Making CodeCity evolve

David Moreno-Lumbreras
Universidad Rey Juan Carlos and Bitergia
Madrid, Spain
dmoreno@bitergia.com

Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
Madrid, Spain
jgb@gsyc.es

Valerio Cosentino
Bitergia
Madrid, Spain
valcos@bitergia.com

Abstract

CodeCity is a software analysis tool with the goal of visualizing software systems as interactive, navigable 3D cities. They rely on the city metaphor, that uses the layout of the city in order to visualize different metrics about software systems. There were other implementations of CodeCity among the years but there are no modern options that contemplate the representation of time evolution of the city. We propose a solution to this time evolution, developing a web version of CodeCity that works with any device that has a web browser. Then, we analyze different projects and show the time evolution of their cities. Finally, we conclude with some needed features and drawbacks that are going to be considered in future works.

Index terms— CodeCity, data visualization, virtual reality, web, 3D

1 Introduction

Codecity [1] is an approach of a 3D visualization which creates cities that look real, due to the combination of layouts, topologies, metric mappings applied at an appropriate level of granularity. It depicts object-oriented software systems as habitable [2] cities that one can intuitively explore. Codecity settles on the city metaphor because it offers a clear notion of locality, thus supporting orientation, and features a structural complexity that cannot be oversimplified. Codecity represents classes as buildings located in quarters representing the packages where the classes are defined. The metrics that Codecity uses are the number of methods (NOM) mapped on the building's height and the number of attributes (NOA) on their base size, and for packages, the nesting level mapped on the quarter's color saturation:

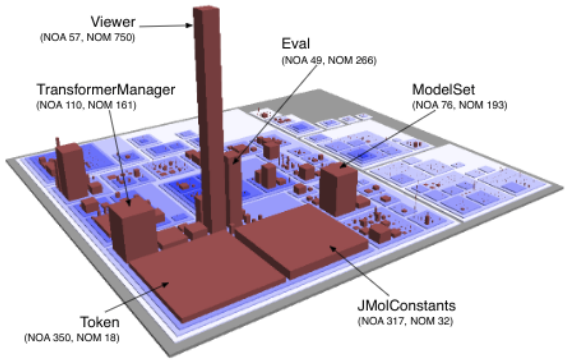

Figure 1: Example of the Jmol Java package city analyzed with Codecity

One of the known limitations of this approach is that CodeCity is developed using an old framework (SmartTalk) and it does not have support for several devices. We rely on web development and we are going to solve this limitation developing a version of CodeCity using web technologies and open source

In: D. Di Nucci, C. De Roover (eds.): Proceedings of the 18th Belgium-Netherlands Software Evolution Workshop, Brussels, Belgium, 28-11-2019, published at http://ceur-ws.org

software in order to make the tool more universal and every device that has a web browser could visualize these 3D cities. In the next sections, we detailed the development and the technologies used.

On the other hand, in terms of interaction, we argue that this visualization type can get more value and better comprehension adding it a Virtual Reality environment, allowing the user to tour and navigate into the city.

Another of its limitations is that Codecity doesn't follow a fixed layout for its blocks/quarters, it means that every time a code city is generated, the layout and the position of a package could change. This would be an important drawback in terms of time evolution analysis because the city would change every single time that it's generated. We are going to solve that, making the city evolve among the time, increasing and decreasing the area and height of the buildings as the metrics represented changes among the time, fixing the position of the buildings, fixing the same position of each building while the time snapshot changes.

Moreover, Codecity is strongly bonded to analyze the program structure. We argue that the city metaphor can be more than just analyze code, like analyze the contributions, users, and other kinds of metrics related to the community. This is one of the future works (section 4) that is going to be analyzed.

