Disco Planet — COM-480 Process Book

Aleksei Kashuba Ivan Yurov Ekaterina Svikhnushina
Department of Computer Science, EPFL, Switzerland

I. Introduction

Modern means of communication and transportation have brought the world closer together. With many cities turning into melting pots, cultural, social, and personal motives drive people's curiosity to explore how diverse backgrounds and experiences intertwine and blur geographical borders. Since music plays a central role in cultural identity, our project strives to provide these insights by looking into the musical preferences of people around the globe.

Many researches and music enthusiasts analyzed music-related datasets before. Every Noise at Once[1] project presented a 2d-map of music genres, providing a possibility to listen to exemplary pieces. Several other studies investigated music profiles of cities [1, 2], geographical regions [3], and countries [4]. However, to the best of our knowledge, no prior work has ever presented an easy-to-use, engaging, and impactful interactive visualization of musical phenomena.

For the Disco Planet project we employed the dataset retrieved from Spotify, the most popular music streaming service in the world in 2019 [5]. The three visualization views address the following major questions: 1) What music genres are popular in different corners of the world? 2) How various cities are similar or different in their musical preferences? 3) How do numerous genres and sub-genres of music relate to each other? We expect that both the general audience and more inquiring music lovers will engage with our website and learn curious insights during the interaction.

II. Data

We retrieved our main dataset using the Spotify API.[2] The dataset contains information about artists' popularity in the cities of the world. In particular, it shows the top 50 cities in which people listen to the given artist as well as the number of listeners. It also provides information about music genres and their associations with the artists. Additionally, we computed the clustering of different music genres, which is described in more details further in this process book.

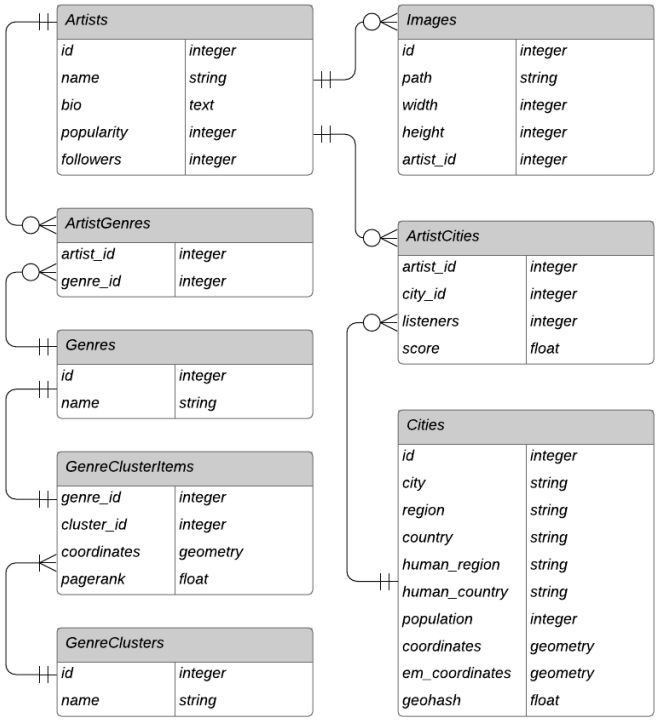

Fig. 1: Entity-relationship diagram for our project database.

To visualize the data on a map, for each city in the main dataset, we retrieved geographical coordinates from the World Cities Database.[3] Further, we collected the population information of each city from the World Cities Population dataset[4] to be able to normalize the listener counts.

The entity-relationship diagram of our database is illustrated in Fig. 1. In total, the database contains 58 502 entries in the `Artists` table, 3 823 entries in the `Genres` table, and 3 277 entries in the `Cities` table.

