University of North Carolina Asheville
Journal of Undergraduate Research
Asheville, North Carolina
December 2022 or May 2023

LandMarked3D: An Open Source 3D Terrain Modeling Application

Patt Martin
Computer Science
The University of North Carolina Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisor: Dr. Kevin Sanft

Abstract

While the quantity of publicly accessible geographic data has increased, many software packages that interact with that data are expensive and have steep learning curves. Landmarked3D is an open source software application that provides a simpler workflow for data retrieval and generation of geographical maps and three-dimensional (3D) terrain models. This application collects satellite imagery and elevation data from the Google Maps API using user-provided coordinates. The program allows a user to define polygonal areas within a location to denote different layers. In addition, a user can import and tweak foliage objects based on a set of parametric equations. The application demonstrates its utility by modeling a golf hole where each defined layer corresponds to different ground conditions (e.g.: tall grass vs sand traps). As the application improves, this software can be extended to aggregate terrain data from different sources to model any type of terrain for a range of application domains.

1. Introduction

In recent years, the availability and quality of geographic data have grown rapidly. This growth provides new opportunities for understanding and analyzing our world. However, many software packages that interact with this data are expensive and require steep learning curves. These obstacles limit accessibility for the researchers and planners who lack the time to commit to learning these tools. LandMarked3D seeks to position itself as an open-source application with a simplified process for data aggregation and terrain modeling.

The primary goal for the development of LandMarked3D was to improve upon a prototype. This proof of concept would retrieve satellite imagery and define regions with a graphical user interface (GUI) and the command line. By rebuilding this prototype, LandMarked3D is able to eliminate the need for the command line while providing an

organized development environment to rapidly create additional features that can be utilized for simulation projects.

2. The Application in Practice

LandMarked3D's graphical user interface (GUI) is divided into four key regions. 1) The viewport renders user-defined regions, sample points from datasets, and satellite imagery. The viewport also supports user input through the mouse and keyboard for an interactive experience. 2) Right of the viewport, the inspector is a detailed editor for fine-tuning parameters or changing which type of interactive tool is active for interacting with the viewport. 3) To the right of the inspector, the inspector toolbar provides a means of switching the active inspector. 4) Under the previous three key regions, the status bar displays application statistics (i.e., number of loaded assets, cursor position in different coordinate systems), and the status of the backend if any tasks are running (i.e., downloading data, generating images).

2.1 Creating a New Project

For the purposes of this walkthrough, the user will select the golf course pictured below in Figure 1 as their location. The user's goal is to retrieve satellite imagery and elevation data of the location, define relevant features of the location, measure the amount of variance from walking through the location to obtain elevation data from the Global Positioning System (GPS), and export all collected and generated data to run a simulation of a golf ball in another computer program.

Figure 1. Satellite Imagery: A course of a golf course that will be used for demonstrational purposes.
Source: Google Maps Static API

Upon launch of the application, the user will be prompted to open a previous project or create a new one. By choosing to create a new project, the user will need to select the services to download data from. First-time users may be prompted to provide an application programming interface (API) key for a guaranteed service that provides satellite imagery. Next, the user will be prompted to provide two latitude and longitude points, corresponding with the north-west and south-east corners of a rectangle. While the backend downloads data from the selected services, the user will receive feedback to be assured that the process is in progress.

2.2 User Defined Objects

After successfully pulling satellite imagery, the user is presented with the main editor workspace. Users are able to define multiple types of objects for different purposes. Through the viewport and inspector, the user is able to fine tune these objects for their needs, as shown below in Figure 2.

