Group 3 Final Report

Section A:

Game concept: To what extent did your game concept change from initial concept to what you implemented? If it did change, how did it change and why?

Our initial game concept was largely implemented. We envisioned a multi-player maze-based battle royale that relied heavily on aspects of the maze to produce interesting gameplay. The style of gameplay we initially imagined was implemented and we were able to add many maze-based features such as footprints, proximity-sound, and maze-altering abilities. There were some ideas we had early on such as disappearing walls, weapon types, reloading, and ammo limits that we ended up not having time to implement or ran into issues with during testing, but we believe we got close to the game we brainstormed in Week 1.

Design: How does your final project design compare to the initial design, and what are the reasons for the differences, if any?

From the beginning we liked the idea of having a simple, geometric style for our playable characters and environment because we would be able to design it in the amount of time we had, and we liked the aesthetic. We accomplished this in our final project design.

Schedule: How does your final schedule compare with your projected schedule, and what are the reasons for the differences, if any? (You should be able to glean this from your status reports.)

We followed the order of the schedule, but around Week 4, when it came time to integrate, things got delayed. It took a few weeks to completely finish integration due to our lack of design planning beforehand and the complexity of the game. We also did not coordinate on things like compilers and project set up environments, so when it came time to build integrated versions of the project in Visual Studio, we faced individual issues that took an unnecessarily long time to sort through.

Section B:

What software methodology and group mechanics decisions worked out well, and which ones did not? Why?

Our code followed the separation of concerns design principle, which made it easier to test and discover issues. We also had a good choice of initial libraries. Boost worked well for networking, and the model loading library did the job.

In retrospect, we should have had more frequent group meetings. We always met at least once a week, and had smaller meetings in between when needed, but more regularly scheduled meetings likely would have helped us coordinate better as an entire team and be more efficient overall. Better coordination on environment setup and design structures would have led to an easier integration process and less delays in the middle of development.

Which aspects of the implementation were more difficult than you expected, and which were easier? Why?

Integration was more difficult than expected due to our team using different programming environments. Loading in 3D models using ASSIMP was also much more difficult than expected, especially for the animations. There was very little online documentation on how to play back animations for specific file types. Separating the code into client and server responsibilities was also more difficult than expected because almost all of our code was written initially on the client. We also ran into some confusion and misunderstanding within the team about what needed to be client-side and what needed to be server-side. Another issue we encountered was with our network message delimiters due to misunderstanding Boost documentation.

Initial 3D world generation was much faster than expected.

Which aspects of the project are you particularly proud of? Why?

We are proud that we were able to create a fun game that basically followed our vision from the start. We are also happy with depth we were able to add to the game, such as footsteps, proximity audio, and abilities. Some of the models and animations, such as the chests, look nice and are great additions to the game.

What was the most difficult software problem you faced, and how did you overcome it (if you did)?

Our differing environments and Visual Studio setups prevented us from effectively working on the project for a long time. Many of us weren't even able to build the project for weeks because the setup would take hours and was far too machine specific. These issues slowed down our development and shifted our attention from mechanic-building to tedious configuration sorting. We overcame it by debugging each person's configuration manually, switching over to alternate installation methods, and streamlining our installs.

If you used an implementation language other than C++, describe the environments, libraries, and tools you used to support development in that language. What issues did you run into when developing in that language? Would you recommend groups use the language in the future? If so, how would you recommend groups best proceed to make it as straightforward as possible to use the language? And what should groups avoid?

We used C++.

How many lines of code did you write for your project? (Do not include code you did not write, such as library source.) Use any convenient mechanism for counting, but state how you counted.

We wrote 7,856 lines of code for the project. We counted GitHub file line counts.

In developing the media content for your project, you relied upon a number of tools ranging from the DirectX/OpenGL libraries to modeling software. And you likely did some troubleshooting to make it all work. So that students in future years can benefit from what you learned, please detail your tool chain for modeling, exporting, and loading meshes, textures, and animations. Be specific about the tools and versions, any non-obvious steps you had to take to make it work (e.g., exporting from the tool in a specific manner), and any features or operations you specifically had to avoid — in other words, imagine that you were tutoring someone on how to use the toolchain you used to make it all work. Also, for the tools you did use, what is your opinion of them? Would you use them again, or look elsewhere? Are there any tools that you used but, looking back, you would avoid?

