Oregon State University
College of Engineering

CS Capstone Technology Review

December 27, 2017

Ancestry Data Viewer

Prepared for

Ashley McGrath _____ _____
Signature *Date*

Prepared by

Team 4
ADVR

Le-Chuan Justin Chang _____ _____
Signature *Date*

Abstract

The Ancestry Data Viewer application requires various types of technology to properly implement. Among these technologies include UI design tools for both the application view, and VR view, and an algorithm to display the data. Various tools are available for these options, and thus, they must be explored to find the best tool for the application.

1 Introduction

The goal of this project is to take a GEDCOM file as an input and to display an organized ancestry tree as an output.

2 2D Visualization

2.1 Overview

For the first piece, we will be examining different options that can be used to display our program in its 2D application.

2.2 Criteria

The application needs to be able to display all the members required information and to clearly represent the relationship between members. The engine should be easy to manipulate so that the information is neat and not cluttered, even when the user is shifting the tree. The engine should have a VR-compatible feature that can be activated from desktop mode.

2.3 Potential Choices

The potential choices for displaying our project are the Unreal Engine 4, Unity Engine 3, and Lumberyard.

2.3.1 Unreal Engine

The Unreal Engine is a popular VR engines and is known for its ease of use. Switching from desktop to VR mode is also supported on this engine, there are published open-sources for this feature included [1]. The mode switching open-source code is among many other open-source data and an abundant supply of tutorial videos can be found throughout the community for help [2]. The Unreal Engine recently launched an enterprise team to focus on non-video game applications [3]. One of the target uses include data visualization. The software has over 20 built-in CAD data plugins that include features like geometry, texture, and camera angles [4]. Other parameter tools include uniform scaling and tiling awareness with objects.

2.3.2 Unity

The Unity game engine is another widely popular software for VR. This engine provides a built-in VR support that can be enabled from desktop [5]. Unity has also built support for non-video games application which focuses on architecture, engineering, and construction [6]. These tools can be used to display desired graphics and visualizations. Objects can be created in both 2D and 3D, and they can have specific details attached to them. Details are done by creating prefabs for the object, which will establish certain properties to each replicated object [5].

2.3.3 Lumberyard

The Lumberyard engine is a newer engine that was released by Amazon, it is still in its Beta version. It is also encouraged to be used for architecture, simulations, and animated movies purposes [7]. Lumberyard includes features like color grading, shading, particle effects, depth of field, etc. Free open source code is also available for the developer, that way they can just focus on changing aspects they want to change. Lumberyard also includes the concept of entities, which allows developers to drag and drop components to build the behavior they wish to see [7]. These components can be grouped together into slices to build more complex entities.

2.4 Discussion

All three technologies share open-source data that can be used to build functions for the application and can handle toggling to a VR application from desktop. Although they all are primarily game engines, they do provide features for non-video game application. Visual graphics are available in each engine however Lumberyard focuses on it the most and can deliver hyper realistic graphics. Unreal Engine and Unity still provide visual features that can be altered such as color and texture. Lumberyard and Unity utilizes grouping features that allows the developer to control properties across entities. Unreal Engine offers plug-ins and tools that can be used to alter entities.

2.5 Conclusion

As a team, we have chosen to use Unreal Engine because of its ease of use and overall responsive community of users which is beneficial for debugging issues. Although the Unity and Lumberyard provide complex handling with entities, our application desires minimal display that can be achieved with Unreal Engine and its built-in plugins. Unreal Engine also provides a larger option for non-video game applications after the launch of the enterprise community.

3 3D VISUALIZATION

3.1 Overview

For the next piece, we will be examining different options that can be used to display our program in its 3D application.

3.2 Criteria

The application needs to be able to display all the members required information and to clearly represent the relationship between members. The engine should be easy to manipulate so that the information is neat and not cluttered, even when the user is shifting the tree. The user should be able to look around the tree to examine neighboring nodes.

3.3 Potential Choices

The potential choices for displaying our project are the Unreal Engine 4, Unity 3, and Lumberyard.

