A blue and black logo

Description automatically generated**Department of Applied Computer Games (DACG)**

**Honours Project - MHW225671**

**FINAL REPORT**

**2023-2024**

**Submitted for the Degree of:**

**BSc Computer Games (Software Development)**

|  |  |
| --- | --- |
| **Project Title:** | **Voxels and Polygons, competing for the standard geometric primitive in 3D Visualization.** |
| **Author:** | **Lewis McIntyre**  **S2028246** |
| **Project Supervisor:** | **Bryan Young** |
| **Word Count:** |  |
| **Aligns with SDGs:** |  |

(**Word count *excludes contents pages, figures, tables, references and Appendices***)

Digital signature here: scan or photograph your signature and paste it below.

**“Except where explicitly stated, all work in this report, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award”**

**Signed by Student: Lewis McIntyre Date: 22/04/2024**

# Abstract

* Complete overview/Concise Summary, from why and how you did it, as well as what you found.

# Acknowledgements

Contents

[Abstract 2](#_Toc161307954)

[Acknowledgements 3](#_Toc161307955)

[1. Introduction 5](#_Toc161307956)

[1.1 The Standard Geometric Primitive – Polygons 5](#_Toc161307957)

[1.2 An Alternative – Voxels 5](#_Toc161307958)

[1.3 Voxel Development 6](#_Toc161307959)

[1.4 Research Question 7](#_Toc161307960)

[2. Contextual Review 8](#_Toc161307961)

[2.1 Performance Enhancing Techniques 8](#_Toc161307962)

[2.2 Current Tools 9](#_Toc161307963)

[2.3 Voxelization 10](#_Toc161307964)

[2.4 Voxels and Polygons 11](#_Toc161307965)

[3. Methods & Design 12](#_Toc161307966)

[3.1 Method Plan 12](#_Toc161307967)

[3.2 Development Environment & Tools 12](#_Toc161307968)

[3.3 Data Gathering 15](#_Toc161307969)

[3.4 Data Analysis 16](#_Toc161307970)

[3.5 Usefulness of Analysis 16](#_Toc161307971)

[3.6 Ethical Considerations 16](#_Toc161307972)

[4. Study Execution 17](#_Toc161307973)

[5. Outcomes 17](#_Toc161307974)

[6. Discussion/Conclusion 17](#_Toc161307975)

[7. References 17](#_Toc161307976)

[8. Appendices 21](#_Toc161307977)

# Introduction

## The Standard Geometric Primitive – Polygons

Polygons are accepted as the current standard geometric primitive for both the 3D Modelling and Video Games industries; they can be defined as any closed 2D shape made of entirely straight lines. Current GPU manufacturers and tech giants have heavily invested into polygons, originating back to the late 1990s when the term was popularized by the GeForce 256 GPU. (TheBat!, 2023) This can be ascribed to their simplicity and efficiency with the many rasterization techniques at that time. However, their most prominent drawback is within its ‘*imitation*’ of real-world objects, as many techniques are only aesthetically, and not physically simulating the mesh. This is more specifically seen in games through the 3D dissection of polygonal models at runtime, where the model holds no data for the new face required to be created.

## An Alternative – Voxels

A suggested alternative, Voxels; also known as volumetric pixels; are a geometric primitive holding data for the entire model, including what cannot be seen. This primitive is used to represent values in three-dimensional space on a grid. Voxels function similarly to physical particles, therefore creating a more sophisticated implementation used to imitate the real world. Often, Voxels are referred to as 3D pixels and have a wide use in Procedural Generation, Particle Simulation, and Destructible Physics. (Zadick, Kenwright, & Mitchell, 2016) (Sekanina, 2023) Voxels are uniquely stored within a grid, allowing efficient usage of Object-Oriented Programming (OOP) and the Entity Component System (ECS), enabling each voxel to hold any number of unique properties. This is shown through the 3D voxel game ‘*MakeFarm*’ by David Szymon Grobert, as each block holds whether it is breakable, and if so, what item should be dropped. (GROBERT, 2023)

Polygons have held their position as the standard geometric primitive in many of the 3D visualization industries for over 20 years. This has resulted in many of the competing primitives to fall behind, as new technologies advance polygons forward. Many rendering practices for these primitives require the conversion of their structures into polygons, a major of which being the Marching Cubes algorithm. (Lorenson & Cline, 1987) Most previous rendering techniques for mainstream geometric primitives are insufficient for modern day usage, as researchers are now aiming to create greater realism. Due to the observation that is Moore’s law, which dictates that the number of transistors within computer chips double every two years, graphics are no longer restricted by our hardware, but by our software approach. A common implementation includes Ray Tracing, a method of emulating the light reflections and refractions of the real world. Ray Tracing is entirely possible using Voxels, although a similar process named Ray Marching is generally used; however, both of which do not require the need to convert to a polygonal mesh. These techniques are potentially creating a new use case for Voxels, with the possibility of gaining a wider acceptance as an alternative to polygons. The improved graphics that come with Ray techniques leads way into the necessity for Particle Simulation and Destructible Objects, creating a greater sense of realism, and both of which voxels are suited towards.