We used OpenGL as our library to render our 3D world. We used ASSIMP to load in models and other 3D assets and animations. All of our models were exported in a GLTF format and were created in Blender. When exporting in the GLTF format from Blender, we made sure to choose the "apply modifiers" option in order to show the mirrored models. I used LearnOpenGL for almost all things model related, specifically these links were super useful. For ASSIMP data structure documentation, http://assimp.sourceforge.net/lib_html/data.html. How to load in models using ASSIMP, https://learnopengl.com/Model-Loading/Model. How to play animations using skeletons and ASSIMP, https://learnopengl.com/Guest-Articles/2020/Skeletal-Animation (skeletal animation using assimp). The ASSIMP documentation is not the best but I think it's a great tool for loading 3D models for a large number of file extensions. There is basically handling for all modern 3D model file extensions. I would recommend using the GLTF format for exporting models as these files can contain animations in addition to regular object models. We also had some issues using ASSIMP with other file types (.fbx, etc.).

We used IrrKlang for audio. The IrrKlang documentation and tutorials give a pretty comprehensive overview of how to work with 3D sounds.
https://www.ambiera.com/irrklang/tutorial-3dsound.html

Would you have rather started with a game engine or would you still prefer to work from scratch?

There were no practical advantages for us starting from scratch but writing code that would normally be taken care of by the game engine was a great learning experience. We would recommend future classes continue working from scratch instead of starting with a game engine.

For those who used a networking library (e.g., RakNet or Boost), a physics library (e.g., Bullet), or a GUI library, would you use it again if you were starting over knowing what you know now? Describe any lessons you learned using it (problems that you had to troubleshoot and how you addressed them) for future groups who may use it. If you did not use a library, judging from the experiences of the groups that did, would you have used it in retrospect?

We used the Boost library (ASIO). We would use it again. It is a powerful library with all the required functionality for our project. We would like to warn future groups that the documentation for the Boost library is lacking. There is rarely an issue with Boost that StackOverflow cannot help with, but it can still be a little frustrating at times.

For your group web pages, we used wordpress. Were you satisfied with wordpress, or would you rather have used some other system for maintaining your group web pages? If you had a choice, what other system would you prefer to use (or even just doing everything on your own using HTML, CSS, and JavaScript)?

We used our own raw HTML, CSS, and JavaScript. For our simple site, it worked well, it was good practice, and it was pretty fun to do.

Edward introduced using a discord server for the class. Would you recommend that we continue using discord in the future (e.g., even if the next class is in person)?

Having a class discord server is a good idea. During the course, our group helped out another team with issues rendering animations and later another group helped us out with we problem we were facing. Since the entire class doesn't meet in person on a regular basis, having a place online for groups to collaborate and discuss common issues may be good.

This quarter was particularly difficult having to do everything remotely. What tools and strategies did you use to collaborate, communicate, and work together?

We used a discord server to message, talk, and collaborate. Email was used to send in weekly updates and screenshots. We had weekly meetings with the entire group in order to sync up on progress, strategy, and concerns. If there was a problem which needed to be addressed in between the meetings, we would have one-on-one meetings where we could help each other solve the issue.

What lessons about group dynamics did you learn about working in such a large group over an extended period of time on a challenging project?

Constant communication is key. We did a good job of being active on Discord, responding to each other when questions came up. We also shared documentation with each other, which helped keep everyone on the same page, especially when it came time to integrate different portions of the project.

Looking back over the past 10 weeks, how would you do things differently, and what would you do again in the same situation?

We would dedicate time at the beginning of the course of coordinate our environment, compilers, and setup. This would have helped us avoid a lot of issues later in the course when we had to spend valuable time figuring out what was wrong with each individual person's machine. We would also hold more regularly scheduled group meetings so we could tackle challenges and discuss differences in understanding, allowing us to work more efficiently.

Which courses at UCSD do you think best prepared you for CSE 125?

