Augmentae
design reality

AUGMENTED REALITY MECHANICAL DESIGN SOLUTION

FINAL REPORT

Team Number	**sdmay18-02**
Client	**Ms. Deeksha Juneja, Founder, E.S.A.R.C. L.L.C. (Augmentae)**
Advisers	**Dr. Thomas Daniels**
Team Members and Roles	**Bhimesh Chauhan – Team Facilitator, Backend & Data Roles** **Cal-Van Vert – Data/Pipelining Engineer** **Lucas Ince – Web Application Back End and Database** **Ryan Luckinbill – Web Application Front End and Database** **Vaibhav Malhotra – Documentation Lead, Backend Developer**
Team Email	**sdmay18-02@iastate.edu**
Team Website	**http://sdmay18-02.sd.ece.iastate.edu/**
Revised:	**April 23, 2018**

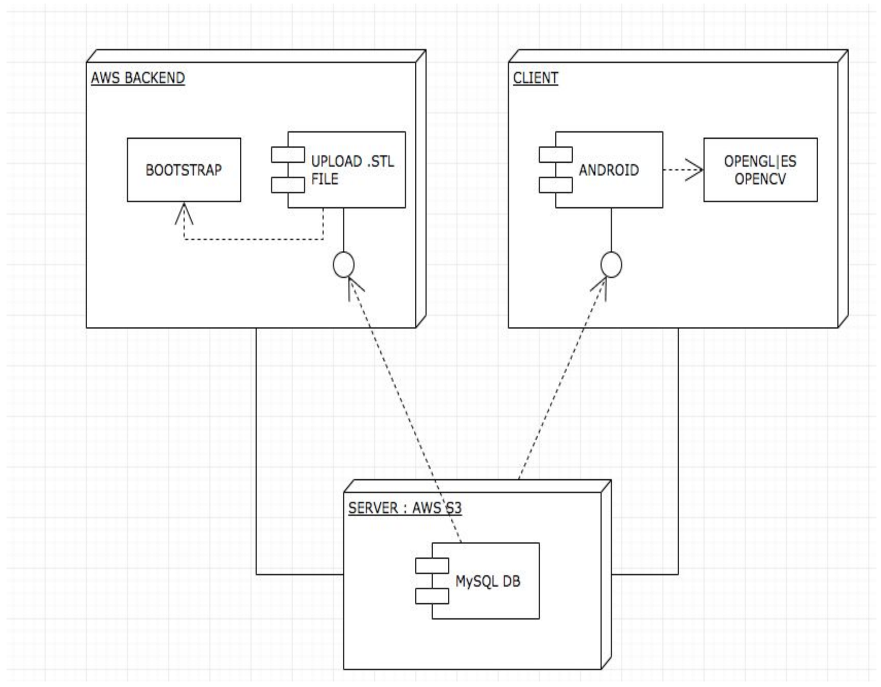
Table of Content

1. Project Design

1.1. Problem Statement and Solution

The augmented reality is one of the leading features in the Mixed Reality industry which has been designed only for entertainment industry or gaming industry. The goal of this project is to let designers showcase their 3D or CAD models to their clients in an augmented reality setup. Designers can achieve this task by uploading their current CAD model to a web application which can later be downloaded and viewed through an augmented reality camera of their android mobile phone. The clients can then interact with the model and provide feedback to the designers.

The overview of the design process / application architecture can be stated as follows:



1.2. Functional Requirements

- CAD designers should be able login to their account on the web app
- CAD designers should be able to upload CAD files from the web app to the AWS server.
- CAD designers should be able to add a client account to the current models.
- Clients should be able to login to their account on their mobile device and web app.
- The mobile app should be able to fetch the CAD models from AWS according to the user account.
- Clients should be able to view their requested CAD models from the mobile app.
- Clients should be able to view project on web

1.3. Non Functional Requirements

- Web and mobile apps should be easy to use and intuitive.
- The web app should not have any dead web pages.
- CAD models should be able to be uploaded from the web app successfully to the AWS.
- The mobile app provides an optional payment processing feature.
- The mobile app should not crash while rendering the CAD models.
- Mobile app is available for Android.

1.4. Resource Requirements

Both hardware and software resources are required to develop and test for this project. We used our own mobile devices to test the CAD model rendering and the mobile app's user interface Design. For software, we used the Android Studio SDK and notepad++ to build the mobile app and the web app respectively. To store the user accounts and CAD model files online, we decided to use the Amazon Web Services as our backend storage and server.

