Indoor3D: A WebGL Based Open Source Framework for 3D Indoor Maps Visualization

Meng Gai, Guoping Wang
Peking University
Outline

• Motivation and Backgrounds
• Our Design
• Algorithm
  o Best view generation
  o Progressive visibility of elements
• Results
• Future Work
• Conclusion
Motivation

Indoor maps are widely used in shopping malls, airports, subways, etc.
Related work

As far as we know there has not been any open source indoor map visualization framework.

Commercial services:
- Google indoor map
- Amap (Gao De)
- weiditu
Related work

Reconstruction from 2D Floor Plans [Lewis and Sequin 1998] [Zhu et al. 2014]
Related work

- indoorGML: A candidate OGC standard for an open data model and XML schema for indoor spatial information.
Our Design

• Web based
• Simple data format for network transmission
• Easy to use and be customized by developers
Data Structure

- Building
- Floor
- Rooms
- public facility spots
Data Structure

Map Data

Building

Floors

Floor

Floor

Floor

Rooms

Room

Room

Room

Points

Point

Point

Point
The original JSON file is usually less than 1MB.
Architecture

MVC based architecture

• Model: The building’s data
• Renderer: WebGL or Canvas Renderer
• Controller: Handling user interactions
The original JSON file without trimming spaces is usually less than 1 MB. The file of a map with seven floors and more than 300 rooms is less than 300KB. Further compression techniques may be applied, such as those rendering maps in 2D. Some of the traditional techniques are only considerable when WebGL is not supported. In the cases, the building's information is not necessary. Moreover, some applications may take advantage of this file structure to supply the geometry shapes (only one in most cases) to present their outlines, while some cases, the building's information is not necessary. Furthermore, we are developing an utility algorithm can be studied. Furthermore, we are developing an utility building reconstruction from 2D floor plans, automatical converting the data to our format. The data structure is extensible, and designed independent of implementation details. It is a function area in the structure is an array. Each element of the array describes a floor. The floor contains an array of points. All of them have their own unique id to identify from each other. Many interfaces for the developers are using id as the parameter in Indoor3D. The id is naturally the ones provided by ThreeJS. There are three kinds of objects: Point, Rooms, and Floors. Their location is represented as a set of numbers to represent a polygon. Each pair of the numbers is a coordinate point. Usually, only one array is used, and the other ones are used for cases of multi-polygons and holes.

The data structure of the JSON file.

The architecture of our framework is designed according to the real structure of a building, so that it can also be used in other indoor maps applications. We design our file format according to the real structure of a building. The file sample file format is shown in Figure 1. The building's name, its address, the latitude and the longitude. The building holds the information of the building, itself such as the building's name, its address, the latitude and the longitude. The floor's properties are also extensible to hold more information. The Floor's properties are also extensible to hold more information. The Floor's properties are also extensible to hold more information. The Floor's properties are also extensible to hold more information. Notice the information are stored in 2D, and it will be extruded to 3D in the back end.
Usage

```html
<script>
    var indoorMap = Indoor3D();
    indoorMap.load("sampledata.json");
</script>
```
<div id="indoor3d"></div>
<script>
    var params = {
        mapDiv: "indoor3d"
    }
    var indoorMap = Indoor3D(params);

    indoorMap.load("sampledata.json", function(){
        indoorMap.setSelectable(true)
            .showRoomNames(false)
            .setSelectionListener(callback);

        var ul = Indoor3D.getUI();
        document.body.appendChild(ul);
    });
</script>
Best View Generation

Selecting the best views for 3D object has been well studied, especially in the CAD field. [Mortara and Spagnuolo 2009]. [Fu et al. 2008] [Hu et al. 2011]

The main idea: maximize the visibility of visual features
Best View Generation

- PCA (Principal components analysis)
- The center of the floor’s boundary $P_1$
- The average center of the rooms $P_2$
- Count the result of all floors
Best View Generation
Progressive Visibility of Elements

4.3 Progressive Visibility of Texts and Icons

For the texts and icons progressively, we employed an priority based strategies for those browsers that do not support WebGL. Based on the idea of progressive enhancement, we have many options for developers to build perfect 3D indoor maps.

