A close-up of a logo

Description automatically generated

CMP6200/DIG6200

Individual Undergraduate Project 2024–2025

**A3: Dissertation**

[Document title]

d



Course: Computer Games Technology

Student Name: Khai Ailyan

Student Number: 22130235

Supervisor Name: Jan Krasniewicz

**Overall Note**

**Guidance Sections**: All guidance text is placed in green boxes like this one. Please ensure that you **remove these sections** before submitting your final document.

**Template Usage**: This document serves as a **general template** designed for common final year project purposes.

**Customisation**: It is crucial to tailor this template to fit the specific needs of your individual project. Discuss the structure and content with your **supervisor** to ensure it aligns with your project's unique requirements.

**Assessment Criteria**: Please refer to the marking criteria outlined in the **assessment brief** for more detailed information on how your submission will be evaluated. You do **NOT need** to include all sections to pass this assessment.

Abstract

A summary of the report (100-300 words), which should fully encapsulate the content of the project, while being informative, interesting and contain appropriate quantitative aspects (e.g., results). It should describe the project in one paragraph to follow introduction, method, results and conclusion. An example is provided below.

# With the advent of ever expanding technology, new games and game systems are created, this causes other games/game systems to go obsolete over time as there is less investement by companies into these prior products. And in Going obsolete these games are often forgotten, therefore this project aims to create a pipeline a methodological graphic tool for reverse engineering video games files as a means of games preservation.

Therefore creating the means for future programmers to decompile, reverse engineer and preserver existing obsolete file formats is crucial in preserving video games systems and games for future hardware and for future use. This paper details the step by step process in reverse engineering file formats using static analysis techniques as well as techniques detailed in the graph, alongside an example of two artefacts using this methodology to reverse engineer and reconstruct obsolete file formats, i.e. .Str and .Rws.

The results show the benefit in following this diagram, as the contents of .Str/.dff/.rws files were reverse engineered and converted to contemporary file formats, such as .objs for the .rws/.dff and for the .Str files its contents exported for further review for programmers and reverse engineers.

Acknowledgements

Identifying those from whom assistance has been received. Use discretion in selecting the most relevant people who have directly helped or influenced the project completion.

This Project would not have been nearly as successfully without the help of Jan Kracniewicz my tutor, with 10+ years of digital forensics experience he aided in the understanding of the .str files as well as the acquisition of the compression algorithm used in .str files, furthermore his pushing for better quality writing with the literature review has improved the report substantially, the improvements in question were introductory pieces for the literature reviews theory and review section to ease readers in.

Furthermore, with the aid of greavsey1889’s Visceral Toolkit “Cauldron” aided in the debugging process of the artifacts made, specifically the .Str tool, further external tools include the dragonFF .dff converter, for its comprehensive syntax and understanding of renderware geometry, this was created by Turk645.

# Introduction

This will clearly state the rationale and objectives of the research and contain much of the same information present in the proposal (e.g., problem definition, scope, rationale, aims and objectives). Begin with a brief introduction to provide preparation for the rest of the report, with a clear outline of what was done and the rationale for the work. Much of the information that you have already written will be utilised throughout this section, however it should be specifically tailored to this assessment point.

Start the introduction by answering the question: What is the subject of the project?

This section of the report shows the outline of the scope of the project, the problem the project aims to solve as well as the rationale for why this problem was chosen along with why and how it needs solving. Furthermore the report details the creation of the nessasary artifacts, the processes involved in making them as well as tutor feedback and assistance in fixing and or improving them.

## Problem Definition

A statement of the problem, with its significance and origin. If applicable, make reference to the company or industry that led to the project definition.

The problem in question is games preservation, games are hard to preserve due to the lack of support of old proprietary formats, that is to say in order for games to be preserved, their contents i.e. files need to be compatible with new hardware. Game console companies try to offset this problem with “backwards compatibility” where their new consoles have the necessary hardware requirements to be made compatible with old games.

However, this is only done on games 1 generation less, that is to say, games from Xbox 360 can be played on Xbox One, but cannot be played on Xbox Series X, as a result statistically games will have to be forgotten somewhere along the chronological chain.

To solve this, creating a learning tool to teach people how to reverse these files is crucial, if programmers know how the file works and its syntax said programmers can convert its contents to something more applicable/contemporary, in this paper its .Str,.Rws and .dff files.

## Scope

This section identifies the boundaries of the project, what was included and what was excluded from the final project. This should be justified and underpinned by research.

The scope ought to encapsulate the aim of the project, that is to say an artifact needs to be created to prove the legitimacy of the abstract diagram, therefore only .Str, .Rws and .Dff were chose to be decoded, reverse engineered and converted.  
  
In contrast to initial scope and planning, the including of .Arc files were dropped, due to time constraints, however given how the lack of .Arc files hasn’t contradicted the use of the diagram nor impeded the results/results of the artifacts, this isnt a problem and the scope remains minimal.

## Rationale

Why has the topic been chosen? This may be because of lack of research in the area, to shed more ideas and opinion, in response to a request, (e.g., from a company, organisation or relevant current issue). What benefits can be identified from completing the project? This should be more than personal interest—you should be able to identify a company, organisation or other defined group that will benefit from the work.

There is little research in reverse engineering and games preservation, especially research in game file syntax and decoding, therefore my rationale for this project was for the increase in viable research in this area as well as to propagate practices that would enable more games preservation.

With the use of the artifact, programmers would find it easier to reverse engineer old proprietary file formats to then preserve, remaster or modify games so that way they do not get forgotten or die out.

Furthermore this is necessary due to more and more game systems becoming more and more obsolete as we go from ps3 to ps4 to ps5, the ps3 has become obsolete and support has been dropped/discontinued

## Project Aim and Objectives

There should be a brief and precise statement of overall aim—what is intended to be attained? There should follow a list, using bullet points, of objectives—the completion of which will lead to the attainment of the aim. The objectives are developed from the aim and can be viewed as incremental stages in the attainment of the aim(s). Bloom’s Taxonomy is useful in writing these objectives (see Moodle site).

The aim of the project is to demonstrate and create a learning diagram that aids in game file reverse engineering, furthermore creating artifacts using the methodology outlined in the diagram is done to demonstrate its effectiveness.

Objectives include:

- Construct rough theories/steps on how to reverse engineer game files, I.e through reading offsets, reasonable numbers, observation of repeatable patterns in data, understanding the header file where it starts and ends.