2 Related works

We are going to separate the related works in two categories, one related to the implementations of the city metaphor and the other related to the time evolution implementations about treemap algorithms. Regarding the city metaphor implementations, Juraj Vincur et all. [3] propose a Virtual Reality city for analyzing software, made it with non-web technologies, Steinbrückner and Lewerentz [4] propose stable city layouts for evolving software systems, using a different layout than a treemap. Getaviz [5] is another tool that uses the city metaphor in order to generate structural, behavioral, and evolutionary views of software systems for empirical evaluation. In terms of interaction with the city, one good example is CityVR [6] that uses the same metrics as the original Codecity but adds interactions with a VR headset using the controllers and the sight direction. Regarding the evolution of a treemap layout, Willy Scheibel et all. propose EvoCells [7], a treemap layout algorithm for evolving tree data, using rectangular areas. Another field of treemap evolution research is the Voronoi treemaps, these treemaps have not rectangular areas, making the treemap more flexible in order to evolve the areas, Avneesh Sud et all. [8] proposes a dynamic Voronoi treemap algorithm that explores this dynamic changes that can be used as an evolution of the treemap.

3 Our approach: BabiaXR and Web CodeCity with time evolution

3.1 BabiaXR Visualizations

There are different technologies based on WebGL that are focused on making 3D scenes in the browser. Specifically, we rely on A-Frame[1], a web framework for building 3D and VR scenes for the web, for the visualization render engine, developing, thus a new version of CodeCity that works in any device that has a web browser and the needed standards included (the most important, WebVR), A-Frame is developed on top Three.js[2].

Before the development of CodeCity, we got used to making some usual 3D visualization in this web 3D environment. Starting with BabiaXR[3], a GitLab/GitHub organization where the developing of the different visualizations are allocated. BabiaXR has the aim of aggregate different components, using A-Frame, that can create different types of chart in a modern browser, the web is a universal environment and any device that has a modern browser that supports WebVR can visualize the charts that the BabiaXR components produce, making it more universal and easy to use. The first step of BabiaXR was to make the most common visualizations, the pie chart, the 3D and 2D bar chart, and the bubbles chart. The development of these visualizations is separated in some A-Frame components, *geosimplebarchart* for 2D bar chart, *geo3dbarchart* for 3D bar chart, *geopiechart* for pie chart and *geobubbleschart* for bubbles chart. Moreover, there are components related to data management, these components can query, filter data and add behavior to the visualization components. There are two components that query data, *querier_json* for querying JSON files and *querier_github* that query the GitHub API. The component *filterdata* can filter the data retrieved from a querier component and the component *vismapper* that maps the data filtered by a filterdata component to geometry attributes of the different charts, it "prepares" the data and save it in the entity that it is defined.

In the repository, there is a user guide where all the information and the steps in order to create a 3D and VR dashboard are defined.

One example of these charts is the one shown in 2:

One of the advantages of A-Frame is that extends HTML, so using just two sentences of HTML with a JSON data can generate a 3D and VR visualization in

[1]https://aframe.io/
[2]https://threejs.org/
[3]https://github.com/babiaxr

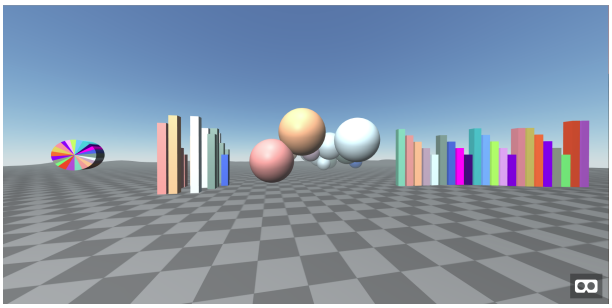

Figure 2: Usual visualizations of BabiaXR

the web environment. For example, the next HTML code and JSON data will generate the figure 3:

```
<a-scene background="color: #A8F3FF" id="AframeScene">
    ...
    <a-entity geo3dbarchart='legend: true; data:
        example.json'
    position="-10 0 0" rotation="0 0 0"></a-entity>
    ...
</a-scene>
```

```
[
{"key":"David","key2":"2019","size":9},
{"key":"Pete","key2":"2011","size":8},
...]
```

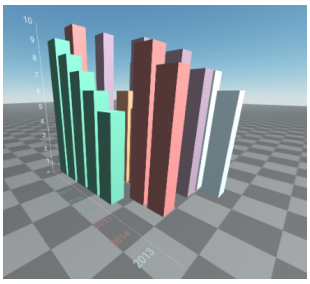

Figure 3: 3D bar chart generated with BabiaXR components

3.2 BabiaXR Codecity

The CodeCity version of BabiaXR is developed with the goal of exploring the known limitations of the original CodeCity, the approach is also inside the components pack, specifically, the components related to the CodeCity visualization are *codecity-block* for blocks and *codecity-quarter* for quarters. The first step in order to create this visualization is the layout selection, we rely on the treemap pivot algorithm [9] for the blocks and quarters positions, it shows a consistent layout and a desired aspect-ratio of the building area (making them the best "real building" possible appearance).