1.5. Risk Identification and Mitigation

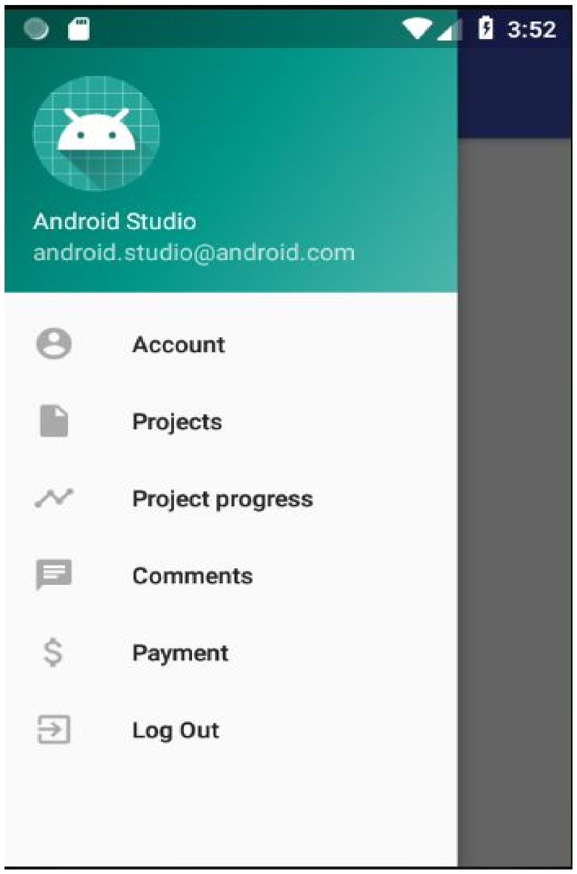
To mitigate the potential risks we may encounter while working on the project, we held meetings throughout the whole senior design semester and used a chat app to communicate with each other. To identify any possible problems, we did many testings while developing our mobile and web app from time to time.

1.6. Functional Decomposition

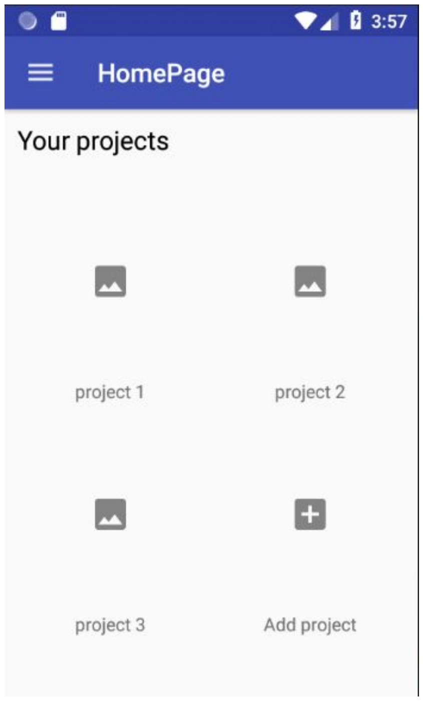
Our product has 3 main parts. The Mobile App, the database, and the Web App. The Web Application is used as a space where the developers who are working on their STL files can receive updates on the project and view important information such as deadlines, the budget, and the previous iteration of the STL file from the database. However, the main use of the web application is to upload new STL files that get pushed to the database. The database houses all information about a project, a company, and the STL files. The last part of the project is the mobile app. The mobile app is able to receive the information from the database and render the STL file for the client to see.

1.7. Mobile App UI Design

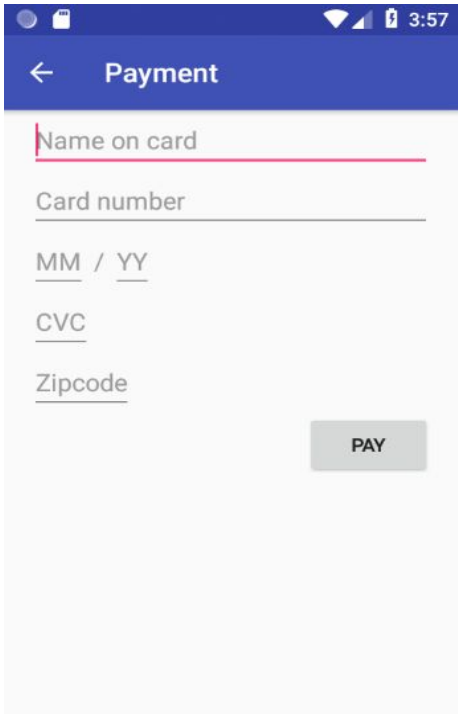
To make our mobile app development easier, we used the default template that can be found in the Android Studio SDK. The design is simple enough by default.