## Voxel Development

Atomontage, a leader in voxel development with a 13-year running micro-voxel engine, are consistently pushing the boundaries of voxel development forward. Atomontage are known for their usage of projection based voxelization, soft-body dynamics and their voxel editor’s recent open beta launch in 2021. (Atomontage, 2023) Daniel Tabar, and Branislav Siles, the founders of Atomontage, shared that the future of voxels lies within cloud services, as with their current implementation, the voxels are relatively inexpensive on both the view, and the server. The future of Atomontage aims to revolutionize both the cloud gaming industry, and the interactive sandbox games genre, and Daniel Tabar has stated that their engine could be compared to “*Roblox + Minecraft + a higher resolution*”. (JTVentures, 2023)

Atomontage is not only invested into the games industry, but also has ties to medicine from one of its most supportive angel investors, Tommy Palm (Siles, 2019). Voxel-based 3D visualization is an incredibly large sector within medicine, with competing software such as ‘*3D Slicer*’, ‘*Voxel-Man ENT*’ and ‘*SolidWorks 3D-Doctor*’. These provides a multitude of tools ranging from CT Lung Analysis to Virtual Training Simulators, all of which involve intensive usage of volumetric data. The usage of voxels in medicine comes from the necessity of creating not only a realistic model, but also for accurate mapping of different tissue types. (Slicer Community, 2023) (Voxel-Man, 2023) (Dassault Systemes, 2023) As voxel techniques are researched, these software could potentially receive updates allowing more realistic, and accurate models for its users. A study within the effect of Field-Of-View and Visual Realism on virtual training tasks found that visual realism may impact the virtual training performance negatively, but that it impacts the future real-life assessment positively. (Ragan, et al., 2015) This suggests that if voxels gain more realistic rendering techniques, that it could positively impact training procedures for users of Medical Simulations.

The games industry is at the forefront of realistic voxel development, a recent voxel success story includes Tuxedo Labs ‘*Teardown*’. Featuring a fully destructible environment, particle simulations of fire and smoke, and an incredible in-house lighting system, it has garnered almost 76,000 positive reviews on steam, gaining the award for ‘*Excellence in Design*’ at the Independent Games Festival in 2021. (Tuxedo Labs, 2023) (Informa Tech, 2023) It is a game that has shown the practicality of voxels in indie game development, and the massive support such a game can receive. According to the Tuxedo Labs modding wiki, the artists used ‘*MagicaVoxel*’, a lightweight voxel art editor, to create all assets for the game. (Tuxedo Labs, 2023) This art editor is still supported, is fully open source and has hundreds of modelling tutorials. It is an incredibly powerful tool within the modelling of 3D voxel environments; however, they require hundreds of hours in development to take form. A more efficient method, used for both rapid prototyping and final model products is LiDAR (Light Detection and Ranging). LiDAR uses pulses of light to rapidly generate point clouds, which are simply a collection of points in three-dimensional space. As suggested by Xu, Tong & Stilla, point clouds can be efficiently converted into a voxel representation, and then can be effectively used for structural analysis, 3D simulation of fluids, volume estimation and pathfinding. Voxels are mentioned to be one of the best primitives used to transfer point cloud data into a higher data format, due to their speed and efficiency, although suggests they may lack accuracy in conversion as compared to some others. (Tong, Stilla, & Xu, 2021) If too much accuracy is lost, this technique may still be useful within rapid prototyping, or give artists a reference prop to work from. (Yuan, Peng, & Zhang, 2018)

## Research Question

The aim of this paper is the investigation of the possible voxel future, as a replacement for polygons as the standard geometric primitive for 3D visualization. The strengths and weaknesses of voxels should be highlighted, whilst further research avenues should be opened and suggested. The research question is:

*“Could voxels replace polygons, as the standard geometric primitive for 3D visualization?”*