3.3.1 Unreal Engine

The Unreal Engine is a popular VR engines and is known for its ease of use. A VR camera setup is included in the software which allows the developer to initialize camera view origins [2]. Depending on how the user should be oriented when using the program, the developer can adjust the initial camera setting. Unreal has its own VR measure units that can be converted to real-world units, so size is more manageable for the VR experience [2]. There are many open-source data and an abundant supply of tutorial videos that can be found throughout the community for help.

3.3.2 Unity

The Unity game engine is another widely popular software for VR. However, camera movement is also very limited [8]. Unity doesnt allow for direct VR camera change, for this to happen a change in position and rotation must be applied to a different object that is the parent of the object. The software also comes with a large option of built-in image effects including Colorful, Chromatica, Amplify Color, etc. Unity focuses on pixel ratio over lens correction, ultimately trading in performance for sharpness [5]. 3D objects also have specific graphic settings that also involves texture and color.

3.3.3 Lumberyard

The Lumberyard engine is a newer engine that was released by Amazon, that is still in its Beta version. High graphics and realistic visuals are highly supported by this software. Lumberyard utilizes gems to enable VR view [7]. Flow Graph nodes are used to control and add properties to the entities being used in the program [9]. The nodes can be used camera angle, image display, and controlling entities.

3.4 Discussion

All three engines can display 3D objects and utilize cameras. Unreal Engine is much more user-friendly with camera orientation. Lumberyard would be the next easiest to use while Unity is the more obvious limited one. Lumberyard and Unity provide a lot of graphic features that can be used to edit entities. Unity even sacrifices performance to be able to enhance sharpness of imagery. .

3.5 Conclusion

As a team, we have chosen to use Unreal Engine because its camera motion view has more options unlike the Unity engine which is much more limited. Lumberyard and Unity are ideal for high quality and complex graphics. Unity also consists of low performance which is something we dont want to sacrifice because we are using simple images.

4 VR API

4.1 Overview

For the final piece, we will be examining different options that can be used to configure the interface between our application and its controls.

4.2 Criteria

The API needs to be compatible with VR software and hardware. It should be capable of handling motion controllers to drag objects around in VR. Basic button configuration should be easy to configure.

4.3 Potential Choices

The potential choices for displaying our project are the Unreal Engine 4, Unity Engine 3, and OpenXR.

4.3.1 Unreal Engine

The Unreal Engine is one of the most popular VR engines and is known for its ease of use. It supports devices and contains built-in plugins to accommodate features [10]. The engine is compatible with a large amount of hardware including the Oculus Rift, PlayStation VR, Steam VR, Google VR, etc. Motion remote controllers setup is built into the software that allows for simple buttons and actions configuration. Events are set up through a blueprint interface displayed in a flowchart-fashion, the structure is simple and user-friendly while still having the ability to perform advance tasks [2]. Once the blueprint events are set up, there is a simulation feature to run tests.

4.3.2 Unity

The Unity game engine is another widely popular software for VR. It is compatible with the following hardware: Oculus Rift, PlayStation VR, Steam VR, Google VR, Gear VR, and Microsoft HaloLens. There are built-in APIs variables that can be used to detect the device being in use [**?**]. In addition, there are other script features available for API specifically. Images in VR are being provided twice, one for each eye, resulting in a performance cost [5].

4.3.3 OpenXR

OpenXR is a cross-platform VR API that connects a VR application from any VR system to access any VR device. This software eliminates the limited access between application and devices. The API is split into two levels of interface: application and device layer [11]. The application layer takes in events and outputs images to display. The device layer takes in controller state and outputs movement. These layers of translation are used to make the VR setup more uniform and available across multiple platforms.

4.4 Discussion

Unreal Engine, Unity and OpenXR support VR and motion remote controllers. In addition, both Unreal Engine and Unity can support a desktop to VR mode. Unreal Engines blueprint features allow for easier configuration by implementing flowcharts, and utilizing drag and drop actions. Unlike Unity which implements the configuration with code, because Unity is heavily code-based. OpenXR is an excellent option for integrating cross-platforms.

