Jim Olyha, Dash Kieler 5/5/2016

Interactive Visualization Final Project Report

Motivation

For this project, we are mainly trying to test whether a “space” model is an effective way to visualize an individual’s music library. In this model, an artist is represented by a star, an album by a planet, and a song by a moon, where the stars are placed in such a way that similar artists are grouped together in space. Many models use a graph or other spatial visualization to show the similarity between artists, but none that we know of visualize the album and track data as well. Our goal is to allow an end user to traverse their music library in an interesting and engaging way, and allow them to better understand the relations between the artists they listen to. Our target audience for this visualization consists of people who wish to explore their music library and better understand the patterns in their musical tastes.

Our main research question is this: what is the best way to visualize a user’s music library, such that they can always see a particular data point in the context of the larger visualization, while also allowing the user to effortlessly explore their music? We believe that in using a galaxy model, the hierarchical nature will allow us to maintain context and provide intuitive ease of exploration.

Related Work

As we have seen throughout this semester, there are countless ways to effectively visualize large datasets. However, most of these visualizations involve aggregating vast amounts of data together to look for trends. Far fewer visualizations lend themselves to the effortless exploration of individual data points within the larger context of the data. The “galaxy”

form of visualization is particularly good at allowing an end user to navigate through the entirety of the dataset, since a galaxy is inherently hierarchical in nature. This form of visualization has been used in many different contexts, including the visualization of large software projects [3].

There have also been many projects that involve the visualization of a music library. Much research has been done into how such music data should be spatially presented to the user, and most papers have concluded that similar music should be grouped together [1]. Of course, this raises many questions about what the metric for similarity is, and whether similarity should be evaluated to compare artists, songs, albums, or some combination of the three.

Several researchers have tried to use a galaxy model to visualize music libraries, with mixed results. Of all of the ones we have found, the primary focus is on displaying the relations between either songs or artists, with extra information about a given node available upon further investigation [2]. We believe this prevents the user from being fully immersed in the library, and thus one of our major goals is to visually present artist, album, and song data to the user in a useful and pleasing way.

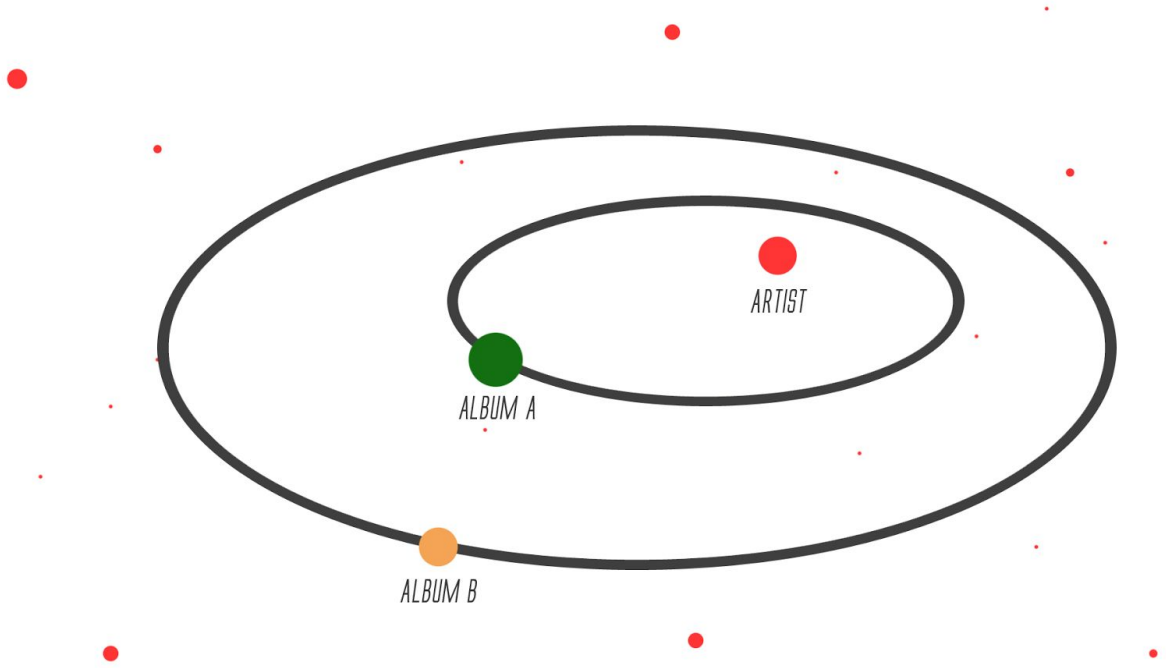
Our primary data sources for this project are LastFM and Spotify. The LastFM service allows a user to archive their listening habits from a variety of streaming services through a process it calls “scrobbling”. This is an ideal source for a music library, since many end users are moving away from downloading music in favor of streaming it. Also, LastFM has a public API, unlike Google Play Music. LastFM stores each song and its metadata as it is played, including the timestamp of each time it is played, could be used to extend the project in the future.