- Secondary Research on .Str files, what are they, who made them, what are they for etc

- Primary Research, going into the .Str via a hex editor, researching the syntax itself by documenting key offsets and behaviours etc

- Find the algorithm required for decompression, then decompress the .Strs contents

- Iterate through each .Str file and export each file

- scan for Geometry files I.e .rws and .dff

- Extract geometry from files, export as .objs

- Parse .Objs into 3d modelling software or game engines to validate the correctness of .Obj and amylase graphical artifacts

## Background Information

A further section of background information will depend on the topic area of the project, but could include hypotheses and theory, which are to be tested in the course of undertaking the project. This is an optional subsection but may be useful in defining the contextual information.

The File syntax of a given file has commonalities between files even if said files are completely different to the originial file, such as examples would include most files having a header file and that header file being at the top of the file. Said Header file has information relating to the file irrespective of what particular file it is, such information could be, file offsets, amount of files insided the file, unique identifiers and ids, the size of the file, dictionarys, flags such as the case with textures and so much more.

Common file/code reverse engineering methodologies includes the use of static and dynamic analysis, these practices involve reading the data itself raw to discern patteran, where they vary is WHEN these are performed, dynamic analysis is observing the file and its contents when its being run by an exterior program whilst static analysis is just reading the file as it is in a hex editor or equivolent, this form of analysis will be the bulk of the project and HexEdit online will be the tool that is used most.

# Literature Review

## Themes

## File Syntax - denotes the rules and structures that data is stored as in a file and is a fundamental to reverse engineering files, an example of which would be .Obj where all of the vertices are denoted with “v” and the vertex normals “vn”, the relevance to the project being file syntax is needed because the .Str and .Arc files structure is unknown, and needs to be known to then create a file converter. Context drives what type of abstract data structures and structures the file uses in its syntax, for instance 32 bit architectures such as operating windows operating systems that are 32 bit, they will have smaller integer sizes compared to 64 bit operating systems.

## Reverse Engineering - Reproducing a proprietary product (i.e files, code and data formats) via the understanding of syntax, this involves understanding the File syntax of the specific file to then reverse engineer OR to decompiler the software code in order to convert it to another language or for a newer system. The ethics and legality is purely contextual, due to many aspects needing to be present for the company of the proprietary software to care such as, the age of the software, commercial impact, whether its used commercially, whether it infringes on copyright or not ( this is because if a file reader is made, are said files that are read then used for future use, that would cause the company to actually care).

## Games preservation - Preserving and maintaining games either through physical maintenance and or protection OR through the preservation of the games software, code and or files. In this context it is converting their respective contents to more contemporary file formats to make it so future operating systems and systems in general can use / play those games.

## ADT (Abstract Data Type) Syntax - Bytes in code with specific structures and rules created for specific purposes, e.g List, contiguous block of memory containing a list of items of a specific type, does not need to be redefined to be bigger than what it already is.

## Compression Algorithms - Algorithms created to make data and files smaller than they actually are for transportation between devices or storage capacity optimisation, compressed items can then be uncompressed to their original size for further use,

## Review of Literature

**Games Preservation:** Games preservation has existed as long as games have needed to be stored, as time progressed the games software separate to its hardware needed to be maintained (Harkai Istvan 2022)(Haydock Christopher 2018) as well as the hardware (Dany Guay-belanger, may 2021)(Widget, Megan A, 2009) , moreover the importance of games preservation cannot be understated due to its cultural impact (Todd Benjamin C 2019)(Henry Lowood 2009) as games and games systems are being threatened with obsolescence such as the discontinuation of ps3. Moreover, the importance is shown with people's desire to preserve games, but can't due to legal restraints (Bachel, Alasdair and matthew barr 2014). Most games preservation today is hardware based and comes in the form of “Archiving” (Camila johansson 2023). There lacks research on a software approach to Games preservation, most of the research refers to hardware and the minimal research that refers to software rarely accounts for the legal issues, especially when it comes to international law, not just america.

**Reverse Engineering:** Reverse Engineering's existence dates to the first thing ever made. However in the context of software, software reverse engineering began in 1975 with the first text editor WYLBUR (Kathi Hogshead Davis, Peter H Aiken, 2000), features of reverse engineering include disassembler, debugger, etc (Abigail A 2021), whilst in contrast (Alessandro Mantovani , Simone Aonzo, Yanick Fratanatonio), refer to the techniques of reverse engineering through their behavior and performance in time. (Anand Gadwal 2011) talks in detail on the step by step process of reversing software from “context parsing” through to “Design Reconstructing Phase”, (Ramandeep singh 2013) refers to the types of reverse engineering such as “data” and “code” reverse engineering, detailing the different types depending on what is being reversed. There needs to be a greater focus on breaking down the individual processes, with example of high level reverse engineering in software, most of the research refers to very low level or do break down the steps enough, as well as on Data Reverse Engineering specifically.

**Abstract Data Types:** Abstract Data types or ADT’s have existed since the 70’s created by (Barbara Liskov, 1974), a standard example of an adt being a “Stack” (John V Guttag 1977) which contains items of a certain data type and allows for the storage of multiple items in a first in last out order of execution. In Contrast to ADO’s (Abstract Data objects) which are more raw and have little functionality (Jean Francois, Rainer Koschke 2000), Vectors in games would be an ADO, another ADT a Queue is commonly used for AI behavior in games (John v Guttag June 1977). Limitations are that the general research goes back to the 60’s and 70’s and the majority of the ADT’s that are common place are incredibly old, whilst the research does show the process of them being made and so could be replicated, they are dated.

**Compression:** Data Compression’s origins come from (Robert Fano, Claude Shannon 1949), then (Huffman Coding 1952) with “Huffman Coding” and (Lempel-ziz-welch) with LZW, Different compression algorithms are used for specific files and data structures (Tito Waluyo Puroboyo 20,17). This in turn leads to the most common use of compression algorithms, compressing text (Amandeep Singh Sidhy 2014). Custom Compression algorithms are created for specific data structures and object types like model files (Mustafa, Ammar 2017; JingLiang Ping 2005; siddeeq mohammed m, rodrigues marcos 2016), said algorithms are specialized to be better optimized for 3d objects. There lacks research into more proprietary compression algorithms, companies often make their own, as a result these compression algorithms that have been documented and research are often non-applicable for reverse engineering and thus for games preservation.

