University of Portsmouth

School of Creative Technologies

Final year Project undertaken in partial fulfilment of the requirements for the BSc (Honours) in Computer Animation

Tools to improve workflow for implementing content for use in Gavin Wade's Game Engine

By

Geoffrey Samuel

367474

Supervisor: Gavin Wade

Project Unit: CTPRO

May 2009

Project Type: Artefact

Abstract

This project looks at designing, developing and implementing tools within Autodesk's 3ds Max; that can improve the workflow when developing game levels for use in Gavin Wade's Game Engine Technology. It will now look into project management models in order to pick the best project management methods for this type of software development task, as well as different Primary Research methods for collecting data. The interviewing of two subjects will create a list of requirements for these tools to make the tools valid and useful for artist. It will look at what industry professional have designed for similar problems, and with this, design the tools around the list of requirements. Each tool will be implemented and, once at a working prototype stage, will be giving to students for 30 days for Real-World testing, where the tools will evolve to create a truly artist-orientated tool. The tools will be tested with new users to ensure the tools are useable from all ability groups. The tools will be evaluated against the requirements and the original aims and goals of the project; with a final conclusion into whether the new tools did improve the workflow for Gavin Wade's Game Engine Technology.

Acknowledgements

I would like to thank all those who helped and supported me through this project, especially my Mum, my Dad and my Brother.

I would also like to take this opportunity to thank my project supervisor for his constant support and feedback through the entire project.

I also want to thank all participants of the project, both from the real-world testing and the novice testing phases, without you guys, the toolkit would not what it is today!

Contents

Chapter 1: Introduction

1.1: Introduction and background

What I propose to do, is to design, develop and implement a tool or tools within Autodesk's 3ds Max that can improve the workflow when developing game levels for use in Gavin Wade's Game Engine Technology. The tools should be aimed at artists who are less code orientated, with the ease of use of a Graphical User Interface designed for them in mind. This could save time by streamlining their workflow and reducing the possibility of human mistakes such as spelling errors on the names of each object and material. The tool or tools designed must be as useable to novice users of the toolkit as to experienced users.

Gavin Wade's game engine includes a 3D engine that he has programmed over many years, which has been used in a commercial environment (Wade, 2007, p. 6) with the PC game "*The Mummy*", developed by Rebellion Games, published by Konami/ Universal Interactive, and is available for final year student's use for their projects.

At present, the method to get content into a format that the game engine can read, requires the content must be formatted through Autodesk 3ds Max, where each object is manually given a specific syntax based naming convention so the exporter can determine how to handle that particular object during the exporting of the level or asset.

As my primary potential clients for using the tool or tools developed from this project are the BSc (HONS) Computer Animation and the BSc (HONS) Computer Game Technology students. The tool or tools should be built upon their current software skill base. This would reduce both the need and time a student would require to learn a new software package. Autodesk's 3ds Max is the taught 3D content authoring package at the University of Portsmouth, so using this software as the base package for my tool or tools would be the most ideal solution compared to a new stand-alone application.

The client for this project will be a mixture of students but also the Game Engine Technology's creator, Gavin Wade, and as the tool(s) developed as part of this project will need to work in conjunction with the framework for the Game Engine Technology set by Gavin Wade.

Autodesk 3ds Max is an award winning industry-standard 3d modelling, animation and rendering package developed by Autodesk. 3ds Max has been used to develop content for both the film and games' industry, including such games as Unreal Tournament 3 by Epic games, and used on such films as Harry Potter and the Goblet of Fire by the Motion Picture Company.

Current industry game editors such as the UnrealEd by Epic games and Hammer by Valve are great examples of what my project could eventually evolve into as a standalone program, as well as solid resources of information on how industry developers have come across similar problems and hurdles and overcome them. These game editors are good examples of what artists and level designers expect and require from their types of tools at an industry level.

The current workflow for implementing content into Gavin Wade's Game Engine Technology is by assigning syntax based naming conventions to objects so to allow the 'MIF' exporter plug-in to process and determine the type of object. An example of this is the following name "PORTAL $Ntest2 $F5 $B2 $C" (Wade, 2007, p. 11) which tells the 'MIF' exporter that the object is to treated as a portal("PORTAL") with the unique name of "test2" ($Ntest2"), zone 5 in front of the portal ("$F5"), zone 2 behind the portal ("$B2") and to set the portal as closed ("$C").

With each object requiring this type of naming convention, the current workflow can be slow and cumbersome when dealing with large and complex levels and environments.

1.2: How I intend to solve this problem

The way I am going to study the results and outcome of this project is by working closely with two level 3 students who are current making a game level using the Gavin Wade's Game Engine Technology. These level students have agreed to provide a feedback loop providing bug reports and information on what new features they would find useful or timesaving. During the early stages I will be interviewing both of the students to find out what tool or features they would find helpful to improve or speed up their own content pipeline into the Game Engine Technology. Taking this data, I would construct a working prototype encompassing all the features and tools mentioned during the interviews. Once the tool or tools are at a stable, working build, I plan to give both students the tool or tools so they can perform real-world testing on the tools.

What this would allow me to do is to get first hand information from the people who primarily use this technology and to study its real world effects, to establish whether it does improve their content pipeline or hinders their pipeline with unnecessary tasks and problems. This would also help identify any features that could help improve the workflow, which were not originally mentioned or discussed during the initial interviews.

To ensure I have studied the initial interviews carefully, and been able to successfully implement a tool or tools that meet the requirements set out by the initial interviews and requirements, I hope to be interviewing both of the students at the end of the real-world testing phase. These results could help validate the usefulness of real-world testing phase for this project.

To help enable a final conclusion I hope to be conducting tests where 3D artists will be asked to construct a small level without the aid of the any tools developed as part of this project, and to construct an additional level with the use of tool or tools to establish how much time the tools could potentially speed up the production pipeline of a game level and to collect any additional concerns or issues.

The data collected by the study could go on to creating one all encompassing tool to help the content pipeline into Gavin Wades Game Engine Technology, and or to raise the question if the amount of time spent developing the tool or tools is beneficial compared with the amount of time saved by artists by using these type of tools.

Upon the conclusion and completions of this project, all the relevant source code and documentation will be made available to all staff and students to use as learning or teaching resource.

1.3: Project limitations

Although not strictly a limitation, the tool or tools developed would be required to work within the parameters set out by the 'MIF' exporter plug-in. This would require all the changes to be made in the same fashion as the current manual method as the code or method of exporting cannot be changed or altered.

A major limiting factor on this project are the academic deadline, set by the Creative Technologies department, which requires all the research be finalised and submitted by Friday 8th May 2009 at 3pm, causing all my research and testing to be done well in advance of the deadline.

As the main users of this project are students, the final outcome of this project would have to work seamlessly on any computer at the university so as to allow any student to take advantage of this technology and the game engine technology.

Previous experience has lead me to believe that students are not the most reliable people, and so each section involving students should be considered risky.

1.4: Project outline

I will start the project looking at different project management models to find the best management strategy to work from.

Once a project management model has been chosen, I will research into different Primary Research methods, selecting the best Primary Research method for the project.

Using the Primary Research method chosen, I will plan out, conduct and evaluate a small survey with students currently using Gavin Wade's Game Engine Technology.

The conclusion from the survey will be used to design the tools, and then to implement the designed tools within 3ds Max.

With the tool(s) designed and implement, I will start the real-world testing phase and re-interview the participants of the real-world testing.

To ensure the tools developed can be picked up by artist, I will be organizing and conducting testing session with game artists with no previous experience using Gavin Wade's Game Engine Technology and evaluating the results and feedback.

Lastly, I will be concluding and evaluating the entire project and suggesting where this work and research could be used to further enhance the tools developed as part of this project.

Now I will research into and choose the best project management models in order to create a project management structure which will work for the type of software being developed, as well as compliment the way and method in which I work.

Chapter 2: Methodology

2.1: Methodology Introduction

With the project problem defined, and with a very basic idea of how I intend to solve this problem, I will now look into project management models in order to pick the best project management method for this type of software development task. Another factor I must take into account when deciding on the project management model is my work style so to ensure that the project management method will benefit from my work ethic and method.

The reason for using a project management model and methodology is that this project will last for 32 weeks from conception to competition, and ensure that the project is split into manageable chunks and is completed on time.

2.2: Project Management Models

Two main project management models for software development that I have used before are the Waterfall methodology and the scrum agile approach.

The Waterfall model is a method where *"…each phase flows naturally into the next phase…"* (Burback, 1997) and has no overlapping phases. This method requires an established plan be analyzed and researched before any building of code is started, to prevent any costly legacy code being created or abandoned (Burback, 1997). Fig 2.2.1 shows a typical waterfall methodology.

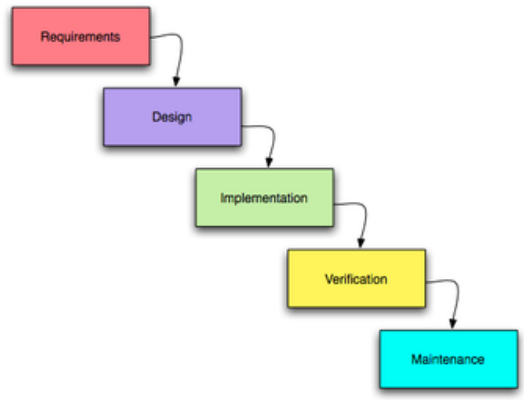


Fig 2.2.1. A typical waterfall methodology approach to software development (http://images.absoluteastronomy.com/images/encyclopediaimages/w/wa/waterfall_model.png)

Scrum agile is a software development methodology devised in the mid-1980's in Japan. Scrum works by having a person overseeing the whole scrum operation, called "the scrum master", who is responsible for the creation of the scrum teams, the product backlog, the spring planning and spring review (Koch, Agile Software Development : Evaluating the Methods for Your Organization, 2004, p. 257). The scrum master would create a sprint plan based off the product backlog and then a scrum team commits to a sprint goal based off the sprint plan, and then commences a sprint. A sprint is s fixed period of time where the team will work to accomplish the sprint goal by any means (Koch, Agile Software Development : Evaluating the Methods for Your Organization, 2004, pp. 258-259).

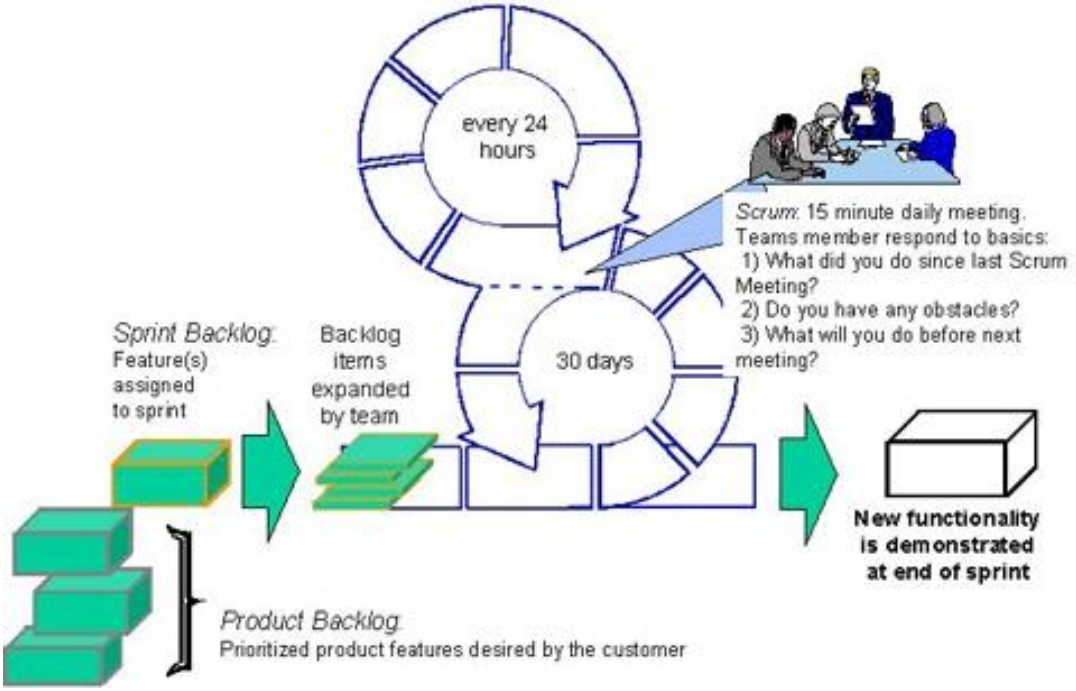


Fig 2.2.2. A scrum agile approach to software development (http://bwwss.baywoodportal.com/Pictures/_w/SCRUM%20Methodology_jpg.jpg)

Another method of software development is Evolutionary Prototyping (Liu & Roussev, 2005, p. 143). Evolutionary Prototyping is "...a method of evolving software and clarifying requirements." (Masse, 1998). What that means, is

that a crucial part of the software evolution life cycle is the test and changing of the program to suite the ever changing requirements of the user. This allows the software to grow in places that users require it to more use in, and, in effect, is driven solely by the user's expectations and needs of the software being developed.

Fig 2.2.3. shows a typical evolutionary prototyping life cycle during a software development process.

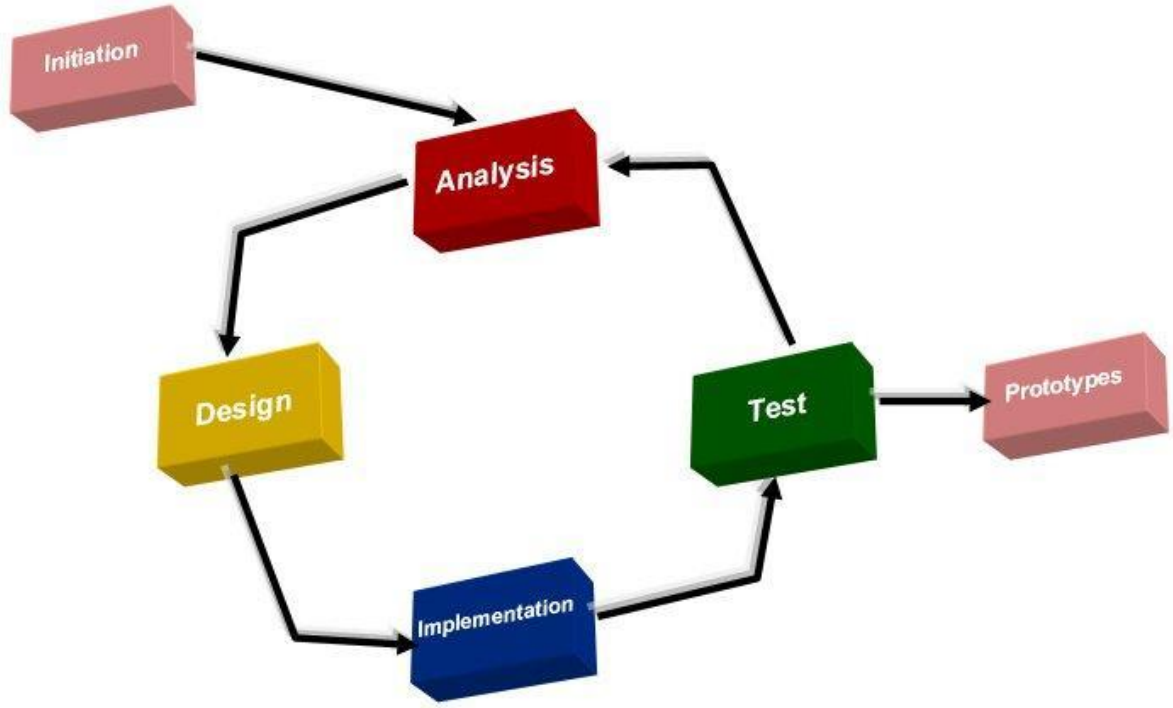


Fig 2.2.3. An Evolutionary Prototyping lifecycle to software development (http://facadetechnologies.com/evolutionary_model.jpg)

2.3: Methodology conclusion

The project management method that best suits my work ethic, in the most part, is a modified Waterfall model, but is strongly based on "*The Standard Waterfall Model for Systems Development*" (National Aeronautics and Space Administration, 2004).

