Department of Electrical and Computer Engineering

Aarhus University

LEGO Database Visualization

Group 13:
Márta Simon - 202001346

May 5, 2022

1 Abstract

The aim of this visualization project was to explore the released sets of LEGO over time, categorizing them by the theme they were released under while taking advantage of the materials we learned in the Data Visualization course. The data was gathered from Rebkrickable (2) and the project was implemented in JavaScript with the D3.js library (1).

2 Introduction

LEGOs are world-famous plastic construction toys manufactured by the privately held Danish company The Lego Group. The toys are usually sold in sets in order to build a specific object with the given building pieces.

Rebrickable (2) is a website made by collectors of LEGO who were interested to know which other sets they could build from the sets and pieces they already owned. They built an entire database from the official LEGO sets, parts and minifigures describing the relationships among them and other features like the theme they were released under or the color of their pieces. The database is maintained daily and is free to use for any purpose. It can be downloaded in csv format from (2).

2.1 The Visualization Problem

The problem which this project aims to solve is to give an opportunity to visually explore the released sets of LEGO over time with their corresponding themes. The database is fairly huge with 18215 different sets in it, so it is important to provide both a greater overview and a more detailed view of it and let the user interact with those views preferably through multiple coordinated views.

The objective or analysis task is data identification to support lookup tasks and comparison tasks on the individual data items. Examples for lookup tasks: "In a given year when did LEGO release the most sets under a certain parent theme?" and "Which were those sets?". An example for a comparison task: "Throughout the years under which parent theme did LEGO release the most sets?"

The subject is the data from Rebrickable. Further details on it in the Appendix.

3 Inspiration

The inspiration of this project was to provide insights in the release history of LEGO and in the evolution of their repertoire. The realization of this project was inspired by Curran Kelleher's exciting stream graph visualization (5).

4 The Process

4.1 DAF: Understand

The target users of the visualization are collectors and fans of LEGO who are interested to see and explore the evolution of LEGO's repertoire. The database consists of many tables, but for this problem only the "sets" and the "themes" tables are needed. The visualization's access modality is a home PC browser.

The design requirements of the project are to

1. give an overview of the released sets over time
2. show the sets in more detail
3. show the hierarchy of the themes and their relationship with the sets.

4.2 DAF: Ideate

To achieve the first design requirement, I chose to use a stream graph to provide a greater overview on how the (root) parent themes have developed over time through the number of sets LEGO released under them. To make the chart more readable, I linked a barchart to it, which shows the exact number of sets which were released in the given year.

To achieve the second and the third requirement, I chose a tree diagram to show the hierarchy between the themes and decided to add the corresponding sets as leaves to the tree. To show more details about the sets, a table was added next to the tree diagram.

4.3 DAF: Make

For the encoding of the data, a categorical color scale was used to distinguish the different parent themes. This encoding will be used both on the stream graph and on its linked barchart. The tree diagram won't need a color scale.

The layout will follow the order of reading (from left to right). Stream graph in the center of attention, on the right from it the linked barchart and below both the tree diagram.

The user will be able to see the details of the stream graph on the linked bar chart by hovering the mouse over on the stream graph. By clicking on the

stream graph, the tree diagram would show up and show the relationships of the themes and their sets.

4.4 DAF: Deploy

As it turned out, the stream graph wasn't expressive at all when all of the parent themes were shown on it at the same time. There are 134 parent themes in total, so checkboxes with the parent theme names were added to the left side of the layout to make the user able to filter on their interests.

To make the stream graph more effective, a focus & context feature was added to it. When the user is hovering the mouse on a specific theme all the other themes get somewhat opaque.

To refine the aesthetics, titles and other labels were added. Both the hovered and the clicked year of the stream graph are shown on the layout with matching titles. The titles are colored with blue and have smaller fonts than the selected attributes to keep the attributes in the focus of the user.

5 The Solution

Everything was implemented in JavaScript using the D3.js (1) library. The stream graph was implemented based on the tutorial (3) and the intended tree based on (4).

The data tables were preprocessed and more suitable tables for the charts were created in Python.