III. Tools Overview

Our visualization is meant for a web browser. Throughout the developmental process, we considered

[1] http://everynoise.com/
[2] https://developer.spotify.com/documentation/web-api/
[3] https://simplemaps.com/data/world-cities
[4] https://public.opendatasoft.com/explore/dataset/worldcitiespop/

the website performance to be of the same priority as visual attractiveness and ease-of-use for the viewers. Slow-loading pages and lagging interactivity are detrimental to delivering compelling user experience. Given the number of entries and connections between them in our database, loading this amount of data in one request did not seem viable. Hence, we decided to split the application into a front-end and a back-end.

A. Front-end

We use React to be able to decompose the app into smaller reusable components. Conceptually React becomes a scaffolding, Three.js serves as a renderer, and D3.js is used very granularly mainly for its brilliant helpers such as scale converters. The 2d visualizations are implemented on top of `html5` canvas with the help of D3.

B. Back-end

To be able to interact with the visualization smoothly, we introduced a GraphQL back-end service allowing to execute dynamic queries. The server-side consists of a Phoenix app written in Elixir, while the data are stored in a PostgreSQL database with the geospatial extension PostGIS.

IV. Project Evolution

Disco Planet offers three distinct views for the users, i.e. *Globe View*, *City Space View*, and *Genre Space View*, to explore different aspects of the data. As our project develops on the subject of music, we tried to use familiar metaphors to help viewers easier relate to and enjoy the experience throughout their interaction with the website. The Earth globe from the *Globe View* was made to resemble a disco ball, and a music player was integrated into each view, allowing users to listen to different representative pieces. In spite of sharing several control elements, each of the aforementioned views visualizes different aspects of the data and underwent unique evolutionary process to reach its resulting appearance. Further in this section we describe the path we took to obtain the final result for each of the three views.

A. Globe View

The *Globe View* is the starting view, and the core part of the visualization meant for a broad audience. It demonstrates the popularity of different music genres in cities around the world. Initially, we thought about visualizing this data on a map. After more thorough consideration, we concluded that 2d space might be too flat to disclose complex and surprising dependencies from the data in a sufficiently engaging and impactful manner. After some additional considerations, we envisioned this visualization as a globe in a 3d space. The first sketch of the visualization is shown in Fig. 2.

1) Scene: The scene is composed of two spheres representing the planet with an atmosphere and a particle system serving as stars. Stars are uniformly distributed around in space at the distance interval (r_1, r_2) and are animated for blinking using shader-powered transform based on elapsed time. The earth sphere has a night texture and a bump map to make an appearance of the terrain more natural.