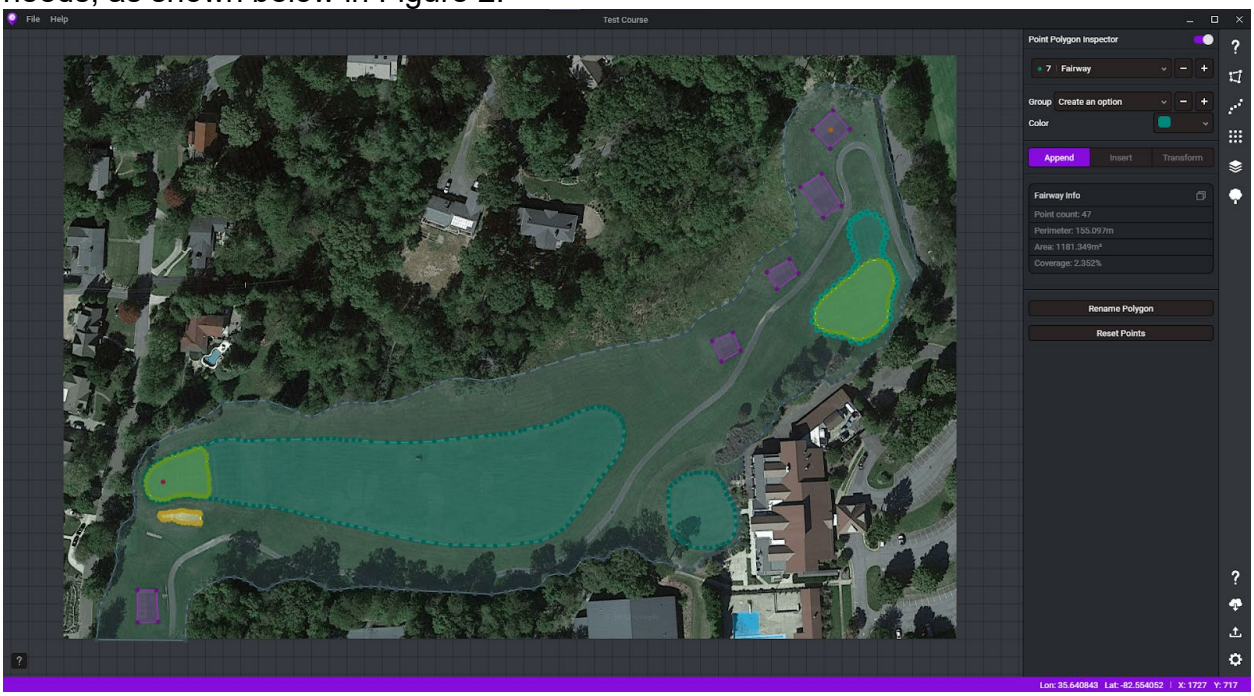

Figure 2. Screenshot of LandMarked3D with the Point Polygon Inspector opened.

2.2.1 Sample Points

Sample Points form the most basic data type that LandMarked3D utilizes. The necessary properties of a Sample Point include a pixel x coordinate, a pixel y coordinate, and an id to uniquely identify itself. Extra properties (i.e., elevation) can be added to the Sample Point at any time through the details property. In addition to the user placing a single point as a point of interest, Sample Points are used as the foundation for the other user-defined objects.

For the golf example, the user will mark the starting position for the course and the general ending position. Once these are exported, they can be referenced to quickly find these key locations.

2.2.2 Point Polygons

Point Paths are an array of connected Sample Points that automatically close themselves. Point Polygons are used to designate specific regions of interest. Point Polygons can also be organized into Polygon Groups that describe disconnected regions with similar properties and aid in project organization.

For the golf example, the user will mark regions with different ground conditions. This includes the green, an area of shorter-cut grass with less friction for a golf ball, and the sand trap, a hazardous area of sand that traps the golf ball.

2.2.3 Point Paths

Point Paths are an array of connected Sample Points that, unlike Point Polygons, do not close themselves. While Point Paths can be defined in the same manner as Point Polygons, Point Paths can be created by importing external data in a CSV format. LandMarked3D will request that the user select latitude and longitude headers. Afterwards, the user may select additional headers. For each Sample Point generated for each non-header row of the CSV file, the data from each additional header will be appended to the Sample Point’s detail property.

For the golf example, the user will create a Point Path and import the GPS data retrieved from a separate application on their phone. After the user selects the latitude and longitude headers, the user will select elevation as an additional CSV header.

2.2.4 Point Fields

Point Fields are an array of disconnected Sample Points that are the result of API requests. While a user cannot directly modify a single Sample Point, they can append an array of Sample Points by completing more API requests and selecting a predefined Point Polygon.

For the golf example, the user will need a higher resolution of elevation data at the green area. This detail will be necessary for the user's simulation to determine a golf ball’s trajectory as it rolls along the ground. The user will select the Point Polygon corresponding to the green area and finalize the details for the API request.

2.3 Visualization Methods

LandMarked3D makes use of D3.js, a JavaScript library, to interpolate data from the project’s Sample Points. The interpolated data can be used to create configurable visualizations. Figure 3A shows a generated contour map with 25 intervals from an elevation from 2277.1 feet (655.05 meters) to about 2149.11 feet (694.09 meters) above sea level. The contour map is also rendered as a transparent PNG to let it be easily

overlaid onto other maps. Figure 3B shows the same elevation data but as a continuous black and white topographic map.

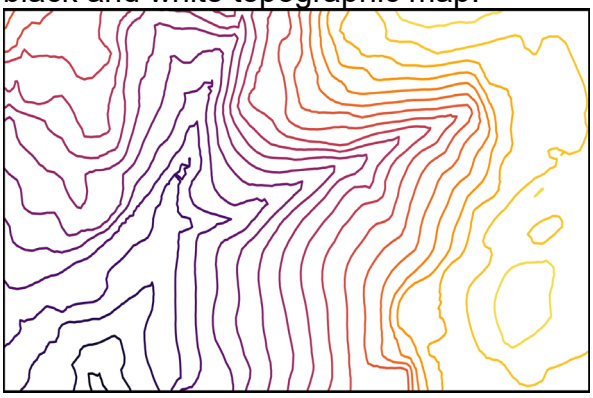

Figure 3A: A contour map of the golf course. Lighter colors correspond to higher elevations while darker colors correspond to lower elevations.
Source: Generated with LandMarked3D with elevation data from Google Maps API.