Scrum Agile methodology should be employed within the modified Waterfall model to conduct sprints on the various “Sprint backlog” of each of the tool or tools (Koch, Agile Software Development : Evaluating the Methods for Your Organization, 2004, p. 30).

During the real world testing phase of the waterfall, an evolution prototyping approach to the software should be applied, as more bugs get discovered and new features get requested. The waterfall model’s timeline was quickly used to create a Gantt chart *(see Appendix A)* for easier understanding of the milestones and deadlines.

These deadlines were made to coincide with my project proposal as well as with the project deadlines and restrictions set out by the Creative Technologies Department.

Part of the Gantt chart, where two processes overlap *(see Appendices A),* are to do with the demonstration of the Artefact to my Supervisor and moderator and the project hand-in dates. This is due to the both of these deadlines being set by the Creative Technologies Department.

For the main development stage, a Scrum Agile approach to software development should be used to create the basic underlying tool structures with the requirements for the chosen sprint backlog (Koch, Agile Software Development : Evaluating the Methods for Your Organization, 2004, p. 30) being made during the tool designing phase. As the development team consists solely of myself, the sprints will be relatively short with a chosen tool to develop during the 2-3 day sprints. This will allow the sprints to focus on that particular tool and work on getting the structure correct and a working prototype fairly quickly. I can see myself moving the sprint backlogs from a tool level to an overall level nearer the end.

The way the relevant data will be collect to make the tool(s) relevant, is by conducting a Primary Research methodology with a small selection of students currently using Gavin Wade’s Game Engine Technology. This would

allow me to get first hand information about the workflow problems connected with implementing content into the Game Engine Technology.

This data would become the main design for the tool(s) based off the common themes between all the data collected, and would address the concerns and needs raised. This list of features will become a “Product Backlog”, and before conducting a Scrum sprint, a “Sprint Backlog” will be created based off the “Product Backlog”.

Once the tool(s) have be implemented to a stable prototype stage, they will be given to the two real-work subjects (Subject A & Subject B) so to grow from an evolutionary prototyping lifecycle to produce a version which is designed and focused around an artist’s workflow.

In order to test the new workflow using the tool(s) developed, I will be conducting tests which will have students creating basic levels. These students would ideally be more art orientated and will be asked to evaluate their time and experience using the new workflow.

With my methodologies chosen, I will now research into how to collect and analyze data, of which will become the basics of my scrum product backlog.

Chapter 3: User Issues

3.1: User Issues Introduction

With the project methodology chosen, and timeline drawn up, the scrum methodology chosen for the programming/ production process requires a list of requirements (Flynt & Omar, 2004, p. 28).

These user requirements are important as it will help develop the tool(s) to meet the needs and expectations of the people who will, ultimately, be using them. With any new product, Primary Research would have to be carried out to determine what is wanted by people who want or use this product (Hague, Adams, & Brace, 2006, p. 4)

To create this list of requirements, and to make the product applicable, and even desirable, to game artist to help streamline their workflows, it became apparent that it would be necessary to get first hand information from game artists currently using the Game Engine Technology.

First hand information collecting, normally called Primary Research, is a vital part of all companies and organizations when dealing with issues such as expanding their current services or introducing a new producers to potential customers (Hague, Adams, & Brace, 2006, pp. 3-4), or even just ongoing Primary Research checking competitors prices.

This type of Primary Research will be vital in creating tool(s) to improve the workflow to Gavin Wade's Game Engine Technology by identifying areas of concerns and reducing the number of repetitive tasks required by the Game Engine Technology.

Currently there are two students (Subject A and Subject B) who are using Gavin Wade's Game Engine Technology for their own projects, and are ideal candidates for the collection of first hand information regarding the Game Engine Technology and its current workflow.

With only two students eligible for first hand data collect, research into the different primary research methods should be conducted to find the best method to get the highest quality results from such a small sample size

3.2: Data collection methods

Primary research methods can be sorted into two different categories, Qualitative or Quantitative (Welsh National Assembly: Department for Education) depending on the weather the data gathered, as a result of the method, is more about the quality of the data or the quantity of the data (Cottrell, 2003, p. 205).

Postal Surveys, Telephone surveys and Face-to-face surveys are examples of research methods that can be categorized as Quantitative research (Welsh National Assembly: Department for Education), as they favour the amount of people asked rather than the quality of the data gathered. In-depth interviews, Focus groups and Case Studies can all be classified as Qualitative research (Welsh National Assembly: Department for Education) methods as they all favour small amount of subjects with more focused, and higher quality results from the data collected.

With a small group of eligible candidates for my primary research, the research into data collection methods should be focused on Qualitative methods to ensure the best possible data from such a small test group.

In-depth Interviews allows for the interviewer and interviewee to create and maintain a dialogue, based loosely around questions allowing for maximum scope of answers (Hague & Hague, 2004, p. 61). The advantages, as states by Hague & Hague, 2004, p.62, are when “Processes need describing in detail” which could, potentially, be usefully when trying to understand the current workflow with Gavin Wade’s Game Engine Technology.

Focus groups is a Primary research method frequency used for both market and medical research (Gibbs, 1997) and can be loosely described as a group interview. (Gibbs, 1997). Focus groups allow for members of the group to listen to each other and to share information which can be used to find the

reason behind their particular view rather than just the view it's self (Scheuren, 2004).

A case study is a method to understanding an issue or problem or to expand current research (Soy, 1997). Case studies work by analyzing a certain number of events, within a limited about of test subjects, to find and understand the relationship between them (Soy, 1997).

With research into the different qualitative Primary Research methods, these methods should be evaluated against the needs of the unit so to get the best data out of the eligible candidates, and therefore enabling us to get the better user requirements.

3.3: Data collection methods conclusion

As the research into Primary Research method has shown, there two main categories for these methods, Qualitative or Quantitative, with Qualitative focusing at smaller participating but emphasizing the quality and depth of the results, and Quantitative which is more focused on the amount of people participating and less on the quality of the data collected.

With only a survey size of two (Subject A and Subject B), the focus of the research in chapter 3.3 is aimed for Qualitative Research Methods.

Three methods of collecting qualitative research data is by in-depth interview, case study and focus groups (Welsh National Assembly: Department for Education).

A case study approach to collecting data could, potentially, generate the highest quality data as it involves filming and watching both subjects use the current workflow with the Game Engine Technology. The drawback of this method is the time that would be required to record and analyze both students could, potentially, take the entire length of the project, as both subjects have a year to create their level, and this research method would only be a fair test with the recoding and analyzing of the entire workflow.

With such a limited survey size, a focus group approach would not be as effect with only two people, as it would not provide enough diversity in the dialog to benefit from it. There is also the factor that having both subject talk about their experience using the current workflow could bias their own personal opinions and views, especially as one subject may find a certain aspect of the workflow difficult or confusing and, without that be a mutual feeling, could go unsaid.

The In-Depth interview would allow enough flexibility for the interviewee to incorporate any comments' or suggestions they feel are relevant, but without the chance to listen to other people's comments and problems, the answers could be shallow, and lacking the reasoning behind the answer.

The research method with the best chance of getting the highest quality of data in a relatively short time is the in-depth interview. This is because of its flexibility with the questions asked as well as allowing for more sensitive issues which the Subject might find confusing or intimidating, as it is these types of issues the tool(s) aim to fix.

Now with the method of collecting the data chosen, the interviews should be planned out, as to maximize the effectiveness of my time with each subject.

Chapter 4: User Requirements

4.1: Planning of the data collection

With the interview method chosen as the most effective way to collect data from the two subjects (Subject A & Subject B), the interview questions should be planned out to help maximum the focus of the data collected.

To establish their background knowledge and area of expertise, the first question would have to be about the degree course their currently undertaking with a second question being about their level of study. These two questions would establish their background knowledge. In order to establish the subject's area of expertise, the third question should be to about what the subject considers themselves in a game's production pipeline.

With their backgrounds and expertise known, the interview should move onto the subject's prior knowledge of game engines and their own personal experience in computer programming. This will allow me to understand how their approaching Gavin Wade's Game Engine Technology, whether as programmers with experience programming game engines or as a game artist with no prior experience in programming game engines. To accomplish this, question four should be about the subject's experience in using game engines and question five should be about the subject's experience in computer programming.

As both subjects' should be new to Gavin Wade's Game Engine Technology, it's important to find out how much of a learning curve there is, so question six should be about their first impressions of using the Game Engine Technology and to follow that up should be question seven on the subject's views and opinions on "The '.MIF' Manual".

These questions should give me access to what artists are have problems with in the current workflow and generate ideas for tool(s) which can improve that workflow.

The interview should close by asking the subject if there was anything else he or she would like to comment on.

With the interview designed complete, interview questions can be found in Appendix B, the subjects should now be interviewed to collect data before that data is analyzed.

4.2: Collection of the data

In order to ascertain what could help improve the content implantation workflow for students for Gavin Wade's Game Engine, an interview with two current students already using this technology would be the best solution for collecting this Primary Research.

The two Students (Subject A and Subject B), who were interviewed as part of the research, are each game artists with little or no previous experience in computer programming. The interviews were recorded to ensure that there was no confusion or mistranslation between what the interviewee said and how it gets interpreted, as well as provide a record of what was said.

Similar questions were asked for each interview (Appendix B), but the exact question may have differed slightly based of their previous answers. The interview questions were quite broad, and encompassed a few different topics that let the interviewee go into each topic, as detailed as they wanted to. The interview started out by asking questions on their particular background to understand how they were approaching Gavin Wade's Game Engine Technology, if they were approaching the Technology as a programmer or as an artist. After their background was established, they were asked about their own experiences using the Game Engine Technology and any problems they had encountered and if they had any concerns to raise about the content pipeline for the Game Engine.

4.3: Interpretation of the data

The interviews were able to give an insight into the artist's way of dealing with this type of Game Engine Technology. Both were initially frightened by the coding and scripting aspect of using the engine, which highlighted my first task, to find a user friendly way to help the artists.

The first student who was interviewed as part of my research, *Subject A is* a 3rd Year BSc (HONS) Computer Game Technology student. When *Subject A* was asked about his experience in programming answered, *"It's very, very limited…"* as his specialty is as a *"3D Modeller"*. This clearly shows that *Subject A* is the perfect individual to validate my research and my final product, as a 3D artist with little or no background in computer graphics programming and would benefit from using such a tool in his own development.

The second student I interviewed for my research, *Subject B* is a 3rd Year BSC (HONS) Computer Games Technology student, who describes himself as an *"Environment artist"* with a little background in computer programming, which he describes as *"…code a little…C++…mostly Visual Basic"*.

Both interviewees mentioned the "Unreal Ed" during their interview when talking about their own experiences using other game editors, as well not scripting or programming to get content into a game engine.

The main documentation for Gavin Wades Game Engine Technology is a 30 page document called *"The '.MIF' manual"*, which was last revised in March of 2007. I wanted to find out what the two students had initially thought about this documentation especially considering their background as artists. *Subject B* describes *"The '.MIF' manual"* as *"Shocking, doesn't give much help at all really",* but did state *"…scripting was a little bit difficult to start with, but when you pick it up, it can breeze pretty through…"* when asked if the whole coding aspect was daunting without a Graphical User Interface. *Subject A* said this about his experience with *"The '.MIF' manual"; "…constantly going back to the*

'MIF' manual…it's a wall you have to overcome to actually get to used to this engine."

This highlights a potential area for a tool, to remove *"The '.MIF' manual"* from the artist. During the course of the interview, *Subject A* explained how using the current content pipeline it can get *"very, very complicated and it does get very, very messy very quickly"* and when asked why this was; *Subject A* said *"It's definitely the naming conventions…".*

The naming conventions required by Gavin Wade's exporter seem to be an area of problem and hindrance with artist which may be linked with *"The '.MIF' manual"* issue.

As both *Subject A* and *Subject B* have had prior experience using the current creative content pipeline employed by Gavin Wade's Game Engine, I wanted to ascertain if there was anything they felt would help speed up their pipeline. *Subject B* expressed the desire for a *"click save"* feature to save and export the current file out to the engine readable format. The reason behind this feature was to *"…Get the files in and out of the engine as quickly as possible... Continually changing a file you could export something wrong, which might not work". Subject A* referred to an Unreal Ed style *"…asset browser…"* to help "*…make it…more intuitive to use."*

4.4: Conclusion of the collected data

From the interviews, a key tool which could be improved the workflow is a way to remove the need for artist to keep refereeing back to *"The '.MIF' manual"* for syntaxes, as well as remove the need for artist to enter these syntaxes in manually therefore cutting down on the amount of mistakes created by human spelling errors.

Other features which could be implemented to help artists create content, could be an asset management system to help artist keep track of all their assets as well as a simpler exporting solution to speed up the exporting of a mesh or level.

With the user requirements established, research should now be conducted into current Game Engine Technology and its related subjects, such as current industry level editors and common interface themes of 3ds Max. This would ensure the tools developed keep the continuity of 3ds Max's Graphical User Interface.

Chapter 5: Related Technologies

5.1: Relevant information in subject area

After the initial talk with Gavin Wade about the lack of tools to help artist and programmers use his Game Engine Technology, I went to look at how to tackle this problem; both from what research has been conducted in this particular area and what industry professionals, presented with the same problems, have designed.

The research started by looking into current game engine tools and game engine editors used by both industry veterans and by the modification community. Wikipedia provided a rather large list of level editors (Wikimedia Foundation, Inc, 2009) for all types of games, both current generation game engines and game engines from the last two decades. Cross checking these games with level editors with release dates, and generating a list of the four most recent level editors to establish what industry artists and level designers are expecting from these types of tools.

Of these four level editors, is UnrealEd by Epic Games which is developed for the game engine, used and developed by Epic Game, the "*Unreal Engine*" (Epic Games, 2008). The Unreal Engine has been used commercially on several games on different gaming platforms with games such as "*Gears of War*" by Epic Games and "*BioShock*" by 2K Games as well on the Nickelodeon TV show "*Lazytown*" by Lazytown Entertainment (Epic Games, 2009). UnrealEd is focused at *"….professional game developers."* (Flynt & Booth, p. 2), and so is a prime example of what an industry level game editor should be able to do, as well as how game development should work using this type of tool.

On the list of the four most recent level editors, is the Hammer Editor for the Valve Corporation's game engine called "Source" (Zerbst & Duevel, 2004, p. 668) . The Hammer Editor is used by the Valve Corporation to create their games (McTaggart, 2004), and is part of the Source engine Software Development Kit (SDK) and has been used by the Valve Corporation to create

“Half-Life 2: Episode Two” (Valve Corporation, 2007) as well as “Half-Life 2: Episode One”, both developed and published by the Valve Corporation.

