Status Report 1

Group
Bjarte K. Larsen *Student*
Morten Omholt Jensen *Student*
Jørgen Hanssen *Student*

---

iOS

What's been done?

- logging in using the streambim credentials.
- Fetching all the projects that you have access to and displaying them in a list.
- Fetching all the buildings to the project.
- fetching the 3d information and parsing the protobuf files into usable data.
- Displaying the 3D information with correct material information.
- Improving memory consumption on the ios platform by merging nodes together.
- switching and toggling on layers (ARK, RIV, RIE, etc..)
- minimap rendering.
- Event system, to pass events between the view controllers.

What's next?

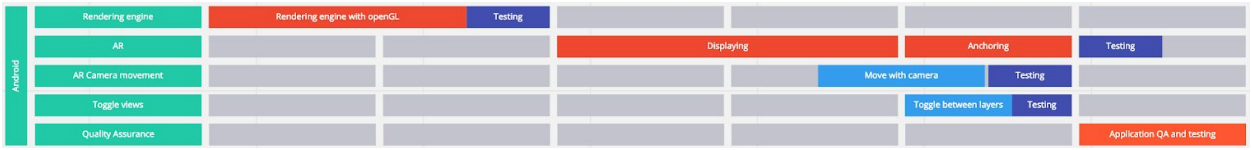
- minimap is not synced correctly with the 3d camera movement, the translation is off.
- Find walls and map them to a geometry instance so they can be matched against the building geometry. (Used for intial calibration)
- figure out if it's possible to work with feature points instead of just ARK models(Architecture models) with this we can match the RIV and RIE models aswell and not

just the ARK models, since RIV will consist of beams and RIE contains the electrical system.

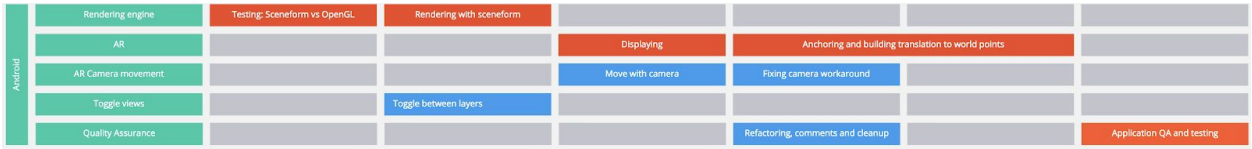
- Correctly anchor the building models to the anchor points found by the camera.
- Improve memory consumption further

Android

Initial Roadmap



Current Progress



What's been done?

- Settled on Sceneform and Scenekit instead of a custom rendering engine in OpenGL and Metal.
- We were able to request and parse rendering data from Rendra's API such as material files, octree manifest (info) files and the octree node data which is used to render the nodes in a 3D environment.
- logging in using the streambim credentials.
- Fetching all the projects that you have access to and displaying them in a list.
- Fetching all the buildings to the project.
- fetching the 3d information and parsing the protobuf files into usable data.
- Displaying the 3D information with correct material information.
- Improving memory consumption on the ios platform by merging nodes together.
- switching and toggling on layers (ARK, RIV, RIE, etc..)
- minimap rendering.
- We were not able to implement free camera movement using the in-built Sceneform camera node. We therefore implemented a workaround moving and rotating the model instead of the camera. We fixed this because we realised that the workaround caused difficulties when translating positions between the 3D models to the minimap and the AR rendering view.
- We were able to navigate the scene in AR in sprint 2: building scaling was correct and surprisingly good tracking. This was tested at Rendra HQ using a model of their HQ, which scaled perfectly to the real building.
- We have implemented a best practice MVVM architecture, recommended by Google which includes data flow, networking and local storage through databases.
- Added custom tracking (trackables) for planes, edges and corners. This tracks walls, roof and floor of a room and calculates intersections (edges and corners) and stores these.
- We implemented a minimap using Rendra's tile map server. However it has not been implemented using proper tile scaling or translations to the 3D model. Current version

uses one big tile in the minimap containing the whole model in 2D and the initial positions are hardcoded.

What's next?

- Use trackable information to map a building to the real world
- Use trackables to create anchor points at corners in a real world building
- Fix minimap to translate properly between 3D model and 2D tilemap. Also use Rendra's endpoint to get smaller tiles based on zoom level and stitch the tiles together so that the resolution would be higher when zooming in and out of the tile map.
- Implement rendering and AR engine as React-Native module
- Redesign and clean up UI
- Consider edge cases for AR mapping such as; the building is half-done or missing real world layers and so on.
- Memory Optimizations
- Application QA and testing
- Write the thesis report

General improvements

User interface

The user interface is different from platform to platform, this is something we need to make uniform. To solve this We have been looking into using react-native as a intermediate cross-platform layer for all the UI code and any code that we are not required to keep native. This comes with some trade-offs and some positive sides, allowing us to write and maintain one interface, but it will give us some problems when it comes to managing and sharing data between the two layers, and javascript in its nature is async, and native code is not. This requires a bit of research and testing and we will spend one work week to get this up and running to see if it is feasible.

Memory management

Currently we load the entire high-res model into memory. This is not good for performance at all when we load larger models. The largest building in the streambim app at this moment is around 4gb in memory thus being a huge problem for android due to heap constraints and IOS due to memory constraints. We will at some point look at streaming the models down as we need them and just keep what you can see and the closest models to you in memory at a given time. As long as we translate the scene by world position when we move it around there should be no issues with streaming and adding models as we go even if you move around inside of the building in AR mode.

Tile map

Currently we load the entire high-res model into memory. This is not good for performance at all when we load larger models. The largest building in the streambim app at this moment is around 4gb in memory thus being a huge problem for android due to heap constraints and IOS due to memory constraints. We will at some point look at streaming the models down as we need them and just keep what you can see and the closest models to you in memory at a given time. As long as we translate the scene by world position when we move it around there should be no issues with streaming and adding models as we go even if you move around inside of the building in AR mode.