Pangeo Technology Research

Game Engine

Unity and Unreal are two of the most popular game engines in the VR development industry today. Determining which one is better suited to our project requires us to look at each of the engine's evaluation when it comes to support, initial learning curve, graphics, and terrain generation.

Support
The mentors at L3Harris as well as Dr. Rebinitsch has a background working in Unity and has offered the VRtualize team their resources to help with the accostomation to the engine. That is a large benefit to the team considering none of the members have ever developed a VR application before. There are also a lot of tutorials made by the Unity community while there are less available tutorials for Unreal.

Learning curve
Unity was founded in 2004 with an aim to make game development accessible for everyone. So right from the get-go, an intuitive interface was a large priority when designing the game engine. C# is used for scripting which makes it very simple to learn and use.
Unreal's interface is packed with features but with a mass of additional pop up windows and buttons, it's easy for a beginner to get overwhelmed with all the features. Unreal Blueprint is used for scripting and has an interface that is more geared towards designers than programmers.

Graphics
Unity's material editor is very limited. It supports normal maps, occlusion maps, and base textures but it has very limited parameters to adjust and customize.
Unreal is packed with tool presets that work right out of the box. It features post-processing that is fully integrated, providing wide options for grading, lens flares, volumetric effects, and other components that gives the system a similar feel to what is used in the film industry to give the application a smooth cinematographic feel. It also has a fully decked out material editor that utilizes node graphs to create and customize materials with live feedback within Unreal itself.

Terrain Generation
Unity has an editor to create terrain but to actually make the terrain look realistic, an asset download must be made.
Unreal comes with a great terrain editor right out of the box.

https://www.gamedesigning.org/engines/unity-vs-unreal/

https://gametorrahod.com/objectively-comparing-unity-and-unreal-engine/
http://cgicoffee.com/blog/2018/01/unity-real-time-rendering-vs-offline-cgi
https://iopscience.iop.org/article/10.1088/1755-1315/20/1/012037/pdf\

Key Parameters	Weight	Unity3D	Unreal4
Support	10	5	2
Learning Curve	7	4	2
Graphics	7	3	5
Terrain Generation	7	4	5
Scores	**N/A**	**4**	**3.6**

Elevation Data Source

USGS:

- Have an expert to ask questions
- We have experience pulling data by hand
- Multiple forms of data quality available (1/9 arc-second data, ⅓ arc-second data, ⅔ arc-second, 1 arc-second, etc.)
- Does not require an account to access data
- Does not have have any image data for locations
- Based entirely in North America

Bing Maps:

- Requires an account
- Has “free-use” limits in place to limit data downloading without payment
- API with good documentation that only requires calculation of a quad-key
- Includes images related to locations for texturing terrains
- Stores data about different resolution levels at different altitudes
- Has data about areas around the world

L3Harris proposed USGS and Bing Maps as potential primary sources for downloading topological data. After careful consideration, we elected to use the Bing Maps database for collecting all of our information. We initially believed that USGS would be a better database to use, but after exploring Bing Maps more, we found it to be more useful. Both databases offer

the ability to view areas at different resolutions, but Bing Maps has far more levels of resolutions than USGS. As well, USGS only covers areas located in North America, primarily located in the continental United States, whereas Bing Maps has data for almost all parts of the world. Bing Maps also hosts image data for all areas at each resolution level, making the database a consolidated location to get all of the data we need for the database. Lastly, collecting the data from both locations is possible via web APIs, but by using Bing Maps we are able to collect all of the elevation data in a uniform binary blob and simplifies the process of processing data and injecting it into our database, whereas USGS requires downloading a large zip file, extracting it, finding the specific data file we need, and inserting that file's data into the database. It is noted that USGS is a free service that does not require any user accounts and does not have any limitations on data usage, while Bing Maps does require a user account and instills limitations on data downloads and API calls per second. However, Bing Maps allows for the usage of accounts oriented for education purposes and the upper limit for these call limitations is very large and shouldn't cause problems.
https://docs.microsoft.com/en-us/bingmaps/rest-services/elevations/get-elevations
https://viewer.nationalmap.gov/basic/

Key Parameters	Weight	USGS GIS	Bing Maps
Elevation Data	8	5	5
Web API	7	4	3
Free Use	5	5	3
Scores	N/A	4.6	3.6

Data Types

Please note, this section is a remainder from our implementation that utilized the USGS database and has been kept purely for legacy purposes. This section is not applicable to our current implementation.