Another game editor from the list is the Sandbox 2 level editor by Crytek for use with their game engine *“Cryengine 2”*, which has been used in the game “*Crysis*” developed by Crytek Frankfurt and published by Electronic Arts in 2008. The Sandbox 2 editor gives its users access to all the dynamics and objects within the Cryengine 2, allowing complete customization of the game level within a graphical user interface (Crytek GmbH, 2008).

The last game editor on the list of the top four most recent level editors is the tTool (LittleBigPlanet Wiki, 2009) game editor developed by Media Molecule Ltd for their game “*Little Big Planet*”. Unlike the other level editors mentioned, the tTool level editor is a built in feature of the game is only capable of creating and modifying user made levels rather than creating any new game play enhancements or changes to game play mechanics.

An article in GAMASURA entitled *“Integrating Maxscript and .NET systems” written by Shea McCombs and Kevin Rabun*, talks about using 3D Studio Max as a level editor taking full advantage of the power of C#.

The current literature for Gavin Wade’s engine, a 30 page document called: “*The ‘.MIF’ manual*” explains the process and the syntax based naming convention used by the Gavin Wade’s Game Engine Technology.

“The ‘.MIF’ manual” represents a client for my project as it explains to the user everything that my tool(s) will need do, effectively replacing *“The ‘MIF’ manual”*, and so would need to do everything stated in the manual as well as any other points raised by the two students who will be interviewed.

5.2: Current Game Engine Technology

The Current Game Engine Technology that is being used for this project is Gavin Wade's Game Engine Technology. This Game Engine Technology has been used in a commercial environment and that the technology is "very experimental" (Wade, 2007, p. 6).

Game levels are typically large areas filled with many different objects, with each of these objects being made up of polygons, and each polygon having to be processed by the computer in order to be drawn. A trick to speed up the processing of the level is by dividing the level into different zones and displaying the zones based off the player's view of a portal (Tim Sweeney (Epic MegaGames, Inc.), 1999). Portals are "windows" (Wade, 2007, pp. 11-12) which draws a zone that can be seen through a portal. This would allow only half of the map to be drawn if the play is in the centre of the map, as he can only see half of the map on screen at once.

Another trick used to speed up the render of a game level is the use of Hardware instancing. What this does, is tell the graphics card that you want to draw the same object multiple times in the same draw call (Roecode, 2008), which allows the graphics card to store that object in memory for quicker calling and drawing of the object.

With the release of Microsoft's DirectX 8.0 came the introduction of the vertex and pixel shaders (St-Laurent, 2004, p. 3). These types of shaders are used extensively within modern gaming engines, such as by the Unreal Engine (Epic Games, 2008) and the Valve Source Engine (Mitchell, 2006, pp. 34, 48). Vertex and Pixel shaders are supported by Gavin Wade's Game Engine Technology as well as .FX shaders (Wade, 2007, p. 15).

One of the last sections in "The '.MIF' Manual" is dedicated to setting up and assigning Physics properties to static objects (Wade, 2007, pp. 17-20). Physics in games has become a staple of game engines with every one of the 4 games listed in Chapter 5.1 boasting a built in physics engine.

Like many game engines on the market today, Unrealscript for the Unreal Engine (Epic Games, 2008), Gavin Wade's Game Engine Technology has a scripting language to help the interpretation of animation and mesh data. These come in the file formats of '.OIS' for mesh object scripts and '.ANM' for animation scripts (Wade, 2007, pp. 27-29).

Users also have the ability to embed an XML file with a mesh mode during the export to the engine's proprietary file format, the '.MIF' file (Wade, 2007, pp. 21-23).

The current workflow for implementing content using Gavin Wade's Engine, as described within "The '.MIF' manual" is by creating all the assets within Autodesk 3ds Max, and giving a special naming syntax onto the mesh name, which will get read and converted by the MIF exporter to be used within the game engine (Wade, 2007, pp. 11-12, 21-23). An example of this is the following name "PORTAL $Ntest1 $F1 $B2" which tells the 'MIF' exporter that the object is to treated as a portal("PORTAL") with the unique name of "test1" ($Ntest1"), zone 1 in front of the portal ("$F1") and zone 2 behind the portal ("$B2"). Material properties are handled the same way, with special syntax names assigning different properties or effects on that material (Wade, 2007, pp. 14-16). An example of material naming syntax is "Borg $Fskinned_diffuse1.fx $D $S" which tells the exporter the name that a .FX shader is being used, with the .FX file name of skinned_diffuse1.fx ("$Fskinned_diffuse1.fx"), that the engine should treat the material as a Death polygon ("$D"), and that the engine should render the material on the second pass as a transparent material ("$S").

5.3: Autodesk's 3Ds Max Interface devices and styles

To help the user feel that the Graphical User Interface for the tool or tools was an integral part of 3ds Max, research into the commonly used interface devices and user interface elements that make up the official interface for 3ds Max was conducted.

Fig 5.4.1 shows the standard 3ds Max user interface, as well as the material editor and the render viewport.

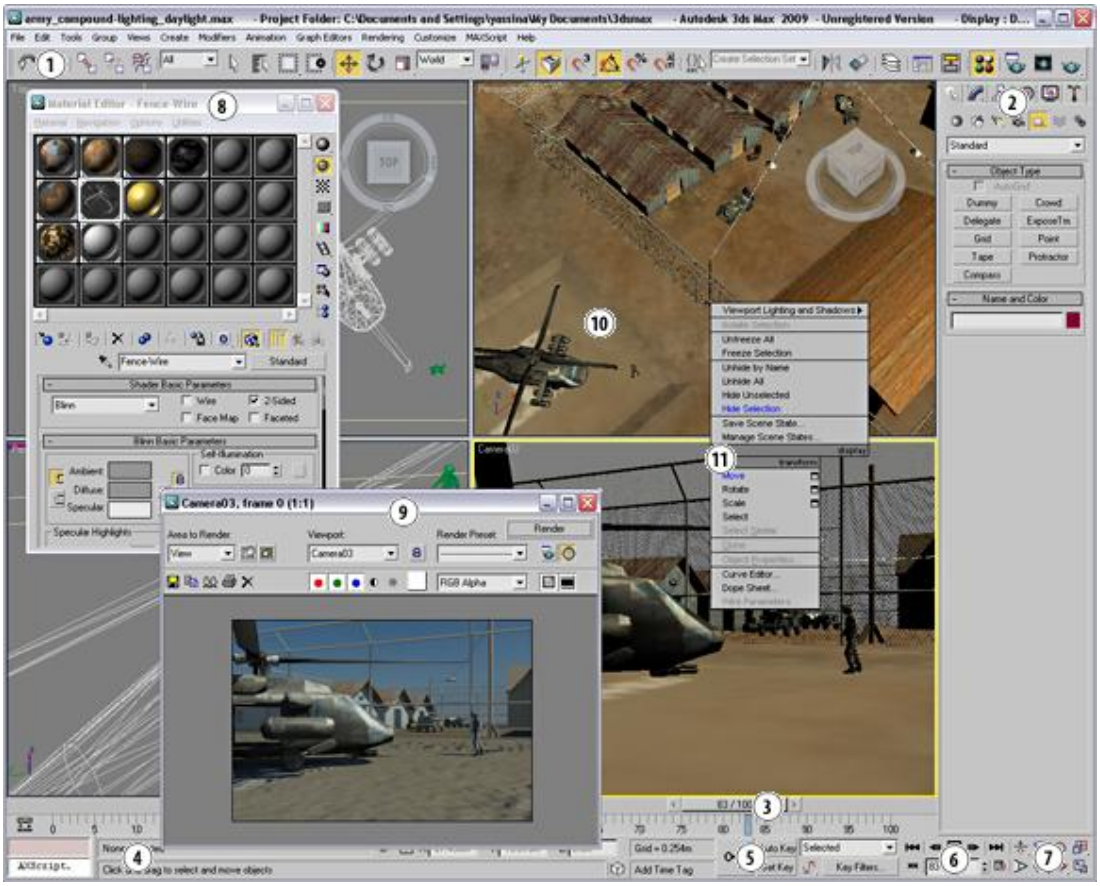


Fig 5.4.1 Autodesk's 3ds Max user interface
*(*http://download.autodesk.com/us/3dsmax/interface_overview/2009/3dsMaxUIOverview.htm*)*

Pictures have been taken of certain areas of 3ds Max's Graphical User Interface in order to establish what interface device they are and what it can do, so as to understand how and where certain interface devices are placed to produce a similar style to that of the default 3ds Max interface.

Fig 5.4.2 is of the Command Panel within 3ds Max, and a good example of the user interface taking advantage of a device known as Tabs (Microsoft Corporation, 2009). Tabs cannot be natively created or handled through the built in script language MAXScript, but through either an ActiveX Tabs control (Autodesk(7), 2006) or the .NET equivalent.

Fig 5.4.2 The Command Panel

A place where a lot of different user interface elements are combined and used is on a Parameter rollout. Fig 5.4.3 is of the parameters for a sphere primitive and as you can see there are 3 different types of interface devices being used. The control with the name “Radius” is a spinner control; (Autodesk(1), 2006) the control with the name “Smooth” is a checkbox; (Autodesk(1), 2006) and the control with the names “Chop” and “Squash” is a radio button (Autodesk(1), 2006). From a layout perspective, the top 8 controls all line up vertically, with the bottom three lining up vertically with each other. Fig 5.4.3 also shows that two of the spinner controls are set as disabled, meaning the user cannot change or edit their values (Autodesk(2), 2006), this is because the checkbox control above has not been set to active.

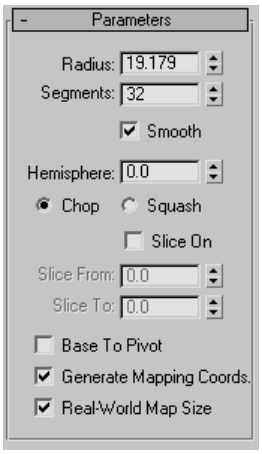


Fig 5.4.3 A Standard modifier parameter rollout.

Chapter 6: Designing the Tool(s)

6.1: Choosing a suitable Programming Language(s)

One of the major elements which will influence the design on the tools and eventually what they will be able to do and how fast they will work, is the programming language chosen.

The main method of programming within 3ds Max is through the C++ SDK that ships with the program. The C++ SDK allows for greater access of the 3ds Max's Application Programming Interface (API) and through this, would be able to integrate any type of plug-ins or tools (Autodesk(8), 2006) .

Another method of programming within 3ds Max is through the built in scripting language called Maxscript. Maxscript allows users to quick design and create plug-ins and automated tools with little experience of programming (Autodesk(5), 2006), but can perform certain operations and functions not accessible through the API (Autodesk(8), 2006).

Since the release of Discreet 3d studio Max 4 in 2000, ActiveX controls has allowed 3ds Max to communicate with other Component Object Model (COM) objects and through this communicates with other Computer Programming languages. Although ActiveX has now been superseded by a faster .NET solution within 3ds Max, COM can still be used to control Maxscript through other means. The most popular of these is through Python scripting (Pletcher, 2008).

The main advantage of Python scripting is that not platform specific and therefore could be used in other applications with minimal changes.

In 2005, Autodesk released 3ds Max 9 which included a new API called "The .NET Bridge". This new API allowed for the communication between Maxscript and .NET code, so a external program or function could be programmed within Microsoft's C# and complied to a Dynamic-link library (DLL) which could be interpreted and run inside 3ds Max (Mr Bluesummers, 2008)

6.2: Comparing the design with the User Requirements

As set out from the Interpretation of the data (*Chapter 4.2)*, there is the need for three main tools to help improve the workflow for implementing content for artists for use in Gavin Wade's engine.

The first of these proposed tools is a method to remove *"The '.MIF' manual"* to let the artists concentrate on producing artwork and less on syntaxes or how a level should be linked together. To solve this, I plan to create a Graphical User Interface so that the user can quickly change the parameters of any object and let the program generate the necessary naming syntaxes required. From my research the easiest way to access and rename objects like this is by using Maxscript or the C++ SDK, and would need to take the form of a modifier or a popup dialog box.

The second tool I propose to create is an asset management browser to allow artists to quickly import user made pre-created nodes, so they can be created and placed in a user friendly environment. This tool would help to create a user interface where *"…everything is there in front of you…"* which was a comment made by *Subject A* during the interview (*Chapter 4.2)*. This tool could also be used as a place to have other functions like a one click export feature. From the research into suitable programming languages, I conclude that this type of tool would require either a C++ SDK plug-in or a C#/Python interface with Maxscript handlers to allow such an interface to call Maxscript objects and functions.

The last of the three proposed tools is an instance helper node to give the user ability to import a static mesh from a .MIF file to allow the mesh to be placed and positioned correctly, but then to get rid of the geometry leaving a simple node with all the transformation matrices for the engine to use at runtime. Although this tool is not the result from any response in the interviews, this tool would fill the gap between the naming conventions and the asset browser but having a node designed for the handling of instance objects, but with the automatic naming syntax generation.

6.3: Design each tool

6.3.1: The Naming Tool

The Naming Modifier should be designed so it will be able to handle the renaming of world meshes, portals and zones, as well as dealing with material. As the Naming tool would be specific to each mesh, containing data which could be used to generate the correct naming syntaxes as well as positing in the hierarchy as required.

From a technical point of view, there are two main ways to create, modify and save the necessary data to generate to the correct naming syntaxes. The first of which is by building upon the built in Object Orientated class system built into 3ds Max. This can be accomplished by creating new modifier plug-in, which, as a class, will allow for specific data to be stored for that object with that modifier (Autodesk(4), 2006)

The other way of saving this type of data would be by writing and reading custom attributes to a particular object. This method would allow for maximum flexibility as it would not require the plug-in to be installed on other machines without causing any errors or problems. The disadvantage of this method is that the interface for this would have to take the form of a pop up dialog box rather than a traditional part of the 3ds Max interface from which the artists are used to working from.

Artists are used to using modifiers within 3ds Max, so using a modifier as the interface for this plug-in, would require the artist to have less of a new interface to learn. However this could cause dependency issues, requiring the plug-in to be installed on a computer. To create a streamlined and user friendly approach, I decided to use the modifier method to take advantage of the build in class system giving me easier access to create and modify various variables within the naming modifier class.

Looking at the requirements of this tool, and the standard interface theme that 3ds Max employs for all its modifiers and option boxes, a draft design for the look of this modifier plug-in was created. The design is modular and keeps

within the themes and standards used by 3ds Max as well as keeping all the necessary options for each type of mesh within a group box for ease of use. As the user changes the mesh type, the options for that particular mesh type would become enabled so the user can change them or set up the parameters.

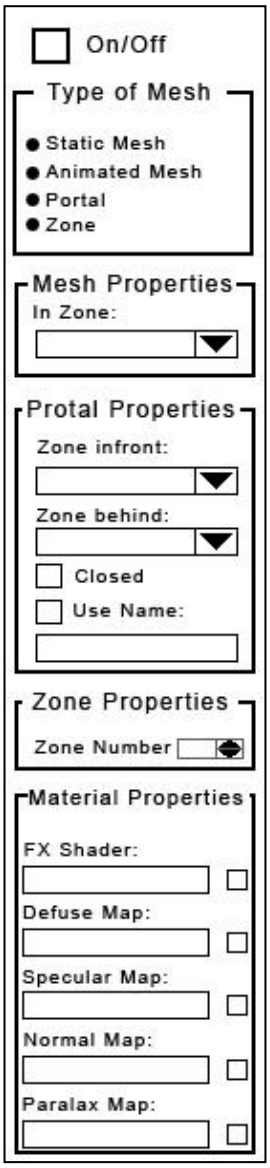


Fig 6.3.1.1 Concept art for Naming modifier.