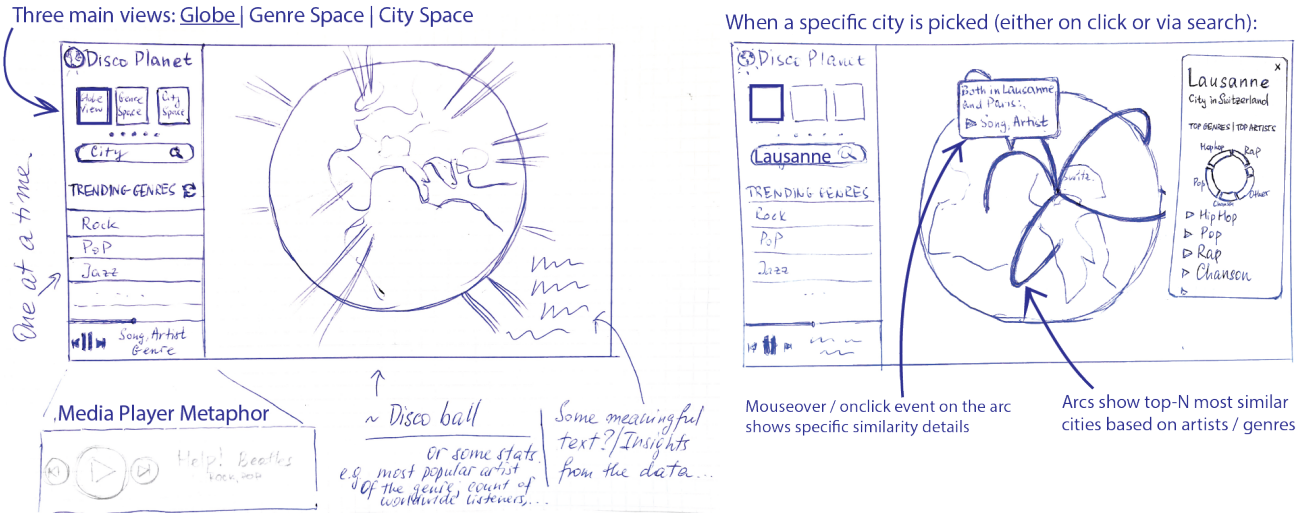

Fig. 2: Sketches of the Globe View, the landing page of our project.

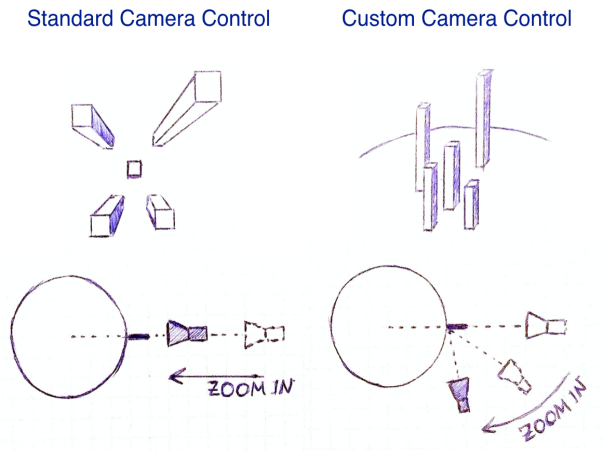

Fig. 3: Standard camera control vs Custom camera control.

2) Navigation: The standard camera control provided by Three.js (OrbitControls) very quickly proved to be unusable for our application: with fixed zoom-speed, the perceived speed of approach grew exponentially as the camera was getting closer to the surface, also the vertical view of the city of interest would not let to compare the height of the genre bars of this city and its neighbors at a glance. The custom component checked both marks by adjusting zoom step proportionally to the distance left to the surface and by moving camera off the path while keeping direction at the point of interest on the planet as shown in Fig. 3.

3) Genre bars: Genre bars were designed to show the level of popularity of a genre in particular cities (Fig. 4). By design the default state of the bar is flat and white, which appears as a bunch of white dots on the globe.

Fig. 4: Genre bars showing the level of popularity of different genres around the world.

When a genre is selected and a city has a statistically meaningful presence of the listeners of this genre, the bar grows to the sky and changes its color according to the current genre-color assignment, defined by Genre Selector widget. Users can choose up to ten genres to see their popularity in comparison. As we have more than three thousand genres on the map to be shown simultaneously, the naive approach of just drawing all these 3D objects in the scene and trying to control their properties would appear infeasible; we needed something more sophisticated if we were going to deliver on the promise of performance and smooth user experience.

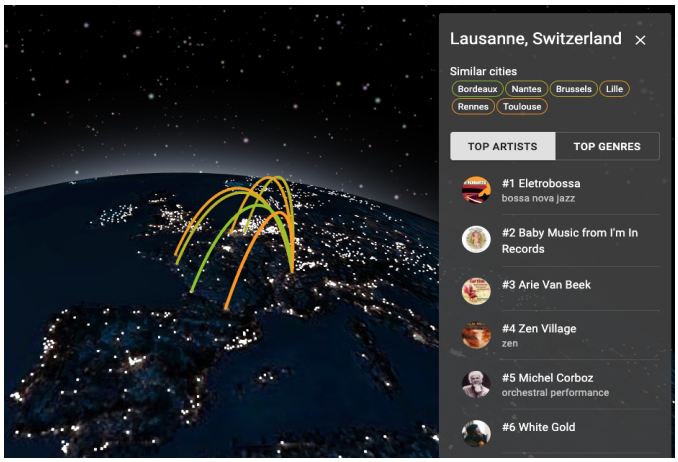

Fig. 5: Arcs to the cities with similar musical preferences from Lausanne.

