DES310 - Professional Project

Name	: Hristo Vuchev (Ritchie)
Student Number	: 1703871
Role	: Gameplay Programmer; Tools Programmer; Designer
Team Name	: The Team
Brief	: #10 - Fight For Freedom

Introduction

A big part of the research done for the project fell on the programmers. Due to the nature of the brief and client, the team was already provided with a fleshed out narrative and setting for the game, as well as a historian if any of the designers had questions related to the time period.

Gameplay Mechanics wise, the team was tasked with researching, designing and implementing two core mechanics. A time rewind mechanic and a conversation mechanic/tool. For those two core mechanics, the programmers decided to split into pairs of two for each mechanic, the author of this report was on the team that worked on the conversation mechanic. On top of those core mechanics there were other tasks and mechanics that needed to be implemented. Tasks such as a player controller, a camera system, an interaction system, an event system and a “looped edges” mechanic. These tasks were split among the programmers based on personal interest and free time. If someone did not have enough work on their core mechanic one week, there were always other tasks they could work on. Everything was communicated through Discord with almost daily calls and meetings, as well as chats.

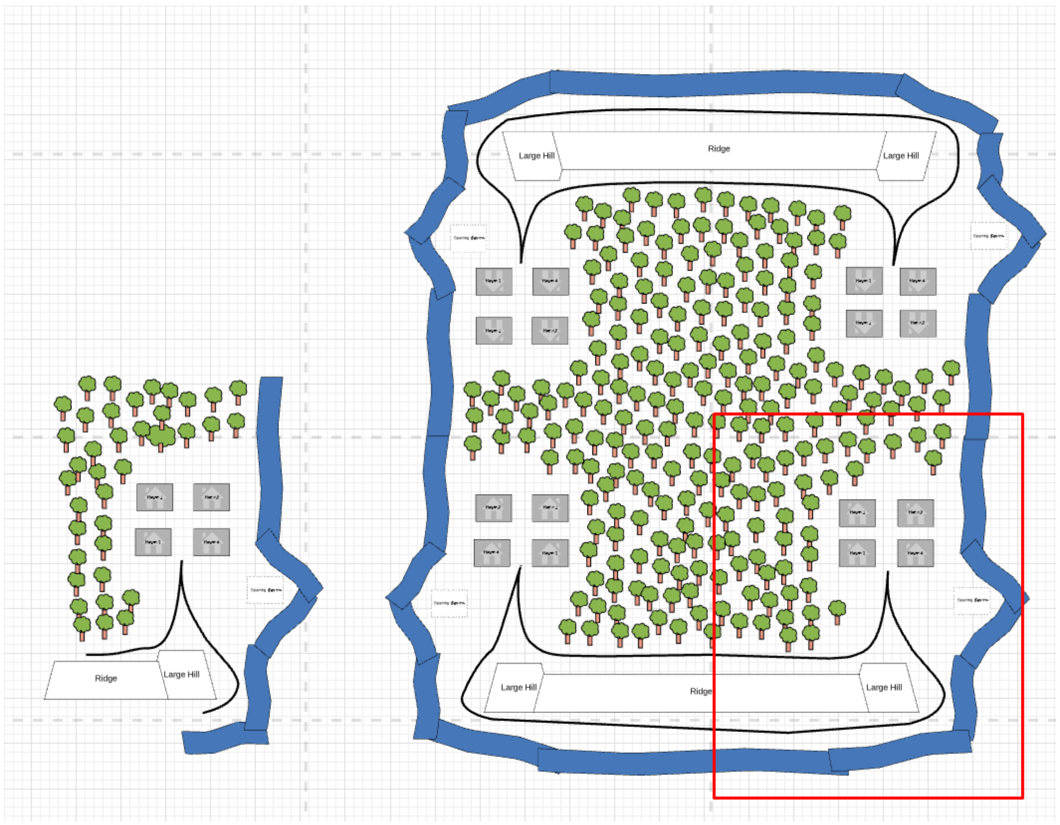
The author has worked on a couple of different mechanics, namely the conversation mechanic/tool, the “looped edges” mechanic, the conversation event system and a debug console. On top of that he has also taken up different roles within the team. He has designed and maintaining the “Coding Standards” document, created the “Git Basics” document and gave a crash course to the team about how to use Git, and due to his previous professional experience with Git, was the person who maintained the repository and helped fix issues that arose. The author also made the blog website which was used to communicate and showcase to the client the progress of the team during weeks where there were no scheduled calls.

Research

Research and prototyping started very early for the team. Due to the client having a clear vision, the team had the chance to start working and iterating quite early. The first 8 weeks were spent researching, prototyping and iterating on the core gameplay mechanics. For the author that meant working on the "looped edges" and the underlying architecture of the conversation mechanic.

Looped Edges

The looped edges mechanic is a mechanic which due to various reasons could not make it into the final build. A lot of research was conducted by the author, but there are limited resources considering such a mechanic has not been implemented in existing games. The goal of the mechanic is to avoid blocking the player from traversing the world using conventional methods such as invisible walls and blocked off paths where it does not make logical sense. For example if the player headed West through the forest and did not stop, he should return to the side of the village where he left. In a sense, a mirrored copy of the village. As shown in the diagram provided by the client below. The small diagram to the left is the village, the big on the right is the village mirrored in all 4 possible directions and where the player should end up if he traversed through the forest. (The river and mountain area should be blocked off)



Initial theory was scaling all game objects by -1 in the appropriate axis, but after research it was made clear that such a method would also invert the normals of the object, and a custom script would have had to be made to recalculate them. That was agreed by the programmers that would be too performance costly, due to it having to be done at runtime.

The next theory and suggestion that was read was rotating and flipping the world instead of scaling it. A demo prototype was also successfully made, but was not implemented due to issues that will be discussed in the “PLANNING & PROTOTYPING WORK” section of the report. A big reason for that was that the further away the world was from the origin, the more precision problems there were, such is the nature of computer science. Those precision problems would break the illusion of the player and as the team learned later in development, would not have worked due to the way the cutscenes are implemented.

The third theory that was researched and attempted was manipulating the player and taking away control from him for a split second in order to reorientate him in the world and for him to go back into the village. That theory did not get far, due to it was unanimously agreed by the team after a prototype was made, that taking control away from the player is just a bad idea.

The final theory which got further into development than the others was to recreate the world as it is in the above diagram, have 4 copies of the world and use unity’s occlusion culling and LOD’s to control what is rendered and when in order to make it performance efficient. That theory fell short due to the amount of extra work needed, and lack of time of the designers to do it. It was also going to be impossible within the time scope of the project to accomplish due to the way the cutscenes are implemented, quadrupling the workload of the team members was just not a smart decision and ultimately the mechanic was scrapped. The difficulties of implementing the mechanic as well as the time scope of the project were discussed with the client and he was on the side of the team. Thus the mechanic was scrapped from the final build.

Conversation Mechanic/Tool

On the conversation mechanic worked the author and Jordan Glendinning. Research conducted was looking into how other people have implemented a conversation mechanic. The author and Mr. Glendinning agreed that it would be best to create their own mechanic and tool from scratch over using an already existing one. Main reasons were that a custom tool would allow for better control over the mechanic and the data. Which was highly valued at the start of the project due to the team not knowing how and what the mechanic will need to be able to do by the end of the project. Creating the mechanic and tool from scratch allowed the team to adapt and iterate over it as it suited the project and eventually ended up in a very stable and designer friendly state, which was the goal. The conversation type chosen is a Node based conversation mechanic, due to its designer friendliness and scalability compared to alternatives (XML/EXCEL/Markup based). Node based also fit the best with the narrative behemoth which was given by the client. It was already structured in nodes and that made translating it into the project easier.

A lot of the early research and push towards the design of the architecture of the mechanic was done by the author. Namely how to control, design, structure, store and load the narrative data behemoth the team had been given and make it scalable. The author successfully convinced Mr. Glendinning in using “JSON” to store the data over alternatives. Reasons for that were good integration with unity for reading and writing to JSONs, the scalability of JSONs, the human readable factor of JSON (which was very important prior to having the graphical tool to visualise the conversations) and the prior experience of the author with JSONs. Although the JSONs are only data containers, there needed to also be a way to visualise and use that data inside Unity. Initially that was done using a class that purely held the data. It served as a middleman between the JSON and the conversation UI. Later it was agreed that Scriptable Objects would serve better as the data containers due to their flexibility. The ConversationSO was made as a template to contain the conversation data, then in Unity with a single press you could quickly create a new conversation and fill it with the needed data, which would then be saved inside its own JSON. Using a class as a conversation data template would have not been as easily achievable.

- Scriptable Objects tutorial used to get it up and running can be found here.
- Tutorial used on reading from JSON in Unity can be found here.
- Tutorial used on writing to JSON in Unity can be found here.

Debug Console

After the first build of the project it was apparent that the team needed a way to reset conversations inside the build, due to not having access to the editor. At that point the author took initiative and implemented a debug console to help with that. The debug console is heavily based on *this* guide, but the code has been adapted to work with the systems of the project. It proved handy when testing builds, but was not used beyond resetting conversations.

