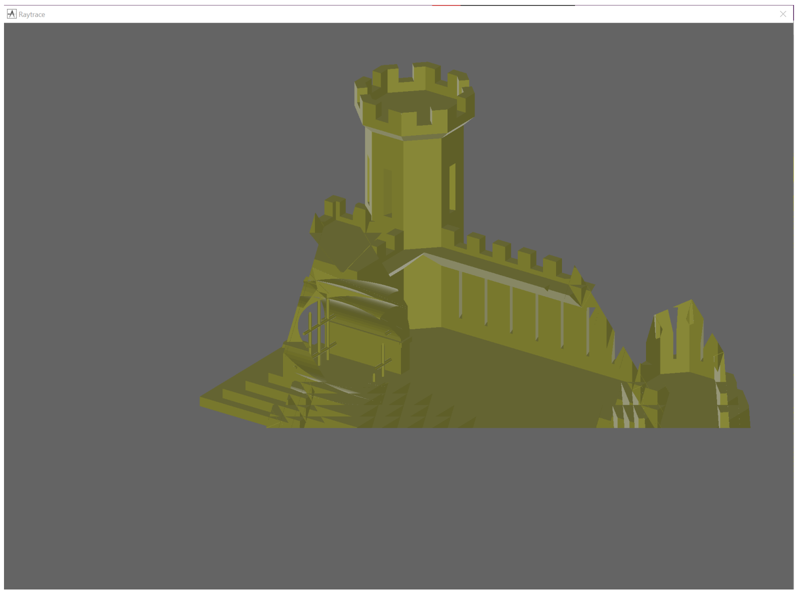


Resizing of primitive

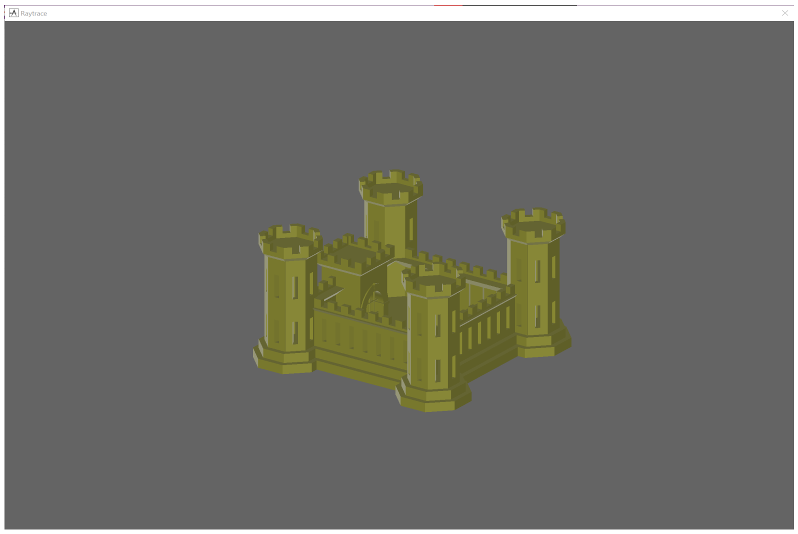
iii. Connect already present primitives implemented in `coreInterface/` to Arbalest under **Create** section. Example: Create `Arb` section, there will be a list of arb primitives ranging from `arb5` to `arb8`.

- **Bugs**

  i. Currently, the **Raytrace** feature is not able to fully ray-trace a simple solid and the same applies to an object which is a combination of other solids. Here is an example:



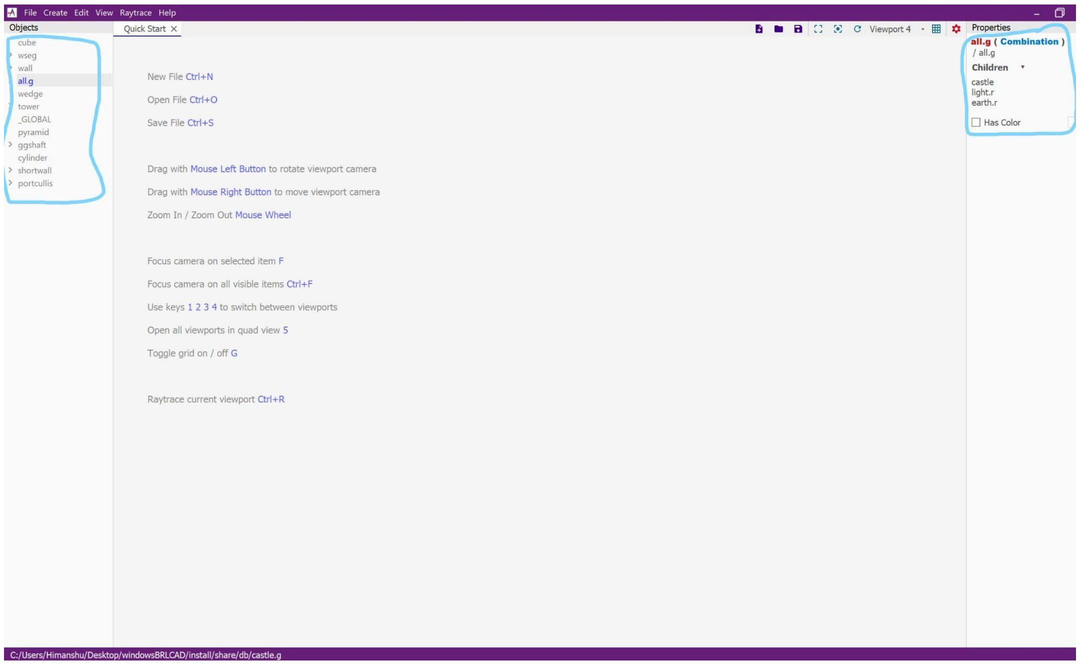
Not fully raytraced object



Fully raytraced

Looking into `void RaytraceView::UpdateImage();` in `RaytraceView.cpp` where `QVector3D imagePoint(column, h - row - 1., 0.);` where for every **Z position**, the starting position is always **0** due to which it is not able to fully ray trace the object. A workaround for z-position can fully ray trace the object.

ii. Closing an existing .g file doesn't clear the **Properties** and **Objects** pane buffers. Here is an example:



cube
wseg
wall
all.g
wedge
tower
_GLOBAL
pyramid
ggshaft
cylinder
shortwall
portcullis

New File Ctrl+N

Open File Ctrl+O

Save File Ctrl+S

Drag with Mouse Left Button to rotate viewport camera

Drag with Mouse Right Button to move viewport camera

Zoom In / Zoom Out Mouse Wheel

Focus camera on selected item F

Focus camera on all visible items Ctrl+F

Use keys 1 2 3 4 to switch between viewports

Open all viewports in quad view 5

Toggle grid on / off G

Raytrace current viewport Ctrl+R

all.g (Combination)
/ all.g
Children
castle
light.r
earth.r

☐ Has Color

C:/Users/Himanshu/Desktop/windowsBRLCAD/install/share/db/castle.g

Buffer remains even after the tab is closed

Looking into `void MainWindow::tabCloseRequested(const int i);` in `MainWindow.cpp` where it shows

```
documentArea->removeTab(i);
if (documentArea->currentIndex() == -1){
    objectTreeWidgetDockable->clear();
    objectPropertiesDockable->clear();
    activeDocumentId = -1;
    documentArea->addTab(new HelpWidget(), "Quick Start");
}
```

If the `Quick Start` tab is not closed and the rest tabs is closed then the buffer still remains because `document->currentIndex()` gives **1** which is not equal to **-1**. A workaround to fix this bug can be to keep a checkbox inside `Quick Start` tab. When the first time the user opens the application, there will be a checkbox if the user wants that application will always start with `Quick Start` or without it.

iii. **Dark Mode** is not compatible with the current UI like the **X button** which means the button is faded out when the application switches to dark mode. There may be more icons that are not compatible with dark mode. A compatible dark theme can fix this issue or have to check specific icons for compatibility in dark mode.

iv. There can be more bugs that I have not seen yet. During the development period if possible I will find more bugs and try to fix them.

- Extend **object-oriented C++ Geometry API**

  i. Some of the primitives are not yet implemented in `coreInterface/` which includes `ars` (Arbitrary Faceted Solid) and `metaball`. Also, the `extrude` is not yet implemented which modifies an `ARB` shape by extruding the specified face through the specified distance to determine the position of the opposing face.

  ii. Currently, there are some primitives that are in `coreInterface/` and not included in arbalest which are **bot**, **pipe**, **sph**, **sketch**, and **nmg**. These primitives can be added to `MainWindow.cpp` under **Create** section. Here is an example of how to add `Pipe` in `MainWindow.cpp`:

```
QAction* createBOTAct = new QAction(tr("Pipe"), this);
   connect(createBOTAct, &QAction::triggered, this, [this]() {
       if (activeDocumentId == -1) return;
```

```
        BRLCAD::Pipe * object = new Pipe();
        QString name = QInputDialog::getText(this, "Object Name", "Enter
object name");
        object->SetName(name.toUtf8());
        documents[activeDocumentId]->getDatabase()->Add(*object);
        int objectId =
documents[activeDocumentId]->getObjectTree()->addTopObject(name);
documents[activeDocumentId]->getObjectTree()->changeVisibilityState(objectId
, true);
        documents[activeDocumentId]->getObjectTreeWidget()->build(objectId);
documents[activeDocumentId]->getObjectTreeWidget()->refreshItemTextColors();
documents[activeDocumentId]->getGeometryRenderer()->refreshForVisibilityAndS
olidChanges();
documents[activeDocumentId]->getDisplayGrid()->forceRerenderAllDisplays();
    });
createMenu->addAction(createBOTAct);
```

I will add rest of the primitives that are already present in `coreInterface/` to Arbalest. I found these are for more advanced modeling, are still in development, or have significant performance implications.