4.5 Conclusion

As a team, we have chosen to use Unreal Engine because it provides a simpler method to initialize the motion remote controllers. This feature is important to us because we want to incorporate these controllers into our program. Although OpenXR would be useful if we had integrated separate applications and devices, we are planning on keeping the hardware and application interaction simple. Unity also has a performance cost to incorporate VR images, which isnt favored for our application so Unreal Engine became the better decision.

5 INTRODUCTION

The Ancestry Data Viewer is composed of many components, three of which have been assigned to each team member to research. Le-Chuan Justin Chang has been assigned to research UI, both VR and 2D, and the algorithm for displaying the data. Yong Ping Li is in charge of researching the data parser, data storage, and data structures to store the data. Monica Sek is in charge of researching visualization for both 2D and 3D view, and VR hardware.

6 OVERVIEW

The User Interface (UI) is what the user will interface with in order to further examine and access additional features in the Ancestry Data Viewer. The UI should, as dictated by the requirements document, be both easy to use, and appear at least somewhat aesthetically pleasing.

6.1 Criteria

The criteria for the tool used is that is must be compatible with both Linux, specifically Ubuntu, and Windows 10. This is the most important criteria for the tool. Another important criteria is how well the tool can be used to create UI, since one of the criteria as referenced in the requirements document is that the UI must be aesthetically pleasing and easy to use. Another criteria is that the UI choice must be compatible with the tool or tools being used for displaying the family tree.

Criteria that is used in the event of equally suited tools will simply be to select whichever tool is easiest to use.

It should be noted that the primary component to fulfilling the usability and aesthetics component of the UI is more likely to be the result of design rather then the tool used to develop it, however, the selected tool is an important component for ease of development.

6.2 Potential Choices

6.2.1 JUCE

JUCE is a partially open-source library that uses OpenGL to generate a user interface. Since it runs on OpenGL, it is compatible with both Windows and Linux, and as an added bonus it is also compatible with Mac [12]. Another feature of JUCE is that it is also compatible with mobile devices [12]. JUCE is free for personal use, though a license is required for larger teams and larger applications [12]. One notable source of JUCE's users are creators of music creation tools, such as FL Studios [12]. There are also various tutorials for using JUCE available on its website [12].

A helpful inclusion with JUCE is the Projucer, which is an application that helps with developing UI [12]. By allowing the developer to manipulate the GUI by hand, be it resizing or changing the colors [12]. It can also allow HTML-like development, except that it uses C++ [12]. Projucer is also compatible with various IDEs, such as Visual Studios [12].

Another aspect of JUCE is that it supports plug-in development, especially for VST, AU and AAX [12]. It is frequently used for various plug-ins for various Digital Audio Workstation [12]. Additionally, JUCE is compatible with mobile devices, both Android and iOS [12]. An aspect that is less direct but helpful and notable nonetheless is that there is a forum for developers, in case assistance is needed from experienced developers [12].

6.2.2 UMG UI Designer

UMG, short for Unreal Motion Graphics User Interface Designer, is a UI development tool for Unreal Engine [13]. It is a visual editor that allows for the creation of menus, and other interface graphics [13]. UMG is bundled and designed for use with Unreal Engine, so there is little need for an external UI library if Unreal is selected to be the exclusive visual development format [13]. UMG has an extensive tutorial available online, and it features an online forum to request assistance [13].

6.2.3 Unity

Unity includes a built-in UI creator, which features an extensive tutorial and forums to consult if needed [14]. There is one large drawback to the Unity UI, which is that it is very resource intensive, to the extent that it runs twice as often as the default Update() function from Unity, which heavily increases the per-frame update time [15]. The UI includes the ability to add most features that are requested from a UI, however, such as built-in buttons, different modes of interaction, and accuracy [14]. The Unity UI creator is also a visual editor, for ease of use, both developing and previewing the finished result [14].

6.3 Discussion

UMG is built for the Unreal Engine, and the Unity UI tools are built for Unity. Both feature an extensive array of tools for their respective game engines, but only Unity's is specifically noted to cause performance issues, though that may not be a large concern, given the data viewer is unlikely to be extremely performance intensive.