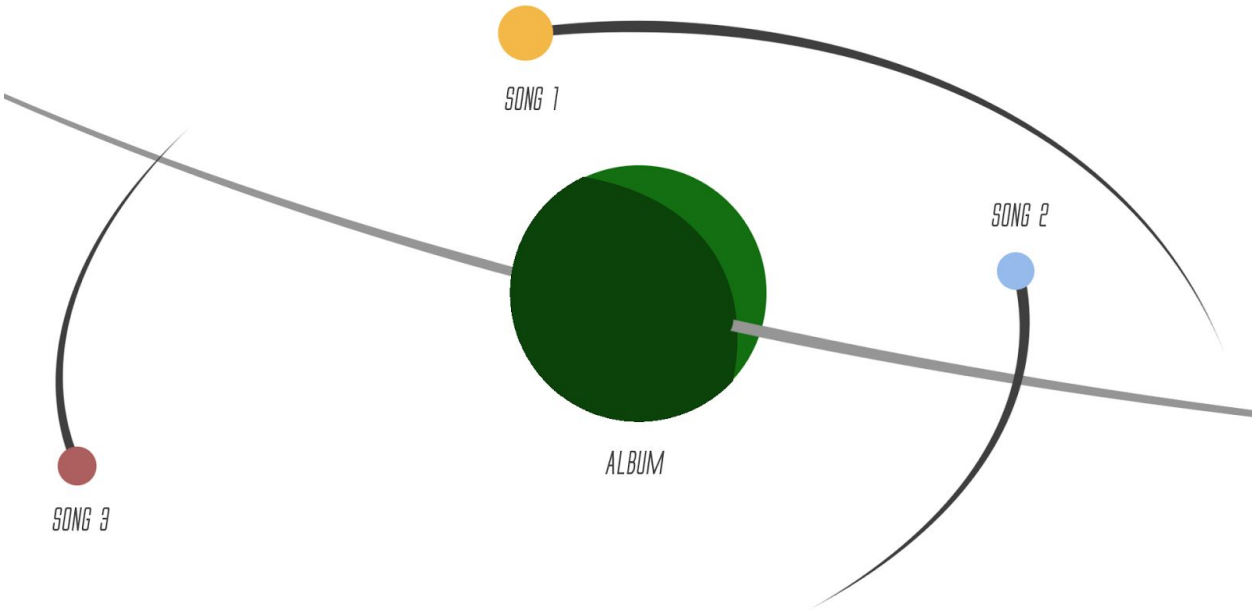
Spotify recently acquired the Echo Nest and its API, which stored vast amounts of metadata for millions of songs and artists. The related artist service was specifically used from the Spotify API to determine an approximation of similarity between artists in our visualization.