6.3.2: The Instance Node

The premise of the instance node is for a dummy node to keep the required matrices so to transform the instance object at runtime. The node would have the ability to import the geometry to represent the in-game meshes. To help speed up the artists level creation process when completing a level design using instance able objects, the instance node should have some sort of optimization for the represented geometry, or to hide the represented geometry complete. This would allow artist to speed up the pre-visualization of a level, arranging instance object using the full represented geometry, and then changing it to only show the instance node instead of the object geometry.

A range of different optimization options and modes should give the artists the full range of control for the level of represented geometry being shown, including very basic blocking methods such as a simple bounding box. All these options would not add much of a computing overhead, but could prove to be a useful feature for artist and level designers.

The Instance Node's Graphical User Interface design should be easy to navigate around and follow the common themes of 3ds Max's Interface. The programming behind it should be designed in such a way as to be the primary way of creating and setting up the Instance Node, by being called and created by the asset management browser. The asset management browser would create an instance node through Maxscript, and would set up the new node with the parameters passed through the asset management browser such as the location of the MIF file and whether there is a corresponding MAX file for the selected MIF file.

The geometry that is generated by the instance node, the represented geometry, should not be selectable so as not to confuse the artists by selecting and transforming the represented geometry rather than the instance node, and should be hidden when its parent instance node is hidden, and deleted when the parent instance node is deleted.

If the node was self-updating, that is to say it would update itself with the latest parameters and mesh's, 1000+ instance nodes could cause a significant overhead and drain on resources, so a button or command to update that particular instance node would be preferable.

Below is the original concept art for the graphical user interfaces for the instance node, taking into account all the design and feature elements as well as the standard 3ds Max interface standards.

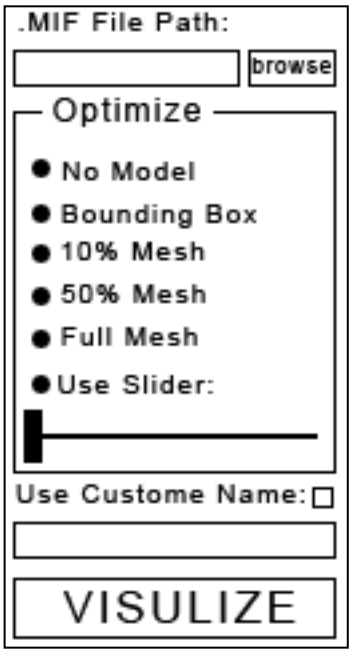


Fig 6.3.2.1 Concept art for Instance Node.

6.3.3: Asset Management Browser

The design for the Asset Management Browser is that it is simple to navigate around the interface and small enough for the artists to have in the corner of their screen to use without obstructing their view of the 3ds Max user interface. The asset management browser should be used as the main interaction for the entire Game Engine Technology as well as the creation point for the instance node. The main feature of the asset management browser will to be to load in directories and to create instance nodes using a selected MIF file from a loaded directory.

As the main user interface for the entire toolkit, it could also be used to house features such as a comprehensive help system for new users and a reference and Maxscript reference for the more advance user. Other features it could include could be such things as a scripting panel, designed to speed up the creation of Game Engine Technology files such as OIS, ANM and XML files with pre-set templates, as well as a material panel to show materials as they would look in-game.

The workflow for this tool should be very simple, to bring in a working directory with the pre-created assets, to be used as instance objects. This would be added to a tree view on the left hand side of the asset management browser. From this directory, a list of all the MIF files will appear in the centre of the screen with details such as; if there is a corresponding MAX file to accompany the MIF file; the full path; and of course the name of the file.

Once a MIF file is selected from the centre of the browser, the details and a small picture will appear in the right hand side. The ‘make instance’ button will would then an instance node with the properties of the selected item. These properties would include such things as whether or not the file has an associated MAX file with the MIF file and the full path to the file.

With these properties passed onto a new instance node, all the user will be required to do is to press the Visualize button on the newly created instance node if they want the mesh to appear, as these properties will be handled to create the correct naming convention required by the Game Engine Technology automatically.

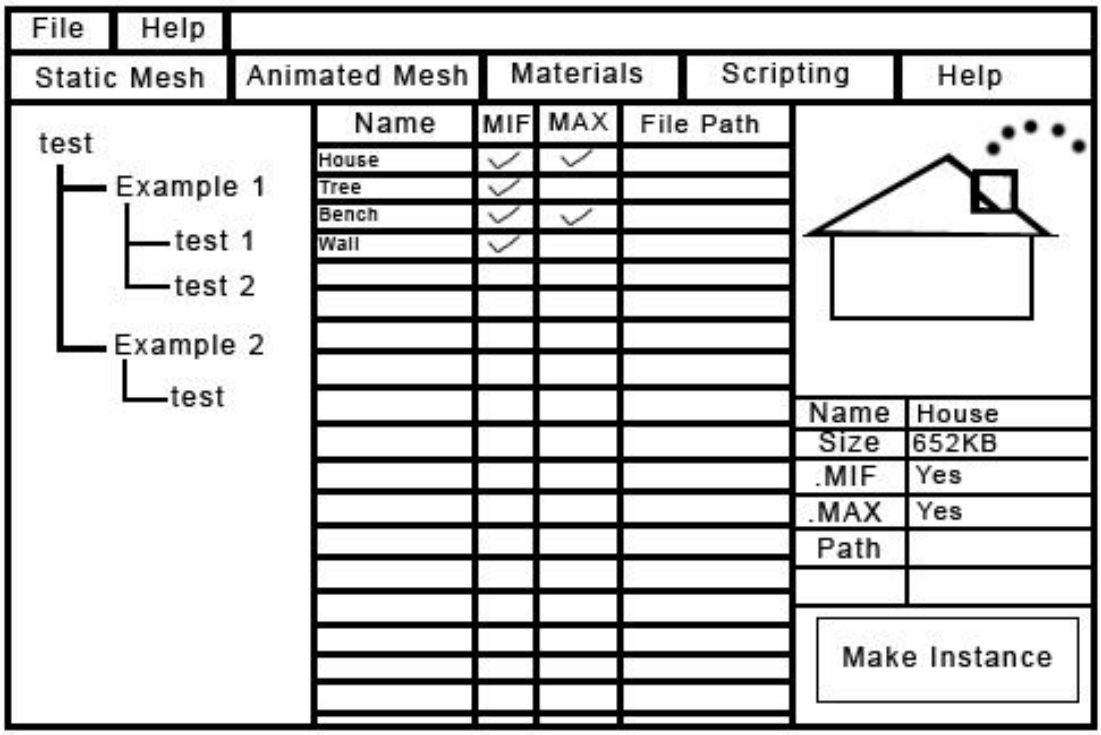


Fig 6.3.3.1 Concept art for Game Engine Technology Asset Management Browser (GETAMB)

Chapter 7: Creating the Tools

7.1: Implementing the Naming Modifier

Using both the required design of the Naming Modifier tool, and the programming languages used in the development of 3ds Max, MAXScript was chosen as it is a simplistic programming language with access to prebuilt functions and class system. As MAXScript is built into 3ds Max, it requires no external compilers, so creating new iterations can be done quicker with near instance results.

Using the MAXScript reference I was able to quickly create a blank modifier that could be applied to all objects inside 3ds Max, and from that I used the rollout creation tool to build the rollout graphical user interface much like the concept design (Chapter 6. 3.1).

As mentioned in Chapter 2.3, a scrum agile methodology was used when creating the tools. With the graphical user interface created, a sprint backlog with all the jobs left to do was generated, effectively linking up the graphical user interface with the main code behind the modifier.

These two day sprints were used to speed up the creation of main bulk of the content with such basic features as; type of mesh selecting; assigning zones, picking Zones; choosing the zone in front and behind a portal.

The code was built as per the MAXScript standards but after the sprint cycle, the code was cleaned up and broken up the code into more logical sections to allow easier adding or removing of features or modes within the modifier.

After the initial sprint, it quickly become apparent that the original design had not been fully thought out and there were rather large gaps in the logic of how this tool would work with regards to materials, and that the types of meshes the game engine uses were not fully researched. The logic gap around the assigning and using of materials was such that it required a complete re-design, and due to the size of the problem, it is covered in Chapter 7.4.

The lack of research into the types of meshes created a very problem, requiring a slight change to the user interface by removing two items, but did not require any change in the main workings of the code. This problem is worth mentioning as it is also reflected in the Asset Management Browser with the names of the original tabs along the top. This oversight could have been avoided with closer attention to the “MIF” manual as well as looking at more examples of how the current 3ds Max MIF exporter works and what it exports.

With the Graphical User Interface for the modifier created and connected with the functions to change the parameters, I started what would be a core feature of the entire toolkit, the Ready for Export function. The Ready for Export function is a function that will go through every object in the scene, hidden or unhidden, frozen or unfrozen to create an array of all objects with the naming modifier. The function would then break down this array into smaller, more focused arrays depending on their mesh type so each mesh type had its own function for converting the data into the naming convention required. This method of keeping the different mesh type conversions as separate functions allowed code to be kept clean whilst still giving the flexibility to add or remove parameters effecting the conversion.

The real advantages of not using external files as part of the toolkit (.DLL files) really shone during the initial bug testing after the tool was built, as any problems could be debugged, solved, recompiled and run in near real-time as the compiling time for this tool took around about half a second. This allowed for the quick turnaround of builds and fixing any bugs or logical errors created during the scrum sprint process, creating a more stable build and encouraging me to add more features to this tool.

Fig 7.1.1 shows the implemented prototype naming modifier.

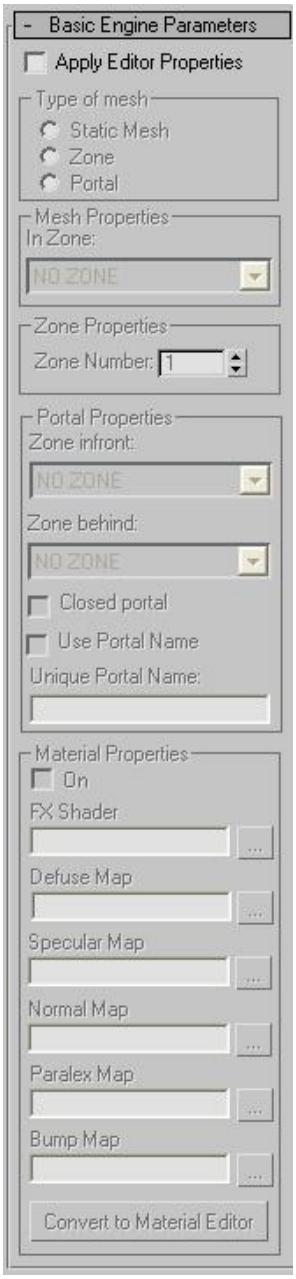


Fig 7.1.1The implemented Naming modifier interface.

7.2: Implementing the Instance Node

Like the Naming modifier, I decided upon MAXScript as the programming language of choice for writing the instance node, as the same advantages that applied for the Naming modifier also apply for the instance node.

The MAXScript reference gave plenty of help for the building of the structure required to create a new dummy class, as well as information in how to set the node up so to only be creatable from MAXScript. Although this code was added whilst writing the initial structure for the node, it was commented out so not to compile or run, in order to speed up the testing of the node, as the Asset Management Browser had not been created yet and linking up the instance node with the buttons within the asset management browser would be dealt with during the implementing of the asset management browser. As with the naming modifier, the creation of the user interface for the tool helped create a sprint backlog of features that were to implement during the 2 day sprints. The features include in the sprint backlog were; Importing a Max file based off the path of a MIF file; optimizing the mesh using the interface options; and setting up a system for the represented geometry mesh to be deleted, transformed and hidden with the parent node.

The Instance Node has a specific purpose with very little options to change, and as such required little debugging or much extension to the existing Ready for Export function to incorporate and rename the instance node based off the settings.

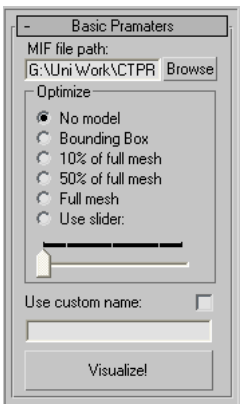


Fig 7.2.1 The instance Node interface.

7.3: Implementing the Asset Management Browser

7.3.1: Implementing of the C# Interface

With the design of the Asset Management Browser requiring a number of control devices not natively creatable from MAXScript but creatable through MAXScript with the .NET extension, I looked at creating the entire interface in C# as a DLL and call this through MAXScript as well as having MAXScript handle any interaction between 3ds Max and the C# interface.

Creating any .NET control within MAXScript requires code for the customUI element placement and all the code to drive this (Neale) and with such an extensive user interface already designed, it would be tedious and time consuming to create such an interface from code. Another way would be to create the interface as a C# form inside a DLL and call it from MAXScript.

With Microsoft's Visual Studios 2005 and using "*C# To MaxScript Tutorial"* (Mr Bluesummers, 2008) as the underlining theory and framework behind the code, I programmed a default form and got this to open up and display within 3ds Max.

One of the advantages of using Visual Studios to create the interface was the more visual aspect of creating the interface as Visual Studios would create all the code for that control device as the position and size that were visually specified when creating the control as well as the parameters you wanted to change. Without this automatic code creation, each line would have had to have been programmed into MAXScript as well as the event handlers for every single control device.

During the design, it was quickly noticed that the Asset Management Browser would be required to control 3ds Max, or more precisely, create an object through MAXScript by passing parameters from the C# interface into MAXScript.

To speed up the processing time of the program, many smaller tasks were scripted into the C# side of the program with public functions that could be

called by MAXScript. This allowed for less code to be programmed within the MAXScript portion of the Asset Management Browser and more within the C# side. An example of this is the New_mesh_tree_node function, which creates a new tree node for the C# tree based off the MAXScript variables for the Directory and Name.

Fig7.1.1.1 is of the prototype C# interface for the Asset Management Browser

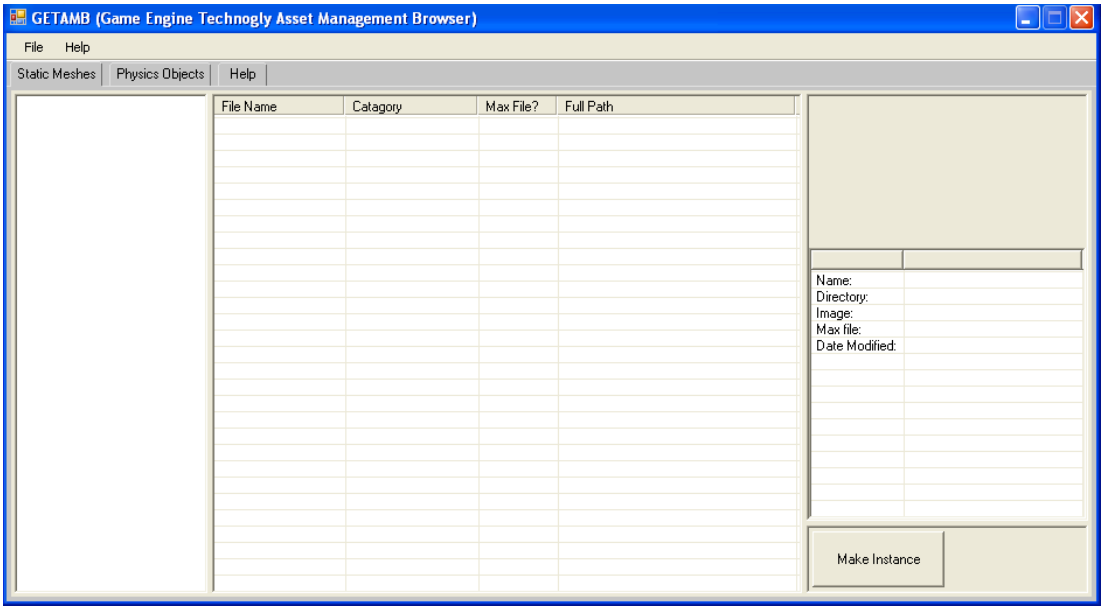


Fig 7.3.1.1 The C# interface for the Asset Management Browser.