Moreover, there is an option for changing the appearance of the buildings for real building models loading gTLF models, making more realistic cities and improving the notion of locality. As A-Frame extends HTML and each building is defined as an HTML tag,

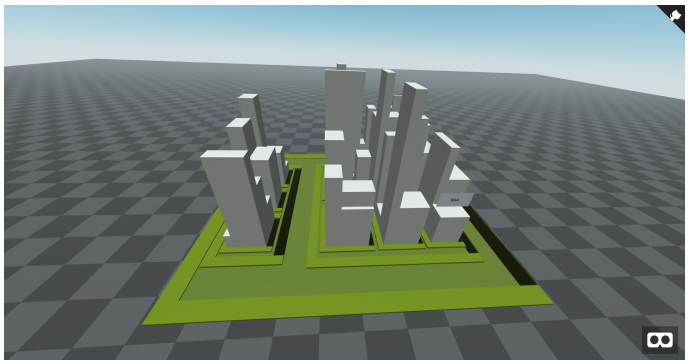

Figure 4: BabiaXR CodeCity examples

we defined a unique identifier for each building in order to change its area and height in the time evolution maintaining the position and solving one of the limitations that the original CodeCity has. In terms of interaction, A-Frame includes by default the possibility of entering the scene in Virtual Reality mode, therefore, the functionality of clicking a quarter with the mouse or a controller if it using a VR device has been implemented in order to show more information about the building/quarter clicked/hovered, showing the name of it as a title on top of it.

3.3 Creating a City

We separate the creation of a city corresponding of a software system in two parts, the first one is related of how we retrieve the data that we are going to visualize as a city, we analyze a repository using Graal [10], Graal leverages on the Git backend of Perceval [11] and enhances it to set up ad-hoc source code analysis. Thus, it fetches the commits from a Git repository and provides a mechanism to plug third-party tools/libraries focused on source code analysis. Moreover, Graal can retrieve similar metrics as the original CodeCity, we use Graal to obtain analyze projects using code complexity analysis (CoCom)[4], retrieving metrics like the number of lines of code or the number of functions that the file code has. Graal stores the data in an ElasticSearch database, once the data is there, we run a Python code included in the BabiaXR components repository in order to get the JSON with the data that will be injected in the CodeCity component of BabiaXR, summarizing:

1. Run Graal specifying the ElasticSearch database and the repositories that are going to be analyzed.

2. Run the code *generate_structure_codecityjs.py* with the time evolution flag and the snapshot argument filled and it will return a set of JSON data with all the information needed of the city.

[4]https://github.com/chaoss/grimoirelab-graal#backends

Once we have run the previous code, in the set of JSON files returned, there is one that has the starting information about the city evolution, called *main_data.json*, it has a structure that defines the data files needed of the time evolution, the sampling days that have been used and the time field where the evolution has been made:

```
{
   "date_field":"field",
   "sampling_days":"180",
   "init_data":"data_X_tree.json",
   "time_evolution":true,
   "data_files":[
      {
         "date":1573001804.136289,
         "file":"data_X.json"
      },
      ...
   ]
}
```

This file is the starting point and has to be defined as the main data when the component is written in HTML. The data files follow the next structure, starting for the main point, that has all the needed buildings and quarters tree in order to create the city:

```
[{
    "block":"name",
    "blocks":[
       {
          "block":"name_child",
          "items":[
             {
                "id":"file_path",
                "area": 1,
                "height":2
             },
             ...
          ]
       },
    ...
}]
```

Then, the next files just have a list with the new values of the buildings:

```
[
   {
       "id": "file_path",
       "name": "file_name",
       "height": 1,
       "value": 2
   },
...]
```