**File Syntax:** File syntax originated in (Russel Kirsch 1957) via a picture of his son, and from that the first file syntax was created alongside it having its own rule set. File syntax varies between different types of files due to each type of file having a different purpose (Kauthar Abdulazeez, sohit agarwal 2021; Samiya Khan;Mansaf Allam 2019), coupled with different structures inside different files (Mike Folk 2003) each type of file is unique in its rules, structure and therefore purpose. The limitations are that there are little research documents/journals that go into proprietary formats and more look into general syntax and older simpler formats like text or image files.

### **Theory**

**Games Preservation:** Games Preservation includes the physical storage of games and their associated hardware (Haydock, Christopher april 2018) (Dany Guay-belanger may 2021), it includes the maintenance of games/games systems themselves through maintaining hardware or Copying contents of games to newer external storage due to bit rot (Haydock, Christopher april 2018), bit rot being when data is stored in a harddrive for too long, the data deteriorates. Also includes the maintenance of software via reverse engineering the games/games systems software to accommodate newer hardware. (Johansson, spring semester 2023), so that it can be played on contemporary systems. This is the approach used in the project, software maintenance is the method of which the game/game files will be preserved, not hardware maintenances.

**Reverse Engineering:** Reverse Engineering in a general sense is when a product/system has been understood mostly in its entirety and recreated. However in this context, Reverse Engineering consists of understanding the syntax of code, a file, hardware architecture/structure (Aremo, Abigail, 2021) etc and trying to recreate it, it's about understanding and recreation. In this context its about understanding the header file of a file, reading the values inside of the header to ascertain data about the file, locating offsets in the file to determine where to go to in the file and where to stop, reading the header file to determine key features of the file such as, the amount of files, of what memory policy/allocation, what structures exist within the file so forth. From this, creating code ( an artifact) to then read the file and write it or its contents into more understandable/contemporary sub-files, if applicable.

**Abstract Data Types:** Abstract Data types allow for data to be stored with specific functionality depending on what ADT it is, a list allows for continuous addition of items of a type, and once fully allocates more memory to the list for further use, arrays are finite and need redeclaring if the array is to full, sets are immutable and so the values inside cannot be modified directly. For this project ADT’s are necessary due to a file effectively being a contiguous block of memory that functions like an array/list, therefore reading line by line and assigning to an array or list variable ADT is crucial in creating an artifact to read/write files.

**Compression:** Compression is the practice of taking in data and making it smaller for the sake of storage and transport. Compression is either lossy, data is lost in compression , first lossy algorithm (ahmed 1974), or is lossless and so data is not lost on compression, the first and commonly used LZW (Lempel-ziv-welch). Depending on the type of data certain compression algorithms are more ideal e.g audio, model files, textures etc (Purboyo, prasasti 2017; Sidhu 2014), this is especially true with model files that have specific compression algorithms ideal for triangles (Mustafa, Ammar, 2017; Rodrigues 2016; Ping 2005).

**File syntax:** File syntax is the structures and rules of the file itself normally shown through the use of a “header file”, which is at the top of the file, file syntax is important as by understanding its structures and rules a programmer can read its data correctly without mishap. Understanding file syntax is crucial for extracting models, audio files, textures etc from games in order to reverse engineer them as the programmer is then able to determine where they are in the file, there size etc.

## Summary

## Games Preservation: Games preservation in the form of hardware preservation is seemingly a problem that has already been fixed, in terms of there is a legal viable solution, however from a software maintenance approach there seems to still be a lack of information and problems with legality, making the project more important due to a lack of emphasis on both software maintenance as well as a more legal avenue.

## Reverse Engineering: The research shows the methodologies and types of reverse engineering, as a result there is a detailed blueprint as to what type of reverse engineering is to be used and the steps to which software can be reversed. Furthermore, for the project, it is data reverse engineering, using text editing tools, reading offsets in memory to memory and identify key structures and repetition in the files.

## Abstract Data Types: Abstract data types are shown to be useful containers for data with additional functionality, artifacts will have to be read as arrays to process line by line characters for binary analysis and data reverse engineering File syntax: File syntax research has shown the importance of header files due to their relation to the rest of the file, moreover file syntax is different for each file type, given this information the project going forward requires better understanding of custom files due to their custom syntax as shown by their respective header files, this will be crucial in decoding the files.

## Compression: The research can be summarized as each data type and or file having its own ideal compression algorithm designed to deal with that specific file, this is important because companies tend to design their own file systems and as a result have their own compression algorithms to better optimise space efficiency. Said algorithm needs to be understood before reverse engineering

# Method and Implementation

The Method And implementation covers both the development life cycle of the artefacts as well as the use of the learning diagram itself in the report, as the artefacts were created using the methodology the diagram shows, therefore validating the diagrams use, wh

## Development Life Cycle

### General devlopment

This covers the artifacts that were made, their general development, tutor feedback, basically an overview of the planning, creation of the artefacts in a macro sense, more detailed work throughs and implementations of the graphs learning material will be illustrated later.

4 Artifacts were created, the .str excracter, the .rws converter , the .dff converter and finally the actual aim and purpose of the project the abstract diagram detailing how to reverse engineer files for the future.

The artifacts development was very linear, without any snags or delays. The artifact (executable programs) originally extracted sections from .str.

Review with supervisor who is well versed in file syntax and decompliation, pushed for the contents to be uncompressed BEFORE the files were extracted.

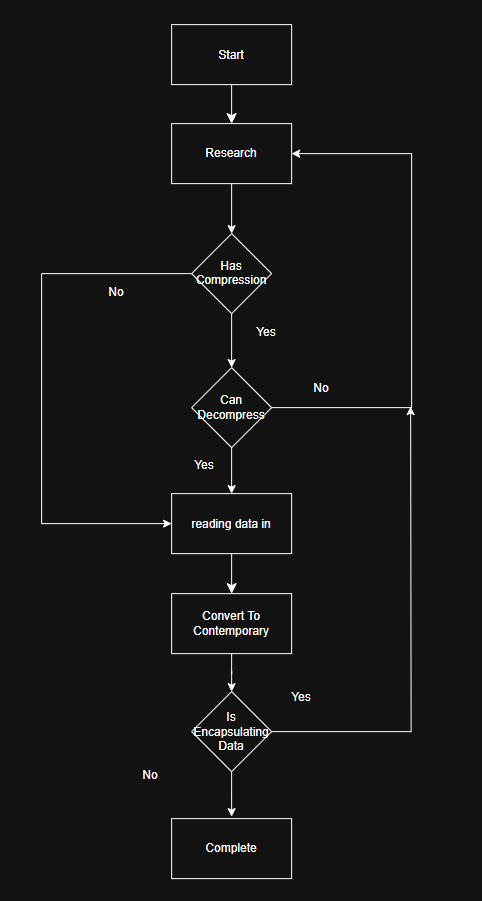
from that further development lead to the sections being uncompressed via the implementation of the “RefPack decompression” algorithm, this enabled files to be successfully extracted as the contents of the files were now actualized, furthermore the output of the sections were compared with similar software that does the similar file viewering, and in contrast the files were identical, meaning this was successful.