# Contextual Review

## Performance Enhancing Techniques

Research within Design & Technology (Telea & Jalba, 2011), Engineering (McNeely, Puterbaugh, & Troy, 2005) and Biomedical Studies (Mason, et al., 2000) suggests a common limitation of large memory consumption whilst using the standard implementation of regular voxel grids. The regular grid structure creates a memory footprint of O(n^3) where n represents the width, length and height. Recent studies have explored the manipulation of a regular grid using warp fields to create dynamic irregular grids, which significantly reduced memory cost; however, this was very specific towards their use case. (Lombardi, et al., 2019) There are many other solutions to this issue, a notable case involving the major Australian software company ‘*Euclideon*’. A video posted in 2010 by the CEO of Euclideon, Bruce Dell, showcases an ‘*Unlimited Detail Technology*’, which garnered mass attention towards an engine which could dynamically render any number of objects in any scene no matter how large. (Quipster99, 2010) It appears to do so by flexibly changing the Level of Detail (LOD) of a voxel grid at runtime. A similar technology was used in Funk & Borner’s research of efficient LOD visualization tools and solved this issue through the usage of Sparse Voxel Octrees (SVOs). (Funk & Borner, 2016) SVOs are a rendering technique, created through the combination of Sparse Voxels, and Octree Voxels. These are defined as the following:

The rendering technique Sparse Voxel removes the data of all unoccupied values. (Figure 1)

The rendering technique Octree Voxel divides the 3D space into a hierarchy of octants, that are recursively divided until a goal is achieve, e.g. the final level of detail is reached. (Figure 2)

A diagram of a circle and a cross

Description automatically generated

Sparse Voxel (Figure 1)

A diagram of a circle and a cross

Description automatically generated

Octree Voxel (Figure 2)

Studies have found that both the rendering methods Sparse Voxel, and Octree Voxel, can be incredibly efficient in the optimisation of non-volatile memory, a mentionable study by Gebhardt, Scott et al. finding both file sizes ranging from 0.0043x and 0.0006x that of their traditional regular grid counterparts. (Gebhardt, et al., 2009) It can be implied that this technique could be further improved by the combination of each technique, to create the Sparse Voxel Octree structure. However, these techniques are not perfect, due to the nature of their compression, models that are more densely populated, with zero empty space or duplication would not receive the same benefits as those without. Therefore, a greater level of compression which can be processed with a more efficient algorithm could help support voxels and increase the feasibility of voxels in more 3D visualization software. (Laine & Karras, 2010)

The costs of both compression and decompression techniques used within voxels can have a heavy cost within the CPU and the GPU, specifically when implementing real-time graphics and dynamic construction and deconstruction of voxels. There are multiple grid traversal techniques used to mitigate this cost, a notable recent usage includes Morton Order, also known as the Z-order curve. (Baert, Lagae, & Dutre, 2013) It is used to convert any set of multidimensional data into a linear sequence for fast traversal, in our case three-dimensional voxel space. Pan, Yucong found when investigating the traversal of Sparse Voxel Octrees using Morton Order, that without the usage of this technique, or one of similar standard, real-time global illumination would not be feasible, especially when working with a voxel grid that is modifiable at runtime. (Pan, 2021)

The goal of Global Illumination is to create a high level of visual realism, which is a crucial component to many of the 3D visualization industries, with notable industries such as product design and advertising needing extra realism. (Dutre, Bekaert, & Bala, 2018) (Kim, Choi, & Wakslak, 2019) This goal can be found using a variety of techniques, although at the highest level of realism it is generally by a form of light-simulating rays. Notable practices include Ray Casting, Ray Tracing and Ray Marching. These techniques are incredibly similar; however, the key technique Ray Marching stands out within voxel development, which traverses the voxel grid using distance estimations. Dreams, a voxel-based game creation system with in-built 3D modelling software uses a form of Ray Marching known as Sphere Tracing, which has the added benefit of smoothing all surfaces hit. (Alex Evans, 2015) Dreams runs entirely on the PlayStation 4 and shows the incredible possibilities of voxels as a base for model storage.

## Current Tools

Technology within games constantly expands the possibilities of 3D visualization, the tool Unreal Engine by Epic Games standing out as a main contributor. Although Unreal Engine was initially created as an in-house engine for the First Person Shooter ‘*Unreal*’ (1998), it quickly gained notoriety and opened for developer use, and then public usage in late 2009. (Kao & Wang, 2023) Unreal Engine is a 3D visualization tool which is not only used within the Games Industry, but has seen usage in media, with over six hundred usages in film and TV shows, and has in-built templates supporting Architecture, Automotive Design, Simulation and Animations. (Epic Games, Inc., 2023) Voxels are supported inside Unreal Engine through the Voxel Plugin which has both a free legacy version, and an updated paid version which includes a perpetual license for any future version of Voxel Plugin. (Voxel Plugin, 2023) (Unreal Engine, 2023) This plugin supports many of the desired features for voxels, including real-time destruction and creation, procedural generation and physics simulations of terrains. Voxel Plugin functions well with the newest release of Unreal Engine 5, allowing the benefits of both ‘*Nanite*’, which dynamically edits the level of detail in the scene in real-time, and ‘*Lumen*’, which allows the real-time edits of all lighting features. Unreal Engine also allows the usage of a node-based scripting system named ‘*Blueprints*’ which is user friendly and provides designers the ability to create systems without the need for a programming background. (El-Wajeh, Hatton, & Lee, 2022) These suggested tools make Unreal Engine a major competitor as one of the software used within 3D visualization.

