# 3D Visualization Project Report

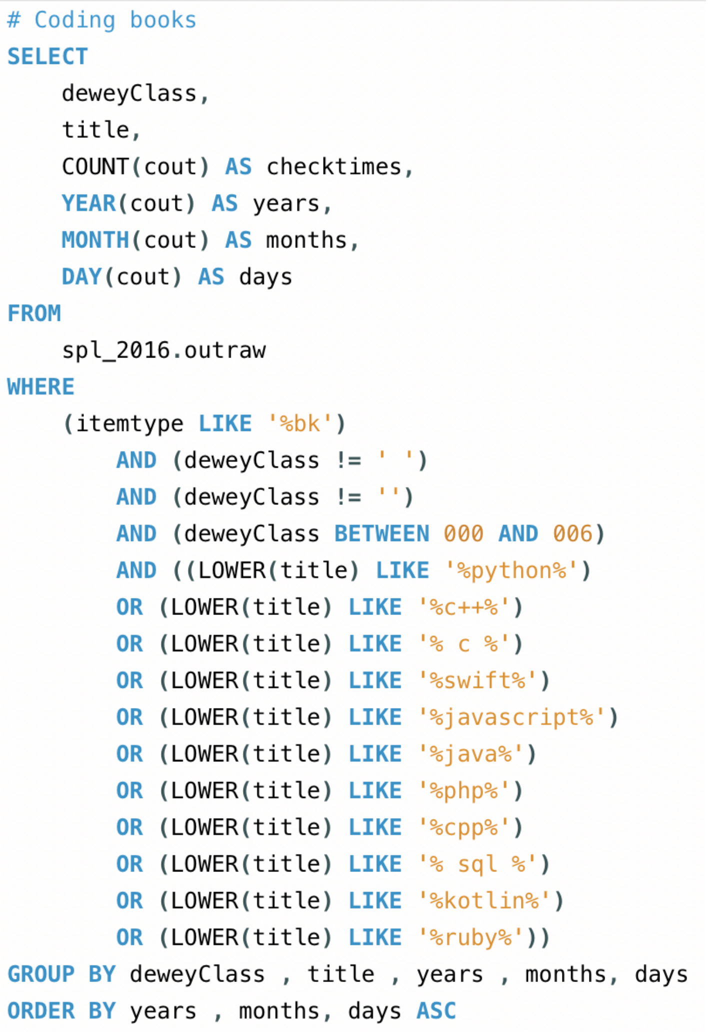
Boning Dong | Feb. 12, 2019

## Concept

In this assignment, I wanted to explore the change of the popularities of different programming languages over the past years. By using the Seattle Public Library checkout records, I fetched, classified and visualized these checkout records to analyze the popularity change of a specific language, and also tried to see if there is any relationship between the trend of different programming languages.