The mobile app has a navigation bar that can direct users to a different page. The user should be able to get to their desire page correctly as the pages are relatable based on their page titles and icons.



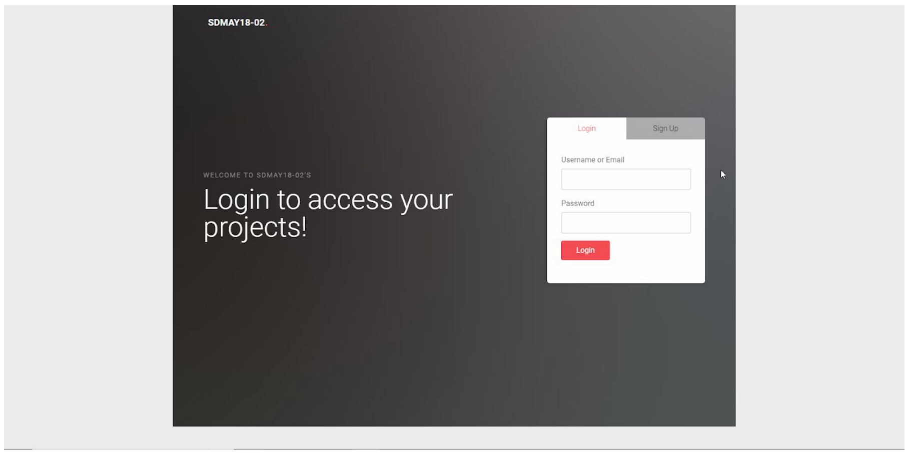
This homepage should be able to show projects that the user is connected to with an image and a project name. User can add or request a project by selecting the “Add project” icon.



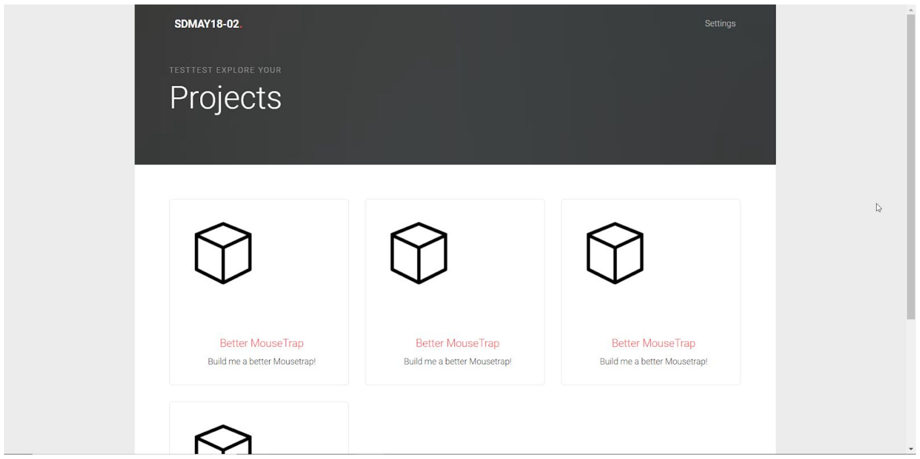
This is the optional payment page that is not fully implemented yet.

1.8. Web App UI Design

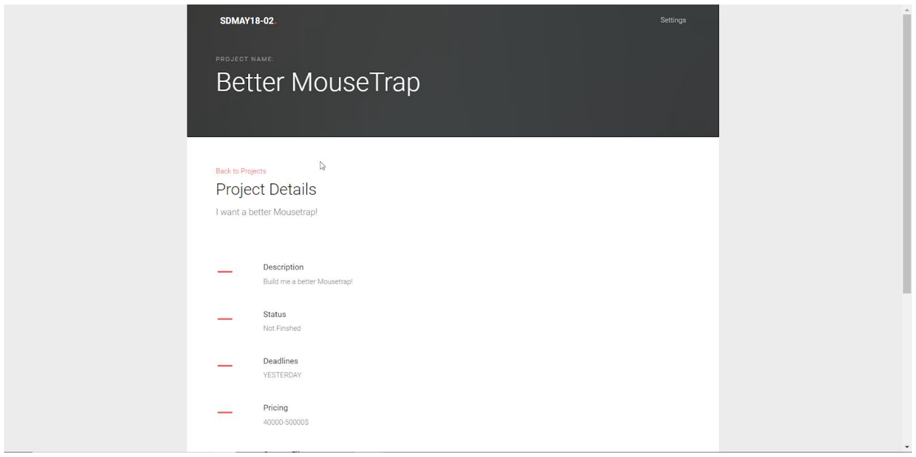
The UI design came from a simplistic idea and was meant to focus on simple functionality and basic use.



