PU 3D MAP

Prof.(Dr.) Kruti Sutaria,
Assistant Professor
CSE Department Parul University Vadodara, India

MANDELA MADHULIKHA GEETHUMA RAJ
CSE AI Parul University
Vadodara, India

***Abstract*—"PU 3D MAP" In the digital age, mapping services have become an integral part of our daily lives. However, when it comes to 3D mapping, particularly in regions like India, the options are limited. This research project delves into the challenge of developing custom 3D maps for areas that are underserved by existing mapping platforms. Unlike conventional 3D maps that primarily focus on popular landmarks, our goal was to create immersive and customizable maps that blend the richness of traditional map services with the versatility of 3D vector mapping. Our journey began by recognizing the absence of public 3D maps covering many regions, with most offerings being 2D maps with labels for prominent locations. In India, the absence of 3D mapping services like Google Maps or Google Earth prompted us to explore the potential of crafting our solution. We envisioned a map that not only provided a three-dimensional perspective but also incorporated features familiar to users of existing map services, including Places, Terrain View, Current Location, and Local Places. The project's development phase led us to a critical crossroads in choosing the right methodology. We discovered that most available methods were either proprietary or closed source, limiting our customization options, or posing significant implementation challenges. As a result, we embarked on an experimentation journey with Unity and Godot, attempting to create vector maps from scratch. Ultimately, we found our solution in Google Vector Maps, which offered a balance of ease of implementation and the flexibility we desired. The project's evolution continued as we transitioned our application from a desktop cross-platform solution to a web-based platform. This transition opened up new possibilities for accessibility and reach. Through this project, we demonstrate the feasibility of creating 3D maps and showcase that, with substantial effort and a deep understanding of web technologies and graphic systems, it is possible to construct a 3D mapping experience by synergizing multiple languages and frameworks. Key components of our solution include the use of Svelte for building web and user interface elements, Google Maps API for obtaining vector data, and with necessary libraries for translating coordinates into a three-dimensional space. Furthermore, we leveraged Three.js for rendering vector maps and seamlessly integrating custom 3D models into the map environment**

***Index Terms*—Custom 3D mapping solution leveraging Google Vector Maps, Svelte, Google Maps API, and Three.js for immersive and customizable mapping experiences in underserved regions like India, transitioning from desktop to web-based platforms.**

I. Introduction

In today's age, maps have become an essential tool for navigation and exploring the world around us. While many popular map services exist, they often provide limited functionality when it comes to three-dimensional representation. Specifically in India, the availability of 3D mapping services such as Google Maps or Google Earth is scarce. Recognizing this gap, our project aimed to explore the potential of creating custom 3D maps with vector-based technology. We sought to incorporate features commonly found in existing map services, such as Places, Terrain View, Current Location, and Local Places. Our goal was to develop a map that not only provided accurate geographical information but also allowed for the placement of custom 3D blocks that could scale dynamically based on the map's view, orientation, and heading. During the research and development phase, we discovered that most methodologies for creating 3D maps were either proprietary and closed source or presented significant implementation challenges. To overcome these hurdles, we experimented with various tools and frameworks, including Unity and Godot, before ultimately settling on Google Vector Maps. We chose Google Vector Maps due to its ease of implementation and the complexity it offered in embedding custom 3D objects onto the map. As we progressed, we decided to transition our application from a desktop cross-platform solution to a web-based implementation. This allowed us to leverage the power and flexibility of web technologies and graphic systems. By utilizing frameworks like Svelte for creating web and UI elements, the Google Maps API for obtaining vector data, and Three.js for rendering vector maps and custom models, we were able to create a functional and interactive 3D map. The purpose of this project is to demonstrate that with significant effort and a comprehensive understanding of web technologies and graphic systems, it is possible to create a custom 3D map.By combining multiple languages and frameworks, we were able to overcome the limitations of existing services and provide users with a unique and immersive mapping experience.

II. Project Methodology

Data Gathering: Gather places and their position coordinates This stage includes gathering different places inside the university like Blocks, stalls and gardens including their coordinates (i.e. longitude and latitude of places). We started using Google Maps as a primary data source to collect positional coordinates from all the Markers. Collecting coordinates is rather an easy job in Google Maps, just with one right click, we get the longitude and latitude of the location.