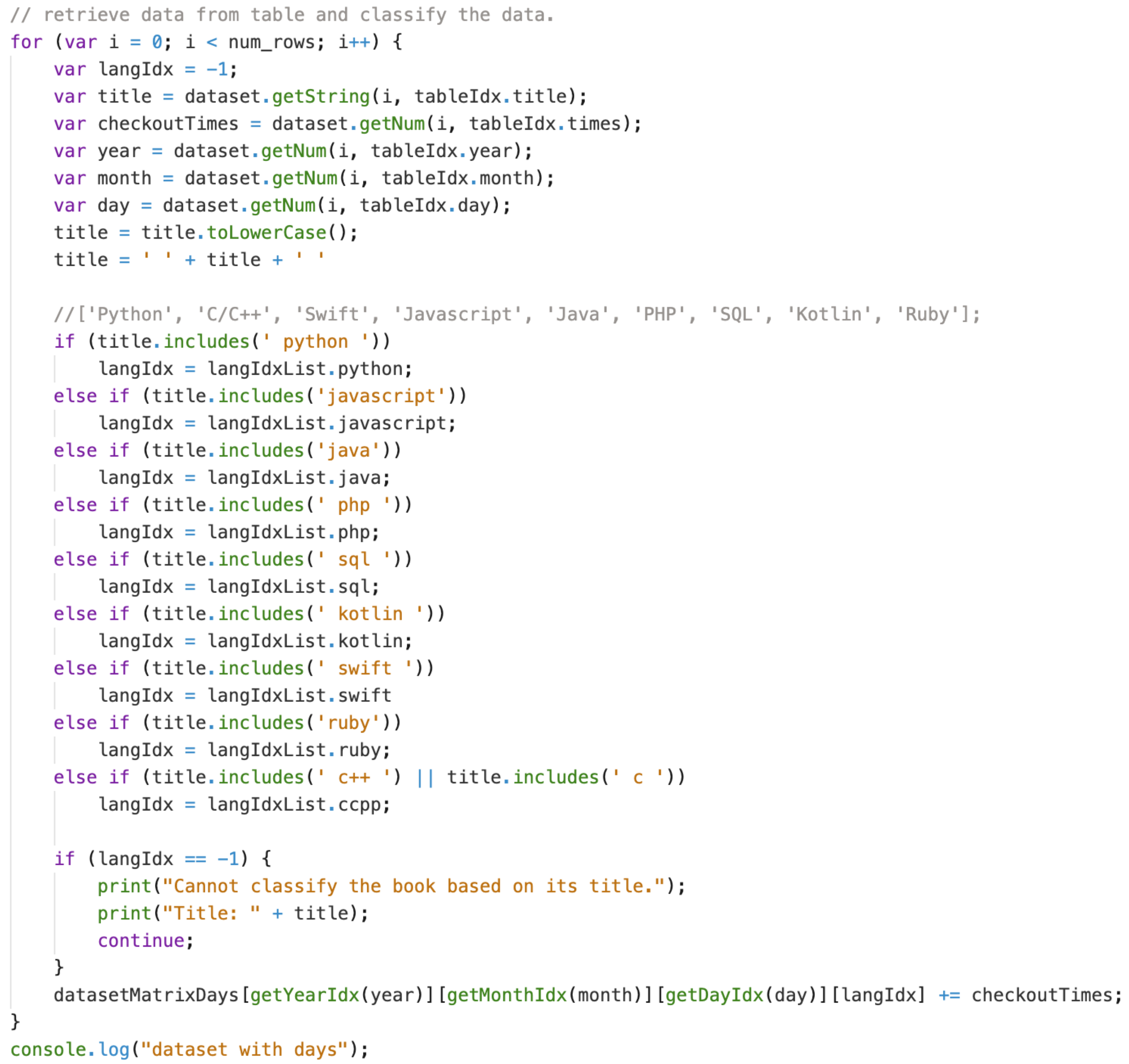
## Query

I tried to search the names of different programming languages to fetch the corresponding data from the MySQL server. Yet the result got in this way include non-programming materials, for example there are a lot of non-programming books contain the word “ruby”. So I also check if the Dewey class is between 000 and 006, which are the Dewey class numbers assigned for computer science books. Here is the query I used to fetch all of the data.



```
# Coding books
SELECT
    deweyClass,
    title,
    COUNT(cout) AS checktimes,
    YEAR(cout) AS years,
    MONTH(cout) AS months,
    DAY(cout) AS days
FROM
    spl_2016.outraw
WHERE
    (itemtype LIKE '%bk')
        AND (deweyClass != ' ')
        AND (deweyClass != '')
        AND (deweyClass BETWEEN 000 AND 006)
        AND ((LOWER(title) LIKE '%python%')
        OR (LOWER(title) LIKE '%c++%')
        OR (LOWER(title) LIKE '% c %')
        OR (LOWER(title) LIKE '%swift%')
        OR (LOWER(title) LIKE '%javascript%')
        OR (LOWER(title) LIKE '%java%')
        OR (LOWER(title) LIKE '%php%')
        OR (LOWER(title) LIKE '%cpp%')
        OR (LOWER(title) LIKE '% sql %')
        OR (LOWER(title) LIKE '%kotlin%')
        OR (LOWER(title) LIKE '%ruby%'))
GROUP BY deweyClass , title , years , months, days
ORDER BY years , months, days ASC
```

To show the popularity of a specific programming language, I also need to classify the results and sum the checkout times of the same language together. But doing that in SQL is too hard, so I determined to process this in Javascript. The way I did that is to create a data matrix that contains the daily checkout times for a specific language. Here is the details of the code which handles book classification.