Plans

Deliverables

The implementation can be broken into three milestones:

MILESTONE 1

- Add code for the feature to select and resize the primitive by the best possible strategy or method after discussion with the mentor.

MILESTONE 2

- Add code for some of the primitives that are not included in `coreInterface/` and then include those implemented primitives to Arbalest in **Create** section.

MILESTONE 3

- Improvement and fixes to the discovered bugs.
- Add documentation for each part, code cleaning, and prepare a summary report of the work done during the period.
  - Update the `README.md` and create `CHANGES.md`.
  - Code clean-up and prepare a summary report for the overall project.

Development Schedule

Following is the timeline summary based on the implementation breakdown above and milestone-oriented code reviews and merge requests:

Timespan	Activity
May 20 - June 12	● Discuss the plan with the mentors to select the best possible strategy to implement the feature. ● Discuss the challenges involved in implementing the feature and fixing bugs. ● Get familiar with the required components of BRLCAD for implementation in Arbalest. ● Get familiar with the **Qt** source code for implementing the feature. ● Particularly discuss **C++11** constructs for writing code.
June 13 - June 19 (1 week)	● Starts writing source code for bringing up the select feature by reading **Qt docs**.
June 20 - July 3 (2 weeks)	● Implement the select feature to select and deselect primitives. ● Prepare tests and demos on various primitives to check if all the select and deselect feature is working or not accurately.
July 4 - July 24 (3 weeks)	● Starts writing source code for resizing the primitive. ● Prepare tests and demos on various primitives to check if resizing works or not perfectly.
July 25 - July 29	● Submit the **MILESTONE 1** Merge Request for review. Work on the change suggested by the mentor(s). ● Update `README` and `CHANGES`.
Phase 1 Evaluation deadline	
July 30 - Aug 28 (4 weeks)	● Starts writing source code for the rest of the primitives that are not included in `coreInterface/`.
Aug 29 - Sept 4 (1 week)	● Write code to connect those implemented primitives to appear in GUI.
Sept 5 - Sept 12	● Submit the **MILESTONE 2** Merge Request for review.

	Work on the change suggested by the mentor(s). ● Update `README` and `CHANGES`.
Sept 13 - Sept 20 (1 week)	● Find the bugs with high severity and try to test those bugs and report the behavior and working.
Sept 21 - Oct 12 (3 weeks)	● Fix the bugs and test if working perfectly.
Oct 13 - Oct 31	● Add the changes to `README`. ● Work on the changes suggested by the mentor. ● Submit **MILESTONE 3** code documentation changes.
Nov 1 - Nov 21	● Take suggestions from the mentor for three milestones. ● Try to improve any changes requested by the mentor.
Final Mentor Evaluation	

There might be some days when I have to take leave for examination and admission during the development period.

Time Availability

I plan to spend **3 hrs/day**, so roughly around **21 hrs/week**. On weekends, I can work up to 4-5 hrs as well. My new semester is going to start on the 20th of April and will probably end on the 31st of July. In between, I have two entrance exams in the first two weeks of June because the dates are not announced yet. Also, I will be having semester exams which may take 2 weeks overall. In any case, I assure BRL-CAD that I can dedicate 21 hrs per week.

Why BRLCAD?

BRL-CAD is like a family to me where I met experienced developers who guided me even if it related to a task during GCI or fixing a bug or applying changes to the codebases. I have been with BRL-CAD since GCI 2019 and still going on and gaining a wonderful experience from mentors. I had chosen this org because mentors are very helpful and responsive and also supportive.

Why me?

I started to learn and contribute to Open-Source back in the 8th grade of school where I am a newbie to code, documentation, and research. Now I am at a stage that reflects open source helps in building and gaining experience. My primary programming languages are **C99** and **C++11-14**. I have knowledge of **data structures** and **algorithms** which includes **trees, binary trees, BST, Priority Queues, Tries, Hashmaps, and Graphs**. Also, I know **OOPS in C++** which includes **delegating constructors, constructor initialization lists, operator overloading, polymorphism, smart pointers, STL, Lambda expressions**, etc. The project programming language requirement is **C/C++** with **Qt** too. Since I am a newbie to Qt but I already started learning it in February of this year and it is mainly C++. Being an open-source enthusiast with the background knowledge above mentioned, I would be very excited to work on this project.