JUCE was not designed for either game engine, and thus should be compatible with most applications. However, JUCE is frequently used for Digital Audio Workstations, which are typically far more extensive than what the data viewer should require.

All three of the tools use visual editors, and all three are compatible with their respective platforms, though JUCE may need more implementation since it is not specifically built to work with a given platform.

6.4 Conclusion

All three tools are sufficient for the data viewer, thus, the visualization tool that is selected for the application is most likely to be the deciding factor between the three UI tools presented. If Unreal is selected, UMG will be selected, if it is Unity, then Unity's UI kit will be selected, and if it is neither, then JUCE will be selected.

7 Overview

The display algorithm is likely one of the most difficult, yet important features for the data viewer. A GEDCOM file can contain an unknown amount of ancestry data, from a single person to thousands. This makes it difficult to display the family tree, due to the fact that it is highly probably that nodes will simply intersect eventually. The primary use of the algorithm is for the tree view of the algorithm, though it could potentially impact the storage capability, as if the algorithm is excessively slow, it could require storing the data in place.

7.1 Criteria

For the purposes of this application, excessive complications of family trees will be ignored, such as incest, adoption, or divorce. The most important aspect of the algorithm is that it must display everything properly, with as few intersections as possible, with ideally no intersections. Compatibility is also an important aspect, as ideally the algorithm should be compatible with how the data is read and stored, though it is possible to slightly modify either the chosen data structure or the algorithm to fit one another, though that may be difficult without fulling understanding every part of the algorithm. A final criteria for the chosen algorithm is that it should also be easy to implement.

7.2 Potential Choices

7.2.1 Ahnentafel

Ahnentafel, German for ancestor table, is a system for displaying ancestry data using numbers. It starts from a child and working its way up [16]. The Ahnentafel is used for a lot of family trees, with at least one set up for the Bush family [16], and sees plenty of usage elsewhere, including one made for King Henry III [17], which indicates the system has been tested extensively. Ahnentafels always start with exactly one child, and can feature potentially thousands of ancestors [16]. The system also numbers ancestors, so that a given ancestor can be found via mathematics [16]. The Ahnentafel is not a pre-made program, however, so it must be implemented and there are no active forums to assist with implementation, though it is possible to find a pre-made algorithm to adapt. It is possible to use the Ahnentafel system, and while displaying nodes, add a quantity of space between each node, so that each node is displayed.

7.2.2 D3 Hierarchy

D3 Hierarchy is a JavaScript library that takes documents that are in hierarchical form and visualizes them [18]. The D3 library also includes a method to automatically convert certain formats, such as CSV files, to a compatible hierarchical format [18]. D3 has multiple display forms, such as tree and cluster views, so it can display the data in multiple ways [18]. Additionally, the library has built-in functions to allow peeking at other nodes relative to a given node, which would allow for traversing through a graph [18]. The library was last updated six months ago, with the developer last appearing in the issues three months ago, so support is likely to be somewhat sparse [18].

7.2.3 Create an Algorithm

Creating an algorithm to display every node allows maximum flexibility, and would allow the algorithm to be completely compatible with the parsed nodes. Additionally, the algorithm can be based on preexisting algorithms, so it is not designed purely from scratch. Creating the algorithm would also allow the extraneous features to be removed, and would allow more features to be added as modification would be significantly easier and could be already designed from the beginning. However, there is no dedicated community designed to assist with the design of this algorithm, and designing and implementing an algorithm, especially one that must work with no overt amount of peer review would be extremely difficult to accomplish successfully.

7.3 Discussion

The easiest algorithm to implement would be the D3, which could be treated as a black box. The main problem when compared to the other two options is that D3 is a JavaScript library, so a JavaScript converter would be necessary. Additionally, depending on the visualization tool, only the algorithm will be necessary.

The Ahnentafel system is fairly simple to implement, being a binary tree, but it is not completely sufficient for the application, as it only works from one child. While it would be possible to combine all children for the root generation, going further up would excessively complicate matters, especially if the GEDCOM contains multiple families linked by marriage.