The solution was the Mesh Instancing, the technique that allows to produce multiple copies of the same geometry using only the power of GPU. To customize the properties of each instance individually, we had to create custom shaders that relied on attributes such as position, rotation, z-scale (to control the height of the bar), and color. We also used the cameraPosition matrix available in the shader to compute the distance to the city and apply x/y-scale to the bar to make its horizontal dimensions appear the same independent of the zoom level. After all optimizations the rendering became extremely smooth while CPU load was negligible.

4) Similarity arcs: When a user selects a city of interest, we clear genre selection and render the arcs to similar cities instead (Fig. 5). The similarity is measured based on genre-popularity vectors of all cities and to make it more interesting, we don't show immediate neighbors, instead, we show cities closest in musical tastes which are at least 500 km apart. The color of the arc represents musical similarity from green, very similar, to orange, less so. We also show similar cities

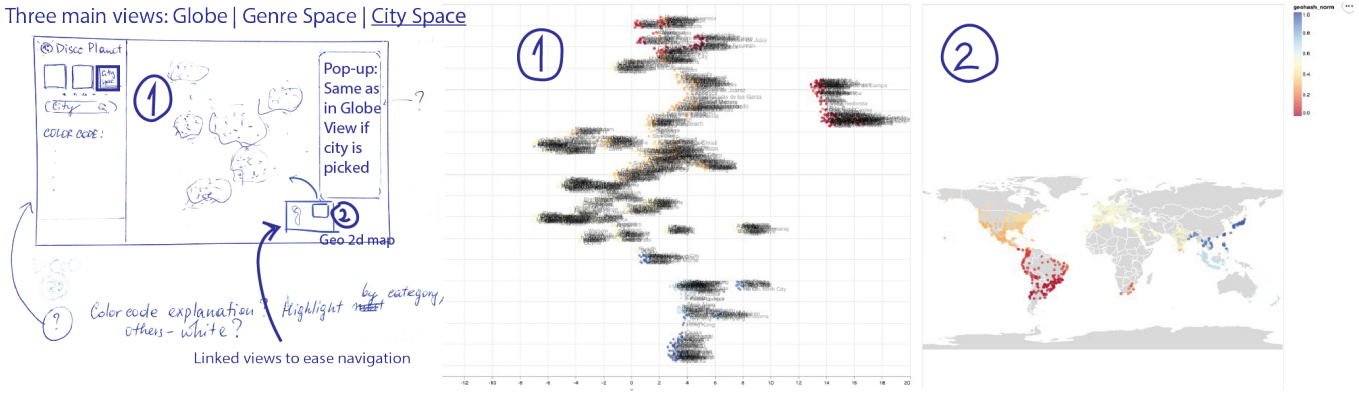

Fig. 6: A sketch and visualization prototype of the City Space View.

in the context panel for quick navigation. One can just jump from city to city based on the musical distance and explore what kind of music is popular there, using Top artists widget.

5) Top (local) artists: When city is selected, we show an "infinite" list of artists sorted by popularity in this city. User can listen to a 30s excerpt of the artists most popular track.

B. City Space View

The *City Space View* was designed for those who are willing to dive deeper into the concept of a vector embedding space. It showcases 1000 of the largest cities of the world arranged on a surface in such a way that if two cities are close together on the surface, then the populations of those cities listen to similar music. Through the use of color and an interactive map widget, we provide information on the connection between the geographical locations of the cities and the location in the music space.