Further development came in then reverse engineering the .rws files contained in the .str to a more suitable format .obj, this in part was due to the fact that there is no 3d model software that will open and or view the .rws as a model, instead RW analyzse was used to view the file as a hierachy, which in turn enabled the syntax of the .rws file to be laid bare, this enabled me to create my own file converter due to the syntax being expressed.

The development of the .rws converter was quick however during development there were problemsi wth the normals that were calculated from the vertices, with guidance from my secondary supervioser, the normals were calculated as vertex normalsa not face normals.

This lead to the successful creation of .objs.

### The learning tool and its implementation/proof of concept

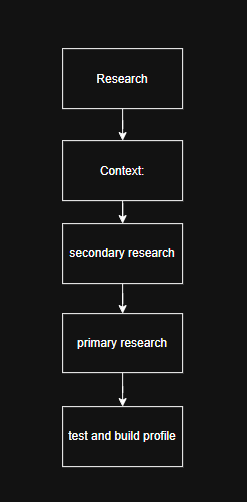


**Figure 1. Shows the Learning tool at its macro-level**

The diagram above is the macro process of which the artefacts were created as well as being the main focus of this project I.e the learning diagram, this diagram enables the aim of the project to be fulfilled as it allows for the successful reverse engineering and conversion of video game assets as a means of games preservation, therefore fulfilling the aim.

As stated prior though, this is a “macro process”, meaning this is applied to the whole process as a whole, it is very holistic, the individiual parts of the diagram will be discussed later in the report, each of their specific detailings and processess.

Generally the stages are researching the file for its syntax/format, determining if it is compressed, then uncompressing its context, converting the file and or its contents to contemporary file types as a means of games preservation, do the files encapsulate more data/files, if so then start the process again to actually preserve the contents of the game data I.e models, sounds, geometry, textures etc.

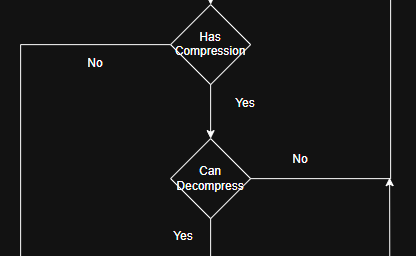


**Figure 2 shows the elaboration of the “Research” node in figure 1**

For the research part of the project as well as the artefacts protaining to .str/.dff/.rws files, both secondary and primary research are needed in order to build a profile of the syntax pertaining to the .str/.dff/.rws files, what this means is conducting secondary research in the form of forumn and wiki searching for the syntax/format of the given file formats to then build a profile, this is done first because there might already be a solution/format available to copy from, why reinvent the wheel.

Primary research will be the bulk of the research depending on how little secondary research there is, primary research involves opening the files into a hexeditor, and reading the data expressed as bytes, this is done in order to read patterns in the data to discern their purpose as well as identifying key offsets and flags that will lead to building a pofile of its syntax.

Specifically the header file is to found, determine where it ends, then read data inside to assertain information about the file, regardless of file this ususally includes, an identifier as to what the file is, the size of the file, dictionary/offset tables that lead to other places in the file, offsets and flags. For the .str file the identifier is “SToc”, then 4 proceeding bytes of what seems to be padding, then an integer 8 that denotes how many sections there are in the file, followed by the integer 8 hardware identifer flag, a big endian 32 integer denoting the start of the section table, then after that its just each section of which each is 24 bytes denoting the compressed and uncompressed sizes of the sections.



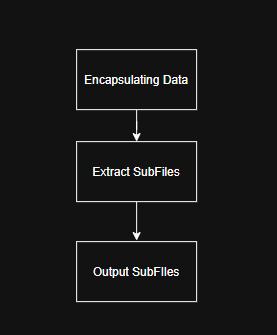
**Figure 3 denotes the compression stage in the reverse engineering pipeline**

This section of the diagram refers to the compression used by companies to reduce the size of their game files to reduce the size of their games on hardware, the compression of a file can be assertained in a variety of ways, the quickest yet unreliable way it so read the magic numbers I.e key byte identifiers of sections of the file, to determine the compression algorithm, this is unreliable because if there isnt alread a resource for this, as a result the programmer cannot do anything with this magic number. However if there is, the programmer has a link back to the originial compression algorithm and can replicate it.

The second way is to read the entropy of the file to determine what kind of compression it uses, as well as experimenting with already existing compression algorithms and types of compression algorithms on the file, then measuring the entropy after the fact, moreover there are only so many types of compression algorithms, huffman, lzw, lx77, deflate, lzma, so experimenting isnt too hard.

Within the context of the project the .Str files were compressed and using something called the “RefPack Algorithm” denoted by the magic number 0x10FB, it’s a type of compression algorithm that uses key size itentifiers to dictate how much it compresses per byte section, it is very similar to LZ77 or LZSS, in which it scans the data and replaces it with a reference depending on if certain flags are met I.e 0x80, 0x40, 0x20 then default.

The .rws and .dff were contained in the .str files, so they needed no further decompression.



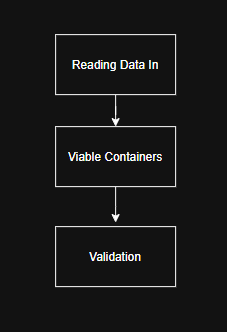
**Figure 4 denotes the encapsulation part of the diagram in more detail**

What this section refers to is the question of does the file encapsulate more files in itself, an example of which would be .Str, the .Str file is a ps3 stream file created by EA, it in itself has no use other than to store more files and to have its own compression algorithm on top of it, the actual files inside of the .str are what matters, as a result the .str file encapsulates more data.