For project design, CSE 110 and CSE 112. For graphics, CSE 167, CSE 168, CSE 169. For networking, CSE 123, CSE 124. CSE 291 (Physics Simulation) was also helpful.

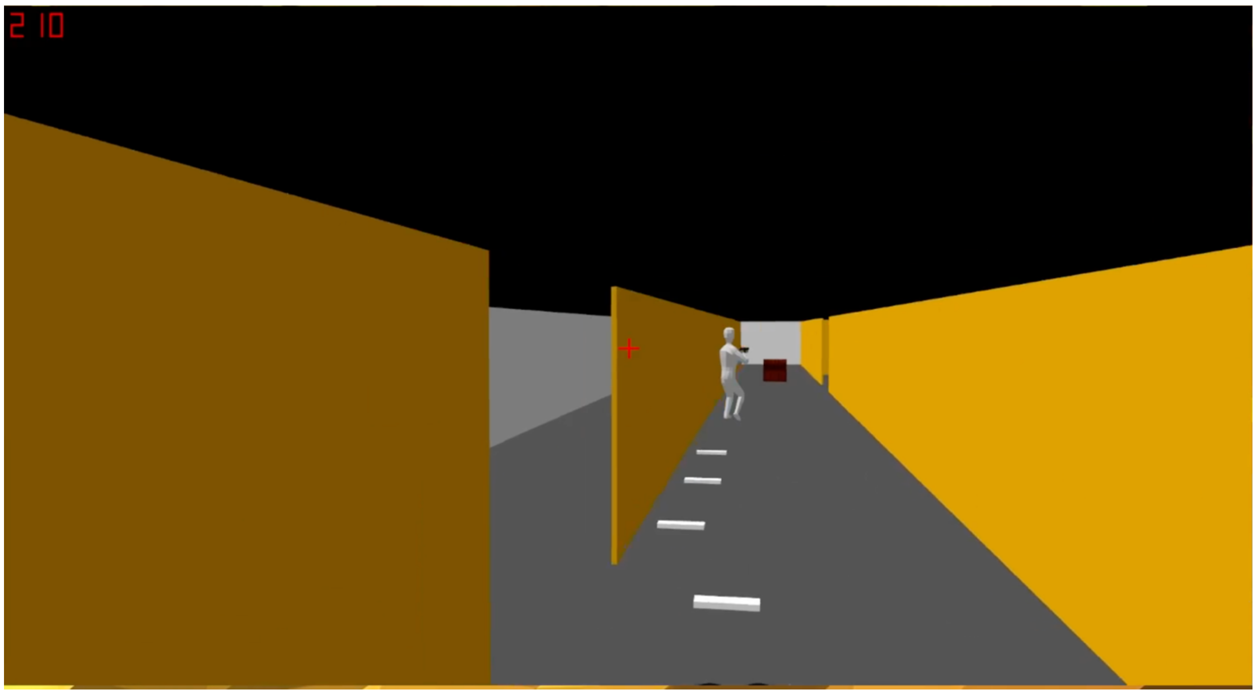
What were the most valuable things that you learned in the class?