Coding Standards Document

The document was based off of this document provided by Jason Weimann and then further extended with the structure and branches in the git repo, how to document the codebase and good practises. The Branch Structure was designed by the author following the methodology he had used professionally prior to the module. The structure decently represents how it could look like in the IT industry as well as it is designed to avoid merge conflicts and “breaking” the project as much as possible using Git.

The Documentation section is based off of the documentation found in the released Command and Conquer Red Alert (1996) and Tiberian Dawn (1995) source code by the creators of the franchise - Electronic Arts. Source code can be found here. The style was adapted to better serve the needs of the project, fields such as “Start Date” and “Last Update” were removed due to that data being available in the github repository.

The “Some good practises” section was just a collection of handy tips due to not all members of the team having had used Unity before and to get them started.

Git Basics Document

No research was done in creating the document due to the familiarity of the author with Git and GitKraken. The goal of the document was to provide a handy guide to designers and artists on the basics of git and gitkraken. GitKraken was chosen as a GUI for the team over Github Desktop purely due to the amount of extra useful features it provides. As well as how much it simplifies basic workflow such as pulling/pushing/merging.

A copy of the document can be found inside the documents folder along with the submission of this report.

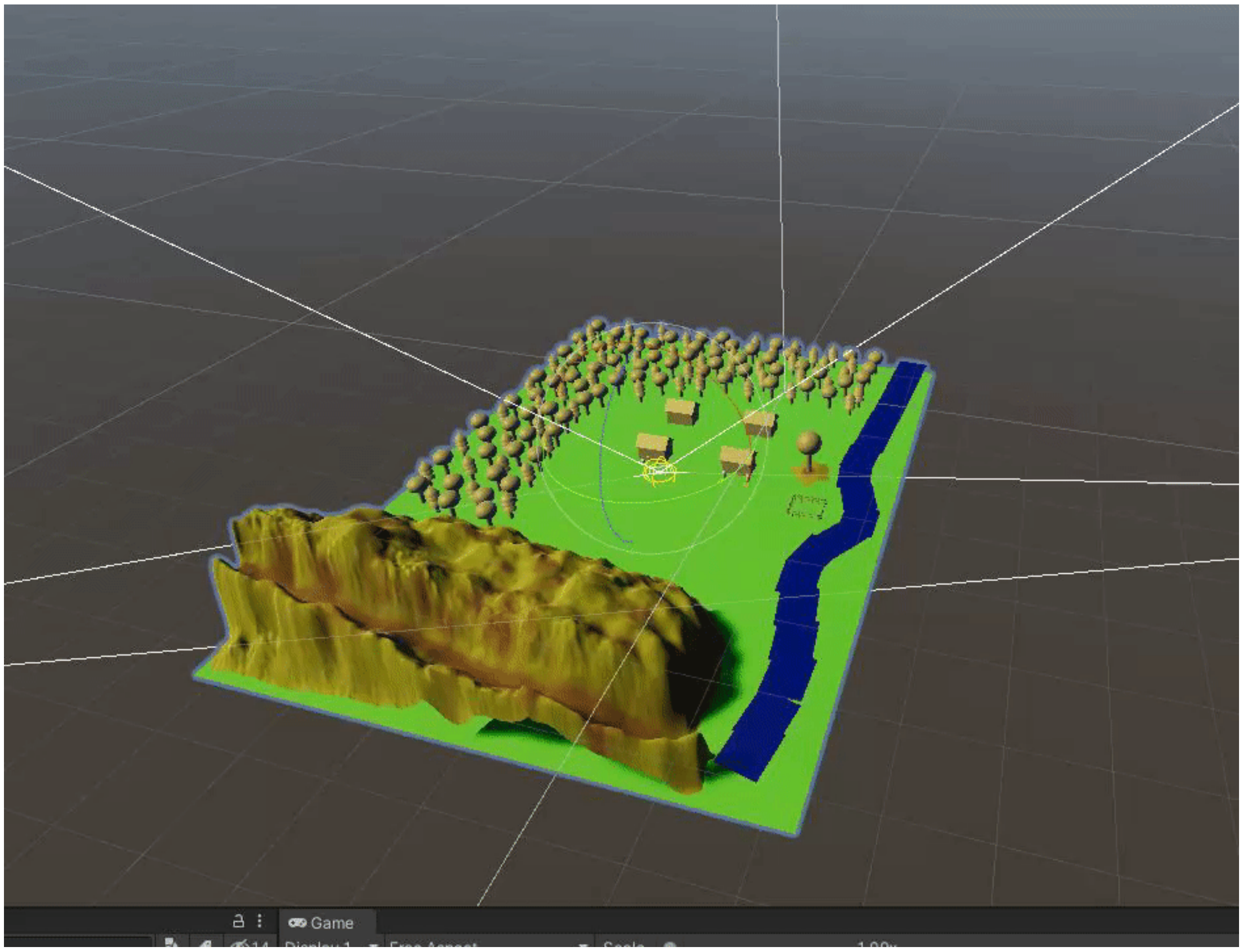
Fight For Freedom Blog

The blog website was created using Hugo and hosted on Netlify. No research was needed for the creation of the blog due to the author being familiar with the technologies and had used them professionally.

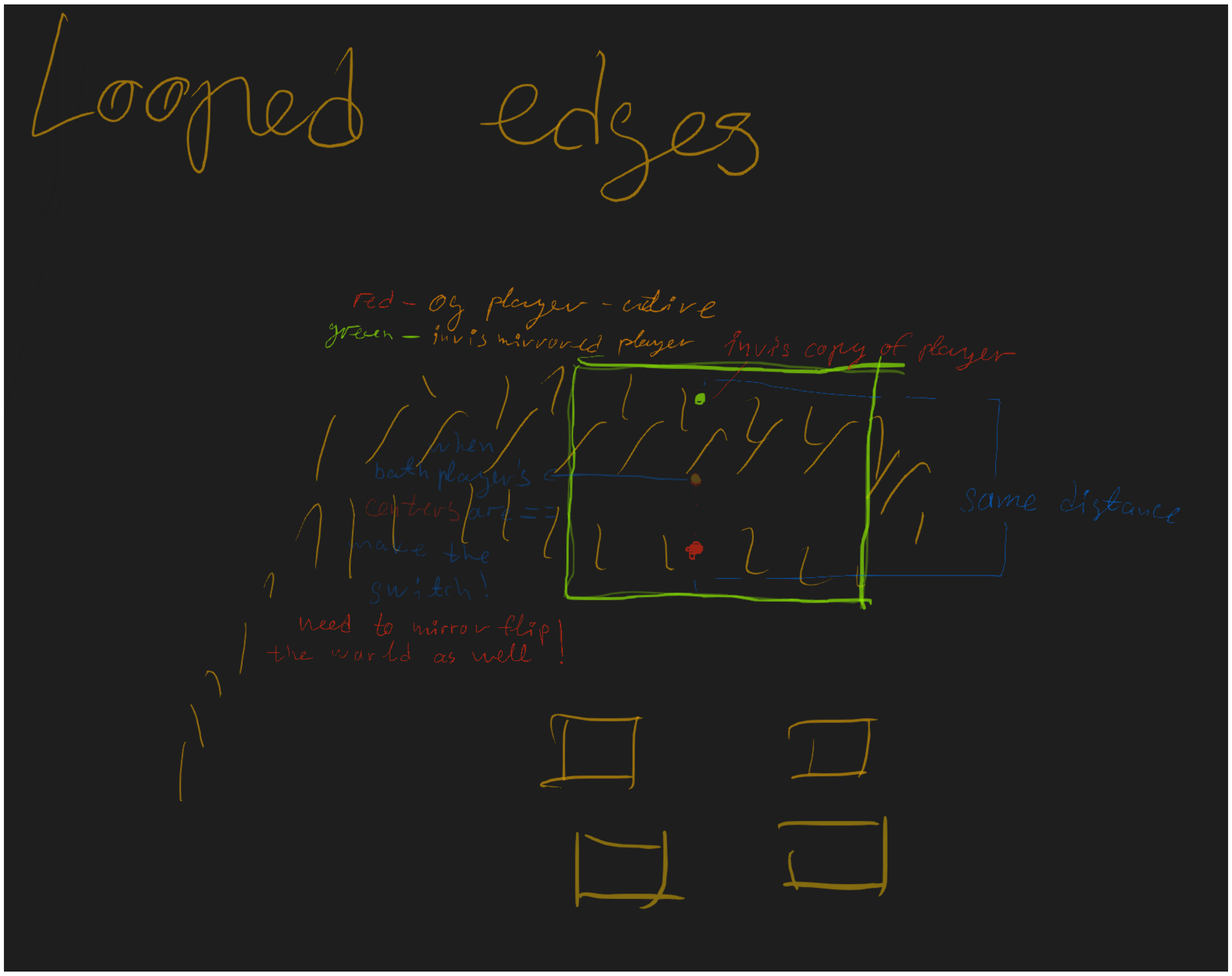
- Link to the blog can be found here.