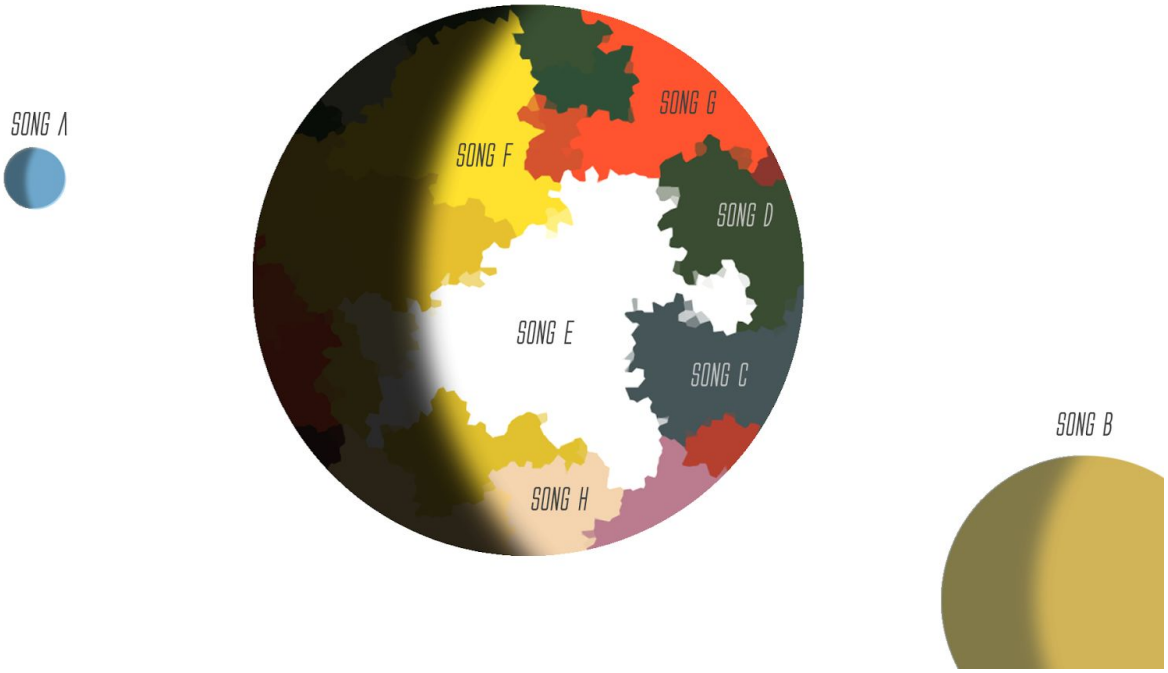
This similarity data was used to create a three dimensional force directed graph of artists using the python iGraph library. We used this library's implementation of the Fruchterman-Reingold force directed graph drawing algorithm to determine where each artist should be placed in 3D space in our final visualization [4].

Initial Design and Evolution

The initial design plans for our visualization can be shown below. We wanted to show the user's music library in a galaxy model, such that a given artist could be seen in the context of the larger collection of similar artists. More specifically, we planned on representing artists as stars, albums as planets, orbiting those stars, and songs as moons orbiting their respective albums. It was also very important that similar artists should be represented as spatially closer together in the galaxy, and this did not change throughout the iterations of our project.