This needs to be known by the reverse engineer because if it encapuslates more files inside of it, those files need to be reverse engineered to now, as a means of game preservation, just extracting the sections and files of the .str does nothing.

Moreover this is shown in the project as well, the .Str files contained sub files inside, those were exported out by means of reading the offsets of each section in the .Str files, and then exporting start to end in each section, this resulted in each of the files of the .str e.g textures, models, binary space partition files, sounds etc, to be successfully exported.

Then from that, the .rws and .dff could then be reverse engineered, of which they were, but that’s later.



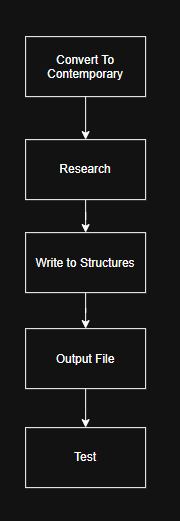
**Figure 5 shows the Data reading section, relates to how data is stored**

This section refers to how the data is then processed once it is read, assuming the file syntax is understood and a profile is built for the given files, the data then needs to be put into viable data containers or ADT’s.

An example would be with .rws and .dff files. They contain geometry and so when converting them to contemporary geometry/model files such as .Obj files, there contents need to be read as Vector3’s and Vector2’s, which are viable containers for their xyz values I.e vertices and normals, then their contents need to be stored in row like ADT structure such as a list or array.

The data can be validated by comparing against multiple different instances of .rws and .dff files, if the size of the file is somewhat proportional to the amount of vertices and uvs, and the values themselves are “reasonable”, which is in reference to the values not being really big or small. This would then

Therefore in the process of reformatting the data, the data needs to be read in appropriate containers and stored in appropriate containers, whether that be Vectors, lists, arrays or other array/row like structures.

****

**Figure 6 shows the final stage of the reverse engineering pipeline**

The most important stage of the diagram is the conversion stage, once the profile has been created and code has been read to transfer the contents into viable containers, the last step is to determine what file format the formatted data will be outputted to.

This involves research into other similar file types to determine the best fit, for example .rws and .dff files are geometry files that use triangle strips, so we can convert them to triangle faces then parse that data into .objs, objs are used because they are very simple to use and contain vertices, uvs and normals which are all of the nesassary values inside .rws/.dff files.

From the research the Formatted data needs to follow its syntax, but because these are being converted to propriatary formats, this part is far easier, due to the ease and access of the syntax of newer file formats such as .obj, in the case of .obj the vertices are stored as “v X Y Z”, the uvs “vt X Y”, the normals “vn X Y Z” and the face indexes are “f vert1/uv1/normal1 vert2/uv2/normal2 vert3/uv3/normal3”.

This section describes the development of the artefact, including design and implementation. This should reflect the progress made in the implementation along with feedback from your supervisor. This section is the first section of the assessment that is completely new to the report.

Remember that success of the project depends upon careful selection of appropriate method (e.g., design, model). A good method increases the validity and reliability of the outcomes. Depending on the type of project, it should cover the choice of apparatus, equipment, and software utilised. It should be possible for another researcher to repeat any experimental or research aspects of the project and expect to obtain the same data.

**In practice this section can be quite large and may often be broken into a number of additional sections, e.g., Methodology, Design, Implementation.** For practical, experimental and technical projects, there may be sections for calculations and analysis for parameterisation or model tuning as needed.

All details should be clearly presented no matter what section structure you have used.

# Evaluation

This section is the second section of the assessment that is completely new to the report.

The evaluation section should provide testing of the artefact and overall project. This will express ideas in answer any research question. Depending on the evaluation chosen, a variety of possible layouts may result. Nonetheless, it is good practice to consider the evaluation section to be divided into two subsections based on the experimental design and the outcomes.

## Evaluation Methodology

Evaluation/Experimental methodology: Here you describe the selected approach to evaluating your design, as well as the motivation for the approach. If this is a standard way of measuring particular phenomena, then it can be motivated through citation. The experimental design of your evaluation will include various subsections possibly including:

**NB: The following sub-subsections (i.e., 4.1.1 through 4.1.3) may not be relevant to your specific project topics, so you should discuss the sections with your supervisor to tailor this to your needs.**

### Evaluation Metrics

The specific metrics being used to assess success.

Success Metrics include:

- .Str reading/input, can .str files be inputted and have its contents successfully read i.e files are valid, files are made valid through being able to be successfully open through other software

- .Rws reading/input, can .rws files be inputted and have its contents converted to .obj successfully I.e can said .objs be loaded in 3d viewing/editing software

- .Obj creation, Objs including vertices, Uvs, Normals and face indices

- Speed of application, when the program is run, how long will that take in comparison to other software like cauldron.

- replicability, the diagram ought to reflect the development progress of the .Str/.Rws artifacts, as that is the aim of the project, therefore from that, future files that are to be reverse engineered need to be able to replicate this progress.

### Baseline systems

Systems under analysis or Baseline systems: The designs being tested apart from the one proposed in the method section. Note that these may also be variants of the proposed approach.

The current baselines systems are segragated and process .Str and .rws/.dff files. That being greavesy1889 Visceral toolkit that processes .Str, the processing of .Str are very quick and all of the subfiles are expressed and downloadable, furthermore there is texture support i.e converts .txd files to .dds

### Dataset

A collection of data that is used to provide reliable consistency in comparative assessments across systems. Depending on your chosen project **this may or may not be relevant.**

**Performance Statistics:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File Amount / Sizes | Project Artifacts  (.str, .rws and .dff) | Visceral Toolkit Cauldron  (only .str) | Python Plugin  (only .rws and .dff) |  |
| 3 files:  - zone01.str (12768 kb)  - zone02.str (9666 kb)  - zone03.str ( 8748 kb) | 72,893 ms  Medal of homer | 2000ms | 1000ms |  |
| 10 files:  - zone01.str (822 kb)  - zone02.str (1510 kb)  - zone03.str(1114 kb)  - zone04.str(668 kb)  - zone05.str(230 bk)  -zone06.str(572 kb)  -zone07.str( 304 kb)  -zone08.str(612 kb)  -zone20.str(446 kb)  -zone22.str(492 kb) | 47,972 ms  Mob rules | 2000ms | 1000ms |  |
| -zone01.str (3298 kb)  -zone02.str(1846 kb)  -zone04.str(2872 kb)  -zone05.str(246 kb) | 25,671 ms  Never quest | 1000ms | 1000ms |  |
| -zone01.str(4074 kb)  -zone02.str(120kb)  -zone03.str(796kb)  -zone04.str (2906kb)  -zone05.str(10 kb)  -zone06.str(10 kb) | 73546 ms  rhymes | 1000ms | 1000ms |  |
| -zone01.str(5574 kb)  -zone02.str(864 kb)  -zone04.str(642 kb) | 23732 ms  Meet thy player | 1000ms | 1000ms |  |
| -zone01.str(10 kb)  -zone02.str(3138 kb)  -zone03.str(2798 kb)  -zone04.str(2228 kb) | 32390 ms  Day of dolphin | 1000ms | 1000ms |  |
|  |  |  |  |  |