The Login page is designed to serve to serve as an access point to your projects and have not unnecessary information.



The projects page lists all of your projects and if you are an admin lists all the projects of your company. Again a spartan design is to allow you to access your projects without unnecessary information overwhelming you.



PROJECT NAME:
Better MouseTrap

Project Details
I want a better Mousetrap!

Description
Build me a better Mousetrap!

Status
Not Finshed

Deadlines
YESTERDAY

Pricing
40000-50000$

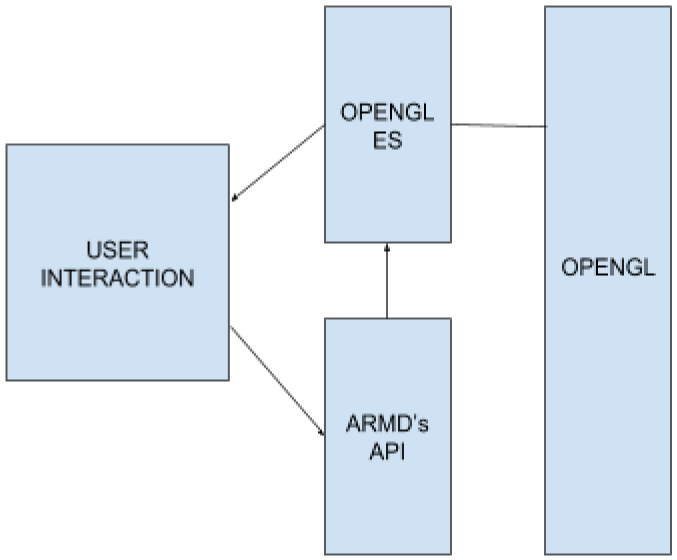
The Project page lists all the information and gives the user all the verbose information. This page shows a picture of the current project and allows new file uploading.

1.9. API Design

After looking into the implementation options and already available API and their limitation we decided to work on creating our own API which would interface with OPENGL (in case of android OPENGL|ES library) to render a model in the application as required by our clients. We looked into various API's like EasyAR, OpenAR and ARCore.

Since none of the API's available would work for our clients we decided to work on creating our own API to implement a model view where the CAD models are parsed and rendered according to various tessellation models as mentioned in the stl files.

The architecture comprises of four main modules with user display / interaction which sends the actions to our API mapping to respective OpenGL ES function and in some cases to the The design model can be represented in the figure as follows:



As we can see in the figure above for the high-level implementation for the API is set of functions that call the respective OpenGL Embedded Systems functions which are wrapped by OpenGL methods. In order to create a native application it was important for us to be able to do some processing for unsupported devices on a lower language level to be handled by a the API running on top of it.

We created the API in such a way to be able to provide maximum device support to the client and also to be able to reduce the number of API calls throughout the lifecycle of the application.

With this API our aim was to provide a scalable and sustainable application which our clients could build upon in the later phases of the product development.

2. Implementation

2.1. STL Parser

STL parser is one of the most important feature in our application as it parses obj and stl type and displays it on the phone’s camera. We used an Android library to which helped us to implement a parser. The parser we implemented can successfully parse an ASCII or binary stl file. The parser then sends the coordinates and the dimensions to the graphics renderer which has been implemented in OpenGL which will in turn process the coordinates and render the model.