1) Data preparation: From the outset of the project, we had the idea to somehow relate the cities based on what music their population is listening to. However, initially, there was no clear path to that goal. Then we realized that the collected data could be represented as a graph, and we could use techniques such as spectral clustering. The matter was complicated by the fact that our graph contained multiple entities: artists and cities. After further research into the machine learning techniques for graphs, we discovered the `node2vec` algorithm [6]. `node2vec` generates random walks on a graph and then feeds them into the `word2vec` algorithm [7] to generate node embeddings. This algorithm inspired us to interpret the normalized listener count as a probability of transitioning from a city node to an artist node. Using this interpretation, we were able to solve our problem of having multiple entities in a graph. Now, if two cities shared an artist in common, we multiplied their probabilities of transitioning to that artist in order to get the probability of transitioning between those two cities. After computing all such probabilities, we fed this data into the `node2vec` algorithm. For each node, the algorithm generates a vector; the number of dimensions can be configured. To be able to visualize a node, we needed to be able to represent it using at most three parameters, one for each dimension. At this point, we began prototyping the visualization using simple tools such as matplotlib (Fig. 7). We wanted to verify our hypothesis that the algorithm produced interesting results.

First, we tried visualizing the output of `node2vec` directly but were dissatisfied with the result. Instead, we opted to produce node embeddings with large dimensionality first and use a dimensionality reduction technique on the output. Two such algorithms were considered: `t-SNE` [8] and `UMAP` [9]. After some experimentation, we concluded that the output of the `UMAP` algorithm was more satisfactory.

2) Conception: After producing node embeddings, we realized that the size of the visualization warranted a level of interactivity; otherwise, the labels were not readable. We began building interactive prototypes of the visualization using Vega-Lite (Fig. 6). From the initial results, it became apparent that there was a connection between the geographical location of the city and its position in the music space. We decided to showcase this connection with the use of color. We used the z-order curve, which can reduce two-dimensional points into only one dimension. Using this technique, we were able to produce a coloring that allows the user to tell

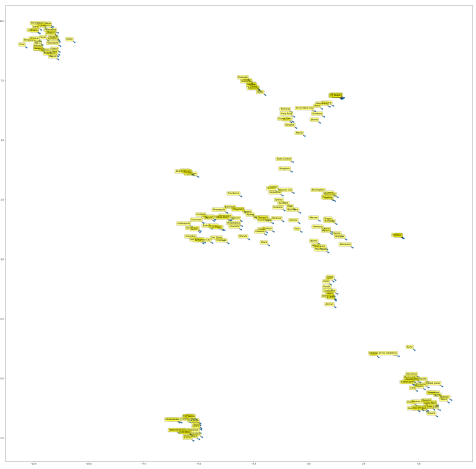

Fig. 7: Node embeddings of 200 cities in matplotlib

in which continent a city is located at a glance of an eye. However, not all our ideas could be implemented in Vega-Lite, such as an interactive map widget. So we moved on to the next stage of prototyping using D3 and the SVG format.

3) Implementation: After building skeletal prototypes using the SVG format, it soon became apparent that the performance of the prototype was not satisfactory, usually rendering at most 30 frames per second (fps). We decided to switch to the `HTML5` canvas to try to improve the performance. Further performance optimizations included calculating which points are currently in the user's viewport and only rendering those. The performance significantly enhanced since modern browsers use GPU acceleration for canvas rendering, and the visualization can be rendered at 50-60 fps depending on the user's GPU. After completing the skeleton, we proceeded to add more features. First, we implemented an interactive map widget that allows users to explore the connection between the geographical location of the cities and their positions in the music space. Now, the user can select a country or a region and see it become highlighted. In order not to overload the user with the labels of the cities, we decided to show them progressively. The labels are shown at different scales depending on the size of the population of the city. The algorithm that we implemented calculates which cities should be visible, then how much space each label takes and lays them out in a greedy manner in such a way that they do not overlap. We also wanted to add interactivity to the music space itself, which became more complicated with the use of canvas. First, we implemented a technique where each interactive element is assigned its color and is then displayed in a hidden canvas. When a user clicks on the displayed canvas, we can deduce which element was clicked by looking up the color in the hidden canvas. However, the technique proved unreliable. Instead, we calculate the positions and the sizes of all interactive elements that are displayed at any moment and see if the position of the click is within the bounds of any one of them.