There are several elevation data types available from the USGS. This includes DEM, ArcGrid, Gridfloat, IMG, and Lidar Point Cloud.

DEM is an open source standard developed by the USGS. It supports different levels of quality and is well documented, but is only available for select areas of the United States. This makes topological generation of the world impossible until more data is added.

ArcGrid is another file format that is used by ArcGIS, a software for mapping and performing analysis on terrain. ArcGIS is proprietary and requires a license to use. In addition, ArcGrid files contain more data than just elevation, so they would take up more space than is needed for this project. ArcGrid is available for all areas of the United States and many other countries though.

Gridfloat is a file format recommended for users with custom applications. This is due to its intuitive file format, using a header file to store metadata and then binary data in a separate file. It only stores elevation in the binary file, so it only provides the data we are interested in for this project. Gridfloat data is also available for all locations in the United States and in many other areas of the world.

IMG is a file format that is not as well documented as the previous options. It has a great integration with the Python library GDAL to interpret it and it is available everywhere that Gridfloat and ArcGrid data is. There isn't as much documentation on this file type and it is not as widely used as ArcGrid and Gridfloat formats.

Lidar Point Cloud (LAS/LAZ) formats are very well documented and offer very high resolution elevation data. There are Python libraries for interpreting the data and both compressed (LAZ) and uncompressed (LAS) file formats are available. The data is not available for the entirety of the United States or any other country. These files are also quite large due to their high resolution and would greatly reduce the number of chunks we could cache for our program.

Our final decision is to use GridFloat due to its simple file format, wide availability, smaller file size, and documentation.

Data Storage

We considered two major methods of data storage, either on the local filesystem or in a database. A local copy of the map data would require terabytes of space on every local machine, but would be the fastest method of access for data due to a lack of network latency. It would also require the local machine running the VR application to deal with the overhead of searching a local disk for the data. The other option would be to store all of our map data in a database. Data would only have to be stored in one location for multiple users on the same network. In addition, finding specific chunks of the map in the terabytes of data would be quite efficient in an indexed database. Databases also generally compress their data, making it even easier to store all the desired elevation data from around the earth. A database would also handle searching data efficiently, reducing processing and search time for specific data. The final decision was to use a database because it did not require multiple copies of the map data for every machine on the local network and the requirement of a caching algorithm by L3-Harris greatly reduces the impact of any network latency for database calls and response.

We also considered the fact that a REST API from Bing or Google being called for data would not require any local storage, but we would still want a local cache of that data to reduce the cost of calls to the API. The final decision was to use the APIs for satellite imagery, but to cache the imagery in our database due to cost of repeated calls. Elevations require many expensive API calls in practice, so we opted to store this data locally in a database.

The decision of which database to use came down to performance, cost, compatibility, and familiarity. There are many great databases available, but the major contenders we compared are Microsoft SQL Server, PostgreSQL, SQL Lite, and MySQL, due to their popularity and because they are well maintained.

Microsoft SQL Server offers great performance, indexing, and scalability. The major drawback from MSSQL is that it is proprietary and doesn't outshine other free and open source databases. PostgreSQL is one of these free and open source databases. It also offers great performance, indexing, and scalability and was one of our top choices for a database. SQLite is public domain and was considered mainly because it doesn't require high performance hardware to operate, but it also doesn't offer the same performance as other databases when put under higher loads. MySQL Server offers high performance, an open source version, indexing, and is easily scalable. The decision came down to PostgreSQL and MySQL due to this, but MySQL was the final choice due to our familiarity with it and its well documented adapters for Python and C#.