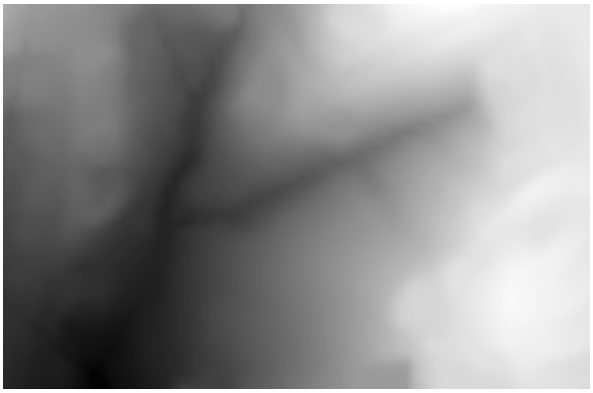

Figure 3B: An elevation map of the golf course. Lighter colors correspond to higher elevations while darker colors correspond to lower elevations.
Source: Generated with LandMarked3D with data from Google Maps API.

2.4 Exporting Data

LandMarked3D provides several options for exporting the data retrieved and generated in the application. While each project file is stored in human-readable JSON, users may also export data as CSV files. LandMarked3D also offers the ability to convert latitude and longitude coordinates into metric or imperial units, using a user-defined location as the origin.

In addition to exporting out as data files, LandMarked3D can also export Point Polygons as black-and-white PNG images. These images can be useful for creating visualizations in other applications or to sample a pixel's position and value to get a boolean output.

Another key feature of LandMarked3D is its ability to export a 3D model of the terrain. The mesh's x and y vertex positions correspond to an even distribution of latitude and longitude points from the location. The z component can relate to any chosen property, such as elevation.

3. Previous Applications

LandMarked3D builds upon two previous applications. The first was a proof of concept that utilized Python to request satellite imagery and elevation data from Google Maps services and used the Open Computer Vision (OpenCV) Python package to define polygonal regions. The output of the application was restrictive, as it could only output the pulled imagery and the polygonal regions as Tag Image File Format (.tiff) image files. If the user needed to make corrections after the application was closed, the user would have to re-request the imagery and redefine all of the polygonal regions.

The next iteration would place great focus on delivering a lenient workflow with the ability to load, edit, and save multiple locations as a traditional application with the Tkinter.py package. After this was achieved, additional features would be developed to generate 2D visualizations with MatPlotLib, 3D meshes of the terrain's elevation with a custom-built tool, and to export all retrieved and generated data into widely available formats (i.e., portable network graphics (.png) for imagery, comma-separated values (.csv) for data, and objects (.obj) for 3D meshes). Limitations with re-rendering UI, specifically with overlapping imagery and polygonal regions, would lead to high CPU and memory usage. Additionally, 3D meshes of locations would require a separate application for viewing.

After the two previous approaches, it was decided to rebuild the second prototype as a web-based application with a new focus on performant user interaction and dynamic visualizations. With the performance ceiling heightened, additional development time can be allocated towards developing computationally intensive features, such as image generation.

3. Software Architecture

3.1 API Extendability

In order for LandMarked3D to aggregate datasets from a variety of sources, building a strict but self explanatory entrypoint for new API pipelines is essential. By requiring a user-developer to wrap all of the new API's specific requirements into a JavaScript class with predefined and abstract functions. This class is reinforced with a TypeScript interface for variables and functions. Some of these properties are sent to the GUI (i.e., name, description), while some are utilized by the backend (i.e., headers and key names that the API provides). This also includes properties that are functions, such as the url generator function, which takes in an array of 2D location points and returns a list of urls to execute to the API service provider. As it is highly unlikely that different APIs will require identical functionality, this approach shifts the implementation to the user-developer.

3.2 Major Technologies Utilized

3.2.1 TypeScript and JavaScript

TypeScript is a programming language that builds on top of JavaScript with a strong type system by encouraging developers to write interfaces, a contract that ensures the properties of a JavaScript Object are consistent (Microsoft Corporation, 2023). Although TypeScript code gets transpiled down into JavaScript code, the stricter nature of TypeScript allows for projects to scale securely with the knowledge that parameters are of an expected and compatible type. This aspect of the language makes it ideal not only for rapid development but also to ensure a lower learning curve when developing custom-built API extensions.

