Oregon State University
College of Engineering

CS Capstone Technology Review

December 20, 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 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.

2 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.

2.1 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.

2.2 Potential Choices

2.2.1 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 [1]. Another feature of JUCE is that it is also compatible with mobile devices [1]. JUCE is free for personal use, though a license is required for larger teams and larger applications [1]. One notable source of JUCE's users are creators of music creation tools, such as FL Studios [1]. There are also various tutorials for using JUCE available on its website [1].

A helpful inclusion with JUCE is the Projucer, which is an application that helps with developing UI [1]. By allowing the developer to manipulate the GUI by hand, be it resizing or changing the colors [1]. It can also allow HTML-like development, except that it uses C++ [1]. Projucer is also compatible with various IDEs, such as Visual Studios [1].

Another aspect of JUCE is that it supports plug-in development, especially for VST, AU and AAX [1]. It is frequently used for various plug-ins for various Digital Audio Workstation [1]. Additionally, JUCE is compatible with mobile devices, both Android and iOS [1]. 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 [1].

2.2.2 UMG UI Designer

UMG, short for Unreal Motion Graphics User Interface Designer, is a UI development tool for Unreal Engine [2]. It is a visual editor that allows for the creation of menus, and other interface graphics [2]. 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 [2]. UMG has an extensive tutorial available online, and it features an online forum to request assistance [2].

2.2.3 Unity

Unity includes a built-in UI creator, which features an extensive tutorial and forums to consult if needed [3]. 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 [4]. 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 [3]. The Unity UI creator is also a visual editor, for ease of use, both developing and previewing the finished result [3].

2.3 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.

2.4 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.

3 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.

3.1 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.

3.2 Potential Choices

3.2.1 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 [5]. The Ahnentafel is used for a lot of family trees, with at least one set up for the Bush family [5], and sees plenty of usage elsewhere, including one made for King Henry III [6], which indicates the system has been tested extensively. Ahnentafels always start with exactly one child, and can feature potentially thousands of ancestors [5]. The system also numbers ancestors, so that a given ancestor can be found via mathematics [5]. 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.

3.2.2 D3 Hierarchy

D3 Hierarchy is a JavaScript library that takes documents that are in hierarchical form and visualizes them [7]. The D3 library also includes a method to automatically convert certain formats, such as CSV files, to a compatible hierarchical format [7]. D3 has multiple display forms, such as tree and cluster views, so it can display the data in multiple ways [7]. 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 [7]. 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 [7].

3.2.3 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.

3.3 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.

3.4 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.

4 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.

4.1 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.

4.2 Potential Choices

4.2.1 UMG UI Designer

UMG UI Designer, short for Unreal Motion Graphic User Interface Designer, is the built in UI tool for Unreal Engine [2]. 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 [8]. The biggest flaw to this, however, is that since it is an in-game object, it does not necessarily follow the player [8]. There is also a large community that exists to assist developers with various tasks using the Unreal Engine [2].

4.2.2 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 [9]. Unity has a large tutorial for integrating VR, including the interface, and a forum to post questions as needed [9]. Additionally, Unity has sample assets to show how a finished UI should look like [9].

4.2.3 Hover UI

Hover UI is a tool for developing a UI with Unity [10]. It is designed to be simple to use yet produce ”beautiful and dynamic” UI [10]. 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 [10]. Hover UI is designed to work with VR Controllers [10], 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 [10]. 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 [10].

4.3 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.

4.4 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 [2]. 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 [3]. 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.

REFERENCES

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

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

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

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

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

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

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

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

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

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

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