The Journal of Open Source Software

terrainr: An R package for creating immersive virtual environments

Michael J. Mahoney[1], **Colin M. Beier**[2], **and Aidan C. Ackerman**[3]

1 Graduate Program in Environmental Science, State University of New York College of Environmental Science and Forestry, Syracuse, New York, USA **2** Department of Sustainable Resources Management, State University of New York College of Environmental Science and Forestry, Syracuse, New York, USA **3** Department of Landscape Architecture, State University of New York College of Environmental Science and Forestry, Syracuse, New York, USA

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

`terrainr` is an R package designed to make it easier to produce immersive virtual environments in the Unity video game engine from real-world spatial data. By providing functions for data access and retrieval, wrangling and format transformation, and import into Unity, `terrainr` improves the speed and reproducibility of developing such visualizations. With the ability to process spatial data in both raster and vector formats, combine data from multiple sources, and rapidly iterate on visualization components, `terrainr` makes it easy for researchers to develop IVEs from real-world data and apply these visualizations to new problems and domains.

Statement of Need

An exciting frontier in spatial visualization is the use of immersive virtual environments (IVEs) as a method for representing real-world locations. IVEs allow users to explore spatial representations in virtual reality (VR), aiding scientific communication by giving users the sense of "being there" within a virtual world (Hruby, Ressl, & Borbolla del Valle, 2019). While many examples of IVEs used in research are entirely manually created, with a visualization designer attempting to develop a representation of a plausible natural environment, recent work has looked at the possibility of representing real-world landscapes by producing IVEs from spatial data (Keil, Edler, Schmitt, & Dickmann, 2021).

However, a major hurdle in the use of IVEs as a way to represent real-world spaces is the gap between the tooling used to handle geospatial data and that used to create IVEs. IVEs are typically created within video game engines, which provide advanced VR capabilities but do not share data models or formats with traditional GIS systems (Hruby et al., 2019). While it is possible to manually transform data produced by one system to formats understood by the other (Keil et al., 2021), this process is slow and imprecise, typically requiring repetitive manual data processing procedures which can be hard to reproduce or debug. The difficulty of transforming real-world spatial data for usage in IVEs, alongside the labor-intensive creation process, is likely a factor in the slow adoption of IVEs as spatial visualizations and the accompanying lack of theoretical research into their usage (Hruby et al., 2019).

This paper introduces `terrainr`, a new package for the open-source R programming language (R Core Team, 2020) which aids in the retrieval, manipulation, and transformation of spatial data for IVEs. By integrating with the most popular geospatial data formats within R and providing tools to automate importing geospatial data such as digital elevation models and imagery into the Unity 3D video game engine (Unity Technologies, 2020),

terrainr aims to make it easier to produce IVE representations of real-world locations.

Package Overview

The `terrainr` package provides functions for the retrieval, manipulation, and transformation of spatial data entirely within R. Package functions are loosely grouped between those which provide access to public domain spatial data from the U.S. Geological Survey's National Map program (U.S. Geological Survey National Geospatial Program, 2020), and those which process arbitrary spatial data for visualization within the Unity 3D video game engine.

Data Access and Retrieval

The `terrainr` package provides a consistent API for programmatic access to public domain data from the U.S. Geological Survey's National Map program from within R, allowing users to download data for areas within the continental United States from 10 separate web APIs. To download data, users specify an area of interest by providing objects of any class from either the `sf` or `raster` packages alongside a number of other optional parameters, including the desired pixel resolution and coordinate reference system of the returned data. `terrainr` will then attempt to download data for the minimal bounding box surrounding the provided spatial data, resampled to the target resolution and converted to the desired coordinate reference system.