Unity is a large competitor to Unreal Engine, holding a 48% global market share of game developers as compared to Unreal Engines 13% in 2020. (Kao & Wang, 2023) Unity has many solutions for 3D visualization, some of which include Architecture, Engineering and Product Design. (Unity Technologies, 2023) A paper on voxel options with Unity involved the usage of mainly native techniques, and ongoing open-source projects supported by the community. (Aleksandrov, Zlatanova, & Heslop, 2022) It was found that voxels are supported within Unity, but many of the tools were left in the hands of the programmer, and within these tools were many limitations regarding large physics simulations. The open-source projects on the other hand generally use some form of octree compression; however, all were rendered using some form of conversion to polygons to then be rasterized. Some of the projects referenced are no longer accessible, or were not maintained and as such are to old to use with the current Unity pipeline.

Unity has the clear support of the community, with tens to possibly hundreds of voxel toolsets with many fully open sourced. There are a few more notable ones, and some of their functionalities should be mentioned. PicaVoxel, known as the TinyVoxel Toolset, was created by Gareth Williams who as of 2021 moved on from PicaVoxel to become a senior developer for SideQuestVR. (PicaVoxel About, 2023) The toolset is fully scriptable, has frame based animating capabilities, includes five demo games created with voxels and enables the import and export of voxel grids. It has taken a quite common modern approach to Voxels, which involves using multiple smaller voxel grids, instead of a singular larger one. (William, 2023) This technique is generally used due to its capabilities within games development, as each ‘object’ can be manipulated uniquely, allowing for player movement and dynamic interactions. A project titled Unity Sparse Voxel Octrees follows the approach of NVIDIA research ‘*Efficient Sparse Voxel Octrees’* (Laine & Karras, 2010)and creates a library that holds and renders Sparse Voxel Octrees. Aside from this, the project does not offer any other features. (Goslin, 2023)

## Voxelization

Voxelization is a term used to describe the conversion of geometric data into any voxel type (Regular, Sparse, etc). It should be noted that the conversion from one voxel type to another can also be referred to as voxelization. (Ma, Buyyounouski, Vasudevan, Xing, & Yang, 2019) Some major examples of voxelization within 3D visualization include Corner-Point grids in Geology, (Chen, Liu, Li, Zhang, & Li, 2017) Point Clouds in Construction (Tong, Stilla, & Xu, 2021) and Polygons in 3D Simulation and Animation. (Lopez-Moreno, Miraut, Circio, & Otaduy, 2017) Each type of initial representation has its own benefits on the converted representation, for instance, polygons will have zero holes and point clouds will have incredible detail. The strongest types of initial representations are those which involve multiple slices of data, particularly used within the Medicine industry through Magnetic Resonance Imaging. Typically, MRI scans use slices of tomographic image data, constructing the voxel grid with each pixel relating to a voxel on the grid. (Bushberg, Leidholdt, Seibert, & Boone, 2012) The benefits of slice of data within voxelization is due to its usage of internal data, as with many other model types of the internal structure is left up to the voxelization method used. Many voxelization methods fill the internals of the newly created voxel grid; however, they may occasionally use surface estimations to generate a hollow structure. There are many voxelization tools for polygons, with various techniques used. The most accessible version is found online within Arjan Westerdiep’s website. (Westerdiep, 2023) It supports over 30 file formats, can convert up to 2048x2048x2048 grids, and has features to change the thickness and size of the new voxel grid.

## Voxels and Polygons

Voxels and polygons hold a point of contention as the standard geometric primitive within some of the specific 3D visualization industries. Through some of the performance enhancing techniques voxels use, and the current tools within 3D visualization, we have found some of the strengths and weaknesses of voxels. From this research, it can be conveyed that voxels hold a major position within 3D visualization industries which require the need of internal data that may not always be seen. A particular industry which requires this data is Medicine with MRI and CT scan data, although it has been suggested that any industry which requires rapid prototyping could voxelize any set of image generated point-cloud data swiftly.