Creating an algorithm specifically for this application allows the algorithm to be fully compatible, capable of taking inputs as is and joining multiple families without overlapping. The glaring flaw, however, is that it must be designed, which is likely to take a large amount of time, and even then, it is likely to miss edge cases. It is likely to have roots in other systems, like the Ahnentafel, however, to hasten the design process.

7.4 Conclusion

Implementing the Ahnentafel system is unlikely to be the optimal choice, due to the fact that it only works for one direct family tree. It would work fine for lineage view, but for the overall tree view, it would be insufficient. Creating an algorithm may be effective for compatibility, but would be extremely difficult to design without overlooking at least some flaws.

Therefore, the most likely tool to use is the D3 library. While it may require implementing JavaScript as a scripting language to the application, at least for reading and converting the files, it is by far the easiest method of implementation, and it generates a well structured visualization, which can be converted back to data for the visualization engine to render.

8 Overview

The UI, as previously discussed, is an important aspect of the program. The UI in VR should behave identical to the UI in application mode, with the exception that its accessibility and user-friendliness is adapted for VR hardware. It should be noted that, while designing a UI for the application view is relatively straightforward, VR view is newer and less standardized, so the user is expected to be much less aware of how to interact with the application using their VR hardware.

8.1 Criteria

The criteria for the VR UI tool remains the same as for application view. It must be capable of creating a UI intuitive, easy to use, and aesthetically pleasing. It must be compatible with the VR framework, which is most likely a game engine, that will be selected, and it must be capable of producing intuitive designs.

If all tools are well suited for the task, then a tiebreaker criteria will be ease of use.

8.2 Potential Choices

8.2.1 UMG UI Designer

UMG UI Designer, short for Unreal Motion Graphic User Interface Designer, is the built in UI tool for Unreal Engine [13]. While UMG is currently the primary UI tool for development with Unreal Engine, and as such is the most compatible

with Unreal Engine. UMG does not fully support VR, but does have an experimental method to create one via importing a pre-existing UI in to a 3D widget, which would make it a very simplistic to convert a normal UI to a VR UI [19]. The biggest flaw to this, however, is that since it is an in-game object, it does not necessarily follow the player [19]. There is also a large community that exists to assist developers with various tasks using the Unreal Engine [13].

8.2.2 Unity

Unity has a built in set of tools for UI development. It features built in VR utilities, including both diegetic, when the UI is in a fixed location that the user can view. The Unity UI tools are the most compatible with the Unity Game Engine, as they are built for Unity. The UI is capable of being moved closer and further away from the user as needed, to simulate the size increasing or decreasing [20]. Unity has a large tutorial for integrating VR, including the interface, and a forum to post questions as needed [20]. Additionally, Unity has sample assets to show how a finished UI should look like [20].

8.2.3 Hover UI

Hover UI is a tool for developing a UI with Unity [21]. It is designed to be simple to use yet produce ”beautiful and dynamic” UI [21]. It uses a prefab, which is a built in Unity objct, that allows developers to add the UI, then modify it with the options that need to be added [21]. Hover UI is designed to work with VR Controllers [21], so it is very compatible with the hardware the application needs. The possible modules for Hover UI includes an on-screen keyboard, and a menu that is attached to a controller [21]. There is little support for Hover UI on its GitHub, and it may be abandoned, however, as its last update was eight months ago and though its developer has been active three months ago [21].

8.3 Discussion

UMG UI is most compatible with Unreal Engine, being developed for it, however, it is by far the least effective, as it is not designed specifically for VR, and is incapable of non-diegetic views. A major benefit is that it is capable of easily converting the UI from 2D to 3D, with minimal effort.

Unity is, by comparison, significantly more powerful for VR UI, as it was designed to be capable. It can do mostly everything that the 2D view can do, and thus can easily be converted from 2D to 3D, though it cannot do non-diegetic views, on the grounds that to do so would be difficult to focus on.

