RV College of Engineering

Go, change the world

Computer Graphics and Virtual Reality (18CS72)

Department of CSE

Virtual Tour

Sanjana Reddy - 1RV18CS145
Varsha Jenni - 1RV18CS183

Problem Statement

- With the COVID-19 pandemic limiting where people can go, hotels, destinations, and travelers alike have felt the effects.

- Tourism has all but stopped, local vacations are often out of the question, and with more people working from home, business travel has also decreased.

Solution

- Virtual Reality is a boon during these tough times!
- Virtual tours, also called immersive tours, are a set of photos linked together so that the viewer gets the full “walkthrough” experience.
- A video tour is a full motion video of a location. Unlike the virtual tour's static wrap-around feel, a video tour is a linear walk-through of a location.
- The main aim to create a simple tourism VR application using **React-360** with dynamic transitions, tooltips, and easy location managing.

Objectives

- The objective of the project is to provide an interactive, immersive and imaginative experience for Chernihiv city in Ukraine.
- It provides a way for a user to explore the city and learn more about it from a remote location.
- They can learn about the distances of the tourist spots from any entered location.

Our app also consists of:

- On click smooth transitions between locations
- On hover tooltips for better city attractions introduction
- Comfortable structure to add new locations, attractions, and transitions

Methodology

Project Structure

- folder *"components"* includes components that used repeatedly
- folder "*modules*" includes Custom Native Modules
- folder *"services"* includes services that work with entities
- folder *"static_assets"* includes different icons, 360 photos, locations, links and photos of attractions
- file *"index.js"* is an entrypoint
- file *"client.js"* is the code that connects browser to the "Runtime"
- file *"index.html"* we are not going to edit it, it only provides a point to mount js code

Methodology

Static assets

- Icons for tooltip and transition
- 360 images
- Places of attraction
- Links for the menu
- Keyboard buttons, audio files, emojis
- Links
- Locations

Data structure

```
export default (locations = {
  ...
   PopudrenkoPark: {
 name: 'PopudrenkoPark',
 img: 'popudrenko_park.jpg',
 transitions: [
  {
   width: 50,
   height: 50,
   yaw: 11.2,
   pitch: 0,
   goesTo: 'CityCenter',
  },
 ],
 tooltips: [
    {
     width: 35,
     height: 35,
     yaw: 14.32,
     pitch: 0.1,
     text: 'Chernihiv
National Bank',
     img:
'national_bank.jpg',
    },
  ],
},
...
});
```

Methodology

State management

To share state we create **WrapperComponent** and wrap all components, that need to know current location, wrap into this component.

In **WrapperComponent** we have the *wrap* function in which we wrap all components. In *changeLocation* function we change the state of all wrapped components and change the background image. This function is called from our **TransitionComponent**.

Methodology

Shared Components

Here we create **TooltipComponent** and **TransitionComponent** that will be reusable many times.

TooltipComponent listens to on Enter and on Exit events and triggers appropriate function. So we will show detailed information (image of the attraction and name) on Enter and show tooltip icon on Exit event. *"hitSlop"* param defines how far a touch event can start away from the view. We need it to avoid blinking caused by fast component resizing. It has two functions onMouseOn and onMouseOut which call resizeTooltip function of tooltipModule.

TransitionComponent has VR-Button and on click it calls *changeLocation* function which changes the location and sets tooltips for the new location.

Entry points

Each **React-360** app has a default *index.js, index.html* and *client.js* files.

In *index.js* we have the **MainComponent** with empty view and we register components using AppRegistry and our wrapper. By registering multiple components with the AppRegistry, you can attach them to a different surface.

In client.js we have default *init* function, where we set set custom native modules to the ReactInstance, render **MainComponent** and set default background as a city center, because our tourism app will start from this location.

index.html is used to load the webpage.

Requirement Specification

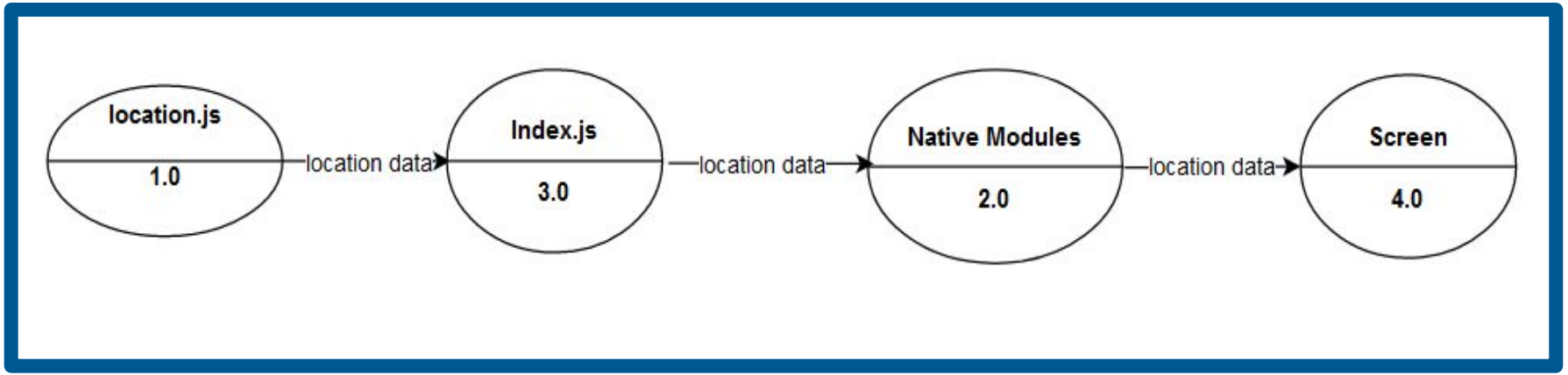
Hardware specifications

- OS: Windows 10 or Mac OS
- Processor: Intel Core i3 or i5
- Memory: 4GB
- Hard drive: 80 GB hard drive space
- VR headset (optional)

2.2 Software specifications

- Node.js
- Web Browser

Data Flow Diagram



Conclusion

React 360 is an interesting way to create 3D web applications which give a VR experience to the user.

This is an open-source framework and therefore is cost-effective to build VR applications.

The library allows one to place and modify objects in a virtual environment. React native uses native controls.

This is better suited for games. Animated library can be used to make animations.

For games, it is easy to develop VR apps.