Prototyping & Development work

Looped Edges

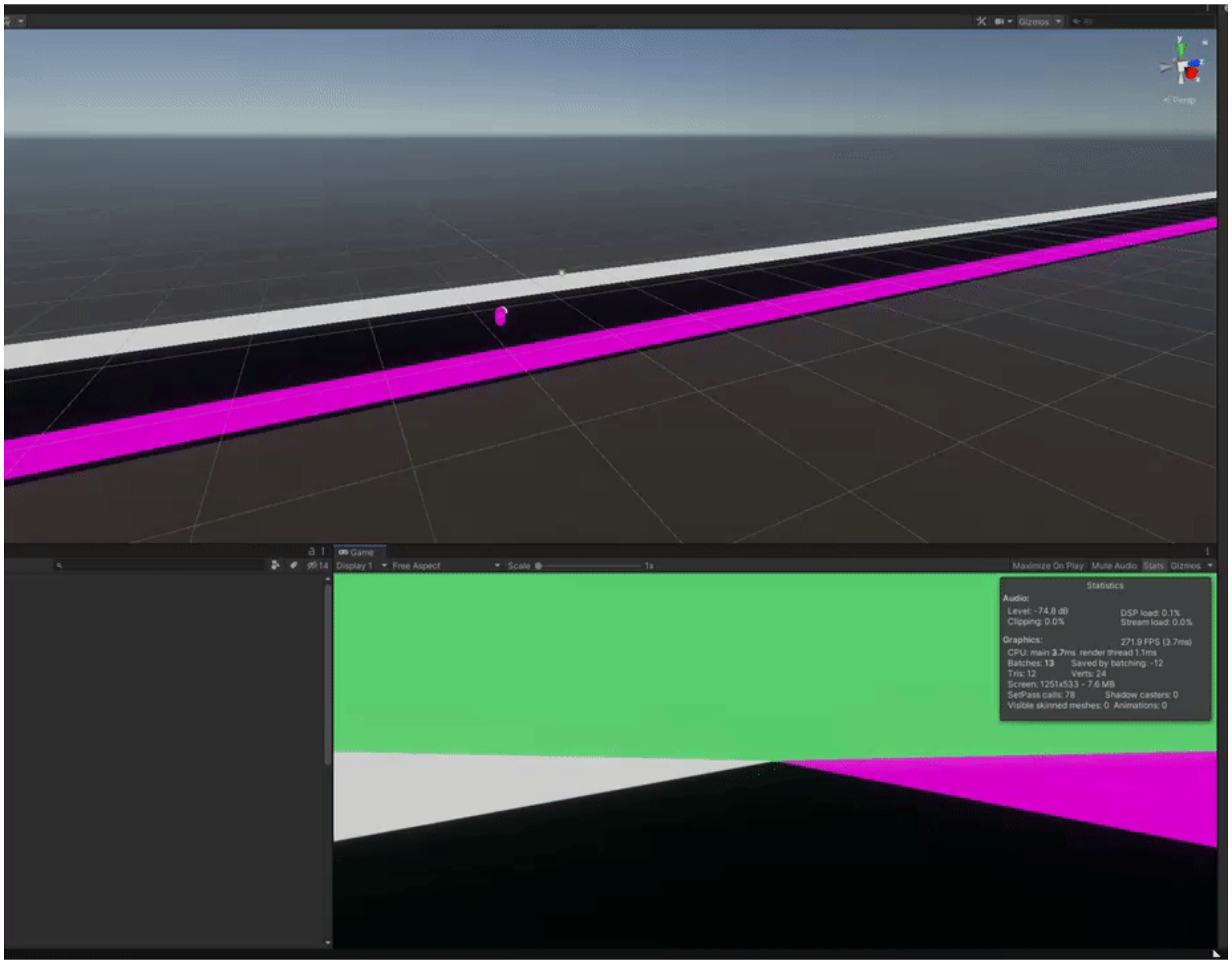


The Gif above showcases one of the issues that were encountered while researching and attempting a rotation based approach for the looped edges mechanic. The issue was related to how Meshes and Terrain are treated in Unity. This did not make the final build.



Looped edges
red – og player – active
green – invis mirrored player
invis copy of player
when both player's centers are ==
make the switch!
need to mirror flip the world as well!
same distance