Hover UI is a Unity plugin, so it is also compatible with Unity. It allows menus from VR Controllers, which make it a very helpful place to store the menu, and allows a good use for the rest of the buttons on the VR Controller. However, due to the fact it has not been updated in a while, there may be compatibility problems in the future if Unity receives a major update.

8.4 Conclusion

The selection of which tool to use for the VR UI is heavily dependent on what visualization tool is selected. It is almost certain that a game engine with prebuilt VR features would be used, thus, Unreal Engine or Unity Engine. If Unreal Engine is selected, then the UMG UI is likely to be the best selection, as it is already fully compatible [13]. If the Unity Engine is selected, Unity’s built in UI tools should be sufficient, as Unity allows developers to bind buttons to do certain things, for instance, opening a menu [14]. As there are few interactions, the advanced tools of Hover UI are not necessary, thus it is more worthwhile to secure the future proofing of the built in Unity UI tools.

9 INTRODUCTION

The objective of our project is to develop a software to generate the family tree with different views from a GEDCOM file. The software also has the functionality of finding the common ancestor between two people. Our team decided to split the project into nine components including the parser, database, data structure, 2D visualization, 3D visualization, VR hardware, 2D UI, 3D UI, and display algorithm. I will be reviewing tools for the parser, database, and data structure. The parser is used to capture data; we need to be able to determine the pieces of information we need in the GEDCOM file and capture the information. The database is used to store the data before we use it because we don't want waste time to repeat the capturing process. The data structure is used to make access to data easier.

10 PARSER

10.1 Overview

The GEDCOM file given contains a lot of information and most of the information isn't needed for our project. The format of the GEDCOM file is complicated, but there are patterns within the file. We will need to find the pattern in the information and develop an algorithm to capture the data that we need. Although the capturing of the data can be performed by a tool call parser, we also have the option of implementing our own parser.

10.2 Criteria

The parser needs to capture only the information we needed and that is the names and their relationship with others. Another criteria for the parser is performance. We need this process to be done at a fast speed because everything else happens after we captured the data. The user will have to wait during this process so having them wait too long is not good.

10.3 Potential choices

10.3.1 Nail

Nail is a parser that is heavily focused on security. It is very secure compared to other parser and this will reduce the effort needed for programmers[1]. Nail can use the protocol grammar to define the data format and the internal object model of the data. Therefore, it can identify the different, unique objects in the data and information related to the object[1]. Nail use other existing parsers to eliminate the notion of semantic actions, so it hides what the data is to be captured from the file with the cost of its performance. There's an authoritative DNS server that can outperform the BIND9 DNS server[1]. This tool is free to share and adapt, but it can't be use for commercial purposes.

10.3.2 Log parser

Log parser is a powerful tool that can perform universal query access to text-based data. These text-based data files include XML files, CSV files, and even file such as event log for your windows operating system[2]. This tool accepts many input formats, and it is possible to create your own custom input format and use it with the log parser engine. The results of the query can also be custom-formatted in text-based output[2].

10.3.3 Creating our own parser

We have the option of not using existing parser and implement our own parser. By analyzing the GEDCOM file and finding the pattern of the data. We could use this pattern and build an algorithm that captures the needed data. The implementation of the parser shouldn't be difficult after we analyze the data and find the pattern.

10.4 Compare and Contrast

Creating our own parser will cost us time and the performance depends on the language that we are using and the algorithm. The best part about creating our own parser is to use the data right away after capturing it, so it doesn't have to output to a file first, and then access the file for data. Nail performs slower than log parser due capturing the data in a more secure method, but our software doesn't have to be done over the Internet so security isn't something we should worry about. Log Parser is a great tool and it allows us to customize input and output.

10.5 Decision

I would choose to create our own parser because the existing parsers might perform some process that isn't relevant and takes more time to capture the data. Nail heavily focus on security but security shouldn't be a problem for our project. Log Parser has these functionalities that allow users to customize input and output and I think it would do unnecessary processes that would lower the performance. Therefore, we should implement our own parser which is simpler and we have more control over it.