4) Iteration: After receiving some feedback on the prototype, we realized that the visualization also required quite a bit of textual explanation to be understood. To make the description more exciting and engaging, we added a number of insightful observations into the left panel together with the links that take the user to the area of the visualization with the point of interest (Fig. 8). We expect that these observations will allow users to build an intuition on how to interpret the visualization.

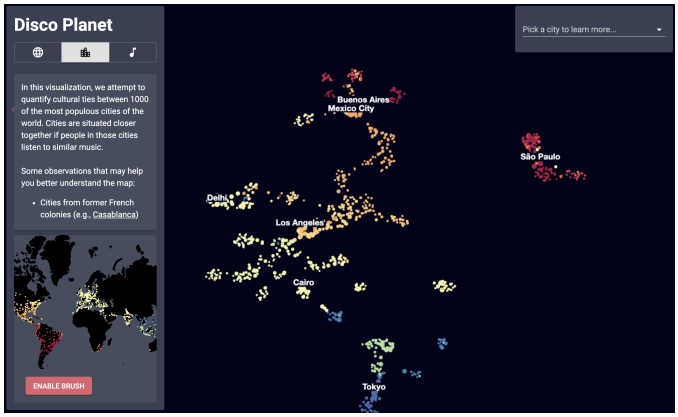

Fig. 8: City Space View showing the layout of the cities along with textual explanations (in the left panel).

C. Genre Space View

The *Genre Space View* is conceptually related to the *City Space View* and shows the music genre clustering for top-1000 genres from Spotify. Initially we were considering visualizing genre relationships in a 2d space, as shown in the first sketch of this view in Fig. 9. As a result of further discussions and data wrangling, we concluded that 3d representation would be more gripping for the viewers and fit better in the overall concept of the project.

1) Data preparation: The idea of the visualization was to show how numerous genres from our dataset

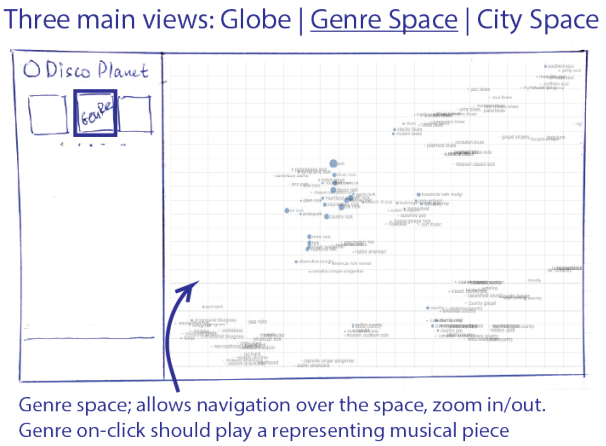

Fig. 9: A sketch of the Genre Space View.

relate to each other and what clusters are formed. We quickly realized that showing the arrangement of several hundreds of genres in some abstract space without any grouping markers would not bring much insight to the user and would make the interaction with the visualization very tedious. Therefore, we decided to explicitly compute the clusters of genres based on the relationships retrieved from the data.

For this purpose, we first constructed a graph of genre associations based on how frequently two different genres appear together for every single artist. Then, we applied the `Walktrap` community detection algorithm [10] to the resulting graph in order to obtain preliminary

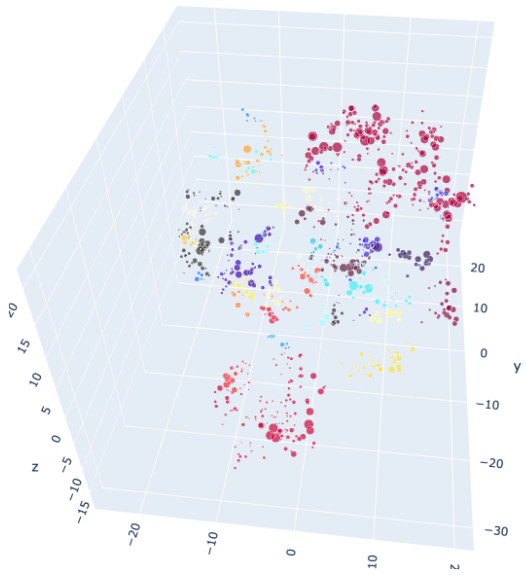

Fig. 10: Visualization prototype of genre clustering in the 3d space with Plotly.

genre clustering. The next step was to "project" the graph of genre associations into a 3d space.