7.3.2: Calling and Handing of the C# within Maxscript

The MAXScript element of the Asset Management Browser was for the creation and handling of the C# form, as well as for any functions which required the manipulation or creation of any object inside of 3ds Max.

There is no documented way of calling MAXScript functions directly through C#, so to accomplish this I used the .NET addevent handler in MAXScript to be able to assign a function to be called when a particular event happens. This was a result of trial and error as, to the best of my research, has not been any documented C# forms that have been integrated in with MAXScript. The idea came from a MAXScript called “AviPlayer” by Puech Yannick (http://ypuechweb.free.fr/Files/AviPlayer.mcr) where he used a

dotNet.AddEventHandler to a .NET timer control. This method worked perfectly, but did require the controls to be declared as global variables.

MAXScript was also used in the communication of different forms as C# uses instances of forms, and as the form was being instanced by MAXScript to create and display a copy, and without any research or documentation to help, seemed to be the quickest way. This was used for the importing of a new directory within the Asset Management Browser, as the import dialog was another form, and this was created and handled by MAXScript

4: Re-designing the Material Pipeline

7.4.1: Assessing and designing a more suitable Material Tool

After the first working prototype of the naming modifier was complete, and the stability of the functions and attributes were being tested, it was quickly apparent that the materials pipeline had not been thought out. The original concept was to allow users set material paths and attributes inside of the naming modifier and to convert them to a material, but during the initial prototype testing this method of creating and assigning materials took longer than using the syntax method.

Because of that reason, I felt the current material pipeline was not sufficient and required a complete redesign, considering the current methods of working with materials and how 3ds Max handles materials.

3ds Max handles materials with a Graphical User Interface called the Material Editor, which handles shader types; procedural mapping and standard bitmaps texture maps (Kalwick, 2004, pp. 283-285). The current method of creating materials for Gavin Wade's Game Engine Technology is to create all the materials within the material editor and to giving it a specific naming convention to tell the game engine what type of properties it is to apply to the material during the run time (Wade, 2007, p. 14).

With the current syntax-driven pipeline working with the standard material editor (Wade, 2007, p. 14), I looked at taking the current standard material

and expanding it, allowing for more parameters to encompass the options that Game Engine Technology uses. MAXScript allows a new scripted plug-in to be based off a current class (Autodesk(3), 2006), extending the current standard material class could be handled by MAXScript.

Looking at the requirements of Gavin Wade's Game Engine Technology and the current syntax-driven pipeline, I drew some concept designs for the User Interface to extend the standard material rollout.

Fig 7.4.1.1 shows the basic concept work for the Material Properties, which will determine what Boolean attributes to give the material. I wanted to use carry on the common theme of storing a group of similar Interface Devices inside a group box. The checkboxes were used as the checkbox control device can only store a Boolean value, which will determine if the material requires that syntax to be applied to the name.

To help keep the User Interface clean, the material properties and the shader properties were split up into different group boxes, and creating a visual difference between the two groups of settings.

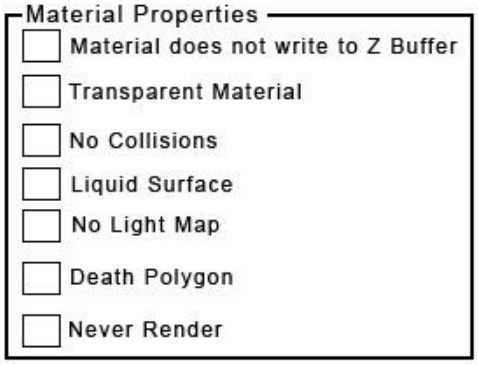


Fig 7.4.1.1 Concept art for the Material Properties section of the Material Extender

Fig 7.4.1.2 is the concept artwork for the shader portion of the extended standard material class. As with the material properties in Fig 7.4.1.1 the User Interface control devices are all encapsulated within a group box in order to keep similar items and devices together. The Shader Properties uses a similar technique as the portal properties in the Naming Modifier, where there is an

edit textbox device for the user to enter a name, but it is set as disabled until the user checks the checkbox above it.

As Gavin Wade's Game Engine Technology can accept three different types of shaders; Vertex, Pixel and .FX (Wade, 2007, p. 15), the concept design utilizes three different checkbox and edit textbox devices, one for each shader type.

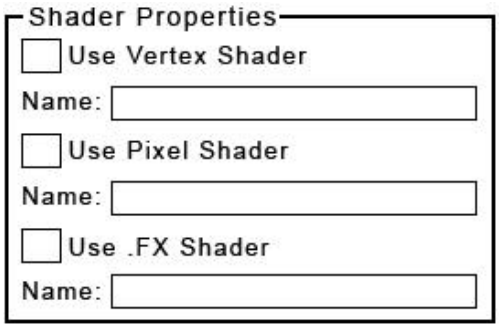


Fig 7.4.1.2 Concept art for the Shader Properties section of the Material Extender

7.4.2: Implementing re-designed the Material Tool

As the design of the new material tool suggests, it will be a MAXScript extension of the standard material class. The advantages of creating a new MAXScript material plug-in as an extension of an existing material is that it keeps all the old material properties, as well as add a new rollout User Interface.

The same framework that made up the naming modifier was used and modified slightly to become a material plug-in. The new script was created using group boxes, checkboxes and edit textboxes. Once the User Interface had be scripted, the User Interface device controls were linked in with a parameter block to hold the values, and the Ready for Export function was tweaked to be able to read these values and to add the appropriate syntaxes.

The inclusion of the enabling and disabling of edit textbox controls used with the shader names requires the running of a script when the rollout is opened. This calculates whether or not the controls should be enabled, as there is no documented way of directly linking in the enabled state to a parameter within

the parameter block. This is accomplished by the scripting of a checkbox with a Boolean value in the parameter block and calculating whether the appropriate devices should be enabled or not from the Boolean value.

One feature I added during the prototyping of this tool was the option for the user to add the '.FX' filename extension to the name of the .FX shader being applied. This feature is not apparent in the User Interface, but if the user does not add the '.FX' suffix, then it will be automatically added, but if the user did add it, it would not be added again.

Fig7.4.2.1 shows how the finished implemented material extension works and fits in with the current material editor and standard material setup and rollouts.

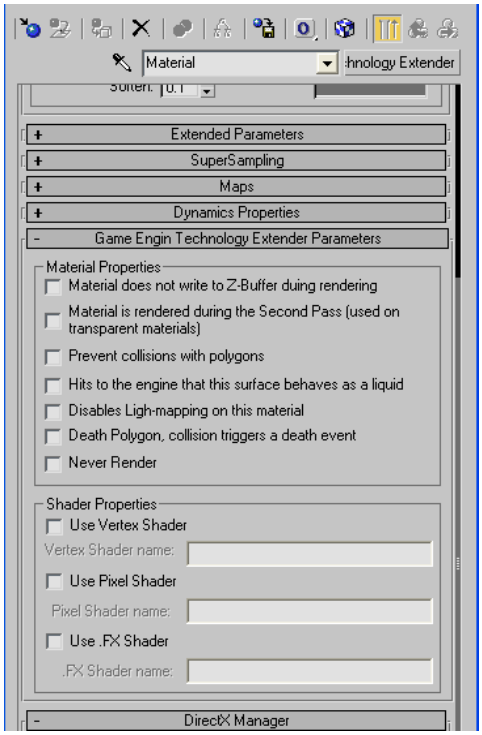


Fig 7.4.2.1The implemented material extender prototype.

7.5: The Initialization Script

As the tools had each been built as separate scripts in order to help in the very early prototyping of each of the tools, however, as the project progressed they stayed as separate scripts. As the working prototypes of each tool were being coded, it became more apparent that I would need a script to control the

tool scripts, to make sure all global properties required by the scripts were created and set up. The script written for this was more of an evolutionary prototype which grew over time to encompass more and more, such as the array of assigned zones used by the instance modifier.

The Ready for Export function was originally written and declared inside of the Naming Modifier, but expanded to work with materials and instance node objects. This generated problems when converting material and instance node objects. When the Ready for Export function was moved over to the new base script, the Initialization program, all problems associated with the function being out of range when being run from material and instance node objects, were remedied.

The initialization script was also a good place for the Call-backs required to handle the instance node's generated geometry to stop users from gaining any access to it. The Call-backs were made to do things such as delete the instance node's generated geometry if the node was deleted, and to keep the instance nodes generated geometry frozen at all times as well as hide or unhide the instance node's generated geometry based on the state of the instance node.

A macro script was also scripted here to allow easy access to the creation of the Game Engine Technology Asset Management Browser.

There are now five implemented tools, they will be collectively referred to as a toolkit as they all require one another to work and run.

Now will the toolkit at a working prototyping stage, the toolkit should be delivered to the participants of the Real-World testing phase to allow for the tools to evolve around the needs of the artists using them.

Chapter 8: Real-World Testing

8.1: Introduction to Real-World Testing

With the design and implementation of the tools completed to a working prototype stage, the tools were distributed to the two level 3 students who had volunteered to participate in the real-world testing, Subject A & Subject B.

The purpose of the real-world testing phase was to allow the tools to be tested in a real-world environment, being used to create levels and environments for Gavin Wade's Game Engine Technology. This would allow the tools to evolve to suit the needs and wants of the artists using them, as well as identify any bugs in the current build versions.

Both of the participants would be able to submit ideas for features they could find both useful and timesaving as well as complain about any current feature though to be annoying or nagging. These complaints and ideas would be fed into a list that would result in a new build of the toolkit each week, implementing all features asked for and removing as many bugs and annoyances as possible. The end result of the real-world testing is a stable toolkit build designed and focused for artists with as many time saving features as possible so to improve the workflow for content into Gavin Wade's Game Engine Technology.

8.2: Planning and participation

The allocation of time for the real-world testing phase of the project, as set out during the project planning phase, is 40 days altogether, but 30 days of real-world testing with a 30 day overlap for the implementation of the results from the real-world testing which will start 10 days after the start of the real-world testing.

The purpose for the real-world testing phases, both the real-world testing and the implementation of the real-world testing, to overlap for the majority of the time is so that the feedback from the participants can quickly and directly be incorporated into the tools developed. The new tools can then be fed back into

the real-world cycle so they can be tested and improved upon. This evolution life cycle methodology will, in theory, see the tools develop from a stable prototype build into a fully working build of the toolkit, designed around the needs of the artist.

The participations for the real-world testing will be the same people who were originally interviewed to establish the user requirements, Subject A and Subject B.

These two subjects are perfect candidates for the real-world testing as their own projects require building a game level for Gavin Wade's Game Engine Technology, so they will be able to push the tools in ways the tools might not have been conceived to do.

With the planning and participants accounted for, the real-world testing phase started.

8.3: Evaluation of Real-world testing

The Real-world testing phase ran for 40 days in total with 30 days for real-world testing and an overlapping 30 days of real-world implementation which started 10 days after the start of the real-world testing. The reason behind the 30 overlapping days of implementation was to provide a constant feedback loop, but the implementation was started and stopped 10 days after the real-world testing began to allow time for the tools to be tested before another build was brought out.

Out of the two students I had hoped would take part, Subject A & Subject B, only one student took an active role during the real-world testing. This was Subject A.

Subject A's feedback during the real-world testing was vital to creating more stable and useful tools. Every bug that was reported was fixed within 6-12 hours from being reported and each week a new build with requested or new features was released and distributed to both Subject A and Subject B via E-Mail.

An unplanned outcome of the real-world testing was the testing of game levels and environments which were using the tools, on computers without the tools. Due to this, new features were designed and implemented to allow for the conversions of new material and mesh types into standard types to allow full compatibilities with non-toolkit computers. One such feature was the 'Convert from Material Extender' function which was designed to collapse Game Engine Technology Material Extender type materials into standard 3ds Max materials with the chosen naming convention.

When the real-world testing was completed, both Subject A and Subject B were re-interviewed, but this time with different questions. The focus for Subject A was his experience of using the toolkit, how he felt about using the toolkit and if there were any problems or parts of the toolkit which were off putting.

For Subject B, the focus of the interview would be to find out why he had chosen not to use the toolkit to aid his project and to establish if there was any feature that put him off from using the toolkit.

When Subject A was asked if he had been using the toolkit, he replied saying *"it's definitely helped with how quickly my levels been progressing"*, and went on to state *"…your tools have definitely sped up the creation process by quite a factor"*. During the interview, Subject A commented on the difference between the new workflow compared to the current workflow *"I did have a go at using the traditional method and, it was slow to say the least"* and again *"it's increased my workflow and general speed"*.

This shows that the new workflow developed through this project is speeding up an artist during the level creation processes and is streamlining his own content pipeline into Gavin Wade's Game Engine Technology.

The next groups of questions for Subject A were about his perception of toolkit's Graphical User Interface and how it blends in, and works with the 3ds Max Graphical User Interface.

I asked how Subject A found navigating the toolkit, to which he replied *"it's very simple to use….nothing particular hidden about it"* and when asked about the Graphical User Interface of the toolkit compared to the Graphical User Interface style of 3ds Max he answered *"it's the same as [3ds] Max really" and "it blends in relatively nicely"*.

As the toolkit *"blends in relatively nicely"*, users familiar with the Graphical User Interface of 3ds Max should be able to grasp the Graphical User Interface of the toolkit, but as with 3ds Max, the pipeline would still need explaining.

Part of the real-world testing plan was to allow users to submit bug reports and request new features be added into the toolkit, so a question was asked on the promptness and regularity of the updates. Subject A's response to this question was *"I've been quite enjoying really being able to say 'Geoff, there's a bug in such and such' and getting a fix within about an hour."*, and shows that all bug reports were dealt with quickly as set out in the real-world testing planning stage.

Subject B had opted not to use the toolkit for their project, using the original method for naming objects and materials within 3ds Max for their level, and for this reason, the question first focused on why Subject B had opted for not using the toolkit.

Subject B's reason for not using the toolkit was *"…it is a new learning curve…and I thought it might [have] slowed me down"*. To understand why Subject B felt like that, the next question was aimed to gaining deeper understanding of this reason. The response to this was *"…make sure it wouldn't alter anything in my level or have to do something a little bit different…".* This highlights a potential cross-workflow problem as users cannot interchange between workflows seamlessly. Subject B went on to say *"…I was a bit unsure of what it might do to the other meshes in the level if I was to export it".* This would show more of a fear of non-compatibility between the two methods especially with the loss of work or time. Throughout the real-world testing phase, there was a lot of emphasis on ensuring the compatibility of the toolkit's end results, but there was no mention, until now, of the desire of a tool to convert data to the toolkit.