## Results

**This section is mandatory.**

Here you will describe the detailed measurements of your system. Which trends appear? Which design performed best across which evaluations? If you have tables or figures that show the performance of your design (and possibly others) refer to these in the text as you explain the output. You may also wish to provide exemplar outputs of the design, which demonstrate the performance of your system, alongside a discussion of the result in the text.

### .Str Input/Output:

.Str files can be put into an “AllAssets” Folder and the .Str files are read, “desectioned”, uncompressed and the files extracted in bulk. Each file has its file extension preserved, each file can then be processed individually depending on context I.e xmls are preserved as xmls and can be read as xmls, meaning files are successfully preserved.

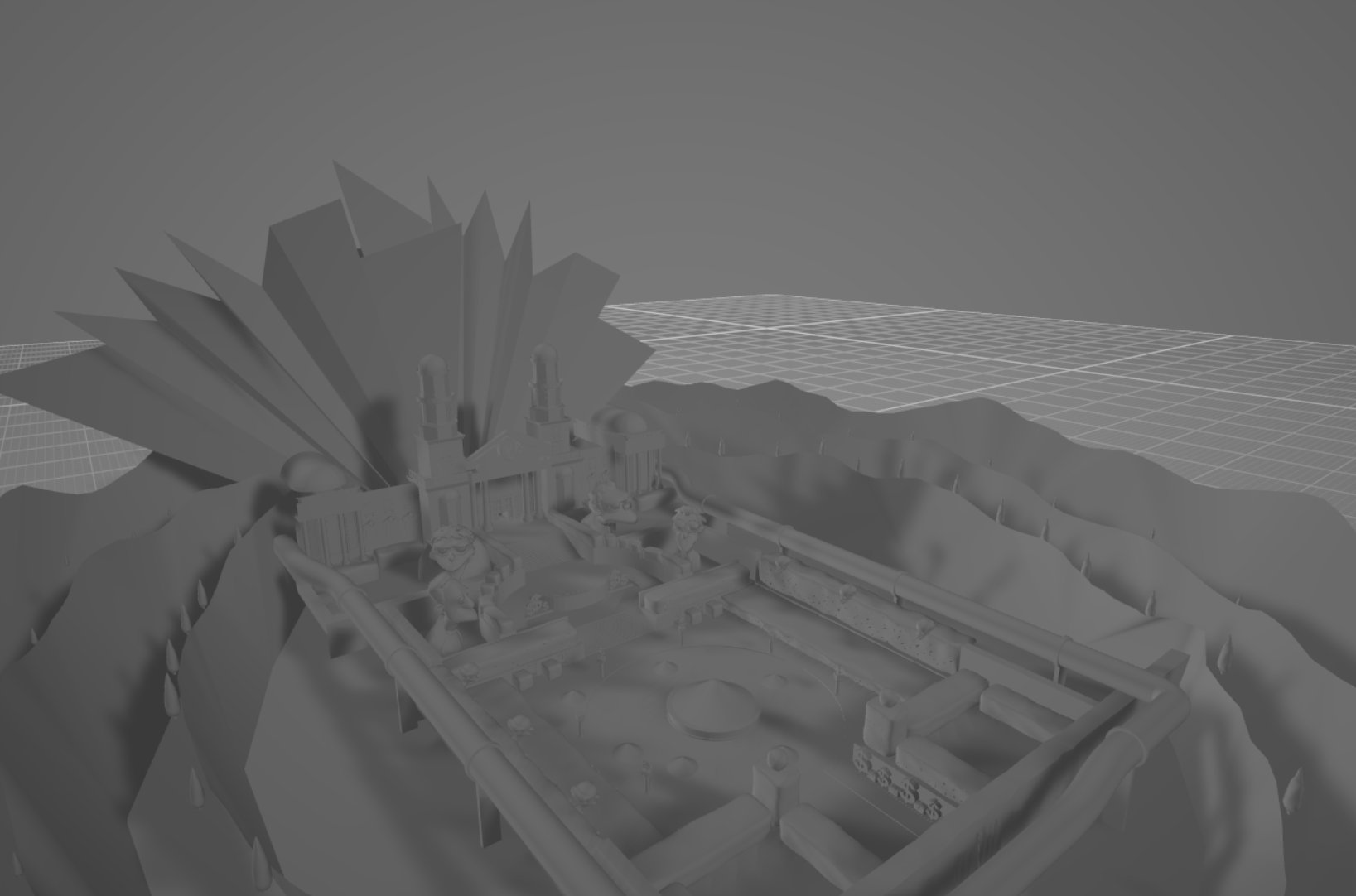
### .Rws Input/Output:

.Rws files are automatically converted to .Obj file formats instantly after .Str files are processed, this ensures user usability, .Rws have their triangle strips converted to face geometry, each of its indicies, vertices and Uv’s are transcribed into text with the correct prefixes e.g “v” for verts, “vt” for Uvs, which are then converted to .Objs, more over Normals are calculated instead of pulled from .Rws files, because .Rws do not contain normals.