4.4 User Interaction

Developers can set a selection listener to Indoor3D to customize the behaviour of the map when something is selected. There are 3 parameters passed to the callback listener: the id of the selected object, the id of the objects to be checked and the type of selection.

4.5 Downward Compatibility

When showing a single room, the room is high lighted to light yellow. When showing multiple rooms, the rooms are selected by users. When showing all the floors, the floors are treated as an integral whole object, while when showing a floor, the rooms are treated as independent objects.

We use the classical ray casting method to check which object is selected. This is useful when the developer wants to create a default selection behaviour of the map.

The algorithm is shown in Algorithm 1. The priority is designed based on the idea that the priority may take the popularity into account and defined as follow:

\[
\text{Priority} = \text{Popularity} + \text{Area} \times \text{Weight}
\]

\[\text{Weight} = \frac{1}{\text{Height} \times \text{Width}}\]

\[\text{Height} = \text{Height of the floor's OBB} \]

\[\text{Width} = \text{Average width of all the rooms in this floor} \]

where \(P_i\) and \(S_i\) are the priority, area and popularity index of the \(i\)th shop respectively.

For example, if the popularity index of every shop is available, the advanced priorities can be designed if there is more information.

For the icons of facility spots, the default priority descending order is set as Table 1. The priority is designed based on the idea that the entrances are the most important. This is coincident with the icons showing navigation information are more important, and the texts and icons progressively, we employed an priority based strategies for those browsers that do not support WebGL.

The overlap of these sprites will bring bad visual effects. To show all the texts and icons at the same time will be a disaster. The view is shown in Figure 7.

The tilt angle should be set to 75 degree in our system, so the margin value is set to 5 pixels to achieve good visual effects. Because the elements may switch between visible and invisible frequently when the user is pivoting. So we add a margin to compensate the visual effects.

Notice that if the element is set to visible as soon as it turns not visible from invisible, there will be an undesired flicking effect. Because the elements may switch between visible and invisible frequently when the user is pivoting. So we add a margin to compensate the visual effects.

The overlap of these sprites will bring bad visual effects. To show all the texts and icons at the same time will be a disaster. The view is shown in Figure 7.
Progressive Visibility of Elements

\[ P_i = w_1 \frac{S_i}{S_{max}} + w_2 \frac{Pop_i}{Pop_{max}} \]

- \( P_i \): Priority of facility spots
- \( w_1 \): Weight of area
- \( w_2 \): Weight of popularity
- \( S_i \): Area of facility spot \( i \)
- \( S_{max} \): Maximum area
- \( Pop_i \): Popularity of facility spot \( i \)
- \( Pop_{max} \): Maximum popularity
Progressive Visibility of Elements

Add a margin to avoid the flicking effect

Algorithm 1: decide the visibility of elements (texts and icons)

1. sort the element by descending priority order
2. for $i = 1$ to $\text{elements.length}$ do
   - $\text{visibility} \leftarrow \text{true}$
   - $\text{margin} \leftarrow 5$
   - for $j = 0; j < i; j++$ do
     - $\text{rect}_i \leftarrow \text{elements}(i).\text{boundingRect}$
     - $\text{rect}_j \leftarrow \text{elements}(j).\text{boundingRect}$
     - if $\text{elements}(j).\text{visible} \text{ and } \text{rect}_i.\text{collide}(\text{rect}_j)$ then
       - $\text{visibility} \leftarrow \text{false}$
       - break
     - end if
   - $\text{rect}_i.\text{shrink}(\text{margin})$
   - $\text{rect}_j.\text{shrink}(\text{margin})$
   - if $\text{elements}(i).\text{visible} \neq \text{true} \text{ and } \text{rect}_i.\text{collide}(\text{rect}_j)$ then
     - $\text{visibility} \leftarrow \text{false}$
     - break
   - end if
   - end for
3. end for
4. $\text{elements}(i).\text{visible} \leftarrow \text{visibility}$
5. end for
Results

