PROGRAMMING LANGUAGES EXPLORER

DATA VISUALIZATION / 2025

VHXPLORE

https://com-480-data-visualization.github.io/VHXplore/

Overview

As students, we regularly interact with a wide variety of programming languages—choosing them for their features, performance, or personal preference. Yet, we rarely stop to consider their deeper stories: when and where they were created, who developed them, how they rose to popularity, or even what inspired their names. Our project aims to change that by creating an interactive and engaging data visualization website that explores the rich and complex landscape of programming languages.

Using historical, technical, and demographic data, we present a comprehensive view of the most influential programming languages since the 1950s. Rather than just listing facts and statistics, our goal is to build a "living museum" of programming languages—one that is intellectually stimulating, visually compelling, and accessible to audiences of all technical backgrounds. Inspired by the interactive spirit of the Musée Bolo, we want our site to spark curiosity and invite exploration, making learning about programming languages not just informative, but genuinely enjoyable. Following are our visualizations highlights:

Temporal trends: When each language was developed and how they evolved over time.

Popularity metrics: Including the number of books published, academic papers produced, and estimated user base.

Geographic origins: Showing the top 7 languages from each country on a world map, offering a global perspective on programming language development.

Technical relationships: Exploring performance characteristics and how languages are interconnected.

Naming origins: Revealing the often surprising or humorous stories behind language names.

The visual identity of our website draws inspiration from the default dark theme of Visual Studio Code. This choice is more than just aesthetic—it serves as a subtle nod to the developer community. By adopting this familiar color scheme, we create an environment that feels both welcoming and slightly humorous, like an inside joke shared among those who've spent long nights in code editors.

By blending data with design, we hope this project provides both **educational value** and **exploratory enjoyment** for anyone interested in the evolution of programming languages.

Our path

Collecting the Dataset

We began by gathering data about programming languages. Our primary source was **pldb**, which includes information on over 5,000 programming languages and 353 features per language. This comprehensive dataset gave us a rich foundation to explore both well-known and obscure languages from multiple dimensions—temporal, geographic, and technical.

Planning the Visualizations

With the dataset in hand, we brainstormed **18 different visualization ideas.** In addition to the ones we implemented, we considered many creative and informative features. These included visualizing **humorous or niche languages** like Brainfuck and HolyC, showcasing **"Hello, World!" syntax** across languages, and comparing **language preferences** between beginners and professionals. However, we recognized that trying to fit all these ideas into a single page would overwhelm the viewer and dilute the experience. Ultimately, we selected **five of the most engaging and educational visualizations** to focus on, balancing variety with clarity.

Sketching and Visual Design

After finalizing our design direction, we sketched our ideas on paper, aiming for visualizations that were not only interactive and intuitive but also aesthetically cohesive. We chose a **VSCode-inspired theme**, using a **dark gray background** with **colors styled to resemble code snippets**. The text, buttons, lines, and visual elements were carefully color-coded based on the default syntax highlighting for Python and JavaScript in VSCode. This familiar design made the visualizations feel natural for developers, while also reinforcing the programming theme.

Implementing our Design

We implemented our visualizations using **D3.js**, focusing on interactivity, clarity, and performance. Each chart was built as a modular component, styled with custom CSS to match a VSCode-inspired theme using syntax-highlighted colors. To enhance user experience, we added hover effects, tooltips, and smooth transitions, allowing viewers to explore the data intuitively. By combining thoughtful design with D3's powerful capabilities, we created a cohesive and engaging way to visualize the history and **diversity of programming languages.**

Sketches and implementations

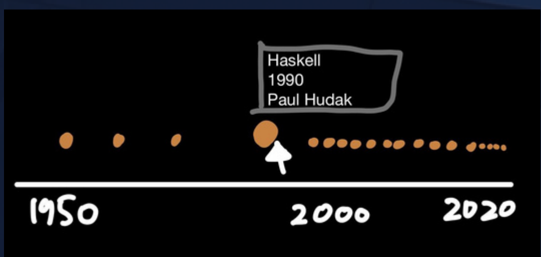
Origin stories

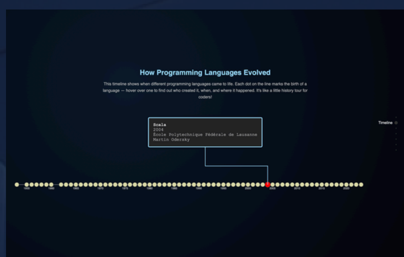
To highlight the often-overlooked cultural and historical background behind popular programming languages, we designed an interactive visualization in the form of a rotating galaxy—The Universe of Languages. Each language is represented as an orbiting planet, with dynamic background stars that respond to mouse movement, enhancing the immersive cosmic theme. Originally, we planned to mix different storytelling formats, but we found that keeping the interaction style consistent across all planets helped with clarity and user flow. As a result, each modal now presents a playful yes/no question that uncovers a curious fact about the language's name or origin. These small, animated interactions bring personality to the abstract concept of programming languages and invite viewers to learn their stories in a way that feels intuitive, memorable, and fun.