Then, just creating a simple HTML file, loading the dependencies needed (from A-Frame, BabiaXR components and the file *time_evol.js*), and defining the parameters of the *codecity-quarter* component, we can see the city of the project analyzed evolving among the time:

```
<a-scene id="scene">
    <a-entity codecity-quarter='items: main_data.json'
          position="0 0 0"></a-entity>
 ...
</a-scene>
```

4 First Applications

The following figures (5, 6, 7) represent the analysis of the project Angular[5]. Each building represents a file of the project and the quarters are defined as the tree folders that the file belongs to. As metrics, we represent the lines of code as the height of the buildings and the area represents the number of functions that the file has. Moreover, the figures represent a selected time snapshot, starting with the time snapshot of the 5th of November, 2019. Then going back 6 months (the 5th of May, 2019) and going back 18 months (the 5th of May, 2019). The demo available in the reproduction package[6]gives more information because there are more time samples and it is possible to see the evolution of the city in a completed and visual way.

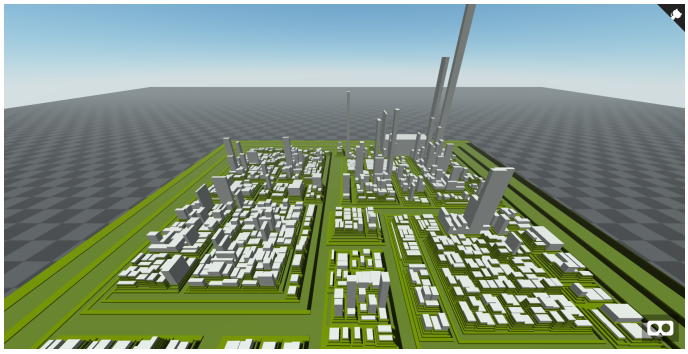

Figure 5: Angular CodeCity of the 5th of November, 2019

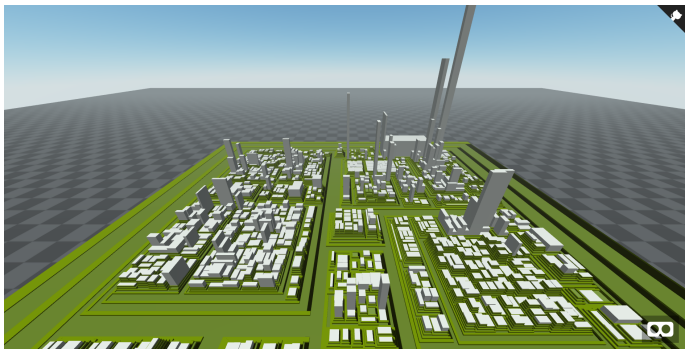

Figure 6: Angular CodeCity of the 5th of May, 2019

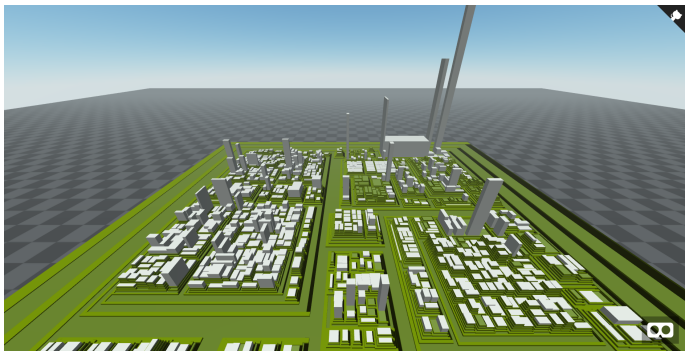

Figure 7: Angular CodeCity of the 5th of May, 2018

We can observe how the city evolves and how the

[5]https://github.com/angular/angular

buildings decrease their area and some of them disappear, as this is an evolution from present to past, some of the files maybe not in the past version, so the space that they filled is changed to empty terrain. For example, focusing on the center of the city, the central quarter is related to some packages that Angular includes in their code, if we observe the evolution, there are some buildings that disappear, those buildings are related to some packages that have been included in Angular in the latest version, their space is not filled in older versions.

There is a reproduction package[6] available in order to reproduce all the steps in a controlled environment.