### .Objs Compatabilities

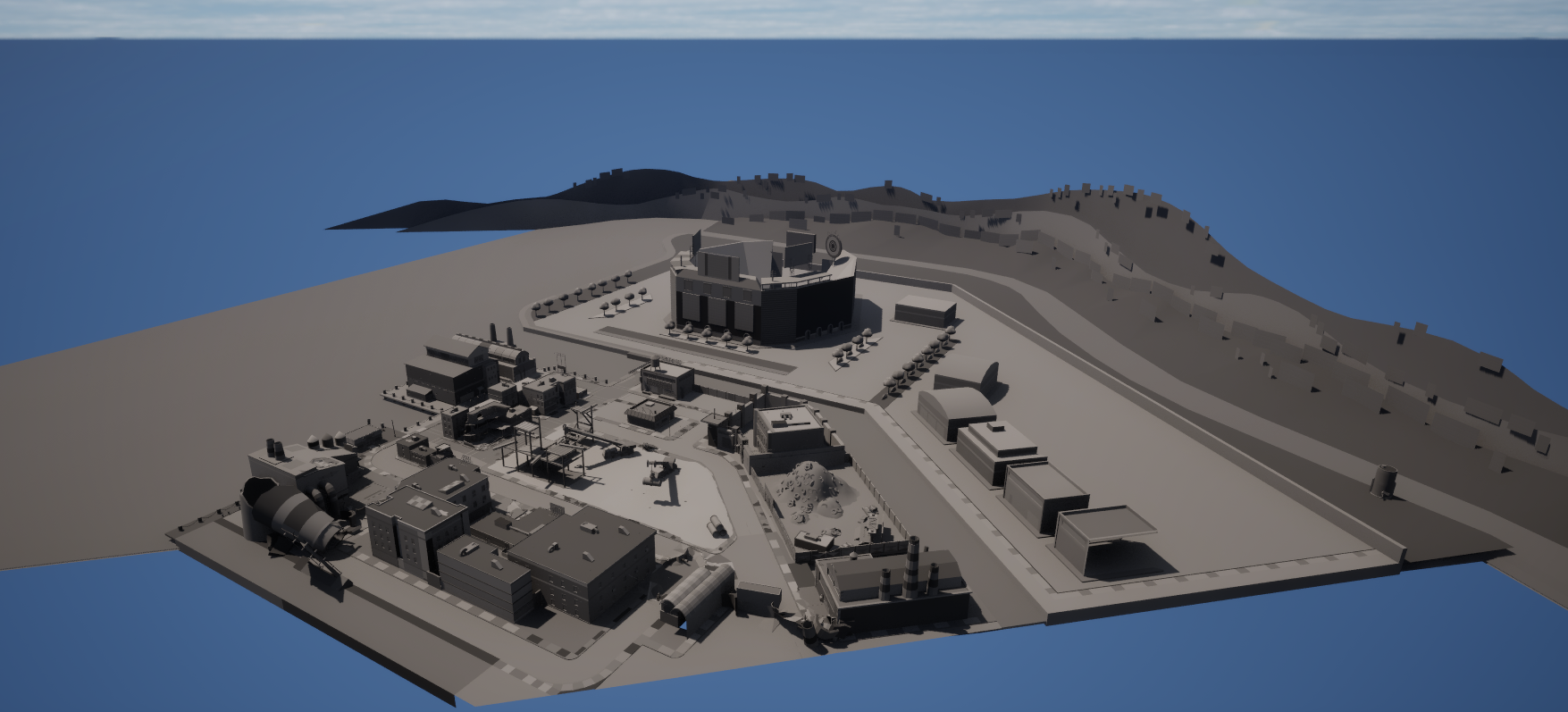
The Final output are a series of .Objs, each .Objs have the correct Vertices in the form of Vs, UVs in the form of Vts and correctly calculated vertex normals in the form of Vns. Each .Objs can be put into .Obj viewer, maya, blender as well as other 3d model rendering software, each vertex, face, edge etc of the geometry can be editied in 3d modelling software.

### Screenshot Proofs



**Figure 7 shows the succesfull creation of the first stage of the second to last level of the simpsons game, of which were .rws prior**

  
  
**Figure 8 shows a statue prop in the never quest level of the simpsons game, of which was a .rws model prior**



**Figure 9 shows the entire of the colossal doughnut fight in its entirety, of which were .rws models prior**

## Discussion

This is a crucial section of the report and should be explored in great depth. The results from the previous subsection are here explained with consideration to the context of the project. This is the area in which you can confirm similarity or difference between trends that appear in your research with that of others that you have discussed in your literature review. You may also hypothesize why you believe certain outputs/phenomena have occurred. This is a deeper analysis in which you piece apart the results to determine the underlying causes of the recorded output.

For business and management related projects, the presentation of findings may be integrated within discussion sections. Limitations of the chosen methods should be identified and ways to overcome them suggested. If compromises have to be accepted, for example in time and cost. Such limitations and problems should be identified together with how they are to be overcome and/or the compromises that will have had to be made.

Depending on the nature of the project, and particularly with certain business topics for which the main outcomes are recommendations on various management related aspects, the results and discussion chapters may be integrated within chapter(s) of findings covering the relevant project objectives. In this case this chapter could be entitled Recommendations.

The proposed File decoders that were implemented were statistically slower than prior file decoder / openers, this is in part due to the reports artifact performing more computations per cpu cycle when processing the files, for instance the python file opener only looks for the specific renderware geometry header then it begins to process the files contents, whilst the artefacts that were created did far more, such as, reading all renderware headers for future work like texture extracting, .Str opening in conjunction to .rws and .dff converting coupled with decompressing all of the .str files contents.

Furthermore the outputs for each of the .objs contain vertex normals in the from of “vn “ rows, in contrast the .rws converter “turk” created, does not have this functionaility due to that persons artefact merely pulling already existing data rather than actually trying to create new data for a more feature rich product, that is to say the artefact ports .dff and .rws to blender, whilst the project artefacts will recalculate the vertex normals for .obj creation and from that they can be put into any 3d software opener with all the functionaility intact I.e verts, uvs and normals.

The results work and are valid due to the conversion setup in the project artefact, as stated prior the geometry type of the .rws and .dff files are triangle strips, said triangle strips cannot be made into .obj unless converted to triangle faces, this is done in both the project artefact and “turks” python artefact, where they differ as stated prior is with the output, but the project artefact converts them to triangle faces then uses those faces to construct face normals then vertex normals.

This is what led to the reliable flatshading as seen in the unreal engine output and the inbuilt object viewer designed by windows. Outputs from the project artefact are therefore correctly lit due to this calculation and application.

The results were also successfully created through proper understanding of file syntax, as shown in the learning tool and methodology, the

# Conclusions

The conclusions should be a short summary of the important results and findings arising from the results and discussion. It is important to ensure that the conclusions address the original project objectives and reflect the main discussion.

You should **not include any new information or discussion** in this section.

The important take aways from this project are, the successful reverse engineering and preservation of the assets provided as well as the demonstration of the usefulnes of the learning diagram. That is to say that the .Str files are decompressed and its subfiles extracted, the .rws and .dff model files were successfully converted to .obj, those .objs can then be put into 3d rendering software successfully. And from this the methodologies used in the diagram were the ones used in creating the artefacts that led to such a big success.