Another theoretical attempt at the mechanic was having a 2nd invisible player copy the exact movements of the player when near the edges. When the 2 players overlap, switch their positions. That was scrapped due to how the client wanted the mechanic to work, in this case that would not preserve the perspective of the player. An example is that if while walking you had a river on your right hand side, when the switch happened, that river would then be on your left hand side, which is not how the mechanic was intended to work by the client. Thus that approach was scrapped.



```
Statistics
Audio:
  Level: -74.8 dB              DSP load: 0.1%
  Clipping: 0.0%               Stream load: 0.0%

Graphics:                      271.9 FPS (3.7ms)
  CPU: main 3.7ms  render thread 1.1ms
  Batches: 13       Saved by batching: -12
  Tris: 12          Verts: 24
  Screen: 1251x533 - 7.6 MB
  SetPass calls: 78          Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

The above gif showcases the most promising attempt that was made for the looped edges. The issue which is not showcased in the above gif is that when this technique was implemented in the demo scene at the time, due precision issues in how floating point operations work in computers, every object would be just “a bit” off of where it should be. Instead of rotating 180 degrees it would rotate 179.3 for example. Over time that would add up and make the world more and more off from its intended orientation in the world.

A solution which could be explored further for future iterations of the project was found recently by the author. This video quickly goes over how the developers of the games Dishonored (2012 and 2016) worked around a similar in nature issue. Although such a solution would still heavily depend whether or not the below mentioned issue with the cutscenes can be resolved.

Despite the above issues and iterations the mechanic was ultimately scrapped for the scope of the project due to a bigger issue involving how the Cutscenes were ultimately implemented. The issue is that after a cutscene had been created, **if an object in the**

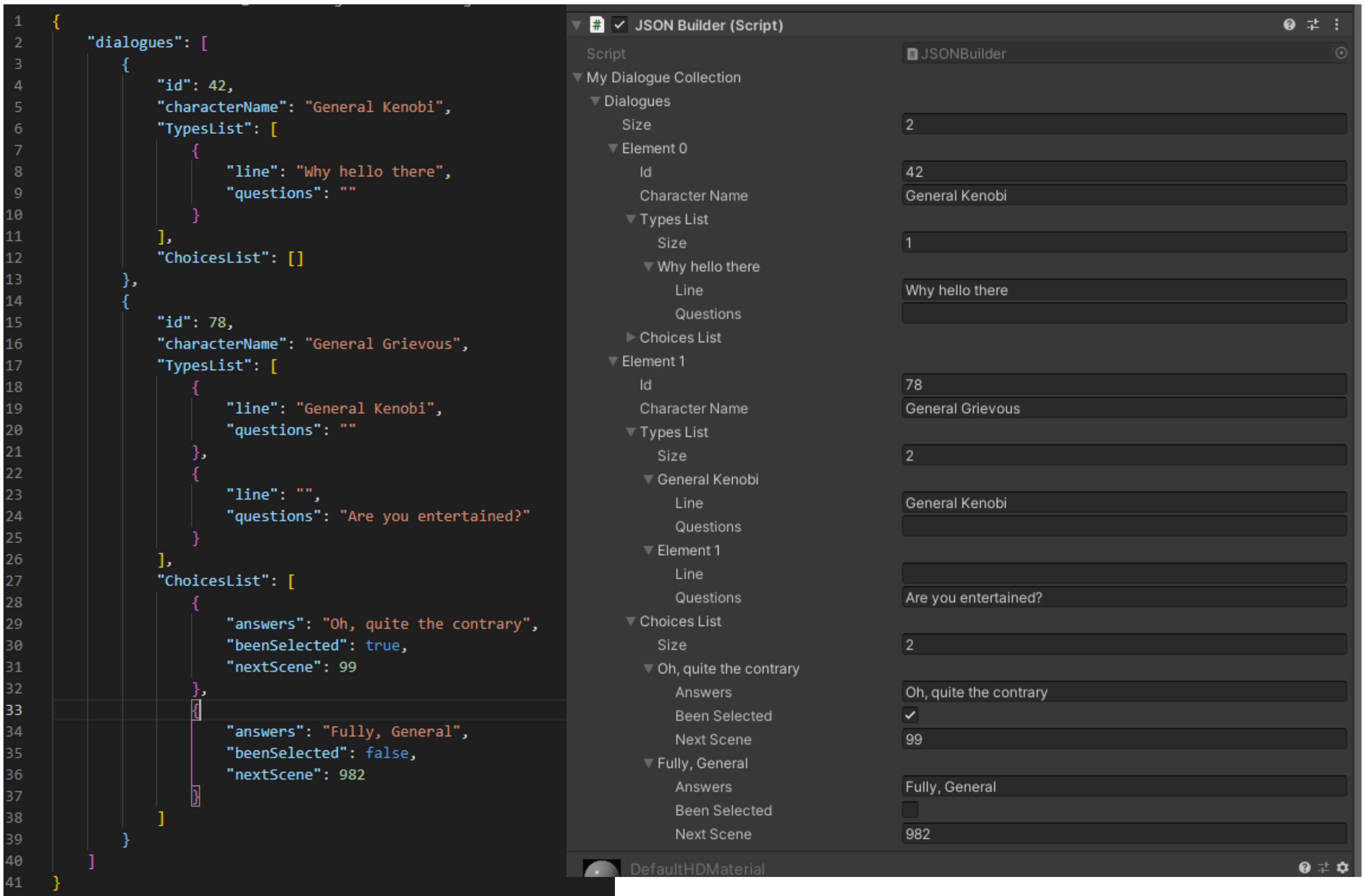
cutscenes was moved in any way it would break the entire cutscene. Due to that reason any implementation of the looping edges that involve manipulating the game world were just not possible. And player manipulation approaches to the mechanic conflict with the flow and gameplay, because the game would have to temporarily take control of the camera.

Conversation Mechanic, Tool and Event System

Conversation Mechanic and Data

The conversation mechanic followed a very iterative approach. It was the first time the author and Mr. Glendinning had undertaken making such a mechanic. The main factor that contributed to the successful implementation was that both members were willing to stay in daily calls together and iron out the underlying architecture and adapt the system to the requirements of the project. Despite going through numerous iterations, sitting down for a couple of hours to discuss how the current iteration could be improved gave clarity when the members sat down to work on implementing the plan. No stone was left unturned and the system undertook numerous re-designs.

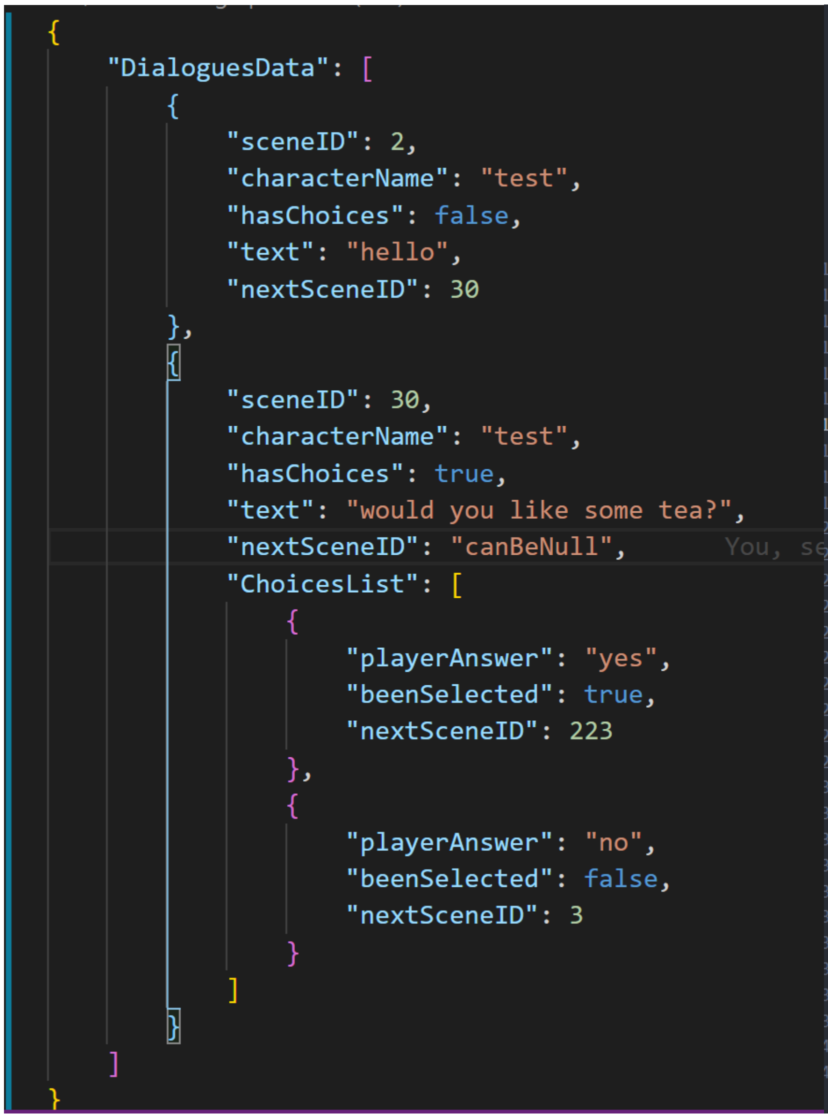
Below are images of some of the sketches used in designing the system as well as some of the gifs at different stages of the process:



```
{
    "dialogues": [
        {
            "id": 42,
            "characterName": "General Kenobi",
            "TypesList": [
                {
                    "line": "Why hello there",
                    "questions": ""
                }
            ],
            "ChoicesList": []
        },
        {
            "id": 78,
            "characterName": "General Grievous",
            "TypesList": [
                {
                    "line": "General Kenobi",
                    "questions": ""
                },
                {
                    "line": "",
                    "questions": "Are you entertained?"
                }
            ],
            "ChoicesList": [
                {
                    "answers": "Oh, quite the contrary",
                    "beenSelected": true,
                    "nextScene": 99
                },
                {
                    "answers": "Fully, General",
                    "beenSelected": false,
                    "nextScene": 982
                }
            ]
        }
    ]
}
```

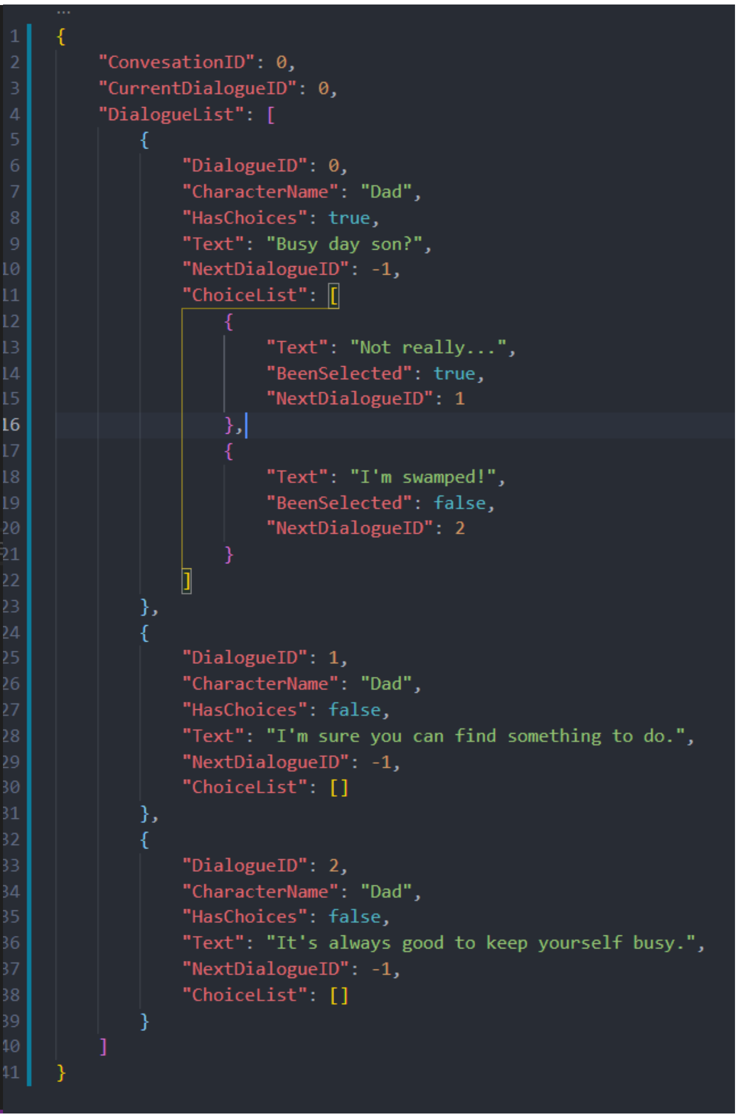
```
JSON Builder (Script)
Script                          JSONBuilder
My Dialogue Collection
  Dialogues
    Size                        2
    Element 0
      Id                        42
      Character Name            General Kenobi
      Types List
        Size                    1
        Why hello there
          Line                  Why hello there
          Questions
      Choices List
    Element 1
      Id                        78
      Character Name            General Grievous
      Types List
        Size                    2
        General Kenobi
          Line                  General Kenobi
          Questions
        Element 1
          Line
          Questions             Are you entertained?
      Choices List
        Size                    2
        Oh, quite the contrary
          Answers               Oh, quite the contrary
          Been Selected         ✓
          Next Scene            99
        Fully, General
          Answers               Fully, General
          Been Selected
          Next Scene            982
DefaultHDMaterial
```

The above images are early prototypes of how the json was structured and how it was visualised inside the Unity Editor using a C# script as a data container.



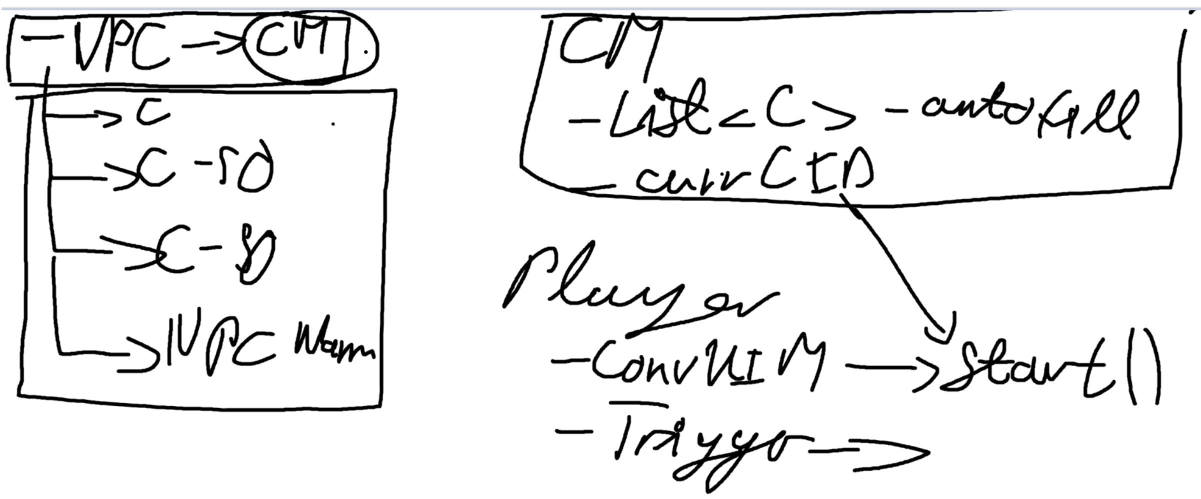
```
{
    "DialoguesData": [
        {
            "sceneID": 2,
            "characterName": "test",
            "hasChoices": false,
            "text": "hello",
            "nextSceneID": 30
        },
        {