I ran out of time to finish this section...

6 Discussion

6.1 Limitations

The visualization supports PC layout only with the resolution of 1920 x 1080. Color blindness is not supported. The colors linked to the parent themes are always changing whenever a new themes added or removed by the checkboxes.

7 Appendix

7.1 Database

The database from Rebrickable had quite a complex structure with multiple tables (shown in Figure 1) and was available in separate csv files. The structure of the data had a data table structure consisting of rows and columns where the columns represented different variables. The tables didn't have any missing values.

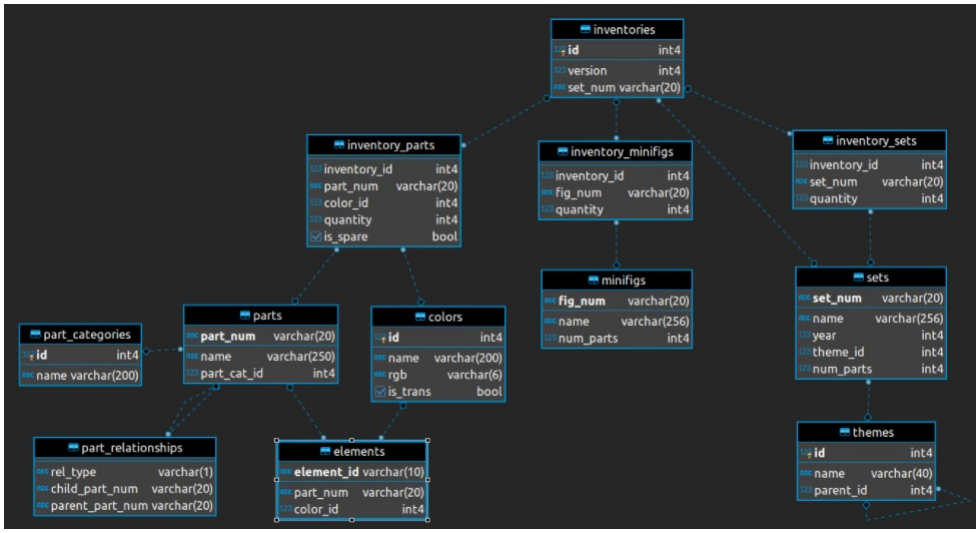

Fig. 1: Schema diagram of the LEGO database

The "sets" table (Figure 2) consisted of 18215 rows and 5 columns: "set_num", "name", "year", "theme_id" and "num_parts".

- "set_num": denotes a kind of ID for the sets (nominal data)
- "name": denotes the name of the sets (nominal data)
- "year": denotes the year the given set was released in (discrete data)
- "theme_id": denotes the ID of the theme the given set was released under (nominal data)
- "num_parts": denotes the number of parts/pieces the given set consists of (discrete data)

The "themes" table (Figure 3) consisted of 535 rows and 3 columns: "theme_id", "name" and "parent_id". There is a hierarchical relationship denoted in this table among the themes.

- "theme_id": denotes the ID of the theme (nominal data)

- "name": denotes the name of the themes (nominal data)
- "parent_id": denotes a reference to the ID of the parent theme of the given theme (nominal data)

set_num	name	year	theme_id	num_parts
31203-1	World Map	2021	709	11695
BIGBOX-1	The Ultimate Battle for Chima	2015	571	9987
10276-1	Colosseum	2020	673	9036

Fig. 2: Snippet from the "sets" table

theme_id	name	parent_id
1	Technic	nan
2	Arctic Technic	1.00000
3	Competition	1.00000

Fig. 3: Snippet from the "themes" table

References

[1] D3.js. URL https://d3js.org/.

[2] Rebrickable. URL https://rebrickable.com/downloads/.

[3] Stream graph. URL https://www.d3-graph-gallery.com/graph/streamgraph_template.html.

[4] Intended tree. URL https://observablehq.com/@d3/indented-tree.

[5] C. Kelleher. Streamgraph explorer. URL https://unhcr.github.io/dataviz-streamgraph-explorer/#types=1-2-3-4-5-6-7.