As a result the aim of the project was made successful as the game assets can infact be put into 3d rendering software like game engines, and from that those games can then be “remastered” using those assets, therefore the diagram acts as a useful tool for the purposes of preserving games and their contents.

# Recommendations for future work

Many projects follow on from previous work and owing to time constraints and the generation of ideas whilst undertaking the work, lead on to the possibility of future work. These recommendations should be summarised briefly.

The future of this line of work will be in creating new methodologies for file decompilation, the diagram acts as a general means of file understanding and decompilcation, but it is abstract in nature, therefore more additions can come in specificicity.

Therefore additions to the diagram for the future will be made, furthermore the diagram is more useful for games/games files for the sake of game/file preservation, therefore if future reverse engineers want to use this for files that are outside of games, this can still be done, but again requires more specificity.

Also all of the shown examples of the learning diagrams successes I.e successful obj creation, are purely geometric, the plan moving forwards is to have texture converting, sound converting, video converting etc, not just geometry, this is in part due to the aim of the project, games preservation, geometric preservation is good for games preservation, but it is not everything in a game.

# References

It is essential that you reference and cite your work correctly. You should ensure all aspects of the project are underpinned by appropriate research cited in the body of the report. Full, correct and appropriate referencing of all sources used in undertaking the project is an essential requirement of a good report and necessary to avoid allegations of plagiarism. Harvard referencing must be used.

Use of, and reference to, a selection of relevant texts, journals and appropriate internet sources should enhance your work, reinforce the validity of your results and findings and demonstrate that you are familiar with accepted knowledge and thinking in the subject area. Reference sources should be selected to be comprehensive, appropriate and current. They should be well integrated with the text and cited in accordance with the University's standard (Harvard) method.

The **[library site provides extensive referencing information](https://www.bcu.ac.uk/library/services-and-support/referencing)**.

**NB: Any use of sources that are not cited or cited incorrectly, may lead to allegations of plagiarism.**

# Bibliography

A bibliography is a list of relevant source texts you have used to undertake the project but not directly cited in the report, in Harvard format.

# Appendices

Appendices, which should have short titles, are separate documents appended at the end of the report. Only include appendices if they are necessary to explain particular details to understand the main report. **Generally, work in an appendix gains no marks directly.**

**You should include a copy of your Gantt chart in the Appendix.**

A report should flow freely and be easy to read. Figures, tables and images should support the content of the report not impinge on it. Do not place any information in the Appendices that can be located using a reference. The Appendix is not is not an opportunity to make a report look thicker. Do not include information that was not referred to in the report. Appendices do not have an introduction and begin with Appendix A if there are more than one. Otherwise, if there is only one, this is called ‘Appendix’. Appendices may include:

• Detailed statistics

• Computer code

• Large diagrams

• Complex graphs and tables

## Appendix A: Dissertation Style and Conventions

The report should be written in your own words and should not contain extended extracts from the work of others. It is possible to use direct quotes, but these must not account for more than 10% of your report. Direct quotes should be identified by using inverted commas and should be appropriately referenced. Additional resources to assist you with referencing can be found on the intranet homepage under Info Links.

The Faculty standard for degree project reports is similar to papers in technical/professional journals. Examples can be found by referring to journals in your field of study.

Producing a readable account requires a logical structure to lead the reader from one discussion point to the next and through from one section/chapter to the next. It also requires that care be taken in spelling, punctuation and grammar. Any significant errors are liable to cause a reader to suspect that the content of the report may also be flawed.

The language for the report should be straightforward jargon-free English, written in conventional style using the conventional third person past tense, and readable by someone familiar with the general subject area, although not an expert in the specific topic.

The following conventions should be used, and care should be taken to maintain a consistent style throughout the document.

## Appendix B: Fonts, Paragraphs and Line Spacing

Aim to maintain a consistent approach throughout. Use Arial font size 11. Use 1.5 line spacing between lines and double spacing between paragraphs (this is done automatically if using the ‘Normal’ style in this template). Do not indent at the start of a paragraph.

## Appendix C: Mathematical Symbols

Mathematical symbols and equations are best entered using a package (e.g., Equation Editor). Equations should be centred and numbered, with the numbers presented in parentheses in the right-hand margin. Additionally, all variables should be discussed in the text.

## Appendix D: Figure and Table Captions

When figures are referred to in the text they should written as: Figure 3.1 (i.e., with a space between Figure and the subsequent numbers), with the 3 denoting the chapter, and 1 denoting the number of the figure within the chapter. The word “Figure’’ should be written out completely (e.g., do not use “Fig”) in all instances of the word. As demonstrated in Figure 9.1, figure captions should appear centred below the figure, with the caption in lower case and an initial capital for first word and proper nouns only.

A graph with colored lines and dots

Description automatically generated

Figure 9.1: Reconstruction scores for interpolations between source and target rhythmic patterns. Results are calculated as a mean of 11000 transformations per each interpolated value of mixing parameter 𝛼.

When tables are referred to in text they should be written as: Table 9.1, (i.e., with a space between Table and the number subsequent numbers. Table headings should appear below the table. The table heading should be typed in the following way:

A number of numbers on a white background

Description automatically generated

Table 9.1: Reconstruction scores (LSD, RMSE, CS) for three baseline models (VAE, WAE-MMD, AAE-ISO) and the proposed AAE-GM approach (Tomczak et al., 2020).

Additionally, if you are incorporating a figure or table from another source, you must cite the source as in the Table 9.1. Both tables and figures must have associated discussion in the text—they should not appear without reference, nor should they only be explained in the caption.

## Appendix E: Text Headings

Headings throughout the report should be consistent as follows:

Main sections and major headings should appear with initial capitals for first words and proper nouns. Leave a space of two lines above such headings and one below.

Section headings should be lower case with capital letters for the first letter of the first word and placed at the left-hand margin. Leave a space of two lines above such headings and one below. Subsection headings can be in italics, leaving a space above and below the heading. Section headers (e.g., 9.2) are available in the Styles Pane.

## Appendix F: Pagination

Starting on the Introduction page, pages should be numbered using decimal numerals (e.g., 1, 2, 3, 4). Pages prior to the Introduction page should have lower-case Roman numerals (e.g., i, ii, iii, iv).