Discussion

The proposed solutions to the known limitations of the original CodeCity has been developed successfully. Using the HTML identifiers of each building fixes the non-fixed location of the buildings, making the same location of each building as the city evolves, just changing the area and the height of each building in each time snapshot. In terms of the interaction, the possibility of adding Virtual Reality to the city and the development using web technologies solve the limitation of the interaction of the original CodeCity, making the BabiaXR version more universal and able to use in any device that has a modern web browser that supports WebVR. On the other hand, we are analyzing similar software metrics of the original CodeCity, it needs to add effort to this research and changes the type of metrics analyzed to new ones that make sense to represent using the city metaphor.

Conclusion and Future Work

This approach is a proof of concept of a tool that has to evolve. We were focused on replicate CodeCity in a modern and open source environment in order to make it more universal and easy to use, in this case, the web environment, and we were focused on the time evolution feature. The first versions of this tool are just the beginning and it needs more effort on the interaction part, we propose to add more functionality to the user interface, for example, adding two types of cities, one smaller as mock-up in order to see the entire city in a small space and take advantage to the location of the modern VR glasses. And the other one is to make a city as the same "size" as an actual city and tour the user into it, for example, driving a car or even flying. Other future work could be the feature of adding more information about the software in the city, with more user interaction than click/hover.

[6]https://gitlab.com/thesis-dlumbrer/repr-pckg-benevol-2019

On the other hand, all the metrics seen were metrics related to modules or packages. Thus, the next step is to look for the right way to represent different kind of data further than software systems structure, using different kinds of metrics. Specifically, we are going to deep in other software development metrics. For instance, the distribution of the commits/repositories, the relationships between the people, the time evolution of the community, etc. Therefore, we have to face the challenge of move this different kind of data to a city, thinking about what will represent the building, the districts, etc. Making the representation easy to understand and easy to answer questions that the common visualizations don't do.

Finally, another research branch is to approach other kinds of metaphor apart from the city, for example, islands and planets.

Acknowledgment

This work was supported by the Spanish Government (IND2018/TIC-9669).

References

[1] R. Wettel and M. Lanza. Visualizing software systems as cities. In *Proceedings of VISSOFT 2007 (4th IEEE International Workshop on Visualizing Software For Understanding and Analysis)* , pages 92–99, 2007

[2] R. Wettel and M. Lanza. Program comprehension through software habitability. In *Proceedings of 15th International Conference on Program Comprehension (ICPC 2007). IEEE Computer Society*, 2007.

[3] Vincur, J.; Navrat, P. ;Polasek, I. VR City: Software Analysis in Virtual Reality Environment. *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017.

[4] F. Steinbrückner, C. Lewerentz. Representing development history in software cities, *2010 5th international symposium on Software visualization (SOFTVIS)*, New York, USA, 2010, pp. 193-202.

[5] Baum, D.; Schilbach, J.; Kovacs, P.; Eisenecker, U.; Müller, R. Getaviz: Generating Structural, Behavioral, and Evolutionary Views of Software Systems for Empirical Evaluation. *2017 IEEE Working Conference on Software Visualization (VISSOFT)*, 2017.

[6] Merino L Ghafari M Anslow C Nierstrasz. CityVR: Gameful Software Visualization. *In-*

ternational Conference on Software Maintenance and Evolution (ICSME), 2017

[7] Willy Scheibel; Christopher Weyand; Jürgen Döllner. EvoCells – A Treemap Layout Algorithm for Evolving Tree Data. *9th International Conference on Information Visualization Theory and Applications (IVAPP)*, Madeira, Portugal, 2018

[8] Avneesh Sud; Danyel Fisher; Huai-Ping Lee. Fast Dynamic Voronoi Treemaps. *Seventh International Symposium on Voronoi Diagrams in Science and Engineering, (ISVD 2010)*, Quebec, Canada, June 28-30, 2010

[9] Ben Shneiderman M; Ordered Treemaps Layouts. *Information Visualization, 2001. (INFOVIS)*, 2001

[10] Valerio Cosentino, Santiago Dueñas, Ahmed Zerouali, Gregorio Robles, Jesus M Gonzalez-Barahona. Graal: The Quest for Source Code Knowledge *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2018

[11] Santiago Dueñas, Valerio Cosentino, Gregorio Robles, and Jesus M. Gonzalez-Barahona. Perceval: Software project data at your will. *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018