A tool to convert data onto the toolkit maybe a valuable feature to ensure a seamless cross-workflow switch, giving artists the choice on which workflow to use, and to allow them to change workflow's at any point in their development pipeline.

From the real-world testing, the toolkit has progressed through seven different builds, one each week starting with 0.01, and has grown in both features and in reliability, and has now become an artist-orientated toolkit.

With the real-world testing concluded, and with a stable and artist-orientated toolkit as the end result, the toolkit should be tested by complete novices all of whom should have no prior knowledge of Gavin Wade's Game Engine Technology, to ensure the toolkit has not been focused solely at the advanced user needs, but of the needs of all users using the toolkit and the Game Engine Technology.

Chapter 9: Novice Testing

9.1: Overview of the novice testing

The aim of the real-world testing phase was to allow the toolkit to evolve into a stable build and to ensure the tools were artist-oriented.

With the real-world testing concluded, the next phase should be about testing the toolkit on game artists of different disciplines, all with no prior experience or workflow knowledge of Gavin Wade's Game Engine Technology.

This would ensure that the tools are easy to use and can improve the workflow for student not familiar with the Game Engine Technology and the 'MIF' Manual, as well as checking to ensure the toolkit was not developed only for the more advanced user, but for all users.

As the toolkit is designed to eliminate the artists dependence on the "The '.MIF' manual", the underlying theory of what goes into a game level still is required, and as such, the workflow has not changed from how it is set out by "The '.MIF' manual".

The theory is important as it describes what a zone is, what a portal is and how a level object should be parented to each other in order to function properly as an optimized level. This type of documentation has been incorporated into the Game Engine Technology Asset Management Browser to allow users to have full and easy access to the level of documentation they want. Unfortunately the documentation has not been fully completed as the main focus of all the prior development was on the development and evolution of the toolkit for a more stable build, and this would have to be taken into account during the planning of the testing phase.

9.2: Planning and participation

In order to test the toolkits appeal and usability to artists with no prior Game Engine Technology, a testing of the toolkit would have to take place.

As one of the aims of the toolkit was for the all the tools to be as useable and accessible to new users, as experienced users, a testing of toolkit with complete novices would be necessary.

The participants of this test would have to be completely inexperienced with Gavin Wade's Game Engine Technology, but as the toolkit is integrated inside of 3ds Max, they would be required to have a basic working knowledge of 3ds Max and basic concepts of computer graphics principles. The participants would have to be completely inexperienced with Gavin Wade's Game Engine Technology to ensure that the toolkit was not just designed for advanced users, but for novice users as well.

This prior knowledge of 3ds Max would mean they could navigate the 3ds Max interface with ease and understand the generic workflow and systems native to 3ds Max.

As the participants of the novice testing would have no prior experience of Gavin Wade's Game Engine Technology, the theory of the technology will have to be taught to the participants so they can understand the different aspects of the game technology and appreciate where the toolkit comes in and the problems it solves.

Once the theory of the Game Engine Technology has been taught to the test group, the participants should be tasked with creating a level by using the toolkit. As the novice testing is evaluating the accessibility and ease of use for new users of the Game Engine Technology, it will not be necessary for them to create a level using the old syntax driven method, as this could, potentially, lead to confusion and extra time teaching the participants the syntax's for each object.

The level created by each of the participants should contain at least five zones with at least three portals, as well as seven instance meshes, seven world meshes and two lights and using at least one material. These criteria will allow for most of the Game Engine Technology toolkit to be used and tested with the most common options and configurations. As the toolkit does not require, or encompass the exporting or running of the level inside of Gavin Wades Game Engine Technology, the exporting or running of the level should not be part of the novice testing.

After the participants have completed their levels, they should be asked to answer a questionnaire on their time using the toolkit. The questionnaire should be designed before the novice testing to allow the testing to run smoothly.

9.3: Designing of the Novice Testing Questionnaire

The questionnaire should be designed to get information from the participants after completing their own level, the rough design and questions should be similar to the questions asked for the Primary Research (Chapter 4.1) but be the later questions should be focused on their experience. Unfortunately, due to a series of unforeseen technical circumstance, the questionnaire was designed at the last minute.

The first question should be related to what the subject considers themselves in a game's production pipeline. This will help establish an area of expertise and background knowledge.

The second question should be to check the subject's prior knowledge on Gavin Wade's Game Engine Technology. The answer to this question should be "None", as this would ensure all the data collected from that participants would be from a novice of the technology.

With the subjects background established, the next group of the questions should be about their experience using the toolkit. For this reason, question 3 should be directly aimed at this by asking the subject how they found using the toolkit. To follow on with their experience with the toolkit, they should be

asked how they found navigating around the toolkit and the toolkit's interface design.

As novices, and approaching the software with little knowledge of the Game Engine Technology, or the workflow employed by the technology, they could, potentially, do something with the toolkit that has not been done before and come across bugs which may not have been spotted. As this maybe the case, question five should ask the subject if there were any problems that they encountered whilst using the toolkit.

The last question should give the subjects an opportunity to leave any other comments about their experience using the toolkit.

With the design of the subject feedback questionnaire complete (Appendix E), the participants should now be tested using the novice testing plan and the feedback questionnaires analysed.

9.4: Evaluation of the novice testing

9.4.1: Intro and background questions

The novice testing had 6 participants, all from within the University of Portsmouth, from different course, of different years and of different specialism. This provided a great opportunity for the toolkit to be tested, bent and broken by complete novices of different disciplines.

The novice testing started off with a 15 minuets theory lecture, talking about the theory behind the creation of a level such as what a portal and a zone are. This was followed up a demonstration of a simple level creation by use of the toolkit to introduce the various tools and workflow to the participants. After this, each of the subjects were free to use a computer with the toolkit pre-installed with the default settings, so they could create their own levels. Once the subject was fished, they were given access to a copy of the feedback questionnaire to fill in.

The completed feedback questionnaire can be found in Appendix E.

Of the six subjects all were from different disciplines; however some had dual disciplines with some of dual disciplines overlapping. These disciplines included Game play designer, Animator, Programmer, Designer, Modeller and Character artist.

None of the participants had any previous experience with Gavin Wade's Game Engine Technology, which is what I had hoped for and expected.

9.4.2: Experience with the toolkit

Of the six participants, five participants said they found the toolkit *"easy"* or *"Useful"* when asked about their experience using the toolkit.

When the participants were asked about the intuitiveness of the interface and navigating round the toolkit, Subject 1 said it was *"As intuitive as MAX….no more complicated…"* and Subject 2 said it was *"…pretty intuitive…only if you already know the order and procedures needed to make a level"* which was an opinion shared by Subject 5 who said *"Missed one of the critical steps, but no errors showed up"*. This highlights a potential problem; the workflow must be learnt by the artist or the workflow must be hardcoded into a level creation wizard, but this would retract from the freedom the artists have for more of a novice friendly toolkit. Subject 4 did state that *"…adding some help files into the final product to assist the user"* and Subject 2 stated *"…would be impossible to figure out without at least some instructions"*, which shows that the pipeline and toolkit workflow could be taught to the artists with more detailed help files and in-depth tutorials.

The toolkit contains a popup message box that will pop up if the user tries to define a zone number that is currently in use. Subject 1 said the warning box *"…got a bit annoying after a while"* but Subject 3 said the message box was *"…helpful"*. This indicates that the purpose of the message box is useful and does its purpose, but the method it uses could be improved and made less intrusive.

When the participants were asked if there were any problems they encountered during the time whilst using the toolkit, Subject 4 and Subject 5 mentioned specific bugs which could be eliminated with more time to

implement failsafe loops, where an error message would appear instead of a 3ds Max MAXScript runtime error.

Subjects 1, Subject 3 and Subject 6 all gave ideas for features they would like to see implemented in the toolkit, ranging from an update button on the material tool to update the material name to a quicker, more streamline method to visualize an instance node mesh.

9.5: Conclusion of the novice testing

The novice testing showed that with a little teaching of the theory of level building for Gavin Wade's Game Engine Technology, complete novices can build a simple level with ease, but with more tutorials and help files on hand to the novices; a more complex level could be achieved.

All the participants in the test group were able to quickly and create a level using the tools developed as part of this project.

With the novice testing completed and evaluated, the toolkit should now be evaluated against the user aims and requirements set out at the start of this project to ensure that the tools developed meets the requirements set out.

Chapter 10: Evaluation against User Requirement

10.1: Re-statement of User Requirements

With the novice testing complete, the tools developed as part of this project should now be evaluated against the user requirements that were established in chapter 4.

When the Primary Research was completed, there were requirements and key problems which the tools developed were going to have to solve. One of these problems was to remove the artists dependences and the need to keep refereeing back to “The ‘.MIF’ Manual” for syntaxes. Another of these problems was the implementation of an asset management system to help artist keep track of their assets as well as a simpler exporting solution to speed up the exportation of a game level or mesh.

There were 3 tools designed to fulfil theses requirements and user needs, a Game Engine Technology modifier, allowing users to graphically set the object’s type and type-specific parameters. This tool was later split into two tools to be handle materials as well as meshes. The purpose of these tools were to eliminate the artist’s dependence on “The ‘.MIF’ Manual” for object and parameter syntaxes.

The second tool to be designed and implemented was the Game Engine Technology Asset Management Browser (GETAMB) which allowed users to graphically pick previously created assets to use in their levels as instance meshes, as well as contain functions like the one click export button.

The last of the tools to be designed and developed to fulfil the user requirements, is the Instance node. As states in Chapter 6.2, the Instance node was not a standalone tool, but to be used in conjunction with the Game Engine Asset Management Browser to hold and handle an instance mesh, and would fill the gap between the Game Engine Technology modifier and the Game Engine Asset Management Browser.

With the problems, user requirements and the tools developed re-stated, the tools should now be evaluated against these user requirements and key problems.

10.2: Toolkit Evaluation

As stated in the chapter 10.1, each tool was designed to fulfil a key problem or a user requirement, and now these tools will be evaluated against these requirements.

In order to help new users feel more comfortable when first using the toolkit, the tools were designed to follow 3ds Max's interface themes and styles. Subject 3 said the *"…interface was really easy to navigate…easy user interface…simple and not too cluttered"* and Subject A, from the Real-World testing phase, said the interface was *"…pretty functional…same as [3ds] Max really…it blends in relatively nicely"* which shows that the interface is simple and easy to use, as well as following 3ds Max's interface themes and styles.

10.2.1: Modifier and Material Extender Evaluation

The first tools to be evaluated are the Game Engine Technology modifier and the Game Engine Technology material extender. The Game Engine Technology modifier was designed with the purpose of removing the artist's dependency on "The '.MIF' Manual" for the syntaxes of objects and properties, as well as to cut down on the amount of mistakes created by human spelling errors. The Game Engine Technology material extender was designed to provide a Graphical User Interface for the quick changing of material properties and shaders in a similar manor to the Game Engine Technology modifier does with objects.

Subject 3 from the Novice testing said *"…gives people without the knowledge of the game engine the ability to create a level without having to know the syntax"* which shows that both of these tools, as well as the Game Engine Technology Asset Management Browser and Game Engine Technology Instance Node, all considerably reduce the artists dependence on "The '.MIF' Manual" for syntaxes. The Game Engine Technology modifier was also said to save artists

time by both Subject A (Follow-up interview) and Subject 3, with Subject A saying *"it's definitely improved the speed of the implementation by far"* and Subject 3 saying *"…definitely better than having to rename every object…it will definitely save you time"*.

The framework for the Game Engine Technology modifier proved extremely flexible, taking only 15 minutes to add and script in the Graphical User Interface, the event handlers and the converting code within the Ready for Export function for a whole new type of mesh. The Game Engine Technology modifier was extended during the real-world testing phase to encompass 3 new mesh types; light, collisions objects and physics objects.

10.2.2: Asset Browser and Instance Node Evaluation

The Game Engine Technology Asset Management Browser and Game Engine Technology Instance Node should be evaluated together as both tools were designed to work together, but both can be used separately if necessary.

The aim of the Game Engine Technology Asset Management Browser was to allow users to graphically pick previously created assets to use in their levels as instance meshes, as well as containing functions like the one-click-export button. The objects the user wanted to import as instances would be generated onto a Game Engine Technology Instance Node to handle the location and transformation matrices, to be applied to the object inside the game engine at runtime as well as the file location and file name which is required for generating an appropriate syntax name for the object.

The Game Engine Technology Instance Node was a tool which was not tweaked much during the real-world testing as there were no additional features that could be updated to it. A quick pre-visualization feature was requested by Subject 6 during the Novice testing phase. This feature was implemented, but did require the function to visualize an instance node to be placed in the initialization script, rather than being in the instance node script.

10.3: Toolkit Conclusion

10.3.1: Modifier and Material Extender Conclusion

If I was to attempt this project again, I would have invested more time in the initial framework of the Game Engine Technology modifier so that each type of mesh had its own rollout interface instead of enabling and disabling controls based off the mesh type. This would remove a lot of code when the rollout is opened and the type of mesh is changed as it currently needs to check which sections to enable and which to disable. This would bring the Game Engine Technology modifier into line with the interface for large modifiers such as the Edit Mesh modifier and the Skin modifier, but the current Game Engine Technology modifier Graphical User Interface is in line with the Cloth modifier and the Unwrap UVW modifier.

An unexpected outcome of the Game Engine Technology modifier is the automation of small re-naming tasks, as each object would get evaluated by the Ready for Export function to give it a naming convention required by Gavin Wade's Game Engine Technology, automating this process.

The Game Engine Technology material extender's frame-work is very flexible and expandable, as new functions to convert the material into a standard material and the function to update the material name were added in less than 10 minutes. Due to the way the class system is used in the creation of MAXScript materials, adding a new properties is relatively simple, requiring no more than 2 lines of code in the Game Engine Technology material extender script and an additional 2 lines of code in the material naming conversion part of the Ready for Export function with those 2 lines also going in the material conversion to a standard material function.

The material pipeline could be optimized further and could be better designed from a programming perspective if the project was to be attempted again. The problems with the current material conversion is that there are two functions, one to convert the name and one to convert the name and convert it into a standard material. As much of these functions perform the same task, they

could have been condensed into one function with different parameters within the function depend on how it was called. This was due to a feature to convert a Game Engine Technology extended material into a standard material which was only request near the end of the real-world testing, and the functions were kept apart to avoid any confusion.

10.3.2: Asset Browser and Instance Node Conclusion

The Engine Technology Asset Management Browser was designed with the idea of being able to house some of the more out-the-way features such as a scripting section and the help features. The scripting section of the Engine Technology Asset Management Browser was removed due to lack of purpose. The Engine Technology Asset Management Browser could have used a greater focus and purpose to it rather than just a collection of functions. At the end of the real-world testing, new placeholder tabs were introduced for features to be added to the Engine Technology Asset Management Browser if time and expansion to the MIF exporter allowed.

The Engine Technology Asset Management Browser was original designed to have the one click export function to help speed up the exporting of a mesh during production, but this feature, if implemented, would have by passed the Gavin Wade's MIF exporter license agreement, and therefore cannot be implemented for possible non-agreement with the terms and conditions of Gavin Wade's MIF exporter.

These features, such as the placeholders feature and the scene browser, are similar to the features also seen in the UnrealEd asset browser system. If I was to undertake this project again, I would have spent more time developing the concept for the Engine Technology Asset Management Browser and the features and tools it would house, as developing new features would have been easier during the implantation stage.