Similarly to the approach taken for the *City Space View*, we first used the `node2vec` algorithm to create an embedding for each genre. Further, we applied the `t-SNE` technique to reduce the number of embedding dimensions to 3. After visualizing the resulting 3d space we discovered that cluster assignment obtained with community detection algorithm matched closely the actual clustering of genres prompted by the `t-SNE` coordinates. Minor manual cleaning and adjustments were performed to bring the clustering to its final state. In total we identified 38 genre clusters, which were used both for the *Genre Space View* and *Globe view* to simplify user navigation and interaction with the visualizations. Finally, we decided to indicate the relative importance (or popularity) of different genres by varying the size of their corresponding sphere radii. We employed the `PageRank` algorithm [11] to compute this metric. Throughout the data processing we were using Plotly to prototype the visualization (Fig. 10).

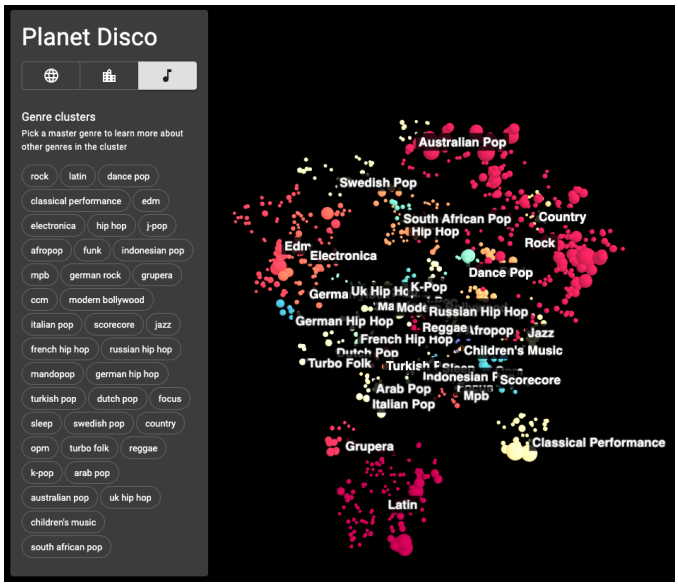

Fig. 11: Genre Space View showing the cloud of all genres.

2) Implementation: Genres in 3D space also use mesh instancing for performance, and their color, luminosity, and transparency are controlled through geometry attributes and executed in shaders (Fig. 11). This view allows for immediate interaction of the user with objects using raycasting (see next section).

3) Navigation: We support two modes of navigation: the user can pick a cluster of interest using buttons on the control panel, or click on the spheres directly. Once

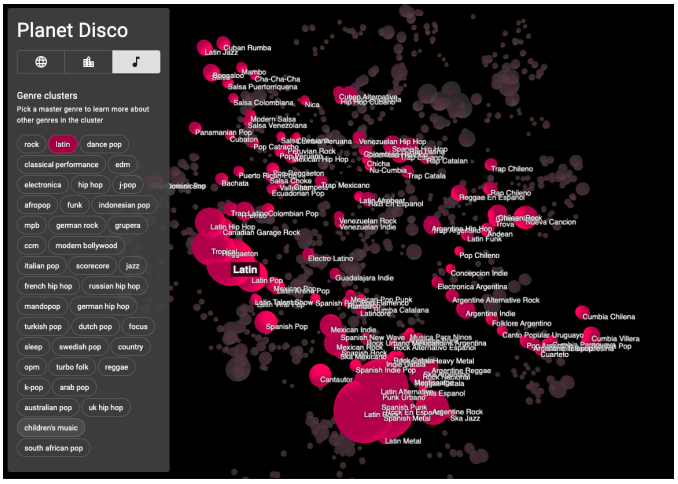

Fig. 12: Genre Space View showing the selection of the Latin genre.

a cluster is selected, the camera changes its pivot point to the center of the selected cluster and further controls allow to fly around that exact cluster, while other spheres become semitransparent and colored neutrally (Fig. 12). With the cluster selected the user can click on the sphere belonging to the cluster to open a Genre Panel and listen to the artists belonging to this genre. Or click elsewhere to clear cluster selection.