Key Parameters	Weight	Filesystem	Database	Rest API
Performance	5	2.75	4.25	4
Latency	5	5	4	3
Search	5	4	5	4
Local Processor Usage	5	1	4	5
Storage	5	1	4	4
Cost	5	5	4.33	3.33
Hardware	5	5	4	3
Software	5	5	5	5
Runtime	5	5	4	2

Source Control

For VCS, there are two types:

- Centralized
  - Single Centralized project source
  - File locking
  - Easier to learn with a mostly linear history logging
- Distributed
  - Distributed clones of project source
  - Branching and merging
  - Complex yet more detailed history logging

While there are many different solutions for both, it boils down to two main systems, Git, and Subversion. Subversion is older than Git, and is really well suited towards large companies with complex projects that may span across human resources of the entire company. With a workforce dedicated to managing the centralized project source and workflow, the entire development process becomes centralized delegated without worry of overlapping work and code conflicts thanks to file locking. However, Git has an overwhelmingly positive reception in the development community right now. While the features of Git make VCS more confusing and complicated, the features serve to give more power and flexibility to the user. Distributed clones of the project give way to potential code conflicts and does not stop developers from overlapping work on its own, but it also allows developers to access the code *offline* and continue to work in situations of spotty or slow internet connections. Branching and merging serve to help with code conflicts, and issue trackers aid developers in delegating work among each other. This puts the workflow in the hands of the developers who *may* be closer to the project code. The biggest reason for Git though is its vast amount of interfacing software solutions to interact with it.

Key Parameters	Weight	CVS	DVS
Source Access	10	3	7
Conflict Resolution	10	4	6
History Tracking	3	5	5
Scores	N/A	4	6

There are a lot of VCS clients, for both Git and SVN. However, VRtualize is most familiar with solutions involving Git, and the most prominent would be GitHub and GitLab. Git's rise to fame is closely coupled to GitHub, but these clients were made to do more than just provide an interface to Git, both clients serve to be full devops solutions, with several different key ideals to achieve this. First, the similarities they share include merge requests, continuous integration, and deployment tools. These enhance Git by reducing work duplication, controlling conflict resolution, and verifying code quality. Next, the differences are what contribute to our end decision, particularly the fact that GitHub has many third party integrations that allow for a large range of extended features. One of these third party integrations is Hygger. Hygger provides more features to the issue tracker, including story points, swimlanes, and burndown charts. While GitLab provides this in its built in issue tracker, it’s only available to its enterprise subscribers, while Hygger keeps this free. Another one would be code review and approvals. Both GitHub and GitLab provide code review tools, and can even enforce rules for merging code, such as number of reviewers required for each file. However, for GitLab users, this is available only to certain subscription levels, while GitHub offers this for free.

Key Parameters	Weight	GitHub	GitLab
Code Review	10	7	3
Issue Tracking	8	7	3
Scores	N/A	7	3

Issue Tracking Systems

Trello
Although free to use with a simple Kanban style workflow, Trello lacks free features like integration to the source control system. Easy to learn, unlimited tasks, boards, and members.

Hygger
Ability to manage sprints, create burndown and velocity charts, and integrate to the source control system. Swimlanes are used to separate each developer’s work progress.
Max of 50 tasks.

ZenHub
ZenHub features direct integration with GitHub. It is designed for agile development and allows for great management of issues and sprints. Burndown and velocity charts are included, as well as customizable boards to manage workflow. There are paid versions for private repositories, but ZenHub is completely free for open source and public projects.

Key Parameters	Weight	Trello	Hygger	ZenHub
Price	10	3	3	5

Issue Tracking	10	3	4	4
Work Flow	8	3	5	5
Reporting	5	0	5	4
Source Control Integration	6	2	4	5
Scores	**N/A**			

Unity Explorable GameObject

Terrain
These GameObjects tend to be the go-to choice for most developers but tend to be problematic when working solely with game scripts. Terrain objects are best used when supplied with a heightmap which is directly applied to the GameObject via the GUI.

Mesh
Meshes tend to be more friendly when it comes to pure script manipulation, though they require slightly more information to set up correctly. There are a lot of tutorials about manipulating meshes available and documentation to help understand how to work with them.

Key Parameters	Weight	Terrain	Mesh
Dynamic loading	10	7	8
Scripting ease of use	8	3	6
Learning resources	5	3	7
Scores	**N/A**	**4.3**	**7**