2.2. Web Application Implementation

For our web application, we started off basing our application around the Amazon Web Service’s database and elastic beanstalk web application. We have 3 parts to our web application; the database, the front end, and the back end. For a good part of the semester, we had our database running off of AWS and were using elastic beanstalk for our web application. However, after a few issues with student credits expiring within the last few weeks, we decided to host the database locally and not use AWS for our web application. Instead, the front end of our web application is all written out of html and css and also hosted locally. We have a login page which then directs the user to a list of their projects. Each project page receives the project’s data from the database and displays it on the page. This page displays the current file and also allows for an upload of a new file. Our backend is php code which connects the html pages with the data from our database using phtml links. This allows for the logic of php to be used with the html of the webpage and fill it with information from the database with standard SQL queries. All three of these together work in unison to receive data from the database and display it to the user.

2.3. AR Implementation

AR Implementation of the Application was one of the most challenging aspects of the project. This implementation was the backbone of the application and it was one of the most important criteria of fulfillment for the client's goals. We started by looking into various available APIs for AR in CAD design and graphics rendering. However, while implementing many of those libraries, we observed that it limited us in many other aspects that were important to us in terms of device support, tessellation etc. We therefore decided to create our own API that would be native to ARMD's application and interface with OpenGL-ES / OpenGL library.

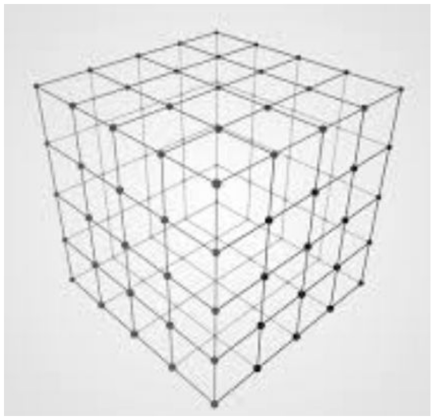
2.3.1. CAD design and Model

The CAD design and Modelling was an important aspect of the AR rendering on the mobile systems, as we found that rendering complex models (polygon rendering) would be taxing on the phones battery life and power management and would crash our application. We therefore decided to create a parser that would parse any file and convert it into a triangular coordinate system. In this way we would lose some quality and still be able to render the basic and most minimally required aspects of the application. We later on decided to render complex models that would require a broader set of function to be able to interface via laptop through an HTTP request and later through USB cable.

2.3.2. 3D Matrix and Space Model

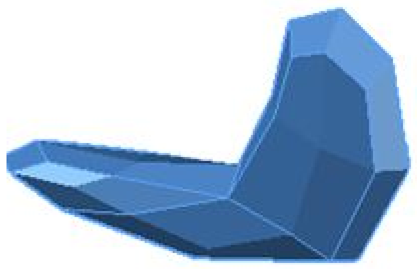
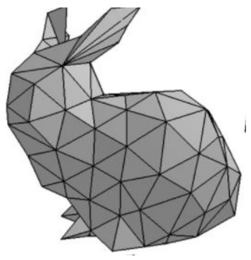
Creating a 3D mesh to render models on was a challenging aspect of the project. Without any reference to create the models we were not able to render model on “true AR” rather than just on the camera view which made relative motion with the model difficult to implement.

To adapt to the model we decided to have a model made out of triangles in space that would then adapt to the coordinates given on for a specific model. For instance, to make a sphere we would first create a model of mesh in space and then adjust each model to make a sphere. As shown in the figure below we were able to adapt the cube mesh model to the sphere model.

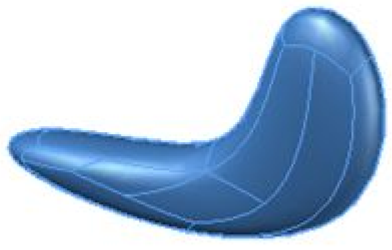
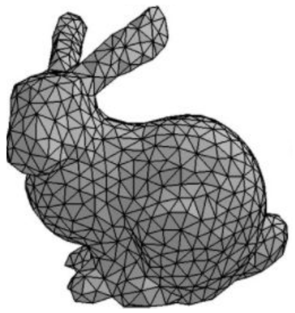


We therefore parse each of the coordinate of the cube mesh (cube mesh is larger than the sphere mesh). We adjust all the coordinates to the one specified in our file considering the center cube as the origin. We then drop all the coordinates that were not needed for the adjustment.

We tested this code with various other models which resulted in making the model very obscure, which was not acceptable for the project. For instance a wave cad model would end up looking something like this which was not something our client was aiming for.