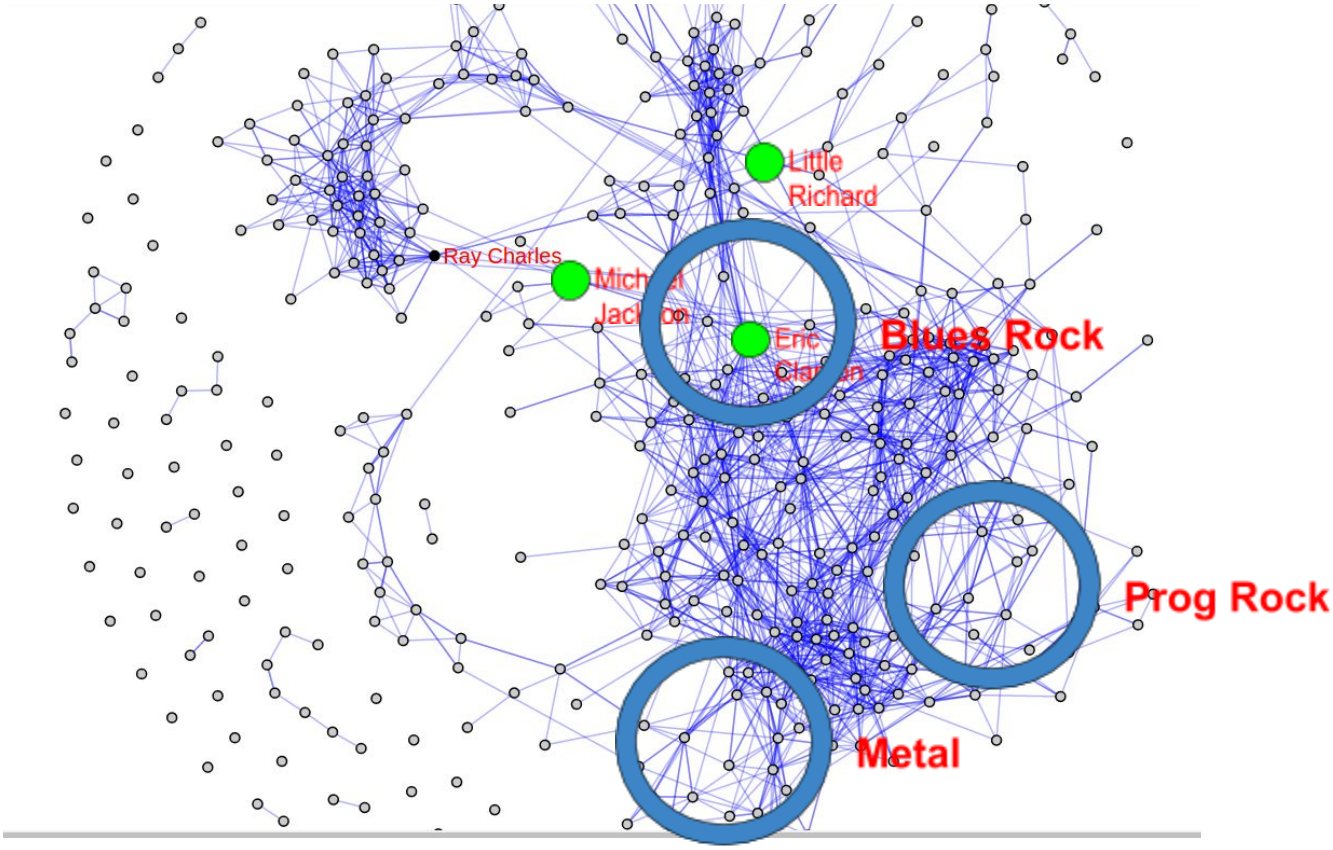






Fairly early on, we also came up with the idea of representing songs as landmasses on the planets (albums) themselves. Since we had access to song play count data, we could also

represent the songs a user liked more as moons, while the less popular songs could be landmasses. This sounded very cool (although also more difficult) so we made it a stretch goal for us to do if we had enough time.



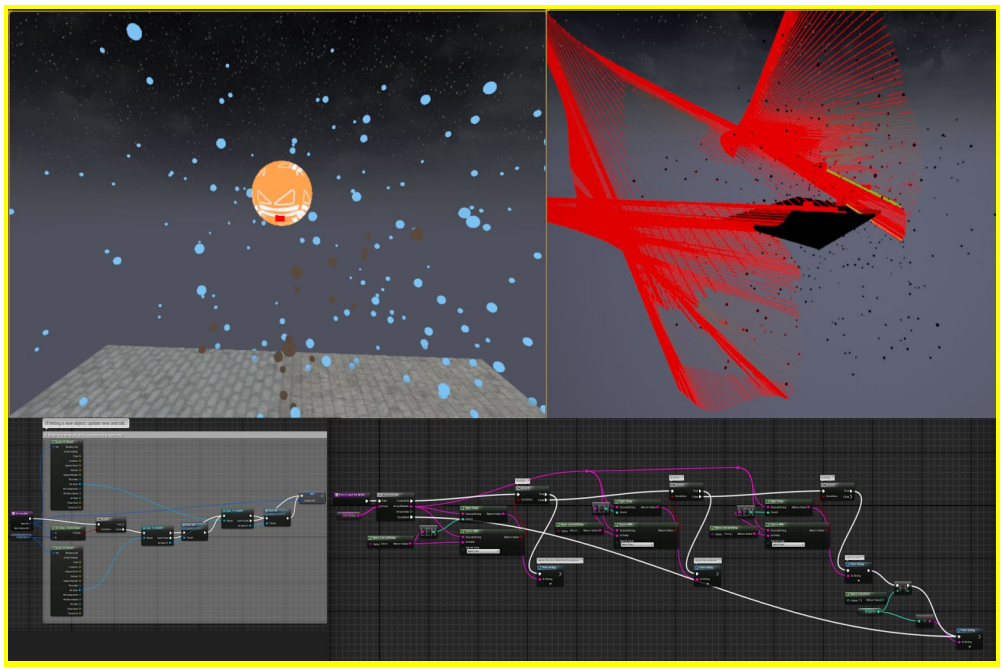
2D debugging graph. Edges represent relations between artists.

As shown above, we created a 2D graph of the artists in Jim's music collection, where the edges represent a similarity between the two artists at its endpoints. This was was used to debug the similarities between artists, and make sure the relations we were showing the end