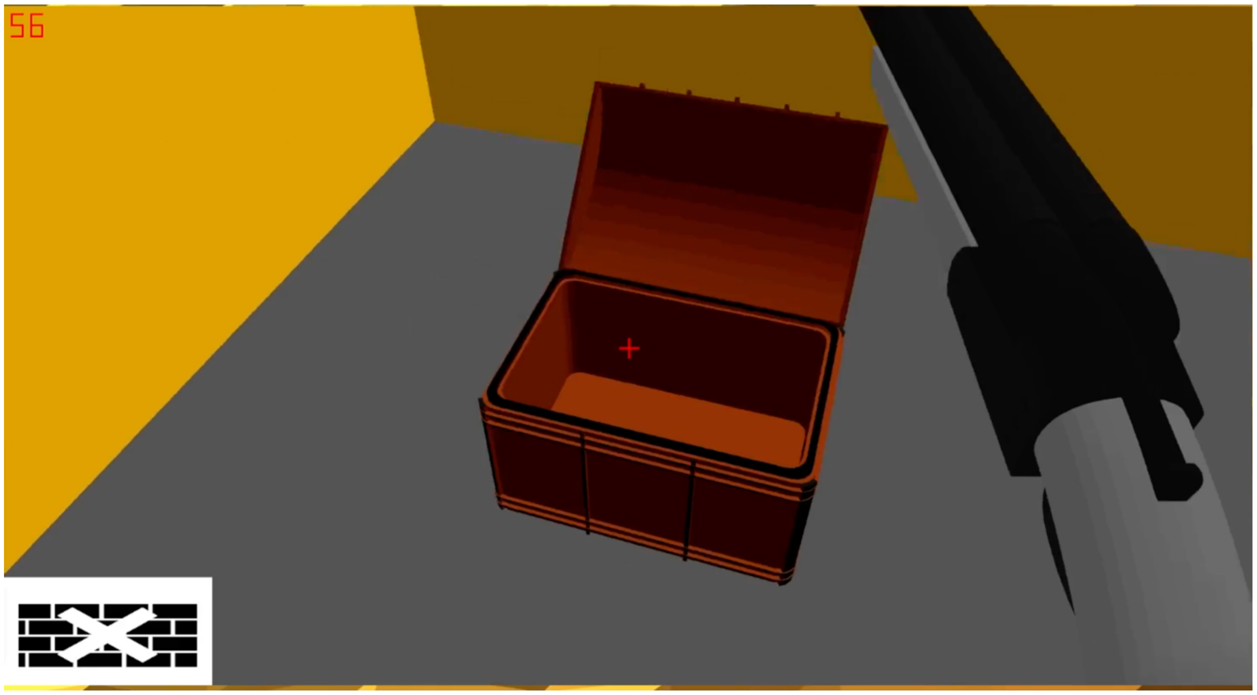
In this class we gained experience in communicating well as a team, coordinating tasks, and planning a long-term project.

[Hwang] Loading in models and playing animations.

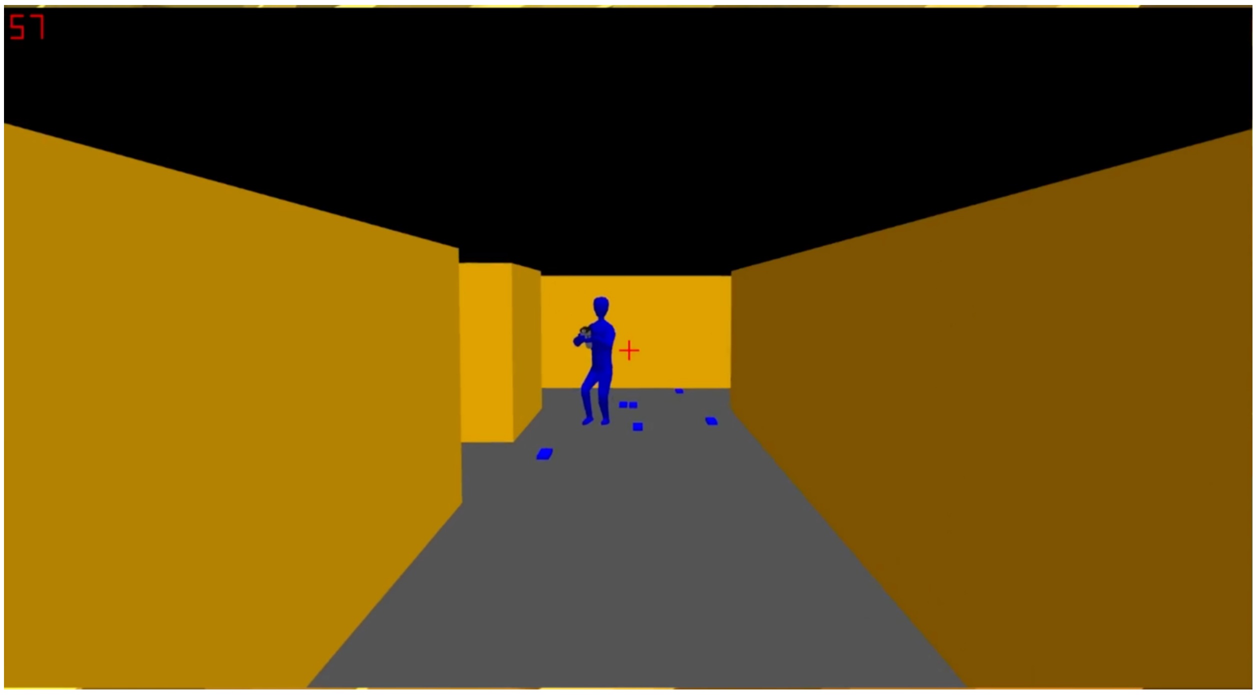
[Nagrecha] Visual Studio.