            "sceneID": 30,
            "characterName": "test",
            "hasChoices": true,
            "text": "would you like some tea?",
            "nextSceneID": "canBeNull",      You, se
            "ChoicesList": [
                {
                    "playerAnswer": "yes",
                    "beenSelected": true,
                    "nextSceneID": 223
                },
                {
                    "playerAnswer": "no",
                    "beenSelected": false,
                    "nextSceneID": 3
                }
            ]
        }
    ]
}
```

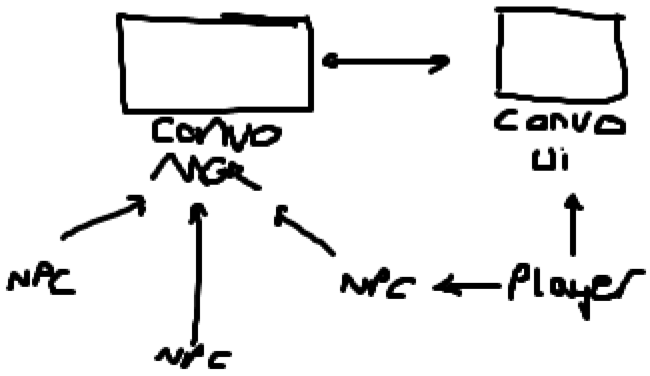