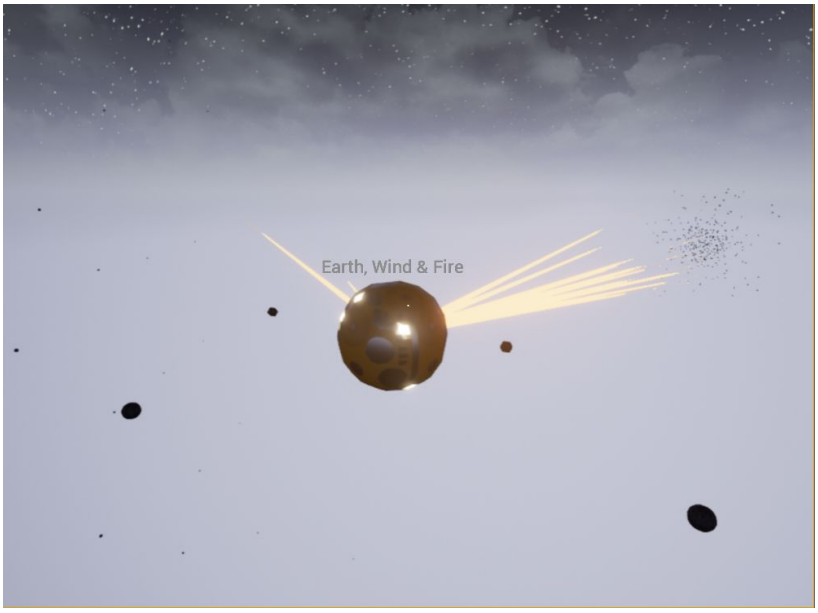
users were accurate. If the relations between artists were weak or did not make very much sense, the user would get the same experience out of exploring our musical galaxy, and the visualization would be a failure. The clustering of the data was a good first indication of correctness. The large cluster on the lower right of the image represents classic rock, which contains the majority of songs in Jim's library. Within this cluster, there are also more granular sub-genres, such as metal (bottom left), blues rock (top), and progressive rock (right). The smaller cluster directly above represents the blues, whose location makes sense since most of classic rock evolved directly from the blues. Finally, the cluster on the left side represents motown and funk, which have their roots in rock and blues, explaining the connections to the other two clusters.

Several artists of interest have also been highlighted in order to point out other elements of interest. For instance, Eric Clapton is positioned directly between the rock and blues clusters because he is one of the most influential artists in the blues rock genre (note that he is also at the center of the blues rock subgenre). Similarly, Michael Jackson is placed between motown and rock, which would be a logical place for 80's pop music to be positioned. Finally, Little Richard is positioned between blues and rock and a bit to the side. Artists in this area generally belong to bands that were active between the 50's and 60's, when popular music was shifting from the older blues influences to the newer and more electric rock and roll. Little Richard influenced some later rock and roll artists (like Buddy Holly and Jerry Lee Lewis), as well as motown groups (like the Four Tops or Stevie Wonder).

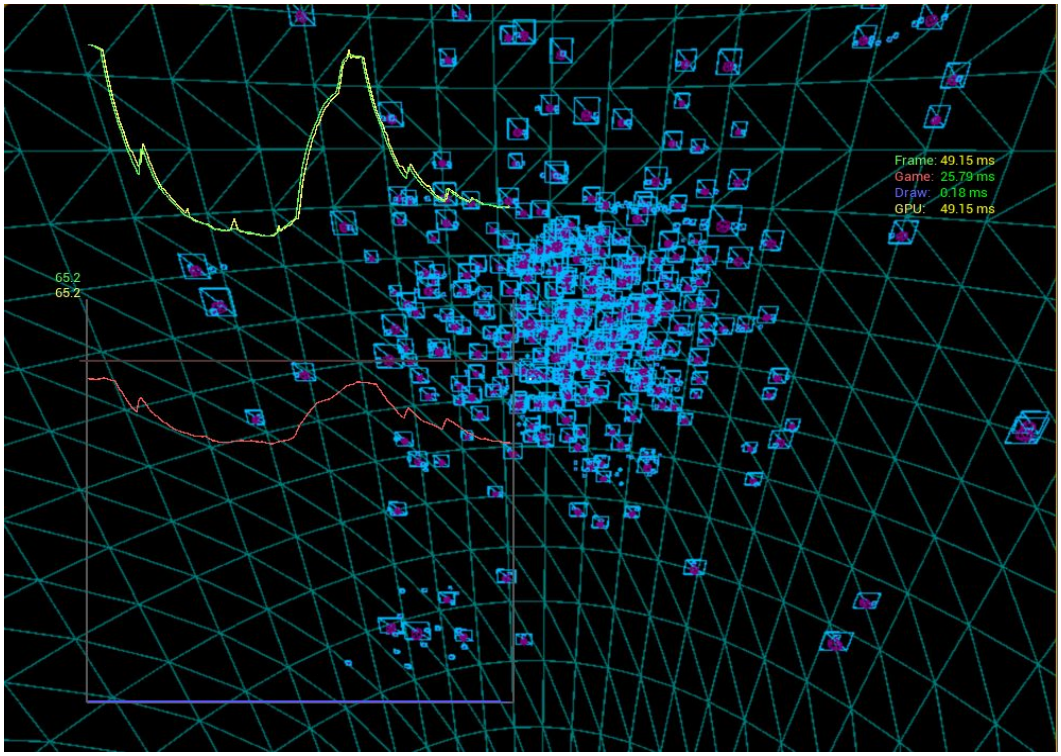
Below are some screenshots of the intermediate results of our visualization. The first examples is of Raycast picking, and shows the path multiple raycasts followed. This screenshot also shows Blueprints, which control object selection and file parsing. Both eventually were refactored.