4) Lighting: The scene has a default ambient light along with far placed spotlight and a dynamic point light that is following cluster selection for creating an illusion that cluster itself is highlighted [12]. The dynamic point light, that we call TrackingLight, has both position and color animated which makes transition between clusters especially spectacular.

V. Discussion

As the time allotted for the project was limited we had to compromise non-priority aspects of the visualizations to ensure the quality of the core deliverable. The known issues include:

- The music playback interrupts if the user switches to another visualization view.
- Some functionality might be unstable in the Safari and Edge browsers.

Solving the listed issues would be a logical continuation of our project. On top of that, we have a number of extra ideas to further enhance the visualizations. For the *Globe View* we could add the possibility to listen to representative tracks for each genre in the left-hand side control widget. Additionally, for the city widget, along with showing the list of most popular local artists, we could also include the list of trending genres. For the *City Space View* we could show the links between the cities more explicitly and demonstrate which artists connect each pair of cities. Another idea, applicable both to the *City Space View* and *Genre Space View* would be to extract the top-artists for the user based on his/her Spotify account information and display where the user would find him- or her-self in each of the spaces respectively.

VI. Peer Assessment

In general, we all agree that the development process went smoothly. We tried to split the work based on each member's experience, strength, and interests and are all satisfied with the results. We maintained constant communication in our team chat and were helping each other when needed. Every one or two weeks we organized an online meeting to discuss the project status and agree on the next steps. Every team member was always prepared for the meetings and contributed productively to the discussions and work. Everyone's opinion was respected and considered during communication. In case of disagreements, we engaged in a constructive discussion, considering the pros and cons of different alternatives to make a rational decision. Overall, all three of us remained flexible and always supported each other's progress.

Below we include the breakdown of the the parts of the project completed by each team member.

Aleksei Kashuba:

Spotify data retrieval and cleaning; analysis of musical preferences of the cities; *City Space View* visualization; music player configuration; overall integration and web hosting configuration; partial contribution to process book.

Ivan Yurov:

Coordinates and population data retrieval and cleaning for the cities; back-end configuration; *Globe View* visualization; *Genre Space View* partial enhancement; website performance improvements; partial contribution to process book.

Ekaterina Svikhnushina:

Visualization sketches; analysis and clustering of genres; *Genre Space View* visualization; usability and interaction logic partial testing; related work review and project documentation preparation (README, report, process book).

REFERENCES

[1] S. Grossman, "What type of music is your city most passionate about?" March 2014, https://time.com/37332/music-preference-maps/.

[2] D. Hauger and M. Schedl, "Music tweet map: A browsing interface to explore the microblogosphere of music," in *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016, pp. 1–4.

[3] C. Mellander, R. Florida, P. J. Rentfrow, and J. Potter, "The geography of music preferences," *Journal of Cultural Economics*, vol. 42, no. 4, pp. 593–618, 2018.

[4] M. Schedl, "Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset," *International journal of multimedia information retrieval*, vol. 6, no. 1, pp. 71–84, 2017.

[5] I. News, "Chart of the week: The world's most popular music streaming services," February 2020, https://www.fipp.com/news/insightnews/chart-week-world-most-popular-music-streaming-services.

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[8] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[9] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *ArXiv e-prints*, Feb. 2018.

[10] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences*. Springer, 2005, pp. 284–293.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[12] P. Beshai, "3d data visualization with react and three.js," January 2020, https://medium.com/cortico/3d-data-visualization-with-react-and-three-js-7272fb6de432.