# Methods & Design

## Method Plan

Throughout the literature review the avenues within voxels have been discussed and the common downfall of voxels, which is that of memory consumption, has been found. As such, it is suitable to change the research question to create a more specified goal and combat the common hurdle that is memory usage. The new research question is:

*“How do polygons and voxels affect the memory footprint within common 3D visualization practices?”*

This new research question should be answered through the comparison of multiple polygonal meshes and voxel grids. The data being compared is the memory footprint of RAM. Some voxel memory saving techniques have been explored through the literature review, namely the Sparse Voxel, and the Octree Voxel. It would be useful to use these techniques; however, due to the complexity of their implementations, it is not feasible to create them within the time constraints of the project. As such, if there are tools available which use these techniques then they should be considered; however, it is not necessary for this project.

To ensure the consistency of each model, an original polygonal model should be used, and then a voxelization algorithm should be run on the same model to create an identical voxel grid.

An aesthetical component of this research should be considered, as within voxelization, the size of the grid of voxels is directly correlated to is blocky appearance. As such, we will edit the grid size during voxelization, such that the majority of the meshes components are visible. Each model will also employ a different artistic style, creating the secondary goal of defining which of the 3D visualization industries voxels may be less intensive in.

## Development Environment & Tools

We are limited in what 3D visualization tools we can use. It would be beneficial to avoid inconsistencies through the usage of different environments for each geometric type, and as such we must use a visualization tool capable of storing both polygonal meshes and voxel grids. The time constraints of this project must also be considered, and as such there should be some familiarity with this tool. Finally, the tool must be able to import assets, which may be created through its own in-built voxelization tool, or by using an external software.

Based upon our findings from the literature review, the decision comes down to between PicaVoxel for Unity (PicaVoxel About, 2023) and the Voxel Plugin for Unreal Engine. (Voxel Plugin, 2023) Due to not only a greater familiarity, but also greater access to its resources, the 3D visualization tool Unity will be chosen for this investigation. Using Unity, combined with the open-source voxel toolset PicaVoxel, we will capture Quantitative data of memory footprint. PicaVoxel does not have any in-built voxelization algorithm, and as such the findings from the literature review should be considered. The literature review found that ‘*Drububu Voxelizer’* is incredibly accessible and efficient, as well as having large voxel capabilities of up to a grid size of 2048x2048x2048 and as such will be used for this project.

The model aesthetical types were chosen based upon common genres within 3D visualization. They were also chosen such that each type portrayed both different levels of detail and base geometry usage. This can be promptly seen within comparing Sci-Fi and Minimalist, as Sci-Fi looks for incredible realism and showcases many cubic designs, were as Minimalist uses very simple shapes which usually create flow. The full list of these types is as follows:

1. Realism (Medicine, Games, Architecture & more) (See Figure 3)
2. Cartoon (Animation, Games, Education & more) (See Figure 4)
3. Minimalist (Architecture, Fashion Design, Marketing & more) (See Figure 5)
4. Sci-Fi (Entertainment, Automotive Design, Aerospace & more) (See Figure 6)
5. Fantasy (Animation, Games, Marketing & more) (See Figure 7)

A close-up of a person's face

Description automatically generated

Digital Emily from the Wikihuman Project (USC Institute for Creative Technologies, 2023) (Figure 3)

A green dinosaur toy

Description automatically generated

Toy Robot Dinosaur (3DWorkBench, 2023) (Figure 4)

A yellow and white table

Description automatically generated

Small Minimalist Table (Ferofluid, 2023) (Figure 5)

A black and silver car

Description automatically generated

Sci-Fi Vehicle (Unity Fan, 2023) (Figure 6)

A stone tower with a window

Description automatically generated

Ruined Round Tower (Dmytro, 2023) (Figure 7)

## Data Gathering

Quantitative data gathering will be employed for the memory footprint of each model. It will be gathered using Unity’s built-in memory profiling system; however, if this proves faulty, a system may be developed on top of this to provide extra memory data. In this investigation, the independent variable is the polygons memory footprint, and the dependent variable is the voxel grids memory footprint. Data will be captured within the table, shown on Figure 8.

|  |  |  |
| --- | --- | --- |
| Model Type | Polygon Memory Footprint (MB) | Voxel Grid Memory Footprint (MB) |
| Realism |  |  |
| Cartoon |  |  |
| Minimalist |  |  |
| Sci-Fi |  |  |
| Fantasy |  |  |

Data Table (Figure 8)

The aesthetical type of each object has been pre-determined before the creation of the object, using tags defined by its creator.