```
{
    "ConvesationID": 0,
    "CurrentDialogueID": 0,
    "DialogueList": [
        {
            "DialogueID": 0,
            "CharacterName": "Dad",
            "HasChoices": true,
            "Text": "Busy day son?",
            "NextDialogueID": -1,
            "ChoiceList": [
                {
                    "Text": "Not really...",
                    "BeenSelected": true,
                    "NextDialogueID": 1
                },
                {
                    "Text": "I'm swamped!",
                    "BeenSelected": false,
                    "NextDialogueID": 2
                }
            ]
        },
        {
            "DialogueID": 1,
            "CharacterName": "Dad",
            "HasChoices": false,
            "Text": "I'm sure you can find something to do.",
            "NextDialogueID": -1,
            "ChoiceList": []
        },
        {
            "DialogueID": 2,
            "CharacterName": "Dad",
            "HasChoices": false,
            "Text": "It's always good to keep yourself busy.",
            "NextDialogueID": -1,
            "ChoiceList": []
        }
    ]
}
```

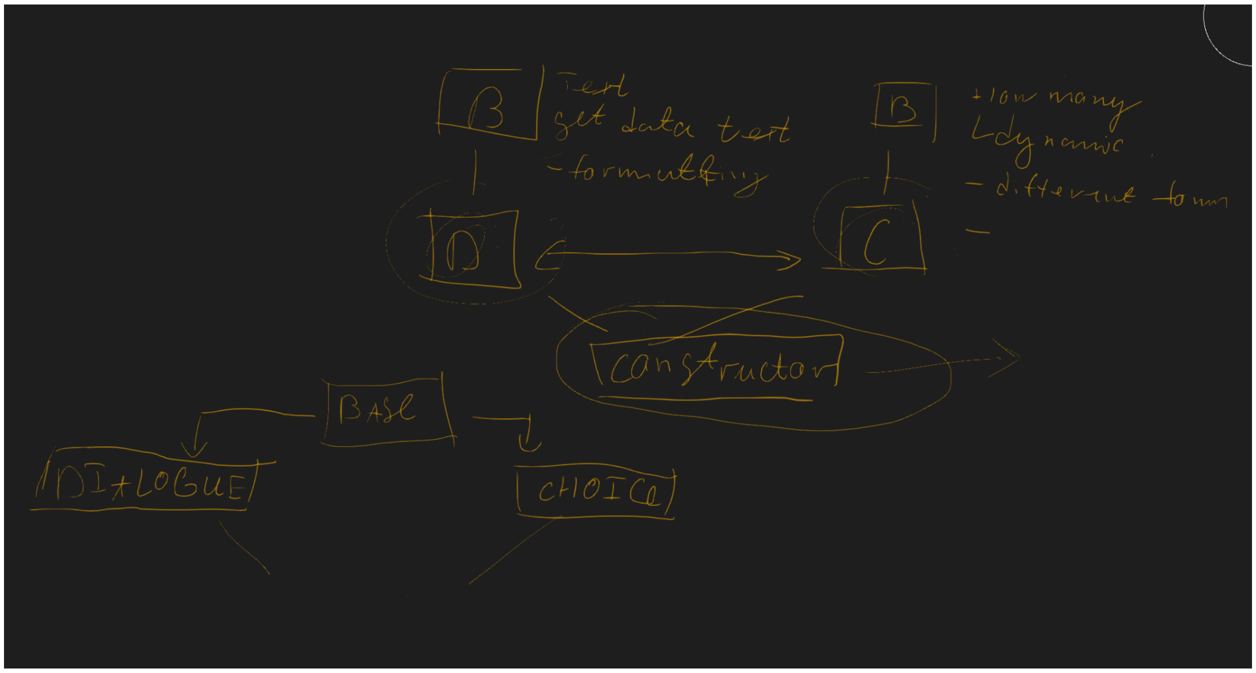