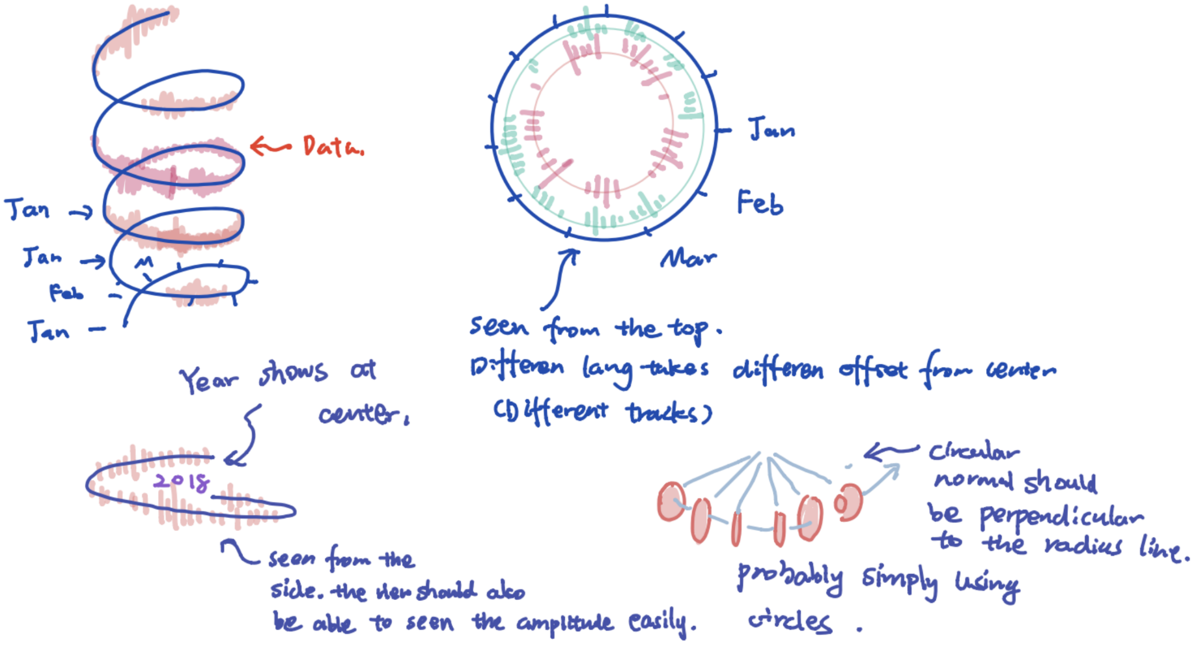
```
// retrieve data from table and classify the data.
for (var i = 0; i < num_rows; i++) {
    var langIdx = -1;
    var title = dataset.getString(i, tableIdx.title);
    var checkoutTimes = dataset.getNum(i, tableIdx.times);
    var year = dataset.getNum(i, tableIdx.year);
    var month = dataset.getNum(i, tableIdx.month);
    var day = dataset.getNum(i, tableIdx.day);
    title = title.toLowerCase();
    title = ' ' + title + ' '

    //['Python', 'C/C++', 'Swift', 'Javascript', 'Java', 'PHP', 'SQL', 'Kotlin', 'Ruby'];
    if (title.includes(' python '))
        langIdx = langIdxList.python;
    else if (title.includes('javascript'))
        langIdx = langIdxList.javascript;
    else if (title.includes('java'))
        langIdx = langIdxList.java;
    else if (title.includes(' php '))
        langIdx = langIdxList.php;
    else if (title.includes(' sql '))
        langIdx = langIdxList.sql;
    else if (title.includes(' kotlin '))
        langIdx = langIdxList.kotlin;
    else if (title.includes(' swift '))
        langIdx = langIdxList.swift
    else if (title.includes('ruby'))
        langIdx = langIdxList.ruby;
    else if (title.includes(' c++ ') || title.includes(' c '))
        langIdx = langIdxList.ccpp;

    if (langIdx == -1) {
        print("Cannot classify the book based on its title.");
        print("Title: " + title);
        continue;
    }
    datasetMatrixDays[getYearIdx(year)][getMonthIdx(month)][getDayIdx(day)][langIdx] += checkoutTimes;
}
console.log("dataset with days");
```