3.2.2 Electron

Electron is a JavaScript framework for building cross-platform desktop applications with web technologies, development tools, and knowledge bases (OpenJS Foundation, n.d.). Electron's architecture works by launching a Chromium web browser and a server process in the background. To handle the responsibilities and restrictions for each process, an Inter-Process Communication (IPC) message broker is used. By utilizing Electron, platform-specific requirements to compile a shippable executable are already met, and the development of project-specific requirements can begin much quicker.

3.2.3 Node.js

Included with Electron, Node.js is an asynchronous server environment that can read and write from file systems (OpenJS Foundation, n.d.). Node.js is utilized by LandMarked3D to retrieve and cache data retrieved from third-party API requests and to serialize and deserialize the application session into a series of files.

3.2.4 React

React is a JavaScript frontend framework for managing application state and reactivity through components. To achieve efficient reactivity, React keeps track of data state with its dependencies to enable it to write to the browser's document object model (DOM) when necessary. This aspect of the language makes it ideal not only for rapid development but also to ensure a lower learning curve when developing custom-built API extensions. By utilizing the browser and React, the viewport can utilize scalable vector graphics (SVGs) with out-of-the-box support. SVGs also come with built-in events to extend for interactivity, which raster graphics do not have.

3.2.5 D3.js

D3.js, or Data-Driven Documents, is a powerful and versatile JavaScript library that enables developers to create dynamic and interactive data visualizations from data using HTML, CSS, and SVGs (Bostock, n.d.). Additionally, D3.js is adept with manipulating geographic data, as it can transform geographic coordinate spaces to make working with pixel x and y positions easier. This makes it an ideal tool for developing location-based visualizations, such as maps and terrain models.

4. Conclusions

In conclusion, LandMarked3D is a promising open-source application that simplifies the process of data aggregation and initial processing. As this project continues to evolve and improve, it holds the potential to be extended to retrieve data from various sources to model and process data that maps to geographic coordinates for a broad range of application domains. The software is currently under development, and the latest executable can be accessed on GitHub through the following link, https://github.com/MaddHatt-PM/LandMarked3D. Interested parties who wish to modify and contribute to the project can also use that link for instructions on setting up their local development environment.

There are several key areas that could be improved upon to make the application even more useful. An area for improvement for LandMarked3D would be the development of tools for efficiently and accurately creating Foliage Meshes. Another potential area for improvement would be the development of a more organized user interface for creating and utilizing foliage surfaces across multiple projects. Additionally, offering more default API services could help expand the range of data sources available to users. Finally, developing the means to compare a property across multiple datasets in 2D and 3D. Finally, providing the means to view and interact with generated 3D models inside of LandMarked3D rather than through an external application would help to further streamline the workflow of users.

5. Acknowledgements

I would like to express my gratitude to the individuals and organizations who supported and assisted the development of this research project. First, I would like to thank the University of North Carolina at Asheville for providing grant money through the Summer Undergraduate Research Program. This grant, in the amount of $1500, was instrumental in enabling me the time to carry out this project's development.

I would also like to extend my thanks to my faculty advisor and professor, Dr. Kevin Sanft, for providing the opportunity to participate in this research project and for his guidance and mentorship throughout the process. Dr. Sanft's flexibility and support allowed me to approach the project requirements independently, drawing on my own experiences and experimenting with new technologies.

Furthermore, I would like to thank Dr. Mark McClure for his special topics computer science class on Dynamic Data Visualization. Without it, I may not have discovered D3.js and ultimately rebuilt LandMarked3D with web technologies.

In addition, I would like to acknowledge the contributions and encouragement of my friends and family who supported me throughout this project's development. With special regards given to my grandpa, William R. Martin III, who has always supported me in my academic and personal endeavors.

Thank you all for your contributions, support, and encouragement. I could not have completed this project without your help.

6. References

Bostock, M. (n.d.). *D3: Data-Driven Documents*. D3.js - Data-Driven Documents. Retrieved April 3, 2023, from https://d3js.org/

Meta Open Source. (n.d.). *Preserving and Resetting State*. React. Retrieved April 3, 2023, from https://react.dev/learn/preserving-and-resetting-state

Meta Open Source. (n.d.). *React: The library for web and native user interfaces*. React. Retrieved April 3, 2023, from https://react.dev/learn

Microsoft Corporation. (2023, March 3). *TypeScript for JavaScript Programmers*. TypeScript: Documentation. Retrieved April 3, 2023, from https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

OpenJS Foundation. (n.d.). *About Node.js®*. Node.js. Retrieved April 3, 2023, from https://nodejs.org/en/about

OpenJS Foundation. (n.d.). *What is Electron?* Electron. Retrieved April 3, 2023, from https://www.electronjs.org/docs/latest/