## Data Analysis

The data within the table found on Figure 8 will be analysed based upon the differing memory footprint between the polygonal mesh, and voxel mesh. The statistical test used with this project is a two-tailed paired t-test, with the Null Hypothesis and Alternative Hypothesis used stated below.

**Null Hypothesis:** There is no significant difference between the voxel and polygonal memory footprint taken up by each model. E.g. the calculated standard deviation is equal to zero.

**Alternative Hypothesis**: There is a significant difference between the voxel and polygonal memory footprint taken up by each model. E.g. the calculated standard deviation is not equal to zero.

If the P-value of this project is less than 0.05 (5%), then we may conclude statistical significance and favour the alternative hypothesis. If the P-value is above 0.05, it must be concluded that the variation within memory footprint is due to random chance, and therefore warrant for the Null Hypothesis. It is presumed throughout this t-test that we are following the assumptions of a standard t-test, such that there is homogeneity of variance, and that the distribution is close to normal. There should be homogeneity of variance due to the choice of keeping the same engine for each memory footprint capture (Unity). To ensure a standard distribution which is normal, the distribution curve will be calculated after data has been gathered.

The secondary objective will also be reviewed, as to look for any links between specific 3D visualization genres, and voxel grid size. This analysis will help highlight industries in which voxels may be more RAM-friendly in, which can then be proposed as new research avenues.

## Usefulness of Analysis

If the statistical tests conclude that the voxel grids memory footprint is less than polygon file storage, this will prove meaningful to the future of voxels within 3D visualization.

If however, we find that voxel grids memory footprint is higher than polygon file storage, this will support existing evidence on voxels limitations within memory consumption.

If the Null Hypothesis is found, and memory of voxel grids is comparable with its equivalent polygonal meshes, then this will suggest further research should be taken within the comparison of these two geometric primitives.

## Ethical Considerations

This project performs a statistical test using quantitative data, and as such, there will be no participants. Due to this, there are no ethical considerations to be reviewed.

# Study Execution

As per Unity documentation, an uncompressed mesh takes approximately 6 bytes per triangle, and 32 bytes per vertex.

6t + 32v

Robot Dinosaur has 59,800 triangles and 30,000 vertices.

6\*59800 + 32\*30000

1318800 Bytes (1.3MB)

Although PicaVoxel does not explicitly state the size of each voxel, consulting the code we find there is one byte for the state, one byte for the size, and three bytes for the colour.

6\*voxelCount

After voxelizing the Toy Robot Dinosaur to a grid of 53x92x128 we get 624128 voxels

6\*624128

3744768 Bytes (3.7MB)

* Details of final design.
* Details of implementation.
* The justification of the design is detailed and supported by how smoothly the study goes.
* Excellent documentation and record of progress.

# Outcomes

* Outcomes of project and evaluation
* Analysis of data constructed to demonstrate your understanding of the results.
* Limitations which lead to evaluating outcomes can be considered.

# Discussion/Conclusion

Discussion

* Very briefly mention purpose of study.
* Discuss work and link back to others discussed in the literature review
* Did the outcome support initial ideas? Did it answer the research question? What is the answer to the question? Does your findings differ from others? If so, why?
* Discuss in context of topic area

Project Critique

* Critically appraise work.
* How successful were you in achieving the project aims given in the introduction?
* What problems arose in the project that could not be readily solved in the time available? Why did they happen and what would you do differently?

Further Work

* Where would you take this research next if you had the funding? What questions should be answered next?

Conclusion

* Summary of project and outcomes, taken from all sections.
* Pull findings together. Is there anything that can be concluded? Who will it be useful to? Was it a success?

# References

3DWorkBench. (2023, December 15). *Robot Dinosaur (Toy)*. Retrieved from Sketchfab: https://sketchfab.com/3d-models/robot-dinosaur-toy-8ce1a6e5ce4d43ada896ee8f2d4ab289

Aleksandrov, M., Zlatanova, S., & Heslop, D. J. (2022). Voxelisation and Voxel Management Options in Unity3D. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 13-20.

Alex Evans. (2015, August 12). Learning from failure. *Advances in Real-Time Rendering in Games. MediaMolecule, SIGGRAPH*, Volume 2.

Atomontage. (2023, May 1). *Home: Atomontage*. Retrieved from Atomontage Web Site: https://www.atomontage.com

Baert, J., Lagae, A., & Dutre, P. (2013). Out-of-Core Construction of Sparse Voxel Octrees. *High-Performance Graphics Conference* (pp. 27-32). Anaheim, California: ACM.