Basically what it does is to check if the title includes any keywords to indicate what programming language it is about, and then save to the corresponding cell determined by the date and the language itself.

## Design Concept


Data.
Jan
Jan
M
Feb
Jan
Jan
Feb
Mar
seen from the top.
Differen lang takes differen offset from center
(Different tracks)
Year shows at center.
2018
seen from the side. the vien should also be able to seen the amplitude easily.
Circular normal should be perpendicular to the radius line.
probably simply using circles .

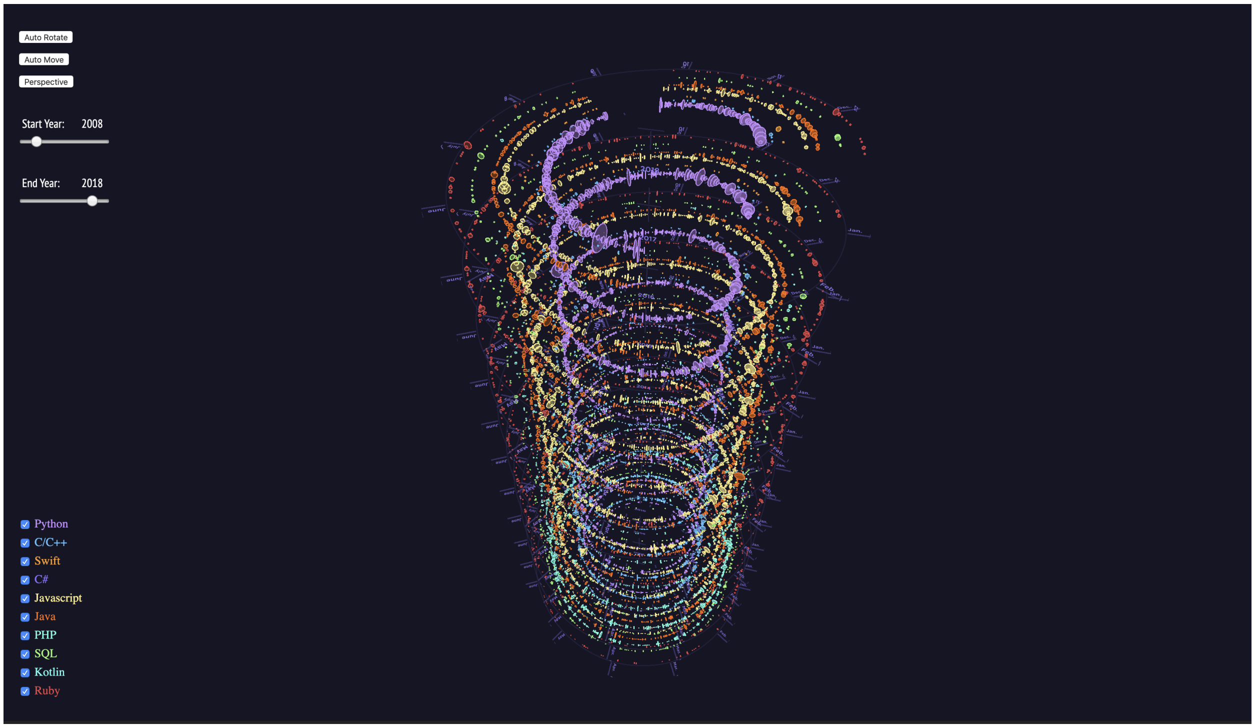
To fully utilize the extra dimensions in 3D space, I am trying to use the angle to represent the month information, and the height to represent the year. So overall the data will be presented following a helix pattern.
To differentiate the programming languages, I colored most of them based on their logo color, and each language will take one radius as their track so that they will not collide with each other.