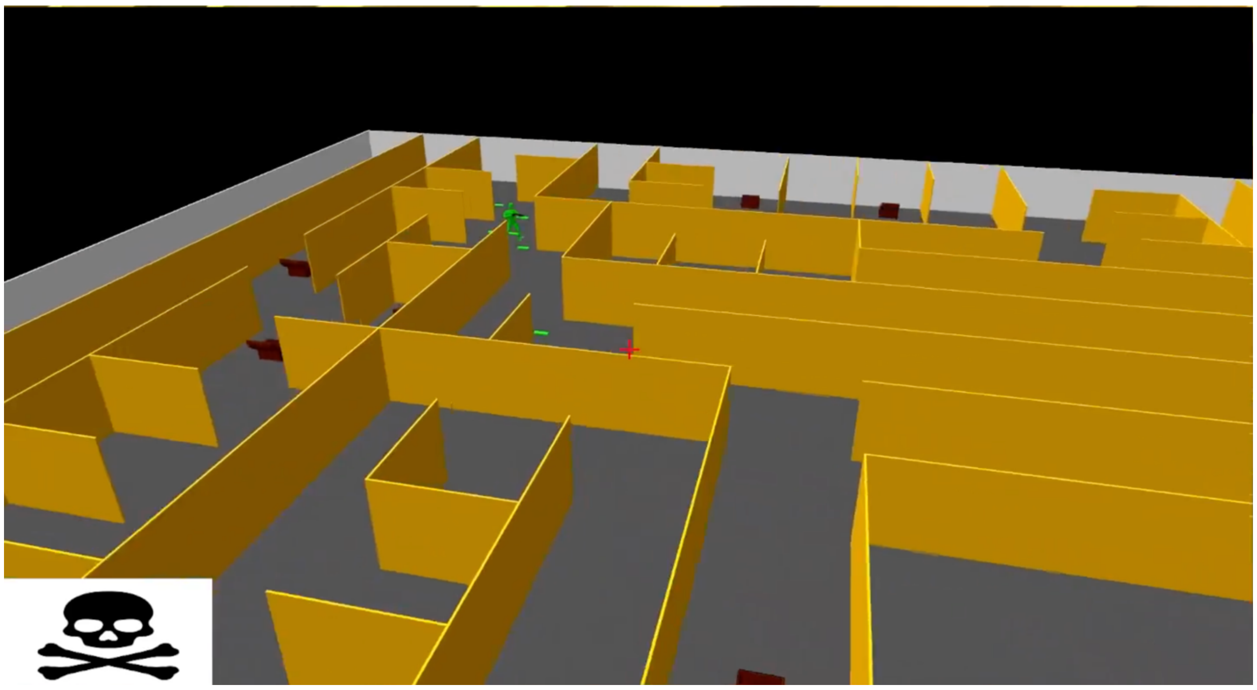
Please post four final screenshots of your game on your group pages for posterity. I will display them on the group web page.




56

.


57



Section C:

What advice/tips/suggestions would you give students who will take the course next year?

Coordinate early and communicate often! Also, test things out on the demo machine as soon as you can, just to make sure things are working the way you expect on unfamiliar machines.

How can the course be improved for next year?

If it's possible, it would be nice if there were more demo machines available in the last few weeks. It gets a little crowded towards the end of the quarter with everyone testing out last minute changes before the demo.

Also, our group also liked using Parsec, so we would recommend having Parsec be the standard instead of VNC.

Any other comments or feedback?

Thank you Professor Voelker and Edward for a great quarter!