Timeline

For the timeline visualization, we mostly followed our original design but made several important refinements during implementation. Initially, the dots representing each programming language were placed slightly above the axis, which made it difficult to associate them with their respective years—especially when multiple languages appeared in the same year. To improve readability and alignment, we adjusted the positioning so that the dots sit directly on the axis.

We also rethought our data filtering strategy. At first, we excluded languages ranked below 100 to reduce clutter, but this approach caused issues when many low-ranked languages appeared in the same year. The tooltip often displayed obscure or unfamiliar names, which added noise rather than insight. To address this, we changed the logic to display all languages per year but only highlight the highest-ranked one. This ensures users are presented with more recognizable and influential languages, making the timeline both clearer and more engaging.





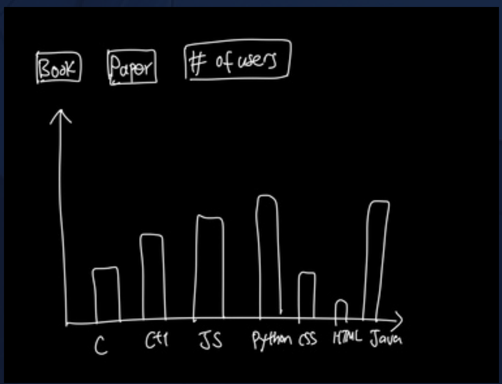
World map

For the map visualization, we followed our original concept but made a few key refinements to improve clarity and inclusiveness. Initially, we planned to filter out low-ranked languages and display only the most popular ones. However, this approach caused countries like the USA to dominate the map, while many others appeared empty or underrepresented.

To better reflect the geographic diversity of programming language development, we revised our strategy. Instead of applying a global rank filter, we now display the top 5 most popular languages per country, regardless of their global ranking. This change significantly increased the number of visible countries and highlighted a wider variety of languages, including many that are lesser-known. As a result, the map became both more balanced and more informative. Now, when users hover over a country, they can discover a richer mix of both influential and niche programming languages from around the world.

Line graph

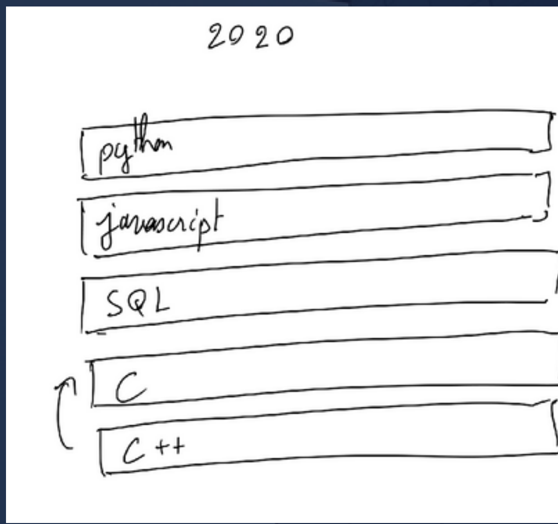
Initially, the chart was designed as a bar chart displaying book count, paper count, and number of users. However, it lacked interactivity aside from a single button, which made it difficult for users to engage with the data. To improve usability, we added a hover-activated toolbox to provide more detailed information. We later transitioned to a line graphto better connect data points, making the chart easier to read and navigate through hover interactions. Additionally, we included a dynamic title to clearly indicate the current metric being displayed, helping users stay oriented after interacting with the buttons.



Popularity

From the start we wanted to do a visualization of the popularity of programming languages over time. It seemed to be an essential part of the programming languages world to be visualized. Starting from milestone 2, we decided to do a bar chart race for this visualization. This choice appeared to be the best option as it would allow to display the popularity of programming languages in a very engaging way.

On the right we show the initial sketch we made for the milestone 2.