Bushberg, J. T., Leidholdt, E. M., Seibert, J. A., & Boone, J. M. (2012). In *The Essential Physics of Medical Imaging Third Edition* (pp. 109-110, 161-162). Philadelphia, PA: Lippincott Williams & Wilkins, a Wolters Kluwers business.

Chen, Q., Liu, G., Li, X., Zhang, Z., & Li, Y. (2017). A corner-point-grid-based voxelization method for the complex geological structure model with folds. *Journal of Visualization*, 875-888.

Dassault Systemes. (2023, December 3). *3D-DOCTOR: Able Software Corporate*. Retrieved from SolidWorks: https://www.solidworks.com/partner-product/3d-doctor

Dmytro, N. (2023, December 15). *Ruined Round Tower*. Retrieved from Sketchfab: https://sketchfab.com/3d-models/ruined-round-tower-debaa8213bd544afaa52432cede8bbce

Dutre, P., Bekaert, P., & Bala, K. (2018). *Advanced Global Illumination.* Boca Raton, Florida: CRC Press.

El-Wajeh, Y. A., Hatton, P. V., & Lee, N. J. (2022). Unreal Engine 5 and Immersive Surgical Training: Translating Advances in Gaming Technology into Extended-Reality Surgical Training Programmers. *British Journal of Surgery*, 470-471.

Epic Games, Inc. (2023, December 15). *Unreal Engine 5.3 brings new potential for media and entertainment projects*. Retrieved from Unreal Engine: https://www.unrealengine.com/

Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Filnet, J., Fillion-Robin, J.-C., Pujol, S., . . . Kikinis, R. (2012). 3D Slicer as an Image Computing Platform for the Quantitative. *Magnetic Resonance Imaging*, 1323-1341.

Ferofluid. (2023, December 15). *Small Minimalist Table*. Retrieved from Sketchfab: https://sketchfab.com/3d-models/small-minimalist-table-f27cf27533fd40d18af35f089267eb87

Funk, E., & Borner, A. (2016). Infinite, Sparse 3D Modelling Volumes. *VISIGRAPP: Computer Vision, Imaging and Computer Graphics Theory and Applications* (pp. 593-605). Rome, Italy: Springer International Publishing.

Gebhardt, S., Payzer, E., Salemann, L., Fettinger, A., Rotenburg, E., & Seher, C. (2009). *Polygons, Point-Clouds, and Voxels, a Comparison of High-Fidelity Terrain Representations.* Environmental Science.

Goslin, A. (2023, December 15). *Unity Sparse Voxel Octrees*. Retrieved from GitHub: https://github.com/xandergos/unity-sparse-voxel-octrees

GROBERT, D. S. (2023). *Implementation of 3D game MakeFarm with infinite terrain model and basic physics in OpenGL API.* Gliwice: Silesian University of Technology.

Informa Tech. (2023, December 3). *Independent Games Festival Finalists & Winners 2021*. Retrieved from Independent Games Festival: https://igf.com

JTVentures. (2023, December 3). *Atomontage: We've solved the problems with 3D that even Google and Meta are struggling with*. Retrieved from JTVentures: https://www.jtventures.cz/

Kao, M., & Wang, P. (2023). *Epic Games Thesis.* Contrary Research.

Kim, K. B., Choi, J., & Wakslak, C. J. (2019). The Image Realism Effect: The Effect of Unrealistic Product Images in Advertising. *Joural of Advertising*, 251-270.

Laine, S., & Karras, T. (2010). *Efficient Sparse Voxel Octrees - Analysis, Extensions and Implementation.* Santa Clara, California: NVIDIA Research.

Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., & Sheikh, Y. (2019). Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM SIGGRAPH* (pp. 1-14). New York, NY, USA: Association for Computing Machinery.

Lopez-Moreno, J., Miraut, D., Circio, G., & Otaduy, M. A. (2017). Sparse GPU Voxelization of Yarn-Level Cloth. *Computer Graphics Forum*, 22-34.

Lorenson, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* (pp. 163-169). New York, N.Y.: ACM.

Ma, M., Buyyounouski, M. K., Vasudevan, V., Xing, L., & Yang, Y. (2019). Dose distribution prediction in isodose feature-preserving voxelizationdomain using deep convolutional neural network. *Medical Physics*, 671-677.

Mason, P. A., Hurt, W. D., Walters, T. J., D'Andrea, J. A., Gajsek, P., Ryan, K. L., . . . Ziriax, J. M. (2000). Effects of frequency, permittivity, and voxel size on predicted specific absorption rate values in biological tissue during electromagnetic-field exposure. *IEEE Transactions on Microwave Theory and Techniques*, 2050-2058.