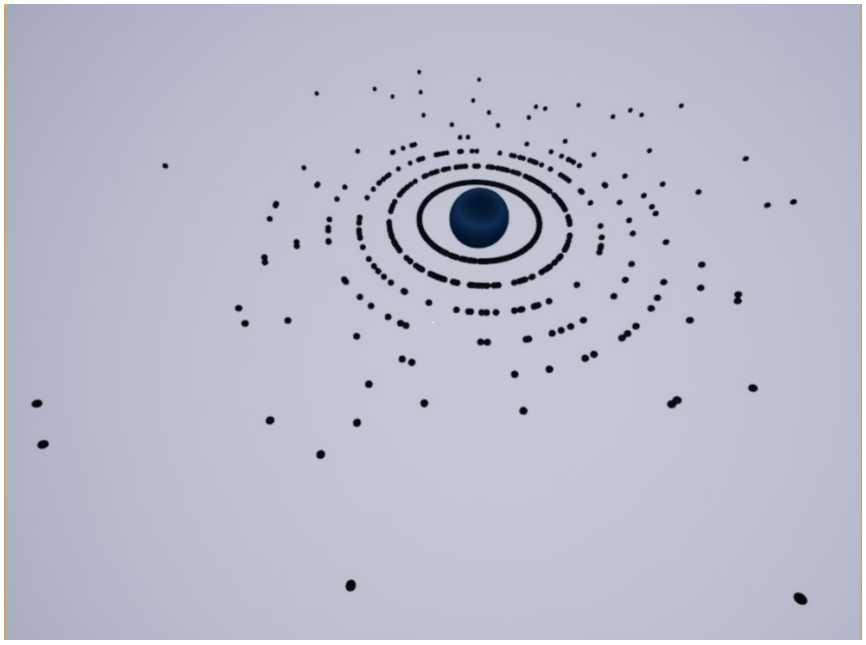
As we processed the data relating to the positioning of the artists in 3D space, there were a few places (on both the data side and the parsing side) where we mislabeled which star corresponded to which artist. An example of the bug that was discovered, where the location of artists didn't line up with their degree of interconnectedness is shown below. The Earth, Wind and Fire star was waaaaay out in the fringes of the graph, despite being very well connected, which should not have been the case. Thanks to our 2D debugging visualization, we were able to determine that it was simply the labeling of the stars that was going awry.



The image below shows some of the debugging stats that were used to identify the major bottlenecks in our visualization process. More specifically, this shows the spikes in computation as the back side of the 3D labels were shown to the camera. These Labels were abandoned for screen space equivalents, both for speed and visibility reasons.



Finally, the image below is an interesting visualization of the number of albums that all of the artists in the data set have. Each subsequent album is spawned with a slight increase in orbital size, so when all artists are spawned on top of each other you can see the statistical spread of the number of albums each artist has in the music library. As seen in the visualization, the majority of artists in the library are represented by only one or two albums.



The design of the project didn't change drastically from the original concept. The main differences between the original concept and the result are as follows. The Idea of showing "smaller" songs of an album as landmasses on the planet/album was nixed, as we had no easily implementable way of producing those land masses, nor ways of ranking songs as larger. We had songs plays, but because of the system involved in converting the songs into lastFM data, all the song play counts were 1-4 in range. Of course, this could be implemented if we did additional future work on this project outside of this class.

Many of the design changes we made were based on the feedback of peers, and these are discussed in more detail in the next section. Such changes included showing selected graph edges in the final visualization and moving unconnected artists farther away from the main clusters.

Feedback and Future Work

The first class feedback we received was from the LMS thread where we proposed final project ideas. One student questioned whether we would use a wide selection of artists in the final product or only use a given user's library. Since our primary goal is to help the end user understand the relationships between the musicians in their own library, we told them we would only be using the artists already in the user's library. Another student suggested we could show the artists over time, where a star's lifetime depends on the years that particular band was active. While this is a cool idea, it may distract the user from the overall connections between artists that we are trying to convey. If we had more time to extend the project, we could possibly add a time selection element, but we did not plan to make it an explicit goal for the final visualization.

