Simulator & ORB-SLAM:

We're done, we have a working code that manages to scan the room and navigate to an exit point, unofficially, the project is complete. With that said, there are still some stuff we'd like to do:

- implement our own exit-finding-algorithm (more on that further down) – We're currently using RoomExit::getExitPoints and it works well enough.
- Add some visualizations – this could be drawing a point at each exit point, drawing a path to the exit, drawing the simulator's camera's path and much more. This, however, is currently our lowest priority as it wouldn't add any "real" value to the project as, as mentioned above, the important part is over.
- Test our software, so far, we ran it manually with a relatively small number of scans (only a sixth of the total field of view) and it worked. This does not mean that the code would necessarily work if we started at a different point, used a different number of scans or ran it a different environment (though this wouldn't need to be checked as according to our Zoom meeting with Liam last week, we'll only work in the simulator)
- Since until now we've used the given RoomExit::getExitPoints function, we didn't need to get the distance of each point (which would be needed for own exit-finding-algorithm) – this means we didn't have an opportunity to try it out, we might encounter some problems with this throughout the week as we implement our own exit-finding-algorithm.

Expansion on algorithms from last week:

Since last week, we have made attempts to alter the RANSAC algorithm to find walls of room, as the probabilistic nature of the algorithm, and the fact that noise of the data can very easily make the solution insufficient, unless we use more computational power for the algorithm, which will then might make the algorithm insufficient computationally.
As far as we have looked at it, the only way to make a more deterministic algorithm that has a different nature then ours, is to use methods such as gradient descent on all of our data. This would obviously make our algorithm even more insufficient. Because of that we have decided to stick with our original algorithm of RANSAC to find walls.
As to find the clusters of exit points, the best algorithm we currently have found is an algorithm known as DBSCAN, which iteratively creates and expands clusters by determining if the number of points surrounding current clusters is high enough. This algorithm has the advantages of creating the clusters based only on their densities and removing the noise from the data.

Implementation of algorithms:

As of this week, we have managed to code the algorithm beside the point clustering, which means that we can assume that in a day or two we would have completely

implemented our algorithm, and after that we can move to testing and improving the baseline of our algorithm.

Our plans for the upcoming week:

Implement getExitPoints – ask Dan

Visualisation + (getDepth – ask Dan) + tests

What we need from you:

We'd like to have a precise due date and expectations of the result of our project this semester.