We therefore moved on to a different algorithm to process the models based on the coordinates given in the stl file. We decide to preprocess the file on submission, stripping the meta data, which was redundant for the specific account and then render the model according to the triangle coordinates permissible by the embedded system version of OpenGL. This helped us in making the process more accurate and close to result we expected as shown below:

2.3.3. Stabilizing the design

The next biggest challenge that we faced in designing the model and rendering is stabilizing it in the space and not merely by rendering it on the camera background. This was really important for the client in order to be able to visualize and interact with the models and be able to comment on specific parts of the models and design. To do this we needed to stabilize the design on a coordinate axis and maintain it there. While we could in general use markers and QR codes to render on the model, our client wanted us to attempt to render model in space with coordinate system selected by client before rendering. This was especially important to make the app easy to use. This was done using marker as explained in the following sections.

2.3.4. Rendering Process

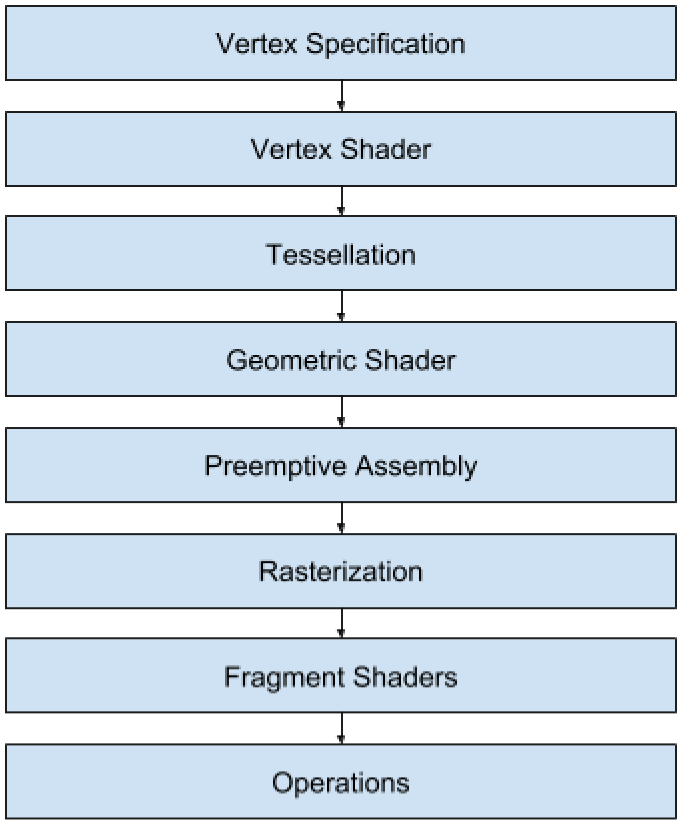
The rendering process is very straightforward using methods from the OpenGL and passing the coordinates to render and making models. Model shaders rendering is done as shown:

```
int shaderProgramId = GLES20.glCreateProgram();
GLES20.glAttachShader(shaderProgramId, vertexShader);
GLES20.glAttachShader(shaderProgramId, fragmentShader);
GLES20.glLinkProgram(shaderProgramId);
```

The shaders given in stl files helps us determine the account of the user and context to render the shaders. There are two different types of shaders that we are expecting to render model. Vertex shader helps us render the vertices of the models starting from one, ending at another. Second, we render a single vertex from the vertex stream and generate a single vertex to output vertex stream. The sample code to execute shaders can be seen below:

```
int shader;
shader = GLES20.glCreateShader(type);
GLES20.glShaderSource(shader, shaderSrc);
GLES20.glCompileShader(shader);
return shader;
```

As shown above the code compiles the type of shader given in the stl headers. This code portion creates an empty shader and then assigns the value as input stream from the program and compiles and renders the object as a whole. The following describes the architecture in much detail.



As shown in the flow diagram above, we see the lifecycle process of each parts design and rendering it on the AR. We collect the rendering data from the stl parsed file and pass it to vertex shader as shown in code before and then tessellation process helps in creating the vertices and then geometric shaders adds the material information on the vertices and rasterize by breaking it into individual fragments for continuous stream rendering using fragment shader which compiles into individual compound object. This in the end helps us view object as it is rendered in the virtualized environment.