The image on the left is one of the iterations for the JSON structure. The image on the right represents the final JSON structure the team shipped.



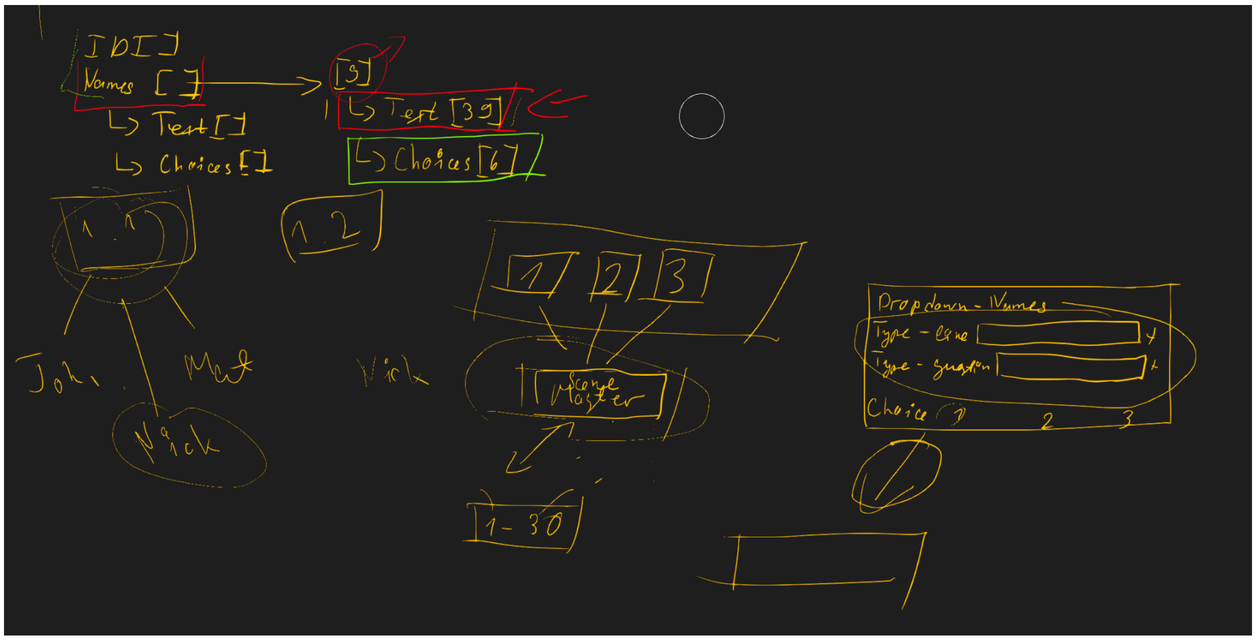
-NPC → CM
→ C
→ C-SO
→ C-SD
→ NPC Name
CM
-List<C> - autofill
- currCID
Player
-ConvUIM → Start()
-Trigger →

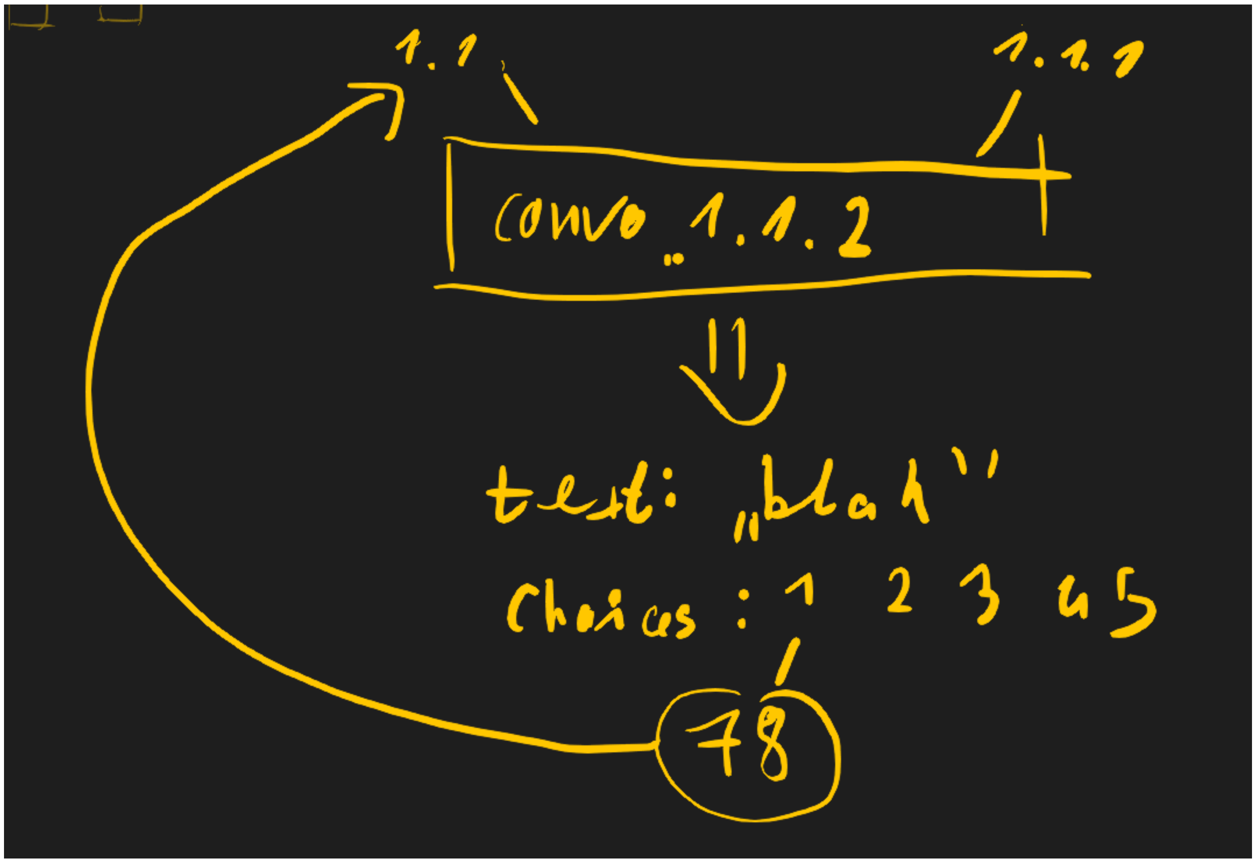


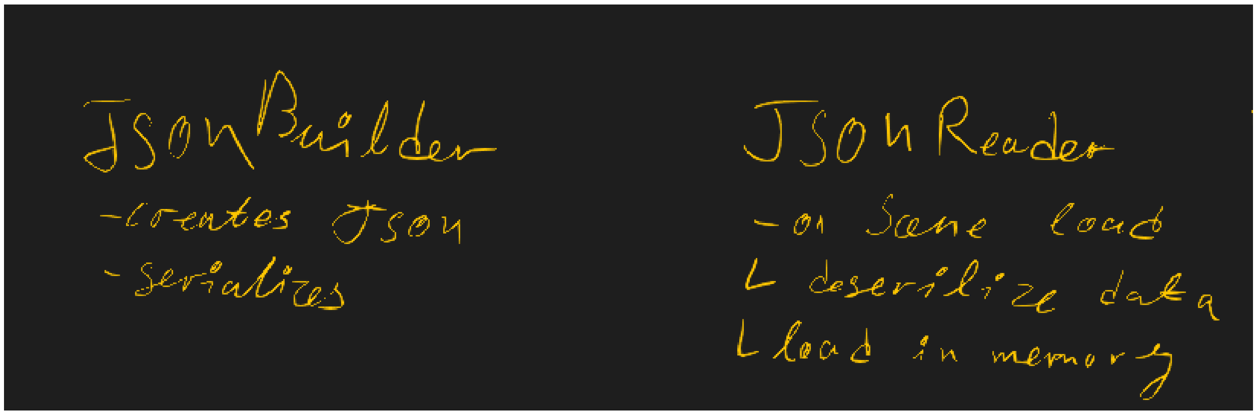
Convo MGR
Convo UI
NPC
NPC
NPC
Player



B
Text
get data text
-formatting
B
How many
L dynamic
- different form
-
D
C
~~constructor~~
BASE
DIALOGUE
CHOICE

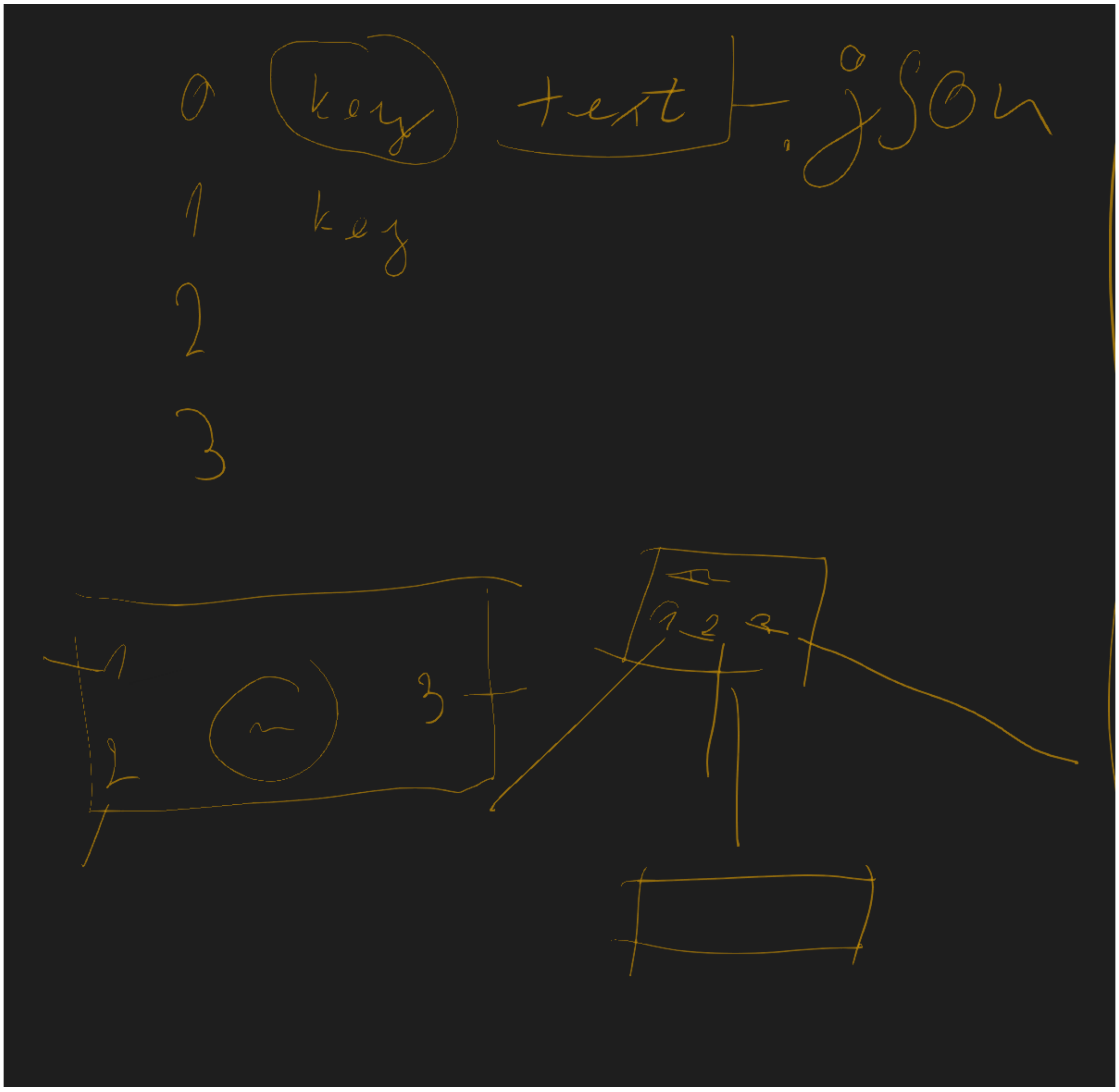

ID[ ]
Names [ ]
↳ Text [ ]
↳ Choices [ ]
[3]
↳ Text [39]
↳ Choices [6]
1.1
1.2
John
Mat
Nick
Vick
1 2 3
Scene Master
1-30
Dropdown - Names
Type - line
Type - question
Choice 1 2 3


1.1
1.1.1
convo 1.1.2
text: „blah"
Choices : 1 2 3 4 5
78

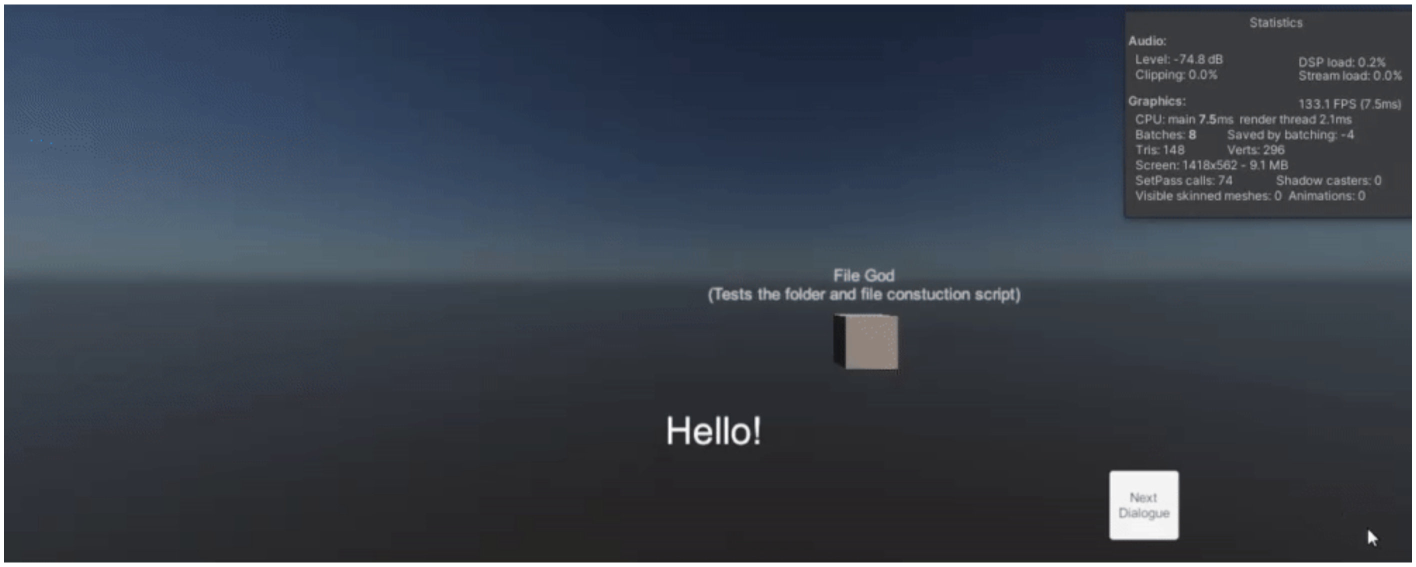


JsonBuilder
- creates Json
- serializes

JsonReader
- on Scene load
  - L deserilize data
  - L load in memory

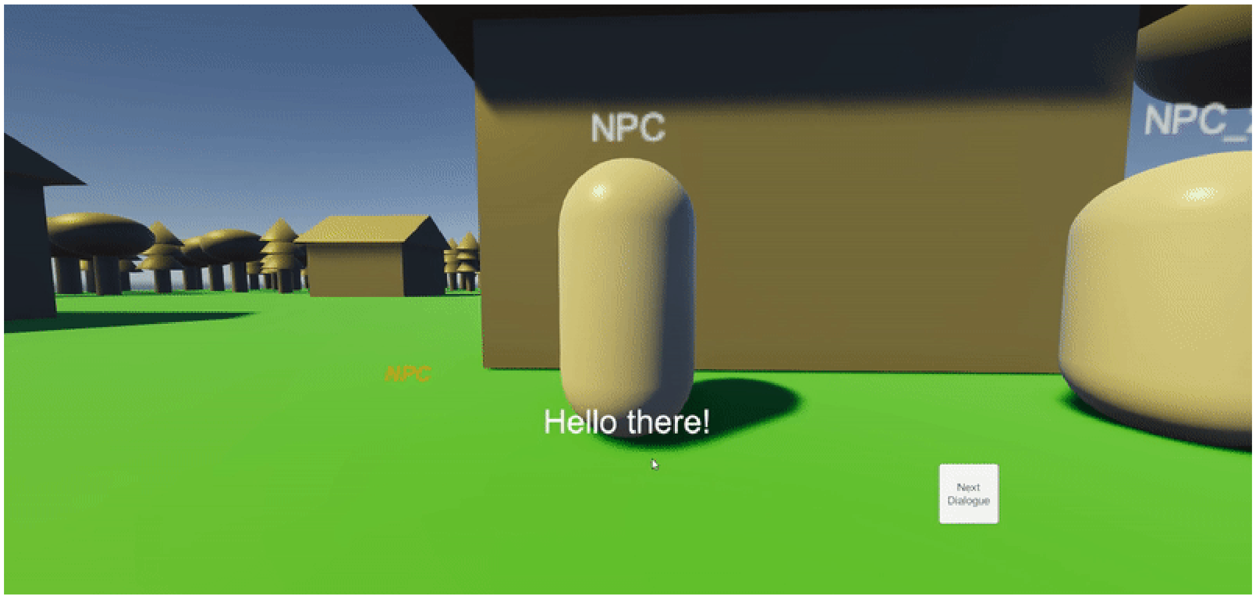


The images above represent various sketches created by the author during calls with Mr. Glendinning related to the conversation mechanic structure.