A. Data Processing and Formatting

To be able to process and fetch data efficiently in code the data has to be formatted to a specific schema. For that, we chose JSON to store data and a special structure to be able to efficiently load and use data.Every JSON data is an Array with every JSON object contacting the details about the includes name, latitude and longitude. This strategy is useful when fetching data back to use With Google Maps API

B. Creating 3D models

For creating 3D models we choose Blender for its handy features to take images as references to build 3D blocks with less hassle. After the construction of models, they will be explored in GLB format (binary file format representation of 3D models saved in the GL Transmission Format (glTF)) which can be later exported to the project for rendering.

C. Implementing Backend of project

Google Maps API provides vector maps with WebGL functionally which enables developers to be able to edit the map for their own needs. we took use of this functionality to place custom 3D models in Google Maps (Lots of the features are undocumented which increased our development time, but it was worth it). Took advantage of map control variables and methods to control orientation,scale, zoom and heading of map.

D. ThreeJS and webGL wrapper

ThreeJs is a library that binds everything and renders a 3D map to the screen. Three JS already simplifies handling 3D content on the web and we made wrappers around those to further simplify the placing of 3D objects by accessing the lifecycle of Google vector map. One of the primary tasks of the code is to create a WebGL overlay view on Google Maps. This is achieved through the use of google. maps.WebGLOverlayView and Google. maps.ThreeJSOverlayView. These objects provide a bridge between the Google Maps API and the Three.js library, enabling seamless integration. The code proceeds to load a 3D model specified in the MODELS object using the GLTFLoader. This model is then scaled and positioned on the map at a defined latitude, longitude, and altitude. The latitude and longitude coordinates correspond to a specific location on the map, while the altitude parameter determines the vertical position of the model. The model's rotation is also adjusted as necessary. The WebGL renderer is initialized and associated with the Google Maps canvas. The onDraw function updates the camera's projection matrix to ensure that the 3D scene aligns correctly with the map. Subsequently, the scene is rendered using the renderer, allowing for the model to be displayed within the Google Map. In conclusion, the provided code successfully demonstrates the integration of a WebGL overlay onto a Google Map. This integration facilitates the rendering of 3D models at specific geographic locations, enhancing the visual experience of maps by adding interactive 3D content. By utilizing the capabilities of libraries such as Three.js and Google Maps API, this code opens up possibilities for creative and informative map-based applications. Further development and customization of this code can lead to the creation of immersive and engaging map experiences.

E. Low cost methods

Around the project, we tried to make the project at a low cost. for now, the only part that costs money for usage is Google Maps API.

F. Front end: Web Page and User Interface

The svelte framework is used to create web user interface. The svelte framework provides ease of implementation features with less development hassle and an easy learning curve. despite other frameworks, we choose Svelte just as a biased choice cause we could write less amount, less confusing code with greater functionality.

G. Tailwind CSS

In building web applications/web apps CSS plays a key role in designing a website. It introduces so much styling to make the user interface look beautiful but CSS itself becomes a hassle when dealing with a project short time.For that, we used TailWind CSS. A utility-first CSS framework packed with classes like flex, pt-4, text-centre and rotate-90 that can be composed to build any design, directly in your markup. means style definitions are down to a minimum code with all the features

III. CONCLUSION

During the development process, we chose Unity as the primary technology but the development time taken for every feature was too high to complete in time before the due date. Later in our cycle, we replaced unity with a new tech stack by choosing JS (Javascript) as a primary language with tools and frameworks like NODEJS (a back-end JavaScript runtime environment), SVELTE (A web framework, VSCODE (IDE), ThreeJS (library to create and display animated 3D computer graphics in web) and Google Maps API Using this stack reduces our expected time to implement a working project increased to 70 percent faster. Stitched Google Maps API and the tech stack to create the project svelte, Google Maps, Three JS, WEBGL, Blender + tinkercad as new requirements we implemented a working website with 3D map and camera.

IV. FUTURE WORK

As the project is aimed to be a development project that shows cases of the Google Maps API and Three JS usage to add 3D Models to the Maps it is visible that for the project to at least become a consumer-grade project it has to go under a lot more development with a lot more time and effort 1. Add more building models. 2. Add more details to the models.