The daily checkouts should be displayed using circles or torus so that the viewer can see the result from any angle.

Also, the user should be able to select the years and languages to be shown, it’s easy for the languages to be compared.

## Final Design & Screenshots

Here it shows the data from 2008 to 2018 for all the programming languages.

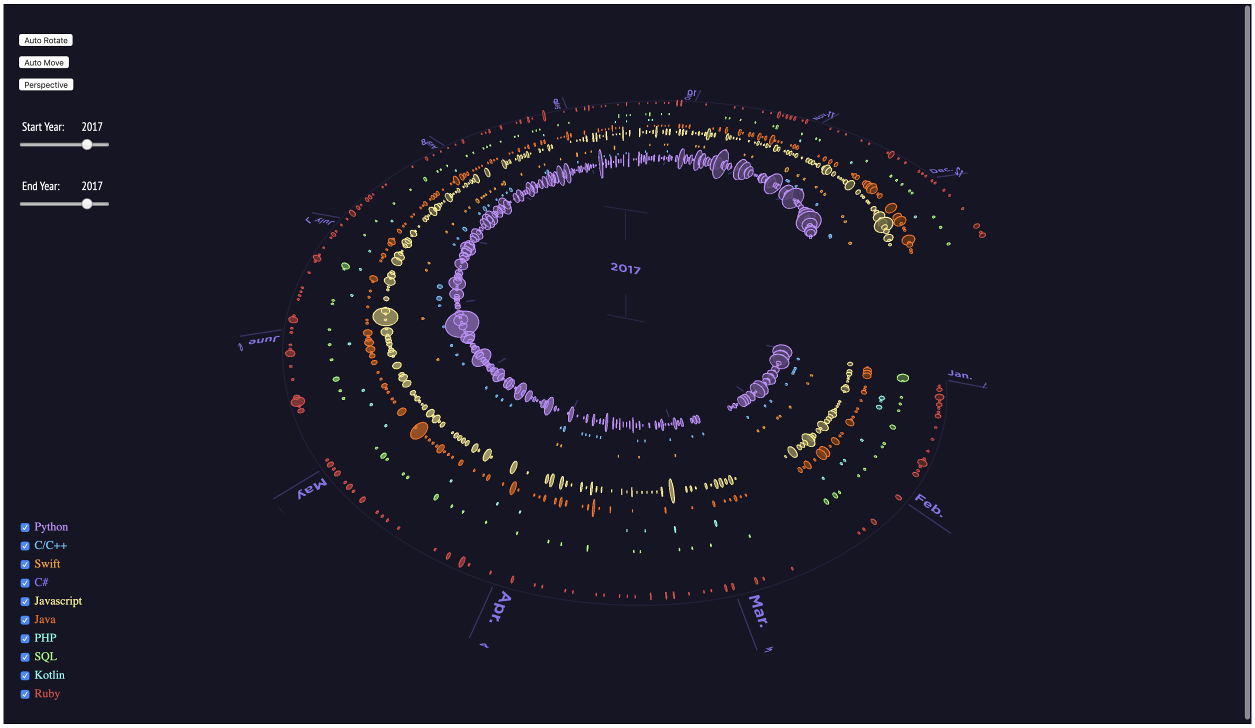


Start Year: 2008

End Year: 2018

☑ Python
☑ C/C++
☑ Swift
☑ C#
☑ Javascript
☑ Java
☑ PHP
☑ SQL
☑ Kotlin
☑ Ruby

Users can select only one year to show, which makes it easier to visualize the relative popularity of these languages over one year.