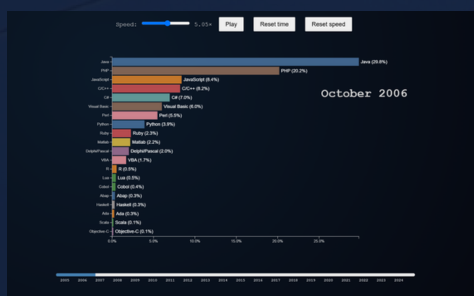
For milestone 3 one challenge we encountered was to find a good quality dataset to do this visualization. The dataset we chose to use is the **Most Popular Programming Languages Since 2004** dataset from Kaggle https://www.kaggle.com/datasets/muhammadkhalid/most-popular-programming-languages-since-2004

This dataset has data which was extracted from another dataset **PYPL PopularitY of Programming Language** https://pypl.github.io .The PYPL Index is created by analyzing how often language tutorials are searched on Google.The more a language tutorial is searched, the more popular the language is assumed to be. The raw data comes from Google Trends.
The Kaggle dataset contains 29 languages with data from 2004 to 2024. For every month and every language it contains a percentage which corresponds to the share of this language in the google searches for programming languages tutorials.

We were really happy with this dataset as it gives information about the programming languages popularity for a wide range of years. No dataset processing has been needed for this visualization.

Here is the final visulalization made with D3.js:

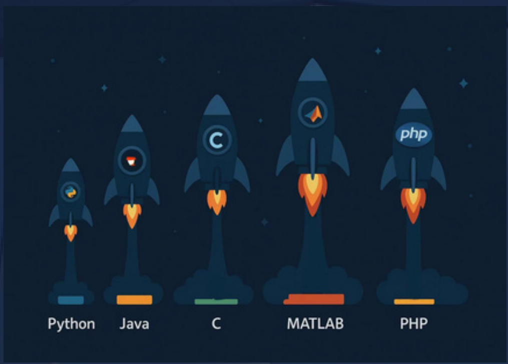
The visualization has a play button which allows to play/stop the animation. We gave the possibility to the user to adjust the speed of the animation as well as two reset buttons: a button to reset the speed and a button which allows to return to the initial date.



We added a progress bar below the bar chart race to give the user a feedback on the progression of the animation.

PERFORMANCE OF PROGRAMMING LANGUAGES

The performances of programming languages was another interesting topic we wanted to explore. Here is the original image from milestone 2 which was an artistic view, not really realistic.



Finding a dataset for this visualization was a challenge as not a lot of dataset about the topic are complete and of good quality. Luckily we found the following website: The Computer Language Benchmarks Game (CLBG) Your paragraph text .This website contains visualizations about performance measurements (execution time, memory usage, etc.) for a variety of programming languages running the same set of benchmark problems (like fast Fourier transforms, matrix operations, etc.).

The original data sets used by the website are located here: https://salsa.debian.org/benchmarksgame-team/benchmarksgame . We used for our visualization the dataset **benchmarksgame\public\data\alldata.csv**. This dataset contains for different **benchmark problems** and different **problem size** the information about the **program size,** the **cpu time** and the **memory** used by the program, for different **programming languages.**
The interesting thing about this dataset is that for a given benchmark problem, problem size and programming languages different compiler options and implementations of the program have been used and for each compiler option and implementation combination the program has been run multiple times to get the cpu time and memory usage.

One challenge we had to face was to create a new data set which gets al the data from **alldata.csv** and aggregates the results to only get the mean and standard deviation of the program size, the cpu time and memory usage indicators, for a given programming language for a given problem and problem size.

This is done by the python script **process_alldata.py** wich produces **alldata_processed.csv.**

Finally with this processed dataset we have been able to produce the following visualization:

This visualization allows the user to select any programming languages he want using the drop down menu. It allows to then show the performances of the programming languages for a certain benchmark problem and problem size. It is possible to sort the algorithms according to their program size, cpu time and memory usage. Finally errors bars derived from the standard deviation computed before allows to compare languages in a rigorous way.

Peer Assessment

Hanbo Shim

IN - Exchange BA6

Programming languages database search

Implemented :
- programming language timelineline
- chart for number of book/paper/users
- programming language world map

Designed contents and format for the process book

Xinhe Gao

SV - Exchange BA6

Visualization idea generation

Implemented :
- intro page
- language origin story page
- whole website aesthetic design

Valentin Vuillon

EL - MA4

Implemented :
- programming languages performance visualization
- programming languages popularity over time visualization

Have helped:
- to find ideas for the project
- to find datasets
- doing the final website
- doing the process book