Creating IVEs

Non-`terrainr` methodologies

An approach for creating immersive virtual environments inside the Unity game engine using real-world terrain data is provided in section 3 of Keil et al. (2021). This approach involves first manually downloading data for an area of interest from a data portal, with the user responsible for inputting their area of interest into a web-based graphical interface or identifying which pre-generated data tile best matches their desired location. This data must then be transformed into a TIFF image and imported into image-editing software, where the user must convert the image into the greyscale colorspace, rescale it to a size accepted by the Unity engine (which only imports square terrains with sides of 2^x+1 pixels, for $5 \leq x \leq 12$) with an accompanying loss of spatial fidelity, then manually exported into the "RAW" format with planar interlacing of color values. Finally, the user must import this RAW file into the Unity, taking care to ensure that options reflecting file bit depth, pixel extent in the X, Y, and Z directions, resolution, system endianness, and terrain orientation options are set correctly. This process results in, as phrased by Keil et al. (Keil et al., 2021), "an untextured terrain excluding additional spatial elements." While the surface will have mostly (though not entirely, due to the image rescaling) accurate terrain heights, no other information (such as other raster imagery or vector geometries) will be represented (Figure 1, left). This process incorporates multiple softwares (namely the data portal, a GIS program for format conversion, image editing software, and Unity) and must be repeated for each tile the user wishes to import, with each tile limited to a maximum of 4,097 pixels in length.

It is difficult, though not impossible to add other spatial data to this surface. For instance, orthoimagery may be added as an image overlay by importing an image file as a Texture2D object within Unity, attaching it to a terrain tile as a TerrainLayer object, and then setting the X and Z dimensions of the new TerrainLayer to the image's side length in pixels, assuming that the orthoimage has been cropped to the same extent and resolution as the terrain tile. The same process may be used to import any imagery, meaning that it is possible to manually add overlays of additional raster data or rasterized vector geometries

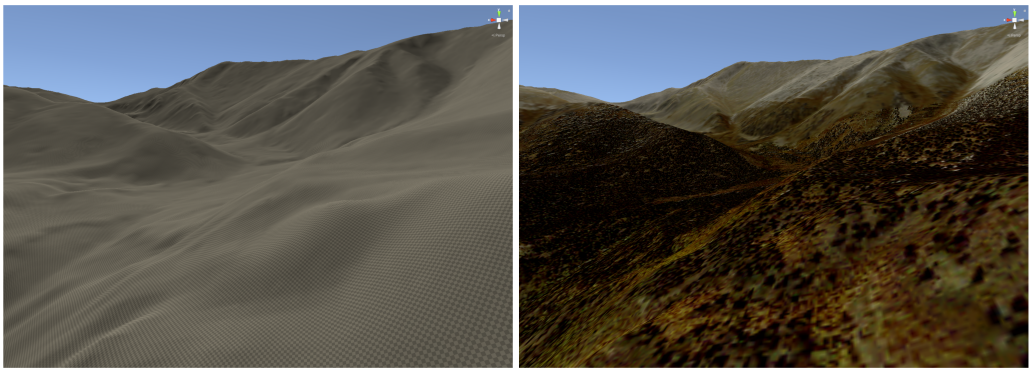

Figure 1: Left: Untextured terrain, as produced via the Keil et al (2021) methodology. Right: The same terrain surface, textured with aerial orthoimagery from the National Agricultural Inventory Program. In both images, areas closer to the camera are rendered in more detail, with individual pixels visible on the left and landscape features such as trees visible on the right. Both surfaces were created using the terrainr R package.

to these scenes. However, the need to ensure that imagery maintains the same coordinate reference system, extent, and resolution as the elevation data throughout all manipulations is a notable obstacle to incorporating geographic data in these visualizations.

`terrainr` methodology

`terrainr` attempts to simplify this process by combining R's advanced geospatial processing ecosystem and image editing packages to handle data transformation, image manipulation, and data transformation, while providing an automated method for importing terrain into Unity which abstracts away the many options and decisions required to import geospatial data. Users create a single-band raster, as well as an optional multi-band raster of the same extent, resolution, and sharing the same coordinate reference system, and save these rasters as any format supported by their GDAL installation (GDAL/OGR contributors, 2021). They then provide these files to the `make_manifest` function, which handles transforming both the single-band raster for importing into Unity as an elevation layer as well as the multi-band image for importing as a corresponding terrain texture. These transformed layers are saved as individual tiles alongside a C# script providing the automated terrain import methods and a "manifest" file listing the dimensions and relative spatial positions of each transformed tile. After copying these files into a new Unity project, users will notice a new "terrainr" menu option in their Unity app menu bar. This menu option produces a terrain import wizard which will read the manifest file, import the terrain tiles into Unity in their proper scale and spatial arrangement, and then attach any provided image overlays as terrain textures.