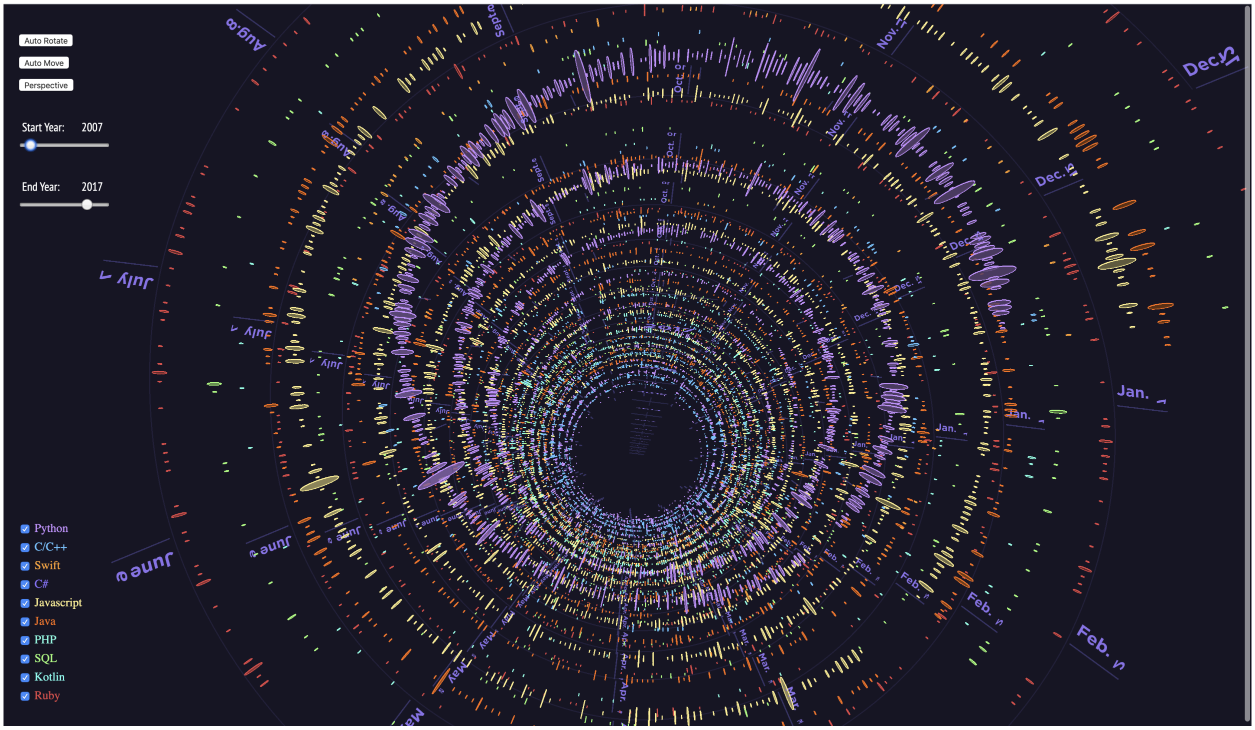


Start Year: 2017

End Year: 2017

☑ Python
☑ C/C++
☑ Swift
☑ C#
☑ Javascript
☑ Java
☑ PHP
☑ SQL
☑ Kotlin
☑ Ruby

Seeing from the top, the user can also see clearly the daily checkouts over the years, because circle diameter size implies the checkout times.