With all of the tools evaluated, the next part will be the conclusion of the overall project.

Chapter11: Project Conclusion

11.1: Re-statement of Initial problem

The original aims and goals of the project, before any research, was to design, develop and implement a tool or tools within Autodesk's 3ds Max to improve and workflow when developing a game level for use in Gavin Wade's Game Engine Technology.

The workflow was designed to be improved by automating simple repetitive tasks, giving the user a simple Graphical User Interface controls to use in the creation of the syntax based naming conventions required and through that, speeding up the assigning of object and material properties for exportation.

A limiting factor for this project, was that the tools developed had to work in conjunction with Gavin Wade's MIF exporter.

The tools developed must be as useable and accessible to new novice users as to experienced users.

11.2: Solving of the initial problem

The initial problem was to design and implement a tool or tools to speed up the workflow for implementing content for use in Gavin Wade's Game Engine Technology. A total for four tools were developed to improve the workflow. These tools were designed and developed after Primary Research was conducted and were evaluated and concluded in the whole of Chapter 10.2 against the design criteria created as a result of the Primary Research.

The tools developed to improve the workflow has *"…increased my workflow and general speed by quite a factor"* as stated by Subject A in the follow up interview. This proves that the workflow was improved in terms of speed. Subject 1, Subject 3, Subject 4 and Subject 6 all commented on how it was quicker and easier that the original workflow. The toolkit also automates the naming of multiplies of the same node, as well as the overall naming of each node based off the input from the Graphical User Interface, although much of this feature goes unnoticed and because of this, it becomes a background feature.

One of the aims of the project was to allow for users of all abilities to use this toolkit to the same effectiveness. In this regard, the tools developed have proven to be relatively novice-friendly allowing for new users to build simple levels once they were taught the basic underlying theory of what goes into a game level for Gavin Wade's Game Engine Technology.

11.3: Project Conclusion

The tools developed as part of the project have all been developed to help aid the artist in creating new content for use in Gavin Wade's Game Engine Technology by reducing the need for referring to "The '.MIF' Manual" for the correct syntax for the object, but also helping to streamline their workflow by creating Instance nodes to handle the displaying of assets to be instanced in-engine as well as an asset management system. The tools developed have been used by a real-world artist to create content for use in Gavin Wade's Game Engine Technology as well as by novices to create their first game level having had only a quick demonstration and lecture of the theory behind the Game Engine Technology.

During the design of the tools, there could have been more emphasis on the programming backbone, expanding similar functions to stop code having to be repeated with only a minor change.

The design stage could also have benefited from more practical use of the current Game Engine Technology to really understand what the artist need when creating their levels. This level of understanding could have proved time saving and insightful when designing the Game Engine Technology material extender and the Game Engine Technology Asset Management Browser.

If the real-world testing phase had more artists taking participating, then the tools developed might be more encompassing, covering topics not mentioned in "The '.MIF' Manual" and even requiring more features to be added to the MIF exporter to meet the demands of the toolkit.

With only one artist using the toolkit developed, it brings in to question; whether it took longer to develop the toolkit than the amount the toolkit saved

this artist? As the toolkit, and all its source code and documentation will be left at the University and will be made available to all staff and students to use as a learning or teaching resource, the toolkit could be used by other staff and students and save them time or inspire staff and students to build and expand this toolkit in order to help more artists.

I think for that reason, the time taken to develop the toolkit has been beneficial, as it may have taken longer to develop than was saved by the one real-world testing subject using it, but it has the potential to help more artists and save more time.

11.4: Recommendations for further improvements

With this project concluded, there is still room for improvement to increase its appeal and capabilities to help improve the workflow and perhaps bring the toolkit to match the functionality of a professional level editor, such as UnrealEd. These features I would like to recommend for further improvement and expansion of the toolkit, are features seen in UnrealEd, such as a simplistic graphical scripting system, giving artists the ability to script simple events without having to use Visual Studio and edit any of the C++ code.

Another feature is the inclusion of Gavin Wade's Game Engine Technology as a viewport render device for 3ds Max, letting artist see what the level will look inside the level, as they build it. This would help create a real-time feedback loop with the artist as they see their level as the player would see it.

11.5: My Personal Development

Looking back at the last 32 weeks of this project, I can see areas I could have improved on, or condensed, but which were not obvious at the time. Areas such as the programming of the initialization script which, in the end, grew from a small script to start all the other scripts, to the longest of all the script, at the very heart of the toolkit.

I learnt a lot about focusing on what artist want, rather than my perception of what I think artists want, as this led to several ideas and concepts which were implemented and later removed as artists did not need them, one example

being the scripting page within the Game Engine Technology Asset Management Browser.

I think I learnt the most about the job and role of a tools programmer, and how the job fits into the production of a game, supporting and improving their pipelines rather than unnecessarily trying to add more confusing elements or packages to the pipelines.

Overall I am very pleased with the outcome and development of the toolkit and how it has evolved, and that this project has been able to help at least one person.

References

Autodesk(1). (2006). *Rollout User-Interface Control Types*. Retrieved March 26, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/

Autodesk(2). (2006). *Rollout User-Interface Controls Common Properties*. Retrieved March 26, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/

Autodesk(3). (2006). *Scripted Plug-ins*. Retrieved March 27, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/

Autodesk(4). (2006). *Updating Scripted Plug-ins*. Retrieved April 24, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/updating_scripted_plugins.htm

Autodesk(5). (2006). *MAXScript Overview*. Retrieved April 23, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/maxscript_overview.htm

Autodesk(7). (2006). *TabStrip ActiveX Control*. Retrieved March 26, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/

Autodesk(8). (2006). *The SDK and MAXScript*. Retrieved April 24, 2009, from Autodesk MAXScript reference 9: http://www.cguu.com/3dsmax/3dSMAX9/maxscript/the_sdk_and_maxscript.htm

Burback, R. L. (1997, July 30). *Classical Waterfall Methodology*. Retrieved 4 17, 2009, from Software Engineering Methodology: The WaterSluice: http://infolab.stanford.edu/~burback/water_sluice/sluice6.25.97/ws/node50.html

Cottrell, S. (2003). *The Study Skills Handbook* (2nd Edition ed.). Palgrave Study Guides.

Crytek GmbH. (2008). *Editor Interface Overview*. Retrieved March 26, 2009, from CryENGINE 2 Documentation overiew: http://doc.crymod.com/SandboxManual/frames.html?frmname=topic&frmfile=index.html

Epic Games. (2008). *Current Technology - Unreal Engine 3*. Retrieved March 24, 2009, from Unreal Technology: http://www.unrealtechnology.com/technology.php

Epic Games. (2008). *Rendering*. Retrieved March 26, 2009, from Unreal Technology: http://www.unrealtechnology.com/features.php?ref=rendering

Epic Games. (2009). *Success Stories - Unreal Engine 3*. Retrieved March 25, 2009, from Unreal Technology: http://www.unrealtechnology.com/success-stories.php

Flynt, J., & Booth, B. *Unreal Tournament Game Programming for Teens.* 2006.

Flynt, J., & Omar, S. (2004). *Software Engineering for Game Developers.* Course Technology.

Gauss, R. (2008, January 24). *Doing it Yourself - The History and Future of the "Level Editor"*. Retrieved March 26, 2009, from Gameing Steve.com: http://www.gamingsteve.com/archives/2008/01/level-editors-wanted.php

Gibbs, D. A. (1997). Issue 19: Focus Groups. *Social Research Update* .

Hague, P. N., & Hague, N. (2004). *Market Research in Practice : A Guide to the Basics.* Kogan Page, Limited.

Hague, P., Adams, K., & Brace, I. (2006). *Introduction to Market and Social Research.* Kogan Page, Limited.

Kalwick, D. (2004). *3ds max 6 Essentials : A Real-World Approach.* Charles River Media.

Koch, A. (2004). *Agile Software Development : Evaluating the Methods for Your Organization.* Artech House, Incorporated.

Koch, A. (2004). *Agile Software Development : Evaluating the Methods for Your Organization.* Artech House, Incorporated.

LittleBigPlanet Wiki. (2009, March 23). *Level Creator Guide*. Retrieved March 25, 2009, from Little Big Planet Wiki: http://littlebigplanet.wikia.com/wiki/Level_Creator_Guide#Level_Size_.26_Grid

Liu, L., & Roussev, B. (2005). *Management of the Object-Oriented Development Process.* IGI Global.

Masse, R. E. (1998). Evolutionary Prototyping: "Add Later" Static Types for Python. *International Python Conference.*

McCombs, S., & Rabun, K. (2008, September 9). *Integrating MaxScript and .NET Systems*. Retrieved March 25, 2009, from Gamasutra: http://www.gamasutra.com/view/feature/3780/integrating_maxscript_and_net_.php

McTaggart, G. (2004). Half-Life 2 / Valve Source Shading. *Game Developer's Confrence*, (p. 19).

Microsoft Corporation. (2009). *Tabs*. Retrieved March 26, 2009, from Microsoft Developer Network: http://msdn.microsoft.com/en-us/library/aa511493.aspx

Mitchell, J. (2006). Shading in Valve's Source Engine. *Special Intrest Group on Graphics and interactive Techniques (SIGGRAPH).* Boston: Special Intrest Group on Graphics and interactive Techniques (SIGGRAPH).

Mr Bluesummers. (2008, October 2). *C# To Maxscript Tutorial*. Retrieved March 26, 2009, from Mr Bluesummers: http://www.mrbluesummers.com/?cat=11&paged=8

National Aeronautics and Space Administration. (2004, 1 29). *The Standard Waterfall Model for Systems Development*. Retrieved 3 2, 2009, from Web Archive: http://web.archive.org/web/20040403211247/http://asd-www.larc.nasa.gov/barkstrom/public/The_Standard_Waterfall_Model_For_Systems_Development.htm

Neale, P. (n.d.). *Max Script / DotNet Listview Tutorial:*. Retrieved March 26, 2009, from PaulNeale.com: http://paulneale.com/tutorials/dotNet/dotNet.htm

Pletcher, A. (2008). python for technical artists. *Game Developers Conference.* San Francisco: Game Developers Conference.

Roecode. (2008, March 17). *GameEngine Development (Part 19, Hardware instancing)*. Retrieved March 26, 2009, from Running on empty: http://roecode.wordpress.com/2008/03/17/xna-framework-gameengine-development-part-19-hardware-instancing-pc-only/

Scheuren, F. (2004). *What is a Survey*. Retrieved April 13, 2009, from American Statistical Association: http://www.webcitation.org/5Zp6d10re

Soy, S. (1997). The Case Study as a Research Method. *Library and information science* .

St-Laurent, S. (2004). *Shaders for Game Programmers and Artists.* Thomson.

The Reek Todd.com. (2008). *Lode Runner Level Editor Tutorial*. Retrieved March 26, 2009, from The Reek Todd.com: http://www.thereeltodd.com/2008/08/lode-runner-level-editor-tutor.html

Tim Sweeney (Epic MegaGames, Inc.). (1999, July 07). *Unreal Zones*. Retrieved March 25, 2009, from Unreal: http://unreal.epicgames.com/Zones.htm

Valve Corporation. (2007). *Source Engine Environments*. Retrieved March 26, 2009, from Valve Business Solutions: http://source.valvesoftware.com/environments.php

Wade, G. (2007, March). The '.MIF' manual. *The '.MIF' manual* . Portsmouth, Hampshire, England: The University of Portsmouth.

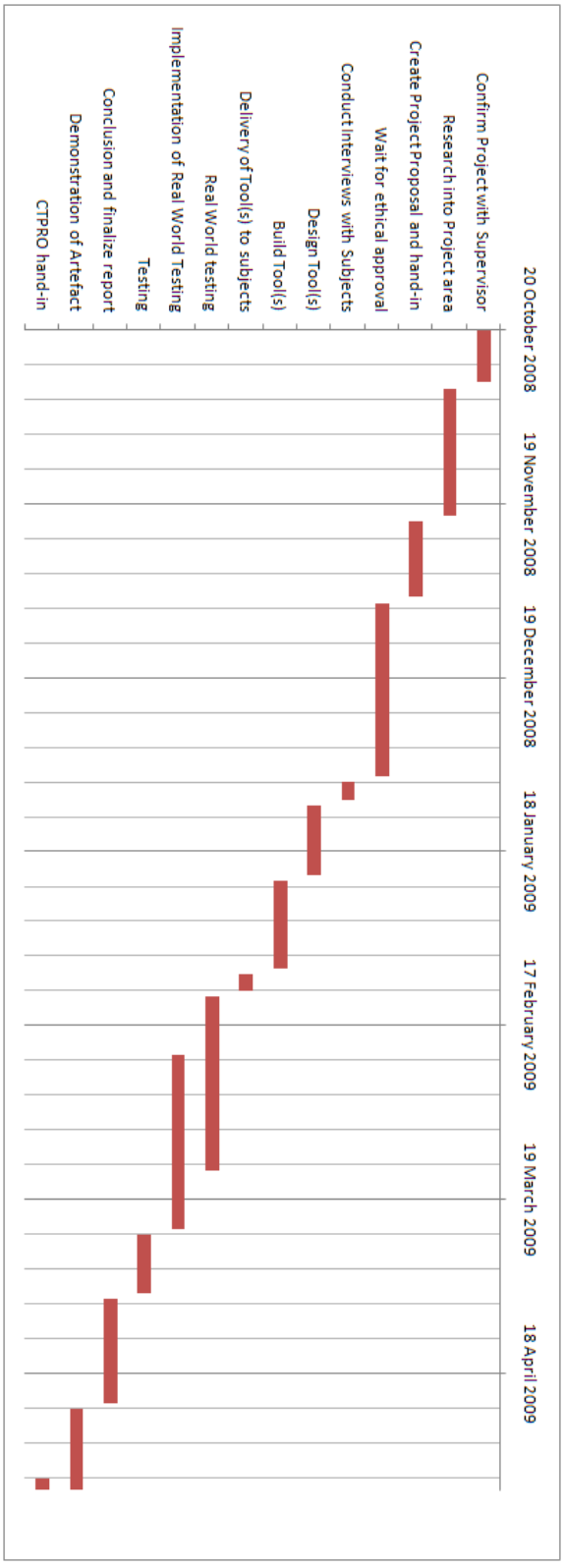
Welsh National Assembly: Department for Education. (n.d.). *Primary Research Methods*. Retrieved November 2007, 2006, from Welsh National Assembly: http://www.elwa.ac.uk/elwaweb/elwa.aspx?pageid=2097

Wihlidal, G., & Whitaker, A. (2006). *Game Engine Toolset Development* (1st Edition ed.). Course Technology Inc.

Wikimedia Foundation, Inc. (2009, March 24). *List of level editors - Wikipedia*. Retrieved March 26, 2009, from Wikipedia: http://en.wikipedia.org/wiki/List_of_level_editors

Zerbst, S., & Duevel, O. (2004). *3D Game Engine Programming.*

Appendix A: Project management - Gantt chart



Appendix B: Interview Questions

- Interviewee's current course.
- Interviewee's course level.
- What the Interviewee considers them selves' within a games pipeline.
- Interviewee experience in using Game Engine.
- The Interviewee experience with programming.
- The Interviewee first impressions of using Gavin Wade's Game Engine.
- What Interviewee views on "The '.MIF' Manual".
- Any other comments on Gavin Wade's Game Engine.

Appendix C: Initial Interview Transcripts

Initial Interview Transcripts - Subject A

Interviewer: "…your course?"