• Three.js for WebGL Rendering
• HTML5 canvas for 2D backward compatibility
Results

Customized Themes

Figure 8: different customized interactions.
(a) Floors selection (b) Add a marker on selected shop. (c) Add a popup dialog is not efficient enough. We provide a 2D version indoor map rendered by HTML5 canvas. The translation and rotation act directly on the canvas DOM element rather than redraw the canvas. In such a way, the 2D map can run smoothly on some older mobile devices.

5 Result and Discussions

5.1 Results

We have tested our framework on several platforms, it can run on a real-time frame on every platform which supports WebGL. Because we redraw the scene only when the view is changed, it runs on a full frame rate when the view is static. When the user is interacting, the frame rate will drop down a little bit. The interacting frame rates of showing the map in Figure 1 on 3 of the platforms are shown in Table 2:

<table>
<thead>
<tr>
<th>Device</th>
<th>OS</th>
<th>CPU</th>
<th>GPU</th>
<th>Browser</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop PC</td>
<td>Windows 8</td>
<td>i5 2.8GHz</td>
<td>NVIDIA GeForce GTX 550 Ti</td>
<td>Chrome 41</td>
<td>50</td>
</tr>
<tr>
<td>Macbook Air</td>
<td>Mac OS X</td>
<td>i5 1.3 GHz</td>
<td>Intel HD Graphics 5000</td>
<td>Chrome 41</td>
<td>60</td>
</tr>
<tr>
<td>Macbook Air</td>
<td>Mac OS X</td>
<td>i5 1.3 GHz</td>
<td>Intel HD Graphics 5000</td>
<td>Safari 8</td>
<td>40</td>
</tr>
<tr>
<td>The new iPad</td>
<td>IOS 8</td>
<td>A5X A5X</td>
<td>Safari</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9: are some different rendering styles which shows the flex-

5.2 Future Work

This is the first step of our work, more future work is required for a complete indoor map framework. First, navigation function is usually necessary for a map application. When the users set the start point and end point in a map, the closest path will be generated. This function is especially useful when the indoor map is registered to the real building. Besides, more customization functions will bring more convenience to developers, such as adding heat map layers onto the map. Moreover, acquiring the map data is an tough but important task. Creating an whole indoor map data manually remains tedious for the users. Developing the algorithm which can convert existing vector plans or raster images to our indoor maps data automatically will help a lot.

6 Conclusion

In this paper we presented our Indoor3D framework. It takes advantage of WebGL to render 3D indoor scenes. We designed a JSON file to store the map structure, and solves several problems such as best view selection and progressive element visibility to achieve better user experience. The framework is designed flexible so that it allows the developers and designers to customize it conveniently. We believe many organizations will benefit from our Indoor3D framework. Airports, subway stations and shopping malls will provide better interactive services for guests and customers with our Indoor3D framework.

The source code and documentation are available from the website [Gai 2015].

Acknowledgements

This research was supported by Grant No. 61421062, 61170205, 61232014, 61472010 from National Natural Science Foundation of China. Also was supported by Grant No. 2012AA011503 from The National Key Technology Research and Development Program of China.

References

Results

Customized Interactions
Future Work

- Location registration & navigation functions
- Tools for acquiring the map data
Conclusion

• In this paper we presented our Indoor3D framework.
• It takes advantage of WebGL to render 3D indoor scenes.
• We designed a JSON file to store the map structure.
• We solve several problems such as best view selection and progressive element visibility to achieve better user experience.
• The framework is designed flexible so that it allows the developers to customize it conveniently.
Thank you!

https://github.com/wolfwind521/indoor3D

Email: gaimeng@pku.edu.cn

Q&A