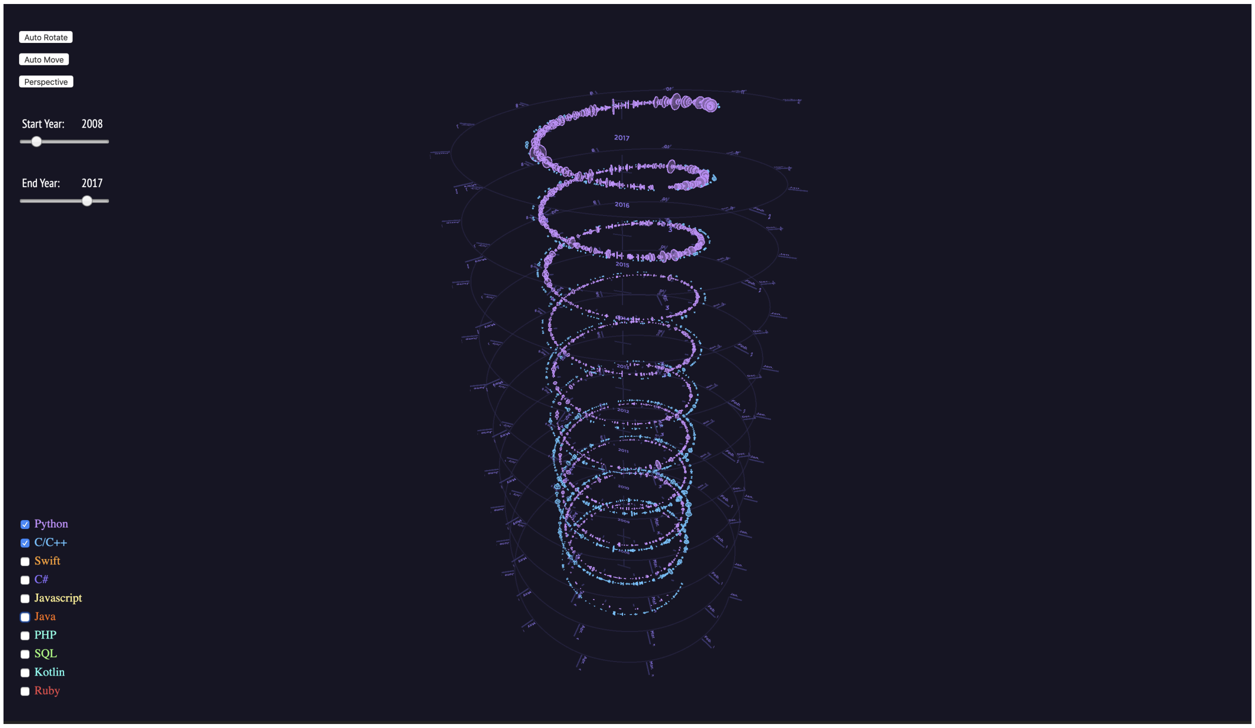


Start Year: 2007

End Year: 2017

☑ Python
☑ C/C++
☑ Swift
☑ C#
☑ Javascript
☑ Java
☑ PHP
☑ SQL
☑ Kotlin
☑ Ruby

Users can turn off other languages to only show the languages they want to compare. For the following picture, I only turned on ‘Python’ and ‘C/C++’, we can see the trend of these two languages over the years. Obviously, Python is getting more and more popular and vice versa for C/C++.



Start Year: 2008

End Year: 2017

☑ Python
☑ C/C++
☐ Swift
☐ C#
☐ Javascript
☐ Java
☐ PHP
☐ SQL
☐ Kotlin
☐ Ruby

## Future Improvement

One problem of p5 or processing is the data is drawn on the screen by CPU, even though after drawing on the screen the rendering work is actually handled by the GPU, CPU actually takes a lot of time to create the 3D objects and calculates their locations. So for my program, the fps is pretty low because so many pieces of data need to be displayed on the screen. I tried to optimize the code but because I cannot avoid the CPU-drawing stage inside the draw() function, there is no way I could boost the speed up. I am thinking for this kinda amount of data, it’s probably good using low-level WebGL or OpenGL to render the 3D objects after it first initialized by the CPU. But this solution probably puts too much burden on the programmer, especially those who are not experienced with computer graphics.

It’s also probably a good idea to add orthographic camera mode because, without the perspective effects, users can view the data within one year and compare the relative popularities easily. I tried to achieve this feature and I have a button called perspective, but when switching from perspective mode to orthographic mode, the camera cannot work properly. So I am thinking probably using two camera systems can solve this issue.