```
Statistics
Audio:
  Level: -74.8 dB                DSP load: 0.2%
  Clipping: 0.0%                 Stream load: 0.0%

Graphics:                        133.1 FPS (7.5ms)
  CPU: main 7.5ms  render thread 2.1ms
  Batches: 8       Saved by batching: -4
  Tris: 148        Verts: 296
  Screen: 1418x562 - 9.1 MB
  SetPass calls: 74        Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

Above is the first functioning prototype of the conversation mechanic. It is a static UI object used to visualise the data from the JSON. The first draft of the UI and interactions were created by Mr. Glendinning. Later iterations of how the data is displayed on the screen are a mixture from both programmers, but the foundation was laid down by Mr. Glendinning.



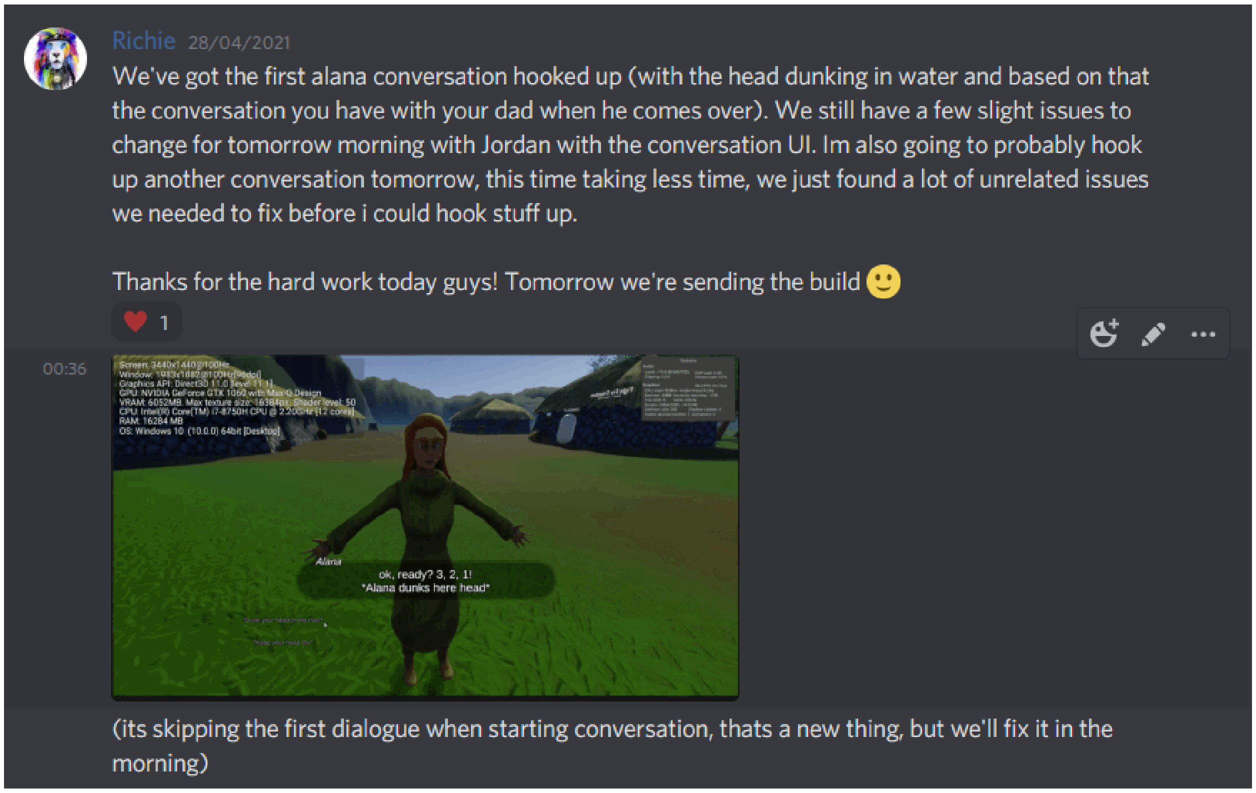
The above gif showcases the next big iteration in the conversation mechanic. Connecting it to the world and to NPC objects. The interaction scripts were developed by Mr. Glendinning, while the author extended the way the data from the JSON is displayed by the NPC, including taking into account if the choice has been previously selected (showcased by that option being coloured in purple).

Conversation Tool

For the Conversation Tool, the author helped test it and give feedback to Mr. Glendinning on how it could be improved and made easier to use.

Conversation Event System

The final piece which was needed for the mechanic to be completed and the author created was the Event System. More precisely, up until the event system, every conversation that an NPC had was its own separate entity and it had no way of knowing, communicating or affecting other conversations. The Conversation Event System addressed that. The implementation is based on a simple manager/listener event architecture. Using a global Event Manager where the designer creates a new event and decides how it would be triggered. Once the event is triggered, each NPC has a listener attached and it listens to its specific events. This system is extensively used throughout the whole project. Choices the player makes constantly send off events and change what conversation the player will have with other NPCs. Due to the nature of the game, rewinding opens up new choices, those choices open up even more conversations. A simple example is that at the very start of the game, if the player has rewinded, he will know that his parents will gift him a sword which opens up a different narrative branch in the conversation. Due to every Conversation being unique, the Mother needs to be notified that the player has rewinded and start the appropriate conversation the next time the player interacts with her. In that new conversation, if the player tells his mom that he knows about the sword, based on his subsequent choices in the conversation, events will be fired off to notify the Father NPC which conversation it should start the next time the player interacts with it. This is all handled by the event system built by the author. It was decided early by the author and Mr. Glendinning that it would be easier to manage and work with if the NPC's were responsible for conversations that are tied to them over having a global manager that stores them or storing them on the player.


Richie 28/04/2021

We've got the first alana conversation hooked up (with the head dunking in water and based on that the conversation you have with your dad when he comes over). We still have a few slight issues to change for tomorrow morning with Jordan with the conversation UI. Im also going to probably hook up another conversation tomorrow, this time taking less time, we just found a lot of unrelated issues we needed to fix before i could hook stuff up.

Thanks for the hard work today guys! Tomorrow we're sending the build 🙂

❤️ 1

00:36

Alana

ok, ready? 3, 2, 1!
Alana dunks here head

(its skipping the first dialogue when starting conversation, thats a new thing, but we'll fix it in the morning)

Critical Reflection

Having had the time to look back on the project and the team I am personally quite happy with how everything went. Everyone on the team pulled their weight, we all communicated quite actively, both through chat and in voice calls. Definitely one of our strong points as a team was the communication and thorough planning of our work.

Our mentor was also very helpful, friendly and always ready to help if any questions arose.

The client was also something I am quite happy with. He was passionate and interested in the project. Always ready to answer our questions regarding the project in depth and we had almost weekly calls, which helped everyone manage the scope of the project and deliver a good product.

Tools wise, the only tool I did not enjoy using was definitely Trello, the shortcomings of trello definitely shine when working on a project of this scale. It just does not handle organising and tracking a project well compared to more advanced alternatives such as Jira/Gitlab/Azure. But nonetheless the team used trello as best as we could and we kept it organised.

In regard to what the finished product looks like I would say everyone is quite happy with how it turned out after considering that this is the first project of this scope we have all attempted. In hindsight and armed with the knowledge I/we now possess there are plenty of things which could be done better and improved. We, the programmers, sat down after we submitted the project and had a talk about how the next iteration of the mechanics we have made could look like if we were to continue working on it. Despite what we have accomplished each mechanic can be improved.

One of the big areas that can be improved, and that is one of the things we had the least amount of time to work on, is how the two core mechanics integrate with each other. The communication between the Rewind and the Conversation mechanics could be heavily improved. Although integrating them together is a whole different architectural task on its own, it can not be done as a bi-product of developing them.

The way we store the conversation data I am quite happy with, but like most such systems it has 1 glaring issue. If we wish to change the underlying structure, that will break all of our already existing conversations. Something I would like to add would probably be an abstraction layer which is more flexible. That would allow us to extend the JSONs without breaking the conversations.

Another thing I would like to improve, but just was out of scope time wise is the Event system as a whole. Currently it is set up to be quite easily expandable and flexible, but is only being used for Conversation events. The event system should be extended to also support the rewind events, which were made separately by the other programmer team which worked on the Rewind mechanic. Another improvement that I would love to make to the Conversation Events specifically is make it more designer friendly. Yes, currently it is entirely drag and drop in the Editor, no writing of code needed. But making a custom inspector for it would go a long way in making it more enjoyable to use, as well as easier. Another improvement the Conversation Events system would benefit from is making the events work cross Scenes. When the system was made I did not know that we have a couple of scenarios where the events should be triggered cross Scenes. When we realised that we have such a situation it was too late to add that functionality to the system and Mario had to hardcode those events into the NPC's Conversation Manager scripts, which is far from ideal.

The Debug Console also has a lot of potential but unfortunately was not used as much as we expected, mostly due to the fact that we made too few builds. If we had a weekly build schedule the console definitely would have come in more handy and been expanded to cover more commands. Despite that, the one command it has was still extremely useful and everyone was happy it was there when we were testing the builds.

In terms of how the finished product compared to existing products. I can definitely say it is still a prototype of an idea with great potential. The biggest issue of the finished project is the lack of things to do apart from talking to NPCs. The prototype is set in a single small area with a lot of conversations. There was supposed to be a combat mechanic as well, but that was agreed with the client to be cut on the very first meeting due to his requirements for such a mechanic alone would take the team the better part of the semester, while the rewind and conversations were much more core to the idea the client had and would showcase his goals better.

I am also glad the blog site I made was well received by my teammates and everyone found it easy to work with. Also the client looked like he enjoyed having that extra layer of communication with us so we can showcase our progress and gave extra talking points during our calls.

In hindsight I am happy that early on I heavily pushed and managed to bring Mr. Glendinning to my side of how we should store and structure our underlying conversation data. It is a system that has proven to be very reliable and scalable for our use case and has caused barely any issues. Later the joint work on displaying the data also went well. Then the work Mr. Glendinning put into making the tool is superb. And finally the event system I made to connect all of our work together also went well, although due to time constraints it could benefit from some extra improvements.

Overall this whole module has been an irreplaceable learning experience. I got the opportunity to work with amazing and passionate people, with an amazing and very passionate client. My biggest fear coming into the module was what kind of team I would end up in, and all of my fears were dispelled immediately after I met the team for the first time and we did a game jam together. Sadly the fate of the programmer is to never be fully satisfied with what they have created and always finding ways they can make it better, and I am no exception, even after having submitted the project I am still thinking of ways I could further improve the systems I and we have made.