We got further feedback on our project during the in class demo session. Pretty much any person who listens to music using their computer would fall under our target audience, but we searched out students who had a similar music taste as Jim so they could better critique the connections we showed between artists. We showed our peers the 2D graph visualization shown above, and asked them if they thought the edges connecting artists were meaningful and interesting, and if it would encourage them to look at their music library in a new and critical way. The responses we received were very positive, and all of our testers found the connections between artists to be meaningful and interesting.

Although we informed our testers that the graph edges would not be explicitly present in the final visualization (they are only used to determine the artists' position in 3D), many testers liked being able to explicitly see the artists related to a selected artist. The testers also said that these graph edges would provide them with a path to explore through the visualization, moving from artist to artist based on the graph edges. We decided that it would be much too busy to show all of the implicit edges between artists. Instead, we decided to show only the graph edges connected to whatever artist the user is currently selecting.

As seen in the 2D graph visualization of our artists, there are many bands and musical groups on the outskirts of the graph that are not connected to any other nodes, either because Spotify has no knowledge of that band, or because that band is very different from every other band in the user's library. We were concerned that the placing of these artists might confuse the user, since completely unrelated artists could end up being placed near one another. We considered adding "dummy" edges between these nodes and other connected nodes of a similar genre, and asked our peers for their input. It turned out that all of our testers were not interested in the unconnected artists in favor of exploring the clusters of artists in the middle of the visualization. They suggested we just push those nodes out and away from the main clusters, and we decided to oblige them.

Lastly, some testers felt we should classify our clusters in some explicit way. For instance, in our 2D example, there are very clear clusters of classic rock, blues, and motown. A few testers thought these clusters should be color coded, so the user could more clearly navigate through their library and have a better understanding of where they are in the larger visualization. We were hesitant to color code our stars, especially because most of our testers liked exploring the artists that served as bridges between multiple genres. Color coding these artists would be difficult, even if there was a gradient scale for which genre an artist fit into best.

Thus we decided that color coding would be detrimental to the experience of the end user, and decided that the clusters of artists themselves should help provide the users with context.

If we were to work on this project after this class, there are a great many ways we would like to extend it. One of the most obvious improvements (and one of the most complicated) would be to allow users to listen to their music directly from the visualization. This is currently outside of the scope of the project, but it would undeniably make the visualization more engaging and useful for the end user. It would be extremely complicated to implement this, not only because of the actual coding of the backend, but because there is no guarantee that the music in their library can be found on another service like YouTube. For instance, if a user was to record their own song and listen to it in Google Play Music (which is then scrobbled over to LastFM), there will be a record that the song in question is in the library, but there is no way to play that particular song from the Google Play Music service. This seems like a corner case, but with increasing issues of copyright being used to control music on online services, it would be very difficult to guarantee any music playback. Of course, all of this could be avoided if the project was implemented with data from iTunes, so all of the song data was available locally on the user's machine. Also, if we were to add media playback to our final product, we would also like to add some searching functionality so a user could jump to a specific artist.

We could also extend our project to present more information about the music library to a user. For instance, the size of planets or stars could represent further information, like how frequently the user listens to that artist or album. In addition, some songs in a user's library are not associated with an album, either due to incomplete data or that song being a single. Given more time, we could illustrate these songs as a comet or asteroid orbiting the star (artist), as opposed to being attached to an album titled "unknown" as it is now.

Features and Implementation

Many of the features of our final product are related to the user interface. Our application features a movement system to explore the stars in a user's music library, as well as a variety of ways for our user to select items of interest, including center of screen selection and brushing. We also feature smoothly interpolated movements between the objects that the user has decided to focus on. This includes the hierarchical and intuitive movement involved with moving between the many layers of a solar system. Our visualization also draws the "similarity" edges between the selected star and the stars that are deemed most relevant to it. The system handles the spawning and deletion of planets and moons as the user focuses on specific artists.

This set of features allows our users to interact with their music library in a way that was previously impossible (as far as we know). By allowing them to see a selected artist in the context of their larger library, they are allowed to glean new insights into their own musical tastes. Furthermore, the users are able to focus on a particular artist and explore the albums and songs created by that artists in the user's own music library.