McNeely, W. A., Puterbaugh, K. D., & Troy, J. J. (2005). Six degree-of-freedom haptic rendering using voxel sampling. *ACM SIGGRAPH* (pp. 42-49). Los Angelos, California: Association for Computing Machinery.

Pan, Y. (2021). *Dynamic Update of Sparse Voxel Octree Based on Morton Code.* West Lafayette, Indiana: ProQuest Dissertation Publishing.

*PicaVoxel About*. (2023, December 15). Retrieved from PicaVoxel: http://picavoxel.com

Quipster99. (2010, February 26). *Unlimited Detail Technology.* Retrieved from Youtube: https://www.youtube.com/

Ragan, E. D., Bowman, D. A., Kopper, R., Stinson, C., Scerbo, S., & McMahan, R. P. (2015). Effects of Field of View and Visual Complexity on Virtual Reality Training Effectiveness for a Visual Scanning Task. *Visualization and Computer Graphics*, 794-807.

Randles , B., Welcher, J., Szabo, T., Jones, B., Elliot, D., & MacAdams, C. (2010). *The Accuracy of Photogrammetry vs. Hands-on Measurement Techniques used in Accident Reconstruction.* Detroit, Michigan, United States: SAE International.

Sekanina, J. (2023). *An Exploration of Algorithms for Real-Time Terrain Destruction.* Brno, Czechia: Masaryk University, Faculty of Informatics.

Siles, B. (2019, June 24). Atomontage Inc.’s Branislav Siles on the Limits of Polygons, the Voxel Future, Streaming, AI and more. (D. Aubrey, Interviewer)

Slicer Community. (2023, December 3). *Slicer: Home Page*. Retrieved from Slicer: https://www.slicer.org/

Telea, A., & Jalba, A. (2011). Voxel-Based Assessment of Printability of 3D Shapes. *Mathematical Morphology and its Applications to Image and Signal Processing* (pp. 393-404). Verbania-Intra, Italy: Springer-Verlag.

TheBat! (2023, May 1). *GeForce 256, GPU Database: Tech Power Up*. Retrieved from Tech Power Up Web site: https://www.techpowerup.com

Tong, X., Stilla, U., & Xu, Y. (2021). Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automtion In Construction*.

Tuxedo Labs. (2023, May 1). *Teardown Modding. Tuxedo Labs*. Retrieved from Teardown game website: https://www.teardowngame.com

Tuxedo Labs. (2023, May 1). *Teardown, Steam*. Retrieved from Steam/Valve Corporation. Websitie: https://store.steampowered.com

Unity Fan. (2023, December 15). *Free Sci-Fi Vehicle 032 F - public domain (CC0)*. Retrieved from Sketchfab: https://sketchfab.com/3d-models/free-sci-fi-vehicle-032-fv-public-domain-cc0-24388eaa5e9647cb8c116afa352b795a

Unity Technologies. (2023, December 15). *Unity Design Visualization*. Retrieved from Unity: https://unity.com

Unreal Engine. (2023, December 15). *Digging Deep: Voxel Plugin 2.0's Next-Gen World Creation Workflows | Inside Unreal*. Retrieved from Youtube: https://www.youtube.com/

USC Institute for Creative Technologies. (2023, December 15). *The WikiHuman Project: Digital Emily*. Retrieved from VGL: https://vgl.ict.usc.edu/Data/DigitalEmily2/

Voxel Plugin. (2023, December 15). *Voxel Plugin Docs*. Retrieved from Voxel Plugin: https://docs.voxelplugin.com/

Voxel-Man. (2023, December 3). *Voxel-Man: Home Page*. Retrieved from Voxel-Man: https://www.voxel-man.com/

Westerdiep, A. (2023, December 15). *Voxelizer*. Retrieved from Drububu: Arjan Westerdiep Portfolio: https://drububu.com

Wilder, M. W. (2015). *An Investigation in Implementing a C++ Voxel Game Engine with Destructible Terrain.* Akron, Ohio: University of Akron.

William, G. (2023, December 15). *PicaVoxel*. Retrieved from GitHub: https://github.com/GarethIW/PicaVoxel

Yuan, T., Peng, X., & Zhang, D. (2018). Direct Rapid Prototyping from Point Cloud Data without Surface Reconstruction. *Computer-Aided Design & Applications* , 390-398.

Zadick, J., Kenwright, B., & Mitchell, K. (2016). Integrating Real-Time Fluid Simulation with a Voxel Engine. *The Computer Games Journal*, 55-64.

# Appendices