On its own, `terrainr` provides for the first time a method for creating reproducible IVEs in Unity from arbitrary spatial data; any raster data format supported by GDAL may be used to produce visualizations, while all decisions and data manipulations involved in the production of the IVE are preserved as simple R code. Additionally, this method allows for IVEs to be produced from real-world data much faster than is possible manually; even when accounting for data download from the National Map, it is possible to produce a new IVE in Unity from whole cloth in under ten minutes. `terrainr` also makes it easier to work with multiple image overlays through its `combine_overlays` function, allowing users to "stack" imagery with controllable opacities in order to produce single images for use as terrain textures. Additionally, users may take advantage of the function `vector_to_overlay` to rasterize vector geometries, obtaining images of the same extent as a reference raster. In this way, users are able to use `terrainr` to represent multiple levels of spatial information within a single IVE; for instance, Mahoney, Beier and Ackerman

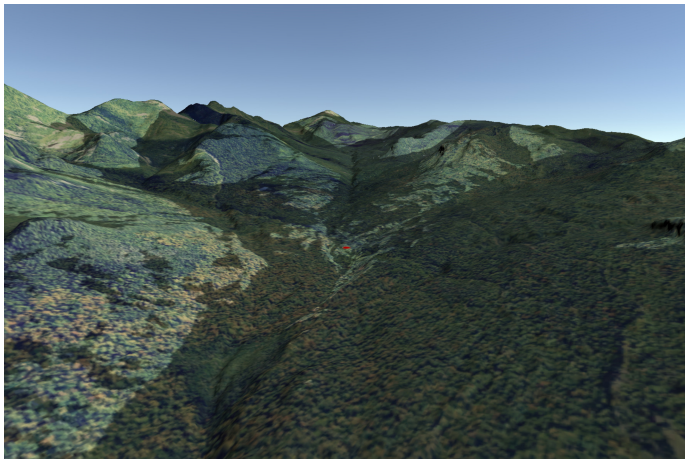

Figure 2: From Mahoney, Beier, and Ackerman (2021), an IVE produced using terrainr showing a section of the Adirondack Park in northeastern New York State, USA. A red dot in the center of the image represents Johns Brook Lodge. Areas which can see and be seen by the lodge (that is, within the lodge's viewshed) are brightly lit, while areas outside the lodge's viewshed are dim.

(2021) used `terrainr` to combine point geodata, a boolean raster of viewshed boundaries, and orthoimagery with elevation data into a IVE of a region within New York State's Adirondack Park (Figure 2).

Acknowledgements

This work was supported by the State University of New York via the ESF Pathways to Net Zero Carbon initiative.

References

GDAL/OGR contributors. (2021). *GDAL/OGR geospatial data abstraction software library*. USA: Open Source Geospatial Foundation. Retrieved from https://gdal.org

Hruby, F., Ressl, R., & Borbolla del Valle, G. de la. (2019). Geovisualization with immersive virtual environments in theory and practice. *International Journal of Digital Earth*, *12*(2), 123–136. doi:10.1080/17538947.2018.1501106

Keil, J., Edler, D., Schmitt, T., & Dickmann, F. (2021). Creating immersive virtual environments based on open geospatial data and game engines. *KN Journal of Cartography and Geographic Information*, *71*, 53–65.

Mahoney, M. J., Beier, C. M., & Ackerman, A. C. (2021). *Interactive landscape simulations for visual resource assessment. Proceedings of the 2021 Visual Resources Stewardship Conference.*

R Core Team. (2020). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Unity Technologies. (2020). *Unity*. San Francisco, United States of America: Unity Software Inc. Retrieved from https://unity.com/

U.S. Geological Survey National Geospatial Program. (2020). *The national map*. Washington D. C., United States of America. Retrieved from https://viewer.nationalmap.gov/services/