2.3.5. Visual SLAM Algorithm

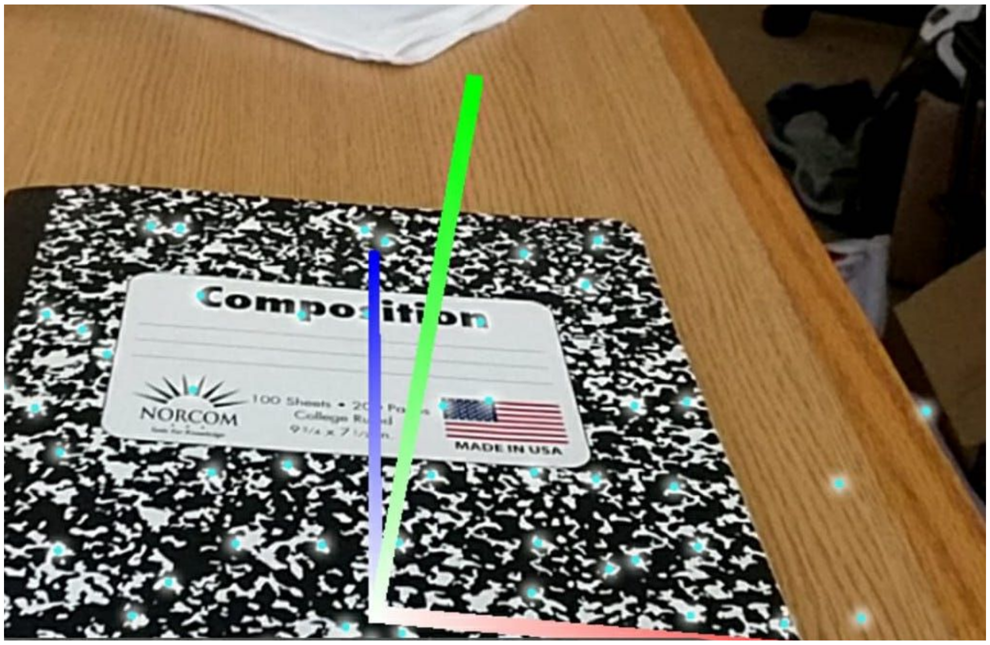
While rendering the model onto camera the requirement was to be able to render it on a 3D axis in space without the use of any code or any physical stimulus. This was accomplished using the marker system used in Simultaneous Localisation and Mapping algorithm. SLAM as the name suggest tries to simultaneously localize (i.e. find position or orientation of) some sensor with respect to some sensor or surroundings while mapping the environment. There are many other ways to map an environment known to us but not all of them are constituted as SLAM. For example marker based tracking is not SLAM because the marker image is known beforehand. The challenge here is to recover both environment map and the positions of camera at the same time.

The important difference between SLAM and other similar techniques is that it operates in real time. This mean that processing on each frame must be finished before the next one arrives, this makes pose of the camera available immediately and not as a result of post stage processing.

The main concept in understanding of SLAM is that it tracks a set of points identified through successive camera frames and using this to triangulate the 3D position, while it simultaneously calculates camera pose using the estimated point location. This helped us develop tracking of rendering model with moving camera giving user an object like interaction with the cad models. Due to these factors even with single camera it is possible to recover the position and structure with high accuracy upto a particular factor.

Another important factor for determining the position previously held by the object is by relocalisation, which is able to cope up with temporary poor tracking performance, which could cause the system to fail. Tracking therefore resumes even after changing the position and view of camera. Code and figure below shows SLAM algorithm creating axis to position object on.

```
Trackable trackable = trackingResult.getTrackable(0);
texturedCube.setTransform(trackable.getPoseMatrix());
texturedCube.setTranslate(0, 0, -0.0005f);
texturedCube.setScale(0.4f, 0.4f, 0.001f);
texturedCube.setProjectionMatrix(projectionMatrix);
texturedCube.draw();
```



2.4. Web - Mobile Application Communication -

For our application to work as whole unit, we had to create REST endpoints for out mobile application to communicate with the web application. The web application's main purpose is to serve as backend unit for the designers where they can upload their current CAD models for their clients respectively. Whenever the designers update their STL file, they can upload it via the web application. When the mobile app is in use, it will request to pull the most recent STL file from the database. The mobile application will then parse the file and render it in front of the phone camera in an augmented reality setup.

3. Testing

Our testing process followed the guideline displayed below. In short, our team built each feature in an isolated environment. We live tested as we developed and had bi-weekly demonstrations to our client to gather feedback. Once a feature was finalized and confirmed by the client, we merged that feature into a master branch to integration test. Finally, we demonstrated the integrated features to the client for final validation and hand over to their team for future steps.