11 DATABASE

11.1 Overview

The database is used to store the data. We need the data to be stored so we can use it whenever we need and don't have to go back to the GEDCOM file every time. Going through and parsing the GEDCOM file will take time and we don't know how long it would take, but we don't want to spend much time on this process.

11.2 Criteria

The criteria are to store in a correct manner to ease the access and use of the data for our software. The data store has to be able to depict the relationships between each person in our data. It should also to able to perform the find common ancestor functionality.

11.3 Potential choices

11.3.1 MySQL

MySQL is a relational database management system[3]. The tool is based on the relational model of data which depicts the relationship between the entities in the data. The data is stored in rows and each with a unique id. The columns will be categories of information about this unique data. In our case, each person will be linked to a unique id and the columns indicate the relationship between the people on our data. For example, we could have a column name "Mother of this person", and inside that column is that person's mother's unique id. It can depict many relationships in just a row of data. It can perform query search for specific data in the database[4].

11.3.2 Neo4j

Neo4j is a graph database. It stores every unique item as a node and the relationship between the nodes[5]. Each person's name will be a node and the relationship between people will be defined in the data. Neo4j takes its input from CSV files to create all the different nodes and relationships[5]. It is able to load up to 10 million nodes so it can deal with large data. This tool can perform queries to search the data that we need which will be useful for the find common ancestor function. It is a lot faster than MySQL due to its queries requiring 10 to 100 times less code[5]. Neo4j is the specialized tool for storing relational data and displaying it as graphs[5].

11.3.3 Not using a database

Not using a database is also an acceptable option for our project. We could create our own way of storing the data locally and doesn't have to get into a database for accessing the data.

11.4 Compare and Contrast

Both the Neo4j and MySQL can depict the relational data. They can perform query search for specific data in the database. Neo4j will take inputs from CSV files which we can create but it will take us time to do so. MySQL is easier for inputting the data into the database. MySQL is able to use online but requires users to log in. Not using a database would be easy for storing the data, we just need to format the data and store it.

11.5 Decision

I think it is best for us to create our own way of storing the data. The data can be easily stored, and we don't have to go through other means to store the data in a database. Storing the data in a database will cost time for manipulating the data and making it cope with the format for the database. There isn't a good reason for doing extra things and make the software function slower.

12 Data Structure

12.1 Overview

The data structures are for making access to data more easily. There are many existing search algorithms and functions that could be used for the different data structures.

12.2 Criteria

The criteria are the data structure needs to make access easy. It also needs to be able to illustrate the different relationships between the people, including parents and children, husband and wife, ex-husband and ex-wife, etc.

12.3 Potential choices

12.3.1 Linked-List

A linked list is a linear data structure, so items in the linked list are actually sequenced. We could implement this like a graph and make one person connect with multiple people. Linked list contains a data field to store data and a link field[6]. In the link field, we can create multiple links to make it into a graph. The links can be named differently in order to determine the relationships. The links are actually pointers pointing from current address to address the next item[6]. The address of items in this data structure is not sequential, that is why it uses links. The benefit of using this structure is that it is easy to add items to the linked-list. Also, there aren't limitations on the number of items to be in the linked-list, unlike arrays, you have to define its size[6].

12.3.2 Tree

Tree is a type of nonlinear data structure. Every item in the tree is called a node. There is one distinguished node being the root of the tree[7]. The other nodes will have a parent node and child nodes. Every node can have a corresponding height and depth determining their place on the tree. Nodes that shared the same parent node are siblings[7]. The binary tree is well known to the programmers. It is a specialized tree that each parent node could only have up to two child nodes[7]. This won't be helpful for our project because people can have more than two children. The general tree is what we needed, but a problem we will have with this data structure is illustrated marital status. We could put the parents on one node, but this would make it more difficult to access the individual nodes/person[7].

12.3.3 Graph

The graph is another type of nonlinear data structure[9]. The graph has two things to it and that is vertices and edges[9]. The vertices represent the objects and edges are the links between the objects. The edges don't only store the link between the two vertices, but it also can store an integer[8]. This integer has a variety of uses, it can be used as the distance between two locations if locations are the vertices of the graph, etc. We can use this integer to identify the relationships between the nodes.