Subject A: "Computer Games Technology"

Interviewer: "And what year are you on please?"

Subject A: "Year Three"

Interviewer: "Awesome, so what would you consider yourself in a games pipeline? Be it a Modeller, Animator..."

Subject A: "3D Modeller "

Interviewer: "Modeller?"

Subject A: "yup"

Interviewer: "ok, so you're coming into this as a 3D Modeller, so do you have much experience in coding?"

Subject A: "It's very, very limited, mostly what I've learnt here at the uni[versity]"

Interviewer: "And what kind of languages do you know?"

Subject A: "Only C++, and as I said, very basically"

Interviewer: "ok, so what about your experiences using game engine technologies, such as professional game engine like the unreal 3 engine or open source engines like the ogre engine?"

Subject A: "right, well I've used a variety of engines. I've used, from varies mod projects I've been on, one of which, multiple ones for the source engine, developed by Valve, one that was developed for the unreal 3 engine, that went nowhere as they usually do, also the Ogre engine which I'm currently

doing, using in RTGROP, Gavin's game engine for example, and there's another one that I can't remember the name of, but that was the first engine I used and it possible the most horrible thing I've ever used."

Interviewer: "ok, so as coming into this as a game artist, with quite a bit of experience using game engines, is your experience using the GUI editors that they come with or creating your own tools or programmers' to work with the engines?"

Subject A: "I'm definitely on the side of working with the GUI stuff because as I said, I am very, very inexperienced when it comes to writing my own code."

Interviewer: "Could you give us an example of a GUI that you use, game engine GUI?"

Subject A: "Unreal 3 editor, is probably one of my favourite ones that I've used so far"

Interviewer: "Ok, so with all the experience, what are your first impressions coming into Gav[in]'s Engine? Using this particular engine?"

Subject A: "It's a nice engine, the only, probably major problem I have with it, is it's not particularly intuitive. That's not so much of a problem because I understand how it work, but I can see how someone that didn't have a particularly great deal of experience if they were coming into this, they were going to have problems understanding how everything works. It's quite daunting when you have to create your meshes and you have to type dollar S one whatever to create a static mesh and then instance it x many times, it can get very, very complicated and it does get very, very messy very quickly."

Interviewer: "Why is that? Why does it get messy, is it because of the naming conventions?"

Subject A: "It's definitely the naming conventions, I mean by different really, the Unreal 3 editor is much, much more adaptive, no sorry intuitive even. As you could, it made things much easier as it had an asset browser there, and u

could just illiterately go ok here is the basic level mesh, and I'm going to plot this this in and its instanced already for you and you haven't got to touch code if you don't want to."

Interviewer: "Ok, so moving on another point now with the MIF manual. You have a copy of the "The '.MIF' manual" in front of you."*

Subject A: "yup"

Interviewer: "When you're using it to export your meshes and to make your level, do you find you have to stop and actually flick through it to find the right syntaxes, or do you find they just start to sink in after a while?

Subject A: "I mean to start with it, because I'm working on this project and one of my friends, [*Subject B]*, is doing something very, very similar, we're constantly talking to each other, saying "what's the syntax for this", "what's the syntax for this" and each one of us seems to have sort of half the picture. So we're constantly going back to the "The '*.MIF' manual" and looking. I mean a while it only natural that it settles in however its still, I mean it's sort of like and ... it's a wall you have to overcome to actually get used to this engine. "

Interviewer: "Any other comments you would like to make about the game engine or the kind of things you would like to see to do with it, to improve its usability?"

Subject A: "It'll be nice to have a similar editor to Unreal 3, I mean that's just personal preference really, I mean I'm sure that there are other people would say, "oh Hammers a much more superior editor", but I mean I personally find, because Unreal 3 is everything is there in front of you, it makes it, it's much, much more intuitive to use, so then, if there was something like that made, it would be ideal for a new user to use."

Interviewer: "Ok, Thank you very much for your time..."

Initial Interview Transcripts - Subject B

Interviewer: "what course are you on?"

Subject B: "Computer games tech."

Interviewer: "And what year are you on?"

Subject B: "[Year] Three."

Interviewer: "Sweet, so what do you consider yourself in a games pipeline?"

Subject B: "Environment artist."

Interviewer: "Ok, What about your experience in code? Are you a coder? Can you code?"

Subject B: "I can code a little. A little Visual C++, small bit, but mostly Visual Basic."

Interviewer: "So V[isal]B[asic] and C++. And what kind of experience working with other game engines, either open source or GUI based like the Unreal 3 editor or the OGRE engine etc."

Subject B: "Just sort of modification game engines like the Source Engine and Unreal Ed, nothing that involves too much additional coding on top of it so I can just drop all my assets into and then go on."

Interviewer: "So you're very much used to using a..."

Subject B : "Standard toolset graphical editor."

Interviewer: "Ok, have you ever used a game engine where you've had to code everything?"

Subject B: "No, Never."

Interviewer: "Ok, that's awesome, so your project, I understand, is using Gavin's engine. What were your first impressions of it?"

Subject B: “Pretty good from what I saw from the start up.”

Interviewer: “Not daunting at all, with the whole coding aspect without a GUI editor?”

Subject B: “On and off really, scripting was a little bit difficult start with, but when you pick it up, it can breeze pretty through, but it is kind of annoying to keep having to type stuff in. Obviously you could speed it up with different parts.”

Interviewer: “Ok, Let’s talk a bit about the “The ‘.MIF’ manual”. Obviously this is the holy bible of Gavin’s Engine from your point of view, what did you think about it when you first read it?”*

Subject B: “Shocking, doesn’t give much help at all really.”

Interviewer: “So would you prefer something that was a lot more concise that kind of crushed it down into...”

Subject B: “There seems to be too much in certain areas and not enough on other concentrating like in instancing... and other sections that were needed.”

Interviewer: “Ok, so what would you want to see implemented into a graphic editor or a new toolset or plug-in to help you, an artist in your own words, use Gavin’s Engine more accessibly?”

Subject B: “Easy access to all my files to get them in and out of the engine as quickly as possible; simple exportation; no real need to change files if I have to. So just kind of like a “click save” does it all for me, that sort of thing because obviously continually changing a file you could export something wrong, which might not work.”

Interviewer: “Ok, anything else you would want to see in there?”

Subject B: “Nope.”

Interviewer: “Ok, thank you very much for your time.”

Appendix D: Follow-up Interview Transcripts

Follow-up Interview Transcripts - Subject A

Interviewer: "so are you still using the game engine?"

Subject A: "yup. yeah its working out quite nicely. I'm getting a good few nice results ".

Interviewer: "Have you been using the toolkit during your time for the last six weeks?"

Subject A: "yeah, it's defiantly helped with how quickly my levels been progressing, from literary nothing to what it is now, which is semi populated. It's actually started to look really quite nice now so I'm quite pleased with that."

Interviewer: "And that's to do with the toolkit?"

Subject A: "Yeah defiantly, I did have a go at using the traditional method, and, it was slow to say the least, so yeah your tools have defiantly sped up the creation process by quite a factor."

Interviewer: "Good. Did you find the installation of the toolkit particularly easy or did you find it very nagging?"

Subject A: "I wouldn't say it was difficult, and I wouldn't say it was nagging, it can be a bit annoying so, it's very simple to install, it's a matter of sticking two files really, one folder and one file into the correct places within the MAX folder, which is pretty easy to do, I can see how some people can have trouble with that, deleting vital files and then complaining to you when things aren't working. So yeah."

Interviewer: "ok, did you find using and navigating the toolkit easy?"

Subject A: "yeah, very, very simple really. Everything's there when you want it, there's nothing particular hidden about it, it's very simple to use."

Interviewer: "Would you say the graphical user interface of the toolkit suits and matches the graphical user interface style of 3D Studio Max?"

Subject A: "yeah, it's pretty functional; it's the same as Max really. It's nothing particular bad about it, there's nothing particular nice looking about it, but that's how Max looks, so it blends in relatively nicely."

Interviewer: "ok, in your opinion, has the toolkit improved your content pipeline with Gavin Wade's Game Engine Technology?"

Subject A: "oh yeah definitely, as I said earlier, it's increased my workflow and general speed by quite a factor."

Interviewer: "Has it only improved with speed, or has there been any other ways it's improved your content pipeline?"

Subject A: "it's definitely improved the speed of the implementation by far. I mean I was able to rough out a level in about a day in oppose to a good week before hand. I don't know if that was just me being slow or the level, but I'm saying your tools have definitely helped me".

Interviewer: "Did you find that updates were regular and timely in accordance with your bug reports?"

Subject A: "Yeah, defiantly. I've been quite enjoying really being able to say 'Geoff there's a bug in such and such' and getting a fix within about an hour, which is very impressive to say the least".

Interviewer: "Have you enjoyed your time using the toolkit?"

Subject A: "Yeah it's been relatively fun. I can't really elaborate any more on that".

Interviewer: "Are there any additional comments or issues you would like to raise within this interview concerning the real-world testing?"

Subject A: “no, there’s no real issues. There a few nagging things, I wouldn’t say they were bug, more annoyances than something. But that’s something to be smoothed out in the testing”.

Interviewer: “ok, thank you for your time”

Follow-up Interview Transcripts - Subject B

Interviewer: "are you still working on the level for your project"

Subject B: "I'm still working on the level, yeah ".

Interviewer: "Have you had a chance to use the toolkit?"

Subject B: "I've used it a few times at the start and I was sort of put off and stopped using it and kept on with the original methods ".

Interviewer: "So what put you off then?"

Subject B: "That it is a new learning curve to go through, and I thought it might of slowed me down ".

Interviewer: "Was there anything daunting about this new toolkit, the tools?"

Subject B: "Not particularly, it's just having to start again and make sure it wouldn't alter anything in my level or have to do something a little bit different and it might just put me out of sync while I was working along through it".

Interviewer: "So would you have preferred some kind of feature to extract the naming convention and create nodes and such based of what you've already done? Take an object, say a static mesh and put a static mesh modifier on that. Do you think that would of helped do you think?"

Subject B: "Well if I knew it wouldn't have any impact on what I'd already created, I definitely think so, yeah. I was a bit unsure on what it might do to the other meshes in the level if I was to export it.".

Interviewer: "So is that the only real concern?"

Subject B: "That's probably my biggest concern, yeah ".

Interviewer: "Where there any features, that you heard about that you did like about the toolkit?"

Subject B: “Yeah the image, ‘insertation’ actually, if it’s still in there. Where you were going to have a screen capture of the mesh, if that’s still in there”.

Interviewer: “Fair enough”

Subject B: “I don’t know why, it is very interesting”.

Interviewer: “Anything else you wanted to comment on?”

Subject B: “That if I had started the project later on, then yeah I would of used more of the toolkit, rather than of kept with the original method as I had already started with the original method. As I said, difficult to break a habit once you’ve started”.

Interviewer: “So did you actually have a chance to use the toolkit then?”

Subject B: “I used it once or twice but not heavily in the main project, so mainly for testing and bugs purposes.”

Interviewer: “So when you were testing it,… how did you find the interface, when you were using an early version of it[toolkit]”

Subject B: “It was a bit on and off when I first used it, but from what I can tell it should be a lot better now in the final release.”

Interviewer: “ok, thank you for your time”

Appendix E: Testing survey

Novice testing survey - Questionnaire

Q1. What role would you consider yourself in a game production pipeline?

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

Q3. How did you find using the toolkit?

Q4. How intuitive is the interface and navigating the toolkit?

Q5. Where there any problems you encounter whilst using the toolkit?

Q6. Any comments you would like to make??

Novice testing survey - Subject 1

Q1. What role would you consider yourself in a game production pipeline?

Gameplay designer

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

none

Q3. How did you find using the toolkit?

Quite easy, would be nice to be able to view the end result, and some functions such as an update button on the material editor would be nice

Q4. How intuitive is the interface and navigating the toolkit?

As intuitive as Max is normally, no more complicated etc.

Q5. Where there any problems you encounter whilst using the toolkit?

Yes but it was because of me forgetting what to do rather than being a fault of the toolkit

The "zone already defined" warning box got a bit annoying after a while.

Q6. Any comments you would like to make??

Very nice, looks a lot easier than the existing method

Novice testing survey - Subject 2

Q1. What role would you consider yourself in a game production pipeline?

Animator

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

None

Q3. How did you find using the toolkit?

The tools were all useful and easy, but would be impossible to figure out without at least some instructions

Q4. How intuitive is the interface and navigating the toolkit?

It's pretty intuitive, but only if you already know the order and procedures needed to make a level

Q5. Where there any problems you encounter whilst using the toolkit?

None. Once I knew what I was doing it was very simple and easy

Q6. Any comments you would like to make??

I would have liked to see the level in the engine.

Cheers

Novice testing survey - Subject 3

Q1. What role would you consider yourself in a game production pipeline?

Programmer/ Designer

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

No

Q3. How did you find using the toolkit?

I found the tool easy to use, nice and simple, quick, point and click. Drop boxes made it easier to use and the error messages that came up to tell me that a zone name was in use were helpful.

Q4. How intuitive is the interface and navigating the toolkit?

I thought that the interface was really easy to navigate and had a nice easy user interface, which was simple and not too cluttered.

Q5. Where there any problems you encounter whilst using the toolkit?

I did not find there to be any problems whilst using this application.

Q6. Any comments you would like to make??

Overall nice, plug-in simple to use, looks like it will save level designers both a lot of time and energy, thought that instead of click off the material and back on and update button would have been better. However I found the plug-in very simple and easy to use, gives people without the knowledge of the game engine the ability to create a level without having to know the syntax.

Novice testing survey - Subject 4

Q1. What role would you consider yourself in a game production pipeline?

Modeller

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

No

Q3. How did you find using the toolkit?

Quite easy to use.

Q4. How intuitive is the interface and navigating the toolkit?

A bit tricky at first, but once you know where the tools are located they were all within easy reach.

Q5. Where there any problems you encounter whilst using the toolkit?

Yes, one during the Instances stage, for some the reason the MIF file path was blank giving an error message. Deleting the instanced object solved this problem.

Q6. Any comments you would like to make??

Quite an easy tool set to use, definitely better than having to rename every object that you use. It will definitely save you time. It might be worth adding some help files into the final product to assist the user.

Novice testing survey - Subject 5

Q1. What role would you consider yourself in a game production pipeline?

Programmer

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

No

Q3. How did you find using the toolkit?

Quite intuitive for those who have had previous experience with the engine.

Otherwise, additional "safety checks" could prevent possible inconsistencies.

Q4. How intuitive is the interface and navigating the toolkit?

Intuitive enough

Q5. Where there any problems you encounter whilst using the toolkit?

Missed one of the critical steps, but no errors showed up, which resulted in corrupted data structures.

Q6. Any comments you would like to make??

Keep up the good work.

Novice testing survey - Subject 6

Q1. What role would you consider yourself in a game production pipeline?

Character Artist

Q2. Have you had much experience using Gavin Wade's Game Engine Technology previously?

None

Q3. How did you find using the toolkit?

Incredibly useful, it sped up the production pipeline considerably

Q4. How intuitive is the interface and navigating the toolkit?

Incredibly useful, it sped up the production pipeline considerably

Q5. Where there any problems you encounter whilst using the toolkit?

Yes, however it was a minor gripe that prevented a much quicker implementation of the level, the problem itself was with the visualization of the instancing node and it slowed down workflow by having to re-visualize every time a node was duplicated

Q6. Any comments you would like to make??

Not at this time