3.1. Unit Testing

The unit testing was done in a TDD way where each unit test was written to fail and made to succeed eventually. The main structural code used looked something like this:

```
TEST(MainLoopExecutesCorrectNumberOfTimes){
    MainLoopState loopState;
    MainLoopUtils::Execute(slamState, 5);
    CHECK_EQUAL(5, slamState.frameCount);
}
```

There were multiple tests for each unit while rendering, rasterization, tessellation and rendering. We had separate unit tests to run on each part individually provided as default model (waves and rabbit). Each validation was done manually by the user after rendering.

3.2. Validation Testing

Validation testing was done with the clients and demonstration. We used the comments and concerns provided by the client to make the necessary changes and discuss challenges in the project we faced. The follow up meetings were lead by progress made from the previous suggestion, merged to master environment and so on.

3.3. Integration Testing

Integration testing was done once the individual models were imported on to the main environment by installing the application remote device and checking the edge cases and then followed by general use. Bug fixes were done once we had found the errors in the report, we debugged the code to address the needed change and corrections were sent for approval for next validation.

3.4. Testing Results

Throughout the process we were able to make significant progress despite of our restriction and challenges. By getting tight cycles of feedback from the client we were able to make progress in terms of developing a scalable and testable application which could address the needs or be the platform on which future application could be built and developed.

4. Future Steps

The future step that can be done in this project to create a marketable product is to refine over the model stabilization and processing of larger files in upwards of 2GB owing to the bandwidth limitation (while using a cable medium). Also,using a better algorithm to render the model could help make the process more efficient. This API is the most basic model on which one could develop further to provide more features and cover more devices as the technology in AR/VR progresses. Next steps could also be in terms of developing application to be able to do live commenting and updates from the solidworks application as a plugin that synchronizes every fixed amount of time. Our team believes that any future work should be consistent to what we have done throughout the semester and should be able to carried out in accordance with the standards set by the client and industry.

5. Appendix I : Operation Manual

5.1. Configuration

Configuration of the camera settings are available for users to do with respect to the resolution they want the objects rendered in and screen size respectively. Users can also configure the time they want to be updated about new project.

5.2. Commenting

Commenting on the CAD files was important part of the design requirement. Users can comment on individual part as a reference and can be viewed by the client in order specified and priority set.

5.3. Updates

To update the CAD model in the mobile app, the client has to pull down on the screen to refresh the models that are available to view from the designers. The client can then load the model and interact with it and view it from different angles in an augmented reality setup.

5.4. Create / Request Project

Clients are able to request new projects from various companies choosing it from the side menu. Then selecting the company and then write project details and deadlines to be discussed on phone or via meeting which can also be requested via same form.

6. Appendix II : Project Changes

In the previous version of this project, our project goal was to create a software that enable users to design and modify a CAD model interactively in a HTC Vive's virtual environment. We wrote a stl file parser in Python and managed to render a simple cube on the HTC Vive without the interactive components. At our previous panel presentation, we were told that the project is too large of a scale to be our senior design project and we might not be able to do it in two semesters. After discussing with Ms. Deeksha, she decided to change our project to the current project.

7. Appendix III : Challenges

7.1. Algorithmic Limitation

One of the major challenges that we faced while developing the application was the size limit on data and processing speed of more complex cad models. This was challenge because many times after rendering the model it would take some time for our process to re-render the model if the camera position changed.

To overcome this challenge we took the help of SLAM algorithm that helped us to simultaneously track the position of the camera and map the environment. This algorithm helped us stabilize the model to certain extent. Another challenge we faced was to fix model at a certain place in the environment which we did using the re-localization process of the SLAM algorithm. We had to match each frame and changes in each frame. If the changes were over the acceptable limits we would know the position has changed. However, sometimes we still faced issue in identifying the proper marker placed in environment, especially in dark lighting situation where we had less data points to rely on.

7.2. File Size Support

Note: Write of overhead size and the data repetition and how to mitigate with the project info

7.3. AWS Issues

We had a variety of issues setting up and using AWS. One of the first issues we ran into issues setting up a multi-user AWS environment. We could not get resources to be shared among two users. In order to fix this we contacted Amazon for support and tried to set up group sharing. Neither of these avenues panned out so we had to go with sharing a single account. In addition both web developers had issues with Amazon supposedly charging us for our free accounts. When it happened to Ryan we moved development over to Lucas's account. When he had these uses the website was moved over to private host by incepost.com.