Many of the initial implementation challenges were related to acquiring and processing our data. Our initial plan was to directly download the necessary music data from Google Play Music, but unfortunately that service does not have a public API. There is, however, a very sketchy unofficial API, and this was originally going to be used. Unfortunately, this unofficial API was geared towards developers who wanted to be able to stream music from Google Play through their own application. This meant there was no easy way to use the API to download a user's entire library, so this too was abandoned. A solution was found with LastFM, which allows music from other streaming services to be archived through "scrobbling". The LastFM service has its own public API that allows for the downloading of all music listening data.

Unfortunately, the LastFM site has recently gone through many architectural changes meaning that some parts of both the python wrapper we utilized and the API service itself were non-functional, which forced us to use a third party script to download the raw data (as discussed later in this report).

Once the working API calls were worked out, we were able to download all artists in the user's LastFM library. These were then used in conjunction with the Spotify API to find the artists most related to each artist. Each artist was added as a vertex in a graph, with each artist relation representing an edge between two artists in the graph. Finally, the Fruchterman-Reingold force directed graph drawing algorithm was used to create a valid layout of our graph in 3D space. This layout assigns an (x,y,z) coordinate to each artist, which was then used to determine the position of each star in the final visualization.

For data collection, a Python script created by Nathan Povo (available on GitHub at https://github.com/nat143/lastfm-data-exporter) was used to download the raw, unprocessed data needed from LastFM. We made a separate script to process the artists data and form the implicit graph needed in the final visualization; this process involved API calls to both the LastFM and Spotify services, and the use of the Pylast and Spotipy python wrappers, respectively. This script leveraged the iGraph Python library, which allows for the easy creation of graphs with a large number of nodes and edges. The iGraph library also allowed us to easily use the Fruchterman-Reingold force directed algorithm to fix the location of the artists in 3D space.

As far as the actual visualization went, we used a variety of existing code sources. By using the Unreal engine and a strong foundation of vector math, string parsing and rendering was simple to build off of. Cursory level set ups linking the player input and controls were also available reducing the time spent simply acquiring input. The other external code used was the

Victory Blueprints plugin by Rama. Most of this functionality consisted of simple functions related to vector math and string parsing. Materials and textures were used from a Solar System visualization that was also made in Unreal.

Perhaps the most important data structure we created was the graph of artists we have previously discussed in detail. This graph was essentially just an extensive setup as input to the Fruchterman-Reingold force directed graph drawing algorithm. This algorithm is a very efficient one that seeks to minimize edge crossings and evenly distribute vertices, but most importantly it can easily be implemented in three dimensions [4].

On the visualization side of things, an attempt was made to utilize a hashMap to get around a limitation where more than 1000000 instructions would time out as an infinite loop. This attempt backfired spectacularly and consumed at minimum 7 hours of our time. A few structs and enums were used and a stack was made to control the player movement through the star hierarchy. Python dictionaries also were utilized to pair up artists and their connections with their locations. This pairing accomplished what was originally intended for use by the failed C++ hashMap.

Work Breakdown

As far as the allocation of work in this project, Jim was primarily focused on the collection and processing of the data, as well as the creation of the debugging visualization, while Dash focused on the visualization of the stars and planets. The data we collected was actually taken from Jim's LastFM library, so he was involved in collecting all of the data from said library and processing it using the Spotify API and iGraph in order to determine where each artist should be placed in the final visualization. Dash then took this data and used it to place each artist in the visualization at the proper place, and used the library data to visualize each artist's albums and

their respective songs. Both of us contributed to the proposal, progress reports, presentation, and this report.

References:

[1] Hilliges, Otmar, Phillipp Holzer, Rene Klüber, and Andreas Butz. "AudioRadar: A Metaphorical Visualization for the Navigation of Large Music Collections."Smart Graphics Lecture Notes in Computer Science (2006): 82-92. Web. 7 Apr. 2016.

[2] Stober, Sebastian, and Andreas Nürnberger. "MusicGalaxy: A Multi-focus Zoomable Interface for Multi-facet Exploration of Music Collections."Exploring Music Contents Lecture Notes in Computer Science (2011): 273-302. Web. 7 Apr. 2016.

[3] Graham, Hamish, Hong Yul Yang, and Rebecca Berrigan. "A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics." A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics (2003): n. pag. 10 Oct. 2003. Web. 7 Apr. 2016.

[4] Fruchterman, Thomas M. J., and Edward M. Reingold. "Graph Drawing by Force-directed Placement." Softw: Pract. Exper. Software: Practice and Experience 21.11 (1991): 1129-164. Web. 20 Apr. 2016.