12.4 Compare and Contrast

The linked-list uses pointers so we need to change the address of the pointer in order to access the data. It is really a mess if we used linked-list in our project because it is very hard to access the data. The tree data structure has problems illustrating the marital status. If we put parents into one node, we would have to make do more effort in accessing the information from individual node/person. The graph is able to illustrate all the different relationships between people through the integer that can be stored in its edges.

12.5 Decision

I would choose graph because it is able to depict all the different kinds of relationships. The linked-list is also able to do the job, but it is very hard to get access to the data. The tree can have only one parent node, so we need to put the parents into one node in order to use this data structure. We will have to put more effort into accessing the node/data. Therefore, the data store in the graph is easier to access and it is able to depict all the relationships easier than other data structures.

REFERENCES

[1] DreaM&,
https://answers.unrealengine.com/questions/262396/switch-on-vr-mode-in-runtime.html

[2] Epic Games Inc,
https://docs.unrealengine.com/latest/INT/Platforms/VR/

[3] Ronnie Dungan,
https://www.seriousgamesindustry.com/wp-content/cache/wp-rocket/seriousgameindustry.com/en/qa-simon-jones-director

[4] Randall Newton,
http://gfxspeak.com/2017/08/22/reveals-enterprise-visualization/

[5] Unity Technologies,
https://unity3d.com/learn/tutorials/topics/virtual-reality/vr-overview

[6] Unity Technologies,
https://store.unity.com/industries/aec

[7] Amazon Web Services Inc,
https://aws.amazon.com/lumberyard/details/

[8] Unity Technologies,
https://docs.unity3d.com/Manual/VROverview.html

[9] Amazon Web Services Inc,
http://docs.aws.amazon.com/lumberyard/latest/userguide/virtual-reality.html

[10] Tom Looman
http://www.tomlooman.com/getting-started-with-vr/

[11] The Khronos Group Inc,
https://www.khronos.org/openxr

[12] ROLI Ltd,
https://juce.com/

[13] Epic Games Inc,
https://docs.unrealengine.com/latest/INT/Engine/UMG/

[14] Unity Technologies,
https://docs.unity3d.com/Manual/UISystem.html

[15] Unity Technologies,
https://forum.unity.com/threads/ui-design-tool-pros-vs-cons-unitygui-guitextures-3d-mesh-or-3rd-party-tools.76937/

[16] D. Eastman,
http://freepages.genealogy.rootsweb.ancestry.com/ jcat2/ahnentafel.html

[17] Tamura Jones,
https://www.tamurajones.net/Ahnentafel.xhtml

[18] Bostock, Mike,
https://github.com/d3/d3-hierarchy

[19] Epic Games Inc,
https://forums.unrealengine.com/development-discussion/vr-ar-development/56432-tutorials-on-umg-ui-in-vr

[20] Unity Technologies,
https://unity3d.com/learn/tutorials/s/virtual-reality

[21] Kinstner, Zach,
https://github.com/aestheticinteractive/Hover-UI-Kit/wiki

[22] Unity Technologies,
https://docs.unity3d.com/Manual/OpenVRControllers.html

[23] Julian Bangert and Nickolai Zeldovich,
https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-bangert.pdf

[24] Microsoft,
https://technet.microsoft.com/en-us/scriptcenter/dd919274.aspx

[25] MySQL,
https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html
[26] W3Schools,
https://www.w3schools.com/php/php_mysql_intro.asp
[27] Tobias Lindaaker,
https://www.slideshare.net/thobe/an-overview-of-neo4j-internals
[28] Victor S.Adamchik,
https://www.cs.cmu.edu/ãdamchik/15-121/lectures/Linked%20Lists/linked%20lists.html
[29] Carnegie Mellon University,
http://www.cs.cmu.edu/člo/www/CMU/DataStructures/Lessons/lesson4_1.htm
[30] GeeksforGeeks,
http://www.geeksforgeeks.org/greedy-algorithms-set-9-boruvkas-algorithm/
[31] Tutorialpoint,
https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm