University of Derby

School of Computing & Mathematics

A project completed as part of the requirements for the

BSc (Hons) Computer Games Programming

IDENTIFY THE ISSUES OF USING A MIDDLEWARE TO PORT AN EXISTING GAME TO LINUX

Sam Albon

sam.m.albon@outlook.com

2009 - 2013

Contents

Abstract
Chapter 0

In a world were more and more computing devices are making their way into our homes, cross-platform development is become ever more important. With the discontinued support of XNA, smaller developers have turned to MonoGame for cross-platform development.

This project looks at the use of MonoGame as framework for porting existing XNA games to Linux. This project explores the process of Porting Space Salvager, an existing Windows and XBox 360 game, to Linux.

This project provides a critical analysis of the application of using middleware, not just as a tool for porting, but also as a tool for cross-platform development in general and its place within games development.

Introduction

Chapter 1

1.1 Chapter Aim

The aim of this chapter is to introduce the overall concept of the project and its place within Computing in general. This will include a brief on the actual aims and objectives of the project, including the reasons why cross-platform development and porting are important in the first place. As no one project can cover absolutely everything relevant to a topic, this chapter will also outline the limitations of this project and the reasons for them. Overall, this chapter is about introducing the project and giving a general outline.

1.2 Importance of Cross-Platform Development

1.2.1 Software Differences

In the home environment there are several different operating systems used, these can be broken down into three categories: Windows, OS X and Linux (Net Applications, 2013). Each operating system is fundamentally different to each other operating system(Informatics Tech, n.d.): creating the necessity of cross-platform development. In terms of games development, the main difference between each operating system is the supported graphics APIs (BioSys, n.d.). Although OpenGL is supported on Windows, OS X and Linux, DirectX is often the preferred choice (Hardwidge, 2011) and therefore makes cross-platform development difficult. There are numerous differences in each main operating system; however this is something chapter two will cover in more detail.

1.2.2 Hardware Differences

Between home computers, games consoles and mobile phones there is a tremendous array of different hardware available. Each type of hardware works differently, although almost all are based on the Von Neumann machine (Gropius, n.d.), what matters most is the instruction set the computers processor executes. Although this is something chapter two will expand on much further, it is suffice to say that one program written for one computer will not necessarily run on another. Cross-platform development groups machines into a few categories and allows you to develop an application which will compile for each of those groups (Goodwin, 2005).

1.2.3 Summary

Cross-platform development is important to software development in general due to the varied software and hardware available. As hardware being used becomes more varied, so too does the operating systems employed by that hardware (Net Applications, 2013). More specifically there has been a rise in complex mobile devices being used; devices which run operating systems vastly different to the most popular home computer operating systems (Net Applications, 2013). When developing an application on a different operating system than the target operating system cross-platform development makes the development process far more expedient, as you can test for non-platform specific bugs on the host machine without having to deploy the application and rely on the remote debugger (Tong, 2011).

1.3 Project Aims

1.3.1 Explore the Difficulties of Porting Existing Games

The main aim of this project is to explore the difficulties of porting existing games to other platforms by specifically porting an existing Windows game, Space Salvager (2 Person Games, 2012), built using XNA (Microsoft, 2010) to Linux using the existing framework MonoGame (MonoGame, n.d.). This example is a good model of the issues surrounding porting due to the key differences in operating systems that video games rely on (Stack Exchange, 2011): namely graphics APIs.

1.3.2 Investigate the Appropriateness of MonoGame

A further aim is to investigate how appropriate MonoGame is for developing/porting for Linux. The areas under scrutiny will be: graphics, player input (*keyboard/mouse and gamepad*), sound and file management (*content loading and data storage*). All four of these areas have previously been managed by XNA using DirectX (Wikipedia, 2013), however according to their website

(MonoGame, n.d.) MonoGame will only manage graphics, player input (*keyboard/mouse*) and file management (*content loading*).

1.4 Objectives

1.4.1 Port Space Salvager to MonoGame

Firstly, port Space Salvager (2 Person Games, 2012) from XNA (Microsoft, 2010) to MonoGame (MonoGame, n.d.). This will involve copying the existing, or linking (Xamarin, 2010), the existing source code to another project on Linux and building that code using the MonoGame compiler. This should be mostly straightforward; however any compiler errors will highlight very accurately the appropriateness of MonoGame within cross-platform development.

1.4.2 Patch Missing Components

The second objective is to locate any parts of XNA which are not currently implemented within MonoGame and patch them. This may involve implementing a Linux specific version of that component, i.e. gamepad support using OpenTK(OpenTK, 2010), or replacing the component with a temporary fix, i.e. video player replaced with a still texture.

1.4.3 Test and Debug

The final objective is to test the game during runtime and debug any issues. This is where most of the platform-specific bugs will arise. Although the game may compile, the behaviour of MonoGame is different to XNA even when both frameworks are run on Windows (Jackson, 2012). The behaviour of MonoGame from Windows to Linux is just as different (MonoGame, 2013), making the behavioral change from XNA Windows to MonoGame Linux even greater. By building a log of all the bugs encountered, and their respective fixes, this project aims to use them as a basis for the report on the specific difficulties encountered for this port in chapter four.

1.5 Limitations

1.5.1 Single OS Support Only

As the game will only be developed and tested on one specific distribution of Linux, its support can only be guaranteed on that distribution of Linux. Although it should work the same on every distribution, with a GUI, the main community support available is from Debian based Linux distributions (directhex, 2012) and therefore cannot guarantee anything built on MonoGame will run on a distribution of Linux not built on Debian.

1.5.2 Partial Implementation

1.5.2.1 MonoGame 3 (Beta)

At the time of writing, MonoGame is still in beta and as yet does not fulfill the entire functionality of XNA 4 (Mono, 2013). MonoGame does not support gamepad input or any of the gamer services (MonoGame, n.d.)and therefore their implementation within the game cannot be guaranteed.

1.5.2.2 OpenTK Issues

The last OpenTK revision was submitted in 2010 (OpenTK, 2012), despite a number of bugs being reported. Specifically, there are a number of issues surrounding the use of OpenTK within MonoGame (Mono, 2013). The limitation with using OpenTK in its current form is the inability to change the resolution of the game window (Mono, 2012);in addition the default window size differs from the MonoGame default resolution (MonoGame, 2012).

Literature Review
Chapter 2

2.1 Chapter Aim

2.1.1 The Intention

The intention of this chapter is to gain a better understanding of the issues surrounding porting/cross-platform development, why they're important, the solutions to them; but finally why developing for multiple platforms is even a priority in the first place. Knowing the issues is nothing without knowing the reasons for why we should even care about cross-platform development.

This chapter will highlight the features of the primary platform, in this case Windows, and draw a contrast with the counterpart features on the target platform, in this case Linux. This will give us a more in-depth view on the issues surrounding this specific case study, which will work as a decent model for wider issues surrounding cross-platform development in general.

2.1.2 Areas of Focus

This chapter will focus on defining what cross-platform development is, as well as what porting is. Once these terms have been grounded, the chapter will move on to the importance of cross-platform development: the problems and the solutions. Lastly the chapter will outline the middleware available for specifically porting Windows based XNA games to Linux, followed by a critical analysis.

2.1.3 Why These Areas?

By defining the terms and exploring the history behind some of the issues we can put the topic into better context. Without knowing the journey that the middleware has taken to get to this stage it is difficult to be analytical about the choices that have to be made when using it. Furthermore, when it comes to making decisions with large impacts on the game architecture it is extremely important to know the ins and outs of the middleware and its dependencies. Without properly understanding both the primary and secondary platforms decisions made in regards to either could negatively impact the other. This would lead to a constant stream of bugs, as one platform specific fix causes another platform specific bug.

2.2 Definitions

2.2.1 Cross-Platform

Goodwin (2005, p. 1) defined a cross-platform program as:

"*...the same code will compile, link, run and behave identically on whichever platform is targeted. Granted, the compiler and tolls will be specific to each machine, and the underlying Application Programming Interface (API) may vary, but the code itself will be identical as both the game and engine should be reusable across each platform; that is, developed once, and deployed often.*"

This is a very good summation of cross-platform development. Essentially cross-platform development is the concept of writing a program which will run on multiple platforms (Wikipedia, 2010), sharing as much source code as possible.

2.2.2 Port

The difference between a port and cross-platform development is simply that a port happens at the end of the development cycle, it is not something that is necessarily planned from the start. Ports often occur after the main release (Giant Bomb, n.d.). With mobile applications from smaller development teams a buggy, largely untested version is released in order to crowd-source their bug-testing (Making Money With Android, 2012). Once those bugs have been fixed the application is then given a full release.

2.3 Cross-Platform Development

2.3.1 Importance

2.3.1.1 Computer Diversity

In 2013 Valve officially released Steam for Linux (Timothy, 2013), with 2.02% of their active users running Linux. As Steam for Linux was only released recently this number is still growing. This move to Linux is part of an on-going increase of games developing shifting its focus to open source operating systems like Linux (Edge, 2013). More people are running Linux distributions (Net Applications, 2013) and with the rise of Linux in consoles, like the OUYA (Routh, 2013) and PlayStation 3 (Sony, n.d.), it makes sense for games developing to move towards supporting Linux more.

2.3.1.2 Console Diversity

There has always been a large array of consoles available; as one console manufacturer disappears another one takes its place (Wikipedia, 2009). Each platform is different to each other, even each proceeding console from the same manufacturer has radically different hardware (Stack Exchange, 2011). The necessity to port a game from one console to a competing console is obvious, but it is also necessary to sometimes port to a newer version of that same console (Stack Exchange, 2011). Because each console has different hardware it is required to develop cross-platform for each console, whereas with Windows, for example, a game for one operating system almost always works with the proceeding operating system (Plunkett, 2012).

2.3.1.3 Smartphone Pervasiveness

Since 2005 the use of smartphones has risen dramatically (Epstein, 2013). With that has risen a variety of operating system, most popularly iOS and Android (Net Applications, 2013). As with the diversity of consoles: smartphone hardware differs from manufacturer to manufacturer (Julia, 2013). Furthermore the way the phone's operating system handles software differs from system to system (Colbert, 2013); therefore in order to get a game out across all devices you must develop it cross-platform.

2.3.1.4 Marketability

With the pervasion of devices increasing throughout most people’s lives, if a project isn't developed cross-platform it could break that business (36creative, 2013). The main bulk of development time and costs falls within developing the game itself (Cross-platformdevelopment.com, 2013). If you use an existing middleware or game engine then the cost to release on to another platform is minimal (Cross-platformdevelopment.com, 2013), particularly in regards to the profits made from multi-platform releases (Anggriawan, 2013). At the end of the day publishers will be looking for this when funding a game and so will players when it comes to crowd-funding: in order to keep up with modern development cross-platform development must occur.

2.3.2 Problems

2.3.2.1 Instruction Set Differences

Instruction sets are the set of commands a processor can handle (Computer Hope, 2013), for every cycle a processor runs it can execute one instruction. The more complex the instruction, the more work the processor can perform in a single cycle (Roberts, n.d.). Before a program is executed it must be compiled into an appropriate instruction set for the processor that computer employs. A C/C++ program compiled for the MIPS assembly language (MIPS Technologies Inc., 2003), for example, will not run on a computer whose processor is designed using the Intel 8086 assembly language (Patrizio, 2013).

Most home computers use a processor based on the Intel 8086 RISC (reduced instruction set computing) (Patrizio, 2013). Most C/C++ applications compiled for home computers are usually commonly compiled for the SEE instruction set (Microsoft, 2005) to ensure they run on most machines. Applications compiled for this language will not run on a processor that does not employ that specific instruction set (Moshovos, 2005) or newer.

Almost all smart phones employ the use of an ARM based processor (ARM Ltd., 2013). Physically, it is a small processor that can run on low power, using the ARM instruction set (ARM Ltd., 2013). Depending on the operating system being used, smart phones can run programs compiled to an intermediary language (Scribd, 2004), like Java (Oracle, n.d.) or C#

(Microsoft, 2003), which are then just-in-time (*JIT*) compiled for the most recent instruction set available on the processor (Oracle, 2013)(Microsoft, n.d.). Applications compiled like this can be run on other hardware, however it depends on the way the operating system executes them as to whether they will JIT compile correctly and run.

Every games console has a different architecture to every other console. The X-Box 360, for example, uses an out-of-order execution tri-core processor using the VMX128 instruction set (Andrews & Baker, 2006), which is RISC (Eisen, et al., 2007). This is in contrast to the PlayStation 3, which employs an IMB Cell Processor using the SPE instruction set, which is also RISC(Blachford, 2005).

With such distinct architecture it is impossible to write one native application that runs on all platforms, let alone interacts with the system software correctly.

2.3.2.1 Graphics APIs

The argument always comes down to DirectX versus OpenGL (Wikipedia, 2013), which in years gone by have usually ended with DirectX being considered better than OpenGL (Hardwidge, 2011) but having to use OpenGL for Linux as DirectX is only available on Windows (Microsoft, 2011). However, recently speculation amongst games developers has grown in regards to OpenGL's performance (Anthony, 2012).

Since DirectX has traditionally been held as better than OpenGL, Combined with Windows having the largest share in operating system usage (Net Applications, 2013), DirectX was the logical choice of graphics API for any computer games developer. As DirectX is available only for Windows (Microsoft, 2011) porting to other operation systems is a major issue. DirectX handles not only the graphics, but also the player input, sound and file management (Wikipedia, 2013).

2.4.2.2 File Management

There are some major differences between Windows and Linux, one of the largest being the structure. The Linux file structure is based on Unix(Wikipedia, 2013), which means everything is mounted to one single file tree (Lee, 2012). Linux filename restrictions are also different: spaces are not supported (Wallen, 2000) and filenames are case-sensitive (Code Project, 2012).

2.3.3 Solutions

2.3.3.1 Emulation

Emulation is unlike porting, since the goal of emulation is to enable one platform to function as another (Giant Bomb, n.d.). Emulation in the past has often required special hardware, however in recent years software emulation has been employed more and more (PC Mag, 2013).

Popular emulators for Windows to Linux emulation include Wine (Wine HQ, 2013) and Cider (Transgaming, 2013). However, with emulation come many issues: most importantly the introduction of numerous new bugs into the program (Hansen, 2009). Furthermore it can introduce what Ross McGrath (2013) refers to as:

"*...giant cockroach sized uber-bugs.*"

In the long run emulation does not seem worth the trouble.

2.3.3.2 Game Engines

Another option for cross-platform development is the use of a game engine. Although not all game engines are cross-platform, there are several popular engines which are. These include Unity, Source, CryENGINE 3 and the Unreal Development Kit (ModDB, 2013). Amongst these only Unity is available for Linux (Unity, 2013).

Although using a game engine for developing a cross-platform game is arguably the most expedient way to develop (Stack Exchange, 2012); porting a game using a game engine is not easy as you have to port the game from its current game engine or framework into the new game engine. As every game engine is different and gives a different feel (Itterheim, 2012) it is impossible to retain identical playability through this port: which is required by the previous definition of cross-platform development.

2.3.3.3 Middleware

GamesIndustry International (2011) described middleware as:

"*...anything which makes it possible to develop, manage and commercialise games.*"

There are many definitions used for middleware; this is an extremely broad definition and one that doesn't best describe the middleware encompassed in this topic. Middleware usually comes down to a set of APIs which can be used to perform complex operations (DiMaggio, 2008).

In terms of cross-platform development multiple middleware can be used for different platforms to perform the same function. For example, SlimDX is a C# wrapper around DirectX for Windows (SlimDX Group, n.d.) and OpenTK is a C# wrapper around OpenGL for multiple platforms (OpenTK, 2010). Both are considered middleware and could be used to perform rendering and other game services for different platforms.

2.4 Middleware

2.4.1 Mono

Mono is a library designed as an open-source implementation of Microsoft's .NET framework that runs on multiple platforms: including Linux(Xamarin, n.d.); this does not infringe on any copyright even though .NET is proprietary to Microsoft and therefore copyrighted (Galli, 2009).Mono was originally developed by Novell as part of supporting Linux and open source projects, however after Novell was bought by Attachmate in 2011 the Mono Project was terminated (Vaughan-Nichols, 2011). Shortly after being made redundant the Mono Project team formed Xamarin and purchased the license for Mono and continued development themselves (Taylor, 2011).

The Mono Project compiler, tools, class libraries and runtime libraries for Linux are licensed under a mixture of MIT/X11 (Open Source Initiative, n.d.) and the GNU General Public License (Open Source Initiative, n.d.); as found described on the Mono website under the FAQ (Xamarin, n.d.). This means they can be used within this project without confliction.

2.4.2 MonoGame

MonoGame is another library, using the Mono library, designed as an open-source implementation of Microsoft's XNA 4 framework (MonoGame, n.d.). Currently MonoGame supports 2D/3D graphics, audio and networking across Windows, Mac OS X and Linux. The gamepad and content pipeline, however, are not implemented. Although the content pipeline isn't a necessary feature, as you can load them directly without being pre-processed through the content pipeline (Mono, 2012).

MonoGame is absolutely free to use for Linux (MonoGame, n.d.), as the project itself is covered by the Microsoft Public License (MonoGame, 2009).

2.5 XNA/MonoGameRuntime Differences

2.5.1 Windows

The largest difference between XNA and MonoGame is the content pipeline, as XNA compiles its resources into a specific XNA format (Microsoft, n.d.); MonoGame cannot compile these resources in the same way (Mono, 2013). Although you can load these resources, it is not possible to load them in the compressed DXT format until the latest release (Mono, 2013). DXT is a lossy compression technique that most graphics cards support on-chip (Intel, n.d.). The compression ratio is so efficient it is essential for video games to employ (Sherrod, 2005); this is due to the vast reduction in time for loading textures on to the graphics device.

2.5.2 Xbox 360

The main difference between the Xbox 360 and Windows runtime for XNA is that the Xbox 360 uses the compact .NET framework (Harbour, 2012). In regards to a difference in the .NET framework, the only issue that's been a major problem is the usage of a SpinLock(Microsoft, n.d.)and Monitor (Microsoft, n.d.). These objects are both used to lock an object or region of memory from being accessed by another Thread (Stack Exchange, 2009). The SpinLock is the most useful for debugging: when you attempt to get a lock from a thread that already holds the lock it throws an exception (Microsoft, n.d.); however with the Monitor this is not the case (Microsoft, n.d.), although multiple lock attempts can be made without an exception being thrown, a single unlock call will release that object for locking by every other thread. The result can be catastrophic, causing memory corruption and re-writes from other threads known as a "race condition" (Stack Exchange, 2009).

2.5.3 Linux

2.5.3.1 DirectX and OpenGL

As DirectX is not supported on Linux (Microsoft, 2011) the way MonoGame handles graphics is very different to XNA. MonoGame uses OpenTK for all its graphics calls (Mono, 2013); however OpenTK does not behave in the same way as DirectX does for XNA and is not as forgiving (Mono, 2012). There are several forks of OpenTK on GitHub; however and it does appear there is a MonoGame specific fork tailored specifically for MonoGame (Mono, 2013).

In terms of parallel rendering OpenGL does not seem to support making render calls from a thread other than the main thread (Stack Exchange, 2012). This is largely undocumented within MonoGame, but through investigation and trial this project has proven it to be an issue when testing MonoGame's ability to change the graphics device state from a different thread. This will become a larger problem when it comes to development, however the easiest solution appears to be to queue the render calls and running them on the main thread before presenting to the screen. Android handles this much the same way, as you cannot modify the UI from a thread which is not the main thread (Guy, 2009). However, this will be explored fully in later chapters.

2.5.3.2 Saving/Loading Files

As mentioned above, the file management system Linux uses is very different to Windows, this may make saving/loading data files an issue if not using the MonoGame content manager. As with most video games, Space Salvager stores player progress to hard drive. It does this via the gamer services within XNA, however MonoGame has not implemented this feature on all platforms yet (MonoGame, n.d.). This will be explored fully in later chapters.

Methodology & Ethics

Chapter 3

3.1 Chapter Aim

This chapter will explain the planned process of how this project will achieve its objectives and aims in detail, as outlined in chapter one. This will include a step-by-step breakdown of the intended actions, how they will be executed and how they will be recorded.

Chapter one explained the *what*, this chapter will explain the *how*.

3.2 Development Plan

3.2.1 Staged Development

3.2.1.2 Change Frameworks

MonoDevelop mimics Visual Studio in terms of solution files, therefore linking the existing source files in the Visual Studio XNA solution for Windows to the MonoDevelop MonoGame solution for Linux should not cause any issues (Delimarsky, 2010). The only references required should be MonoGame, OpenTK, Tao.SDLand Lidgren.Network (Comte, n.d.). These are all included with the MonoGame MonoDevelop package on the MonoGame website.

3.2.1.2 Resolve Compiler Errors

As Mono and MonoGame do not mimic .NET and XNA entirely (Mono, 2013) not all the classes will compile straight away. The resolution for most of these issues will be to use a different method of attaining the same or similar results, as there are always several different ways of performing the same task.

3.2.1.3 Resolve Runtime Errors

As MonoGame and its dependencies don't behave exactly the same as XNA and its respected dependencies (Mono, 2012), there will be some runtime errors/differences. As with resolving the compiler errors, the solution is to log each error in turn and resolve them as expediently as possible. Whatever errors can be bypassed easily will be in order to get a *working* game as quickly as possible. Once this has been achieved this bug list will be prioritised and resolved in order of importance. By using this agile approach (Karkar, 2011) there will always be a *working* game that can be used. There is, however, a major issue with this methodology in that if a bug is found which cannot be worked around or resolved in a decent amount of time, development halts until it has been resolved. In this event the methodology will have to be adapted to the specific circumstance.

3.2.1.4 Implement Platform Specific Features

Many of the runtime errors may have been caused by having platform specific features included only in the Windows or XBox 360 build, and not the Linux build. Those that cannot be bypassed will be re-introduced during the previous stage, however for any others remaining they will be implemented at this stage. Due to the nature of the file system change and other system services, file loading/saving and hardware analysis will fall into this stage.

3.2.2 Platform

3.2.2.1 Linux Distribution

The specific Linux distribution being used is Ubuntu 11.10 (Oneric Ocelot). Although, at time of writing, the most recent release is 12.10 (Quantal Quetzal) there is far greater support for 11.10 as it has been out far longer and therefore arguably more stable (Stack Exchange, 2012).

3.2.2.2 Physical vs. Virtual Machine

The other question, in regards to platform, is whether to run Ubuntu as a dedicated physical machine or as a virtual machine on another system. Both have pros and cons (Stack Exchange, 2011), however given the context using a virtual machine is the better option.

First, this doesn't require setting up a new dedicated machine. Second, both solutions can be run under both Linux and Windows with absolute ease and test for general/platform specific bugs quickly. Finally, it is easier to share resources within a virtual machine environment (Tsakiris, 2008) than between two physical machines (Ubuntu, 2012).
The specific virtual machine software intended for use is VirtualBox(Oracle, 2013), as this works for Windows and has lots of support for running Ubuntu distributions (Ubuntu, 2008).

3.2.3 Development Pipeline

3.2.3.1 Visual Studio vs. MonoDevelop

XNA 4 will only work with Visual Studio 2010 (Microsoft, 2010), therefore it is required that the primary solution remain within Visual Studio 2010. However, MonoGame 3.0 will work with both Visual Studio and Mono Develop (MonoGame, 2013). It would be prudent to keep development within a single IDE, however this would require compiling on Windows and deploying to Linux. Currently there is no decent pipeline for this and certainly no way of debugging that from within Visual Studio. The best option is to create a new solution for Mono Develop on Linux (Xamarin, n.d.)and use the same source files.

3.2.3.2 Space Salvager Source Control (SVN)

All the source code for Space Salvager is stored on a VisualSVN Server, in order to keep historic backups as well as managing distribution for University and home. The source code can easily be managed between Visual Studio on Windows and MonoDevelop on Linux by using this source control. However, there is a known issue with self-signed certificates when using the GnuTLS library on the client (VisualSVN, n.d.). Essentially, the server is inaccessible by most Linux distributions, including Debian. Although there is a fix posted on the VisualSVN help pages, this fix has not worked in this case.

3.2.3.3 Virtual Machine Shared Folders

The solution to the above is to use shared folders between the virtual machine and the host machine (Tsakiris, 2008). This is achieved by sharing the solution folder on the Windows host and the linking the source files into the Mono Develop solution (Xamarin, n.d.); this will ensure any changes are stored centrally for the main solution to use as well. This will allow testing for any bugs generated and see whether they are general or platform specific.

3.2.3.4 MonoGame & Dependencies Source Control (Git)

MonoGame and its dependencies, namely OpenTK, source control are managed by GitHub(GitHub, 2013). Although there is a packaged version of MonoGame for all IDEs and platforms on their website (MonoGame, 2013), it may be necessary to use the latest revision of MonoGame directly from Git. To this end RabbitVCS (RabbitCSV, 2013) will be used in order to maintain the latest copy of MonoGame and, if necessary, its dependencies as they are also mostly managed by GitHub.

3.2.3.5 Development Log

In order to properly maintain a record of the development process and accurately report on the findings everything must be logged accordingly. As mentioned earlier, all bugs will be logged and tracked, however every event that occurs and decision made must also be accounted for in a separate log. This will take on the format of the date, event/decision title, details. This and the bug list will form the basis for chapter four.

3.3 Ethical Considerations

There will be no human subjects used in this project, nor will there be any data collected other than that of my bug list and development log. There are no ethical issues to be considered.

Primary Case Study

Chapter 4

4.1 Chapter Aim

4.1.1 Purpose

This chapter will explain in detail what happened during the development process, including the problems, the solutions, what went according to plan and what did not. Furthermore this chapter will provide an analysis of the process in terms of the project aims and objectives as outlined in chapter one.

4.1.2 Outcome

As stated in chapter one, there were potentially some large limitations with this implementation. Although MonoGame 3 was given a full release during development, the issues with OpenTK took a long time to investigate and resolve, where possible. Where these resolutions were not possible it left the port uncompleted.

The game window size remains at the OpenTK default. Furthermore, any font that is not Arial is rendered unreadable. The issues behind these will be explained in full later in this chapter, as will some of the solutions. The future of these issues will be investigated in the next chapter.

4.1.3 Summary

Porting an existing game is not without its issues. This is largely dependent on the platform and the tools used; however in this case the port from MonoGame to XNA did not cause many issues. The largest issues were centred on OpenTK, the graphics library used by MonoGame. Although this was a foreseen issue, it was not clear until development exactly how much this would impact development.

4.2 Review

4.2.1 Port Space Salvager to MonoGame

4.2.1.1 Setting up the Pipeline

VirtualBox(Oracle, 2013)was used to run Ubuntu 11.10 (Ubuntu, 2013) on a Windows machine, using a shared folder from the host machine (Tsakiris, 2008) to maintain the source code for all builds. Although SVN would've been better for version control Ubuntu would not read the VisualSVN server due to an issue with the GnuTLS certification (VisualSVN, n.d.).

Setting up the shared folder was a simple process of adding the target folder to the shared folder list in the VirtualBox settings; then adding the current user to the allowed user list for that folder. The command used was (Stack Exchange, 2012):

```
sudo usermod -G vboxsf -a [USERNAME]
```

This pipeline worked fine until part-way through development when VirtualBox was patched to the latest version (Oracle, 2013).This resulted in the Ubuntu virtual machine crashing without recovery. VirtualBox should not have been updated, as this is the exact reason the latest version of Ubuntu (Ubuntu, 2013) was not chosen for use. Fortunately, as all the source code was stored on the host machine there was minimal disruption and the new Ubuntu virtual machine was setup without issue.

4.2.1.2 Setting up MonoDevelop and MonoGame

Setting up the environment was very straightforward; all it involved was adding a new repository and running new commands for the synaptic package manager to execute (directhex, 2012):

```
sudo apt-add-repository ppa:directhex/ppa
sudo apt-get install monodevelop
sudo apt-get install monodevelop-monogame
```

However, when the MonoGame 3 came out of Beta in March 2013 (MonoGame, 2013)this version was and replaced with the latest version direct from the MonoGame Codeplex page. This resolved quite a few issues that had already occurred in development which will be covered later in this chapter.

Later in the project it became necessary to explore using the current revision from GitHub itself (Mono, 2013). There were no issues with building MonoGame itself, as it comes with a MonoDevelop solution file already; however when linking the library with another project the compiler threw numerous *internal compiler exceptions*.

After investigation on the MonoGame forums for GitHub (Mono, 2013) and Codeplex (MonoGame, n.d.), as well as general searches for issues regarding MonoDevelop in general: Nothing was found pertaining to *internal compiler exceptions*. However, this was fixed by re-creating the solution as a standard C# Mono solution, without referencing MonoGame at all. By manually adding the MonoGame references: OpenTK, Tao.SDL and Lidgren.Network (Comte, n.d.),to the solution using the custom built MonoGame library. From this we can deduce this was an issue with the compiler attempting to reference to the installed MonoGame package and the custom built MonoGame library.

4.2.1.3 Abstracting Game Execution and Library

In order to abstract the way the game is executed from what the game actually does, all the game logic and engine code was moved to a new solution which builds into a library. This was then linked into the MonoDevelop solution in Linux, with a local solution built as an executable which called the library to run:

```
[STAThread]
static void Main()
{
    using (var game = new Game1())
        game.Run();
}
```

This is in contrast to how XNA defaults to in Windows:

```
        static void Main()
        {
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
```

The only difference to compilation is the use of *STAThread* around the entry point. This attribute modified the way the thread this function is called on runs. In essence, it allows single threaded communicated with the COM objects (Stack Exchange, 2010) (jfo, 2005). It is strange, however, that this is not required for the XNA default for the entry point. This suggests that XNA uses DirectInput for keyboard and mouse input (Microsoft, 2006), whereas MonoGame uses the message pump for keyboard and mouse input (OpenTK, 2012).

4.2.2 Patch Missing Components

4.2.2.1 Mono ManagementObjectSearcher Missing

Space Salvager manages its own threads in a class called *SalvagerServices1*. The entire class is written thread-safe and runs a number of operations equal to the number of logical cores the computer has access to. This number is calculated during initialisation using the *ManagementObjectSearcher2* (Stack Exchange, 2012).

```
        int CalculateCPUCount()
        {
            /* Find the CPU count */
#if XBOX360
            return 3;
#endif
#if OUYA
            return 4;
#endif
#if LINUX
            return System.Environment.ProcessorCount;
#endif
```

1 SpaceSalvagerBeta.Components.Services.SalvagerServices

2 System.Management.ManagementObjectSearcher

After much investigation of the functionality for Mono in comparison to .NET, the closest function found was to query to *Environment* for the physical processor count. As the result is used to determine the maximum number of threads the game can run this result is not truly accurate, as a logical processor core can run a thread on its own (Stack Exchange, 2012); however it is as close a result as can be attained from Mono at the current time.

4.2.2.2 Mono SpinLock Issues

Very early on in development an issue with releasing certain *SpinLocks*(Microsoft, n.d.)was found; after some investigation it was discovered that the *SpinLock* was never locking in the first place. This issue has never in .NET for Windows before, and after some investigation a report was found of someone else having an issue with the *SpinLock* class on Mono and found it worked on .NET for Windows fine (Young, 2012).

The *SpinLock*[3] class is only available from the full .NET framework (Microsoft, n.d.), which is why Space Salvager had previously used its own cross-platform class which wraps around a different mutex depending on the platform:

```
        public class ThreadLock
        {
#if XBOX360
                object mOwner;
#endif
#if WINDOWS
                SpinLock mLock;
#endif
#if LINUX || OUYA
                ReaderWriterLockSlim mLock;
#endif
```

The XBox 360 uses the static class *Monitor*[4] (Microsoft, n.d.); although it does not behave exactly the same, the missing functionality in comparison to the *SpinLock* is mostly used for debugging platform independent issues. For Linux it was decided to use the *ReaderWriterLockSlim*[5](Microsoft, n.d.), instead of the *Monitor*; this is down to it being easier to

3 System.Threading.SpinLock

4 System.Threading.Monitor

5 Systeam.Threading.ReaderWriterLockSlim

debug with more data accessible than the *Monitor*. The*ReaderWriterLockSlim* was chosen instead of the *ReaderWriterLock* simply because it is faster (Rezaei, 2007).

4.2.2.3 MonoGame VideoPlayer for Linux Missing

While loading the game assets and doing some initial configuration/initialisation, Space Salvager runs an opening video sequence; however the *Video6* and *VideoPlayer7* class are both missing from the Linux build for MonoGame (Stack Exchangfe, 2013). This was resolved in two steps, firstly to create a new *Video* class:

```
    public class Video : IDisposable
    {
#if WINDOWS || XBOX360
        Microsoft.Xna.Framework.Media.Video mVideo;
        public Microsoft.Xna.Framework.Media.Video VideoFile
        {
            get { return mVideo; }
        }
#else
        Texture2D mVideo;
        public Texture2D VideoFile
        {
            get { return mVideo; }
        }
#endif

        /* Constructors */

        public Video(string path, ContentManager content)
        {
#if WINDOWS || XBOX360
            mVideo = content.
                          Load<Microsoft.Xna.Framework.Media.Video>(path);
#else
            mVideo = content.Load<Texture2D>("Backgrounds\\StationBK");
#endif
        }
```

The Windows build for this class simply wraps around the existing *Video* class from the XNA framework. However, the Linux build loads a default texture from the game content. The second step was to implement a new *VideoPlayer* class:

6 Microsoft.Xna.Framework.Video

7 Microsoft.Xna.Framework.VideoPlayer

```
        /* Variables */

        public float Volume
        {
#if WINDOWS && !WINDOWS8
            get { return mPlayer.Volume; }
            set { mPlayer.Volume = value; }
#else
            set { mVolume = value; }
            get { return mVolume; }
#endif
        }

        public MediaState State
        {
#if WINDOWS && !WINDOWS8
            get { return mPlayer.State; }
#else
            get { return mState; }
#endif
        }

#if WINDOWS && !WINDOWS8
        new Microsoft.Xna.Framework.Media.VideoPlayer mPlayer;
#else
        Texture2D mTemporaryTexture;
        float mVolume;
        MediaState mState = MediaState.Stopped;
#endif
```

Again, for Windows this is simply a wrapper around the existing XNA *VideoPlayer* class, implementing all the same public methods that Space Salvager has previously used from the XNA *VideoPlayer* class (Microsoft, n.d.). The main difference here however is that the Linux build returns the same default *Texture2D*8 whenever the *GetTexture()* accessor is called:

```
        /* Accessors */

        public Texture2D GetTexture()
        {
#if WINDOWS && !WINDOWS8
            return mPlayer.GetTexture();
#else
            return mTemporaryTexture;
#endif
        }
```

Although the *VideoPlayer* class is only used in one place so this does not impact the game very much; however this is only a work around.

8 Microsoft.Xna.Framework.Graphics.Texture2D

4.2.2.4 MonoGame Texture2D Missing Methods

As part of the collision management system, Space Salvager creates two dimensional polygons from *Texture2Ds*. The *PolygonCollision*[9] class calls *Texture2D.GetData()* (Microsoft, n.d.) which returns an array of the colours in the texture from left to right: top to bottom. Some textures are a collection of sprites as part of a sprite sheet (BeRecursive, 2008) and others are multiple individual textures compacted into one texture in order to reduce rendering time on the graphics card (Stack Exchange, 2012).

Unfortunately MonoGame has not fully implemented the *Texture2D.GetData()* method (Mono, 2012), which has led to a small change in the way the *PolygonCollision* class handles creating the polygon vertex data:

```
/* Convert the image into a boolean 2D array */
bool[,] data = new bool[source.Width, source.Height];
{
    /* Get the colour data */
    Color[] colour = new Color[source.Width * source.Height];
    texture.GetData<Color>(colour);

    /* Covert the array into booleans */
    int index = (source.Y * source.Width) + source.X;
    for (int y = source.Top; y < source.Bottom; y++)
    {
        for (int x = source.Left; x < source.Right; x++)
        {
            data[x - source.Left, y - source.Top] = colour[index].A > 0;
            index++;    /*< Increment the colour index */
        }
    }
}
```

Where the *Texture2D.GetData()* originally returned only the exact region required, this uses the entire texture and simply iterates through the specific part of the array required. This is far more expensive because, in most cases, the result returned is an array that is 2048 x 2048 (*4,194,304 in total*) when you may only want a region of about 64 x 64 (*4,096 in total*). As this is an issue with the SlimDX on Windows (Mono, 2012) this must be an issue with OpenTK for Linux, however the reasons for this will be explored later in this chapter.

9 SpaceSalvagerBeta.Objects.CollisionObjects.PolygonCollision

4.2.3 Test and Debug

4.2.3.1 Upgrading to MonoGame 3 (Release)

While testing the game during runtime I found a number of issues which were resolved by upgrading to the full release of MonoGame 3 (MonoGame, 2013).

The first bug occurred when *GraphicsDevice.SetRenderTarget()10* was called in *DesktopGame.Draw()11,* the *GraphicsDevice* would throw an exception. At first it was believed to be something to do with OpenTK not allowing you to modify the *GraphicsDevice* from a different thread, however after ensuring all *GraphicsDevice* calls were ran on the main thread this exception was still thrown.

The second bug came from loading the game content on a separate thread. When the *TextureManager12* attempted to load the first asset the *ContentManager13* threw an exception, which was resolved by running the content load on main thread.

The final bug occurred whenever the game attempted to load a texture that was equal to 2048 x 2048 in width and height. This is the largest texture size XNA in the Reach profile can handle (Hargreaves, 2010); however OpenGLs maximum texture size is depends on the device (QT Centre, 2011). However, every time the content manager threw an exception. This was resolved by loading the texture as a .PNG (Roelofs, 2013) instead of the XNA compiled content format (Microsoft, n.d.).

After upgrading to the full release of MonoGame 3 (MonoGame, 2013) and reverting all of the changes that fixed these issues no exceptions were thrown, showing MonoGame is getting closer to fully mimicking XNA.

4.2.3.2 GraphicsDevice Issues

---

10 Microsoft.Xna.Framework.Graphics.GraphicsDevice

11 SpaceSalvagerBeta.Games.DesktopGame

12 SpaceSalvagerBeta.Managers.TextureManager

13 Microsoft.Xna.Framework.Content

In most cases, whenever Space Salvager has to create a *Texture2D* the render process is performed on a new thread. However, OpenTK does not allow you to modify the *GraphicsDevice* from another thread (Stack Exchange, 2012). The solution to this was to mimic the way Android allows you to modify the UI from another thread, by queuing the operation and running it on the main thread before the next render is called (Guy, 2009).

```
/* Check the UI thread queue */
while (mUIOperationsQueue.Count > 0)
{
    /* Check the time frame */
    if (elapsed_time.Elapsed.TotalSeconds > max_time)
    {
        elapsed_time.Stop();    /*< Ensure the stopwatch actually stops */
        break;
    }
    else
    {
        /* Dequeue the next operation */
        ThreadData next = mUIOperationsQueue.Dequeue();

        try
        {
            /* Unlock the UI operations queue */
            mUIQueueLock.Unlock();

            /* Run the next operation */
           next.Start();
        }
        catch (Exception e)
        {
            /* Log the exception */
            Game.Log(e.ToString());
        }
        finally
        {
            /* Lock the UI operations queue */
            mUIQueueLock.Lock();
        }

        /* Set the flag */
        mUIOperationsQueueFlags[next].Set();
        mUIOperationsQueueFlags.Remove(next);
    }
}
```

This was achieved by adding a new function to the *SalvagerServices*: the class which already handles the games multi-threading. The function adds the new operation to a queue of operations and returns a *ManualResetEventSlim14* (Microsoft, n.d.), which is set to true once the operation has been completed. Then, during the *SalvagerServices.Update()* the class iterates through the queue executing the operation on the main thread until either the queue is empty or the timer goes off.

4.2.3.3 Rendering non-Arial Text Issues

XNA renders text by using a large sprite sheet compiled from a specific instance of a font; by specifying the point size, font style and character start and end value, XNA will create a sprite sheet to use during runtime (Microsoft, n.d.). MonoGame uses these compiled *SpriteFonts15* exactly the same way, however for the Linux version it doesn't render correctly anything but the Arial font. Whenever any text that is not based on the Arial font is rendered it comes out as square blocks of text. This cannot be down to MonoGame compiling the font, since MonoGame has not implemented the content pipeline yet (Quandt, 2013). Due to the lack of Linux support online this has not been reported, except in the Windows 8 build for MonoGame (MonoGame, 2013).

There seems to be no way of resolving this issue, aside from going through MonoGame and OpenTK to fix these errors where they're occurring. Another option could be to replace all the text with static images; however this completely removed any scalability to the game. The more feasible approach would be to implement my own *SpriteBatch.DrawString()16* (Microsoft, n.d.) function, using a custom texture file created from a font and treat it the same as a sprite sheet (Whitaker, n.d.).

All of these resolutions are quite time consuming, however, and it might be far quicker to wait for MonoGame to patch the issues itself themselves; even if this is an OpenTK issue it might still be better to wait as the primary contributor for OpenTK has recently re-emerged from a four year hiatus (Mono, 2013). An even faster option may be to use another piece of middleware to render fonts instead, for example: *Craftwork Games* (2012) *BMFont Rendering*.

14 System.Threading.ManualResetEventSlim

15 Microsoft.Xna.Graphics.SpriteFont

16 Microsoft.Xna.Graphics.SpriteBatch

4.2.3.4 OpenTK Window Resolution Issues

The largest issue encountered was the inability to change the window resolution size. The default game window size for MonoGame is 800 x 480(MonoGame, 2012), however the OpenTK default window size is 640 x 480 (Mono, 2013). When you change the *GraphicsDevice* width or height the window size does not change, however the render parameters do change. The result is that the game renders to a resolution either too small or too large for the window, starting at the bottom left corner and continuing on it reaches the edge of the window (Mono, 2012).

This can be fixed quite easily by updating to the latest revision of MonoGame on GitHub (Mono, 2013) via the Master trunk. However, this re-introduces issues outlined earlier in regards to updating to the latest version of MonoGame. It seems to be whatever fixes they put in place for the full release of MonoGame 3 (MonoGame, 2013) were removed from the Master trunk.

After exploring a mixture of the full release of MonoGame 3 (MonoGame, 2013) and the OpenTK (OpenTK, 2010) library used by the current revision, it was found this caused some issues as that version of OpenTK is from a different fork than the previous and does not behave the same (Mono, 2013).

The only permanent resolution is to wait until MonoGame and OpenTK have been fixed to work with each other properly. As with the previous issue this may take some time, however it will still be faster than going through the code manually and resolving the conflictions.

4.3 Analysis

4.3.1 Investigate the Appropriateness of MonoGame

4.3.1.1 Lack of Linux Support

MonoGame has been developed to run on multiple platforms: including Windows, Windows 8, iOS, Android, Mac OS X and Linux (MonoGame, n.d.). However, there seems to be a huge lack of support for the Linux build. After looking at the Discussions page on the MonoGame Codeplex website this is clear, with 1,437 posts in Android, 1,870 posts in Metro (*Window 8*): Linux has 178 posts (MonoGame, n.d.).

The further lack of Linux support is inherent from the fact both Window distributions use SlimDX (Mono, 2012), which is has more recent updates then OpenTK (SlimDX Group, 2012); whereas the rest of the distributions use OpenTK (Mono, 2013), the main contributor of which has only just resurfaced (Mono, 2013). The community has been very good at continuing OpenTK development without lead, however this lead to a lot of confusion on which fork to use. MonoGame has its own fork (Mono, 2013), which has been tailored for MonoGames specific use. However there are several forks available, each with different fixes for different things (Mono, 2013) and it is difficult to know which fork to use.

4.3.1.2 Supported Features

As outlined in chapter one, there were several areas of MonoGame under scrutiny. Player input (*keyboard/mouse and gamepad*), sound and file management (*content loading and data storage*) were all successful in the port. None of these needed any reworking, since the MonoGame 3 full release (MonoGame, 2013) they worked without issue. The most surprising feature to work without issue was the file management, since the file structure between Windows and Linux is so different (Code Project, 2012).

4.3.1.3 Graphics Issues

In terms of porting and platform-dependent development graphics is by far the most difficult area for MonoGame, however this is hardly surprising as it's wholly dependent on OpenTK which has not been supported in quite a few years (Mono, 2013).

Although this issue doesn't affect all games developed using MonoGame, specifically for Space Salvager the lack of multi-threaded rendering was a problem (Stack Exchange, 2012). Although in XNA you couldn't render from multiple threads simultaneously (Stack Exchange, 2012), it is possible to render from different threads as long as they are atomic to each other. For OpenTK, however, this is not possible (MonoGame, 2012). The fix for this was to queue those operations and run them on the main thread, however this is something that could have been added to the MonoGame framework itself as it is a common feature that many games could exploit.

An issue that affects all games, however, is the inability to resize the display (MonoGame, 2012). Without this ability video games developed with MonoGame for Linux are forced to use a tiny display (MonoGame, 2012). The size is completely inappropriate for almost any application, let alone a video game. Without this being fixed it's difficult to see MonoGame being used for any game on Linux.

4.3.1.4 Rendering non-Arial SpriteFonts

When it comes to rendering text in MonoGame there are some large issues. After some testing it appears that the only way to render text is to use a pre-compiled *SpriteFont* of Arial (Mono, 2012). As with a lot of issues for the Linux build, there is not much support for this problem. There are ways around this to explore: using middleware to render text, experimenting with different *SpriteFont* compilations, etc. However, it is possible for MonoGame to include this middleware to fix this issue transparently and would be the cleanest way to fix the issue; at a later date it could be replaced with a different fix if necessary and nothing would change from the way *SpriteBatch.DrawString()* (Microsoft, n.d.) is implemented.

4.3.1.5 MonoGame Still in Development

During the development of this project, MonoGame 3 went from Beta to Release:

"*MonoGame 3.0 marks the start of full support for the entire XNA API.*" (Mono, 2013)

The change from Beta to Release did cover almost every feature that Space Salvager uses in XNA, the major issues relating to the MonoGame dependencies (Mono, 2013). Upgrading a tool from one version to another part-way through development is not a good idea, however if there are features you need which are only available in the latest release there is no other choice but to upgrade. In this case, it worked out extremely well. There were no issues created from upgrading, only issues resolved. Most of the issues with MonoGame now lie with its dependencies, namely OpenTK and the graphics specific issues (Mono, 2013).

4.3.2 Explore the Difficulties of Porting Existing Games

4.3.2.1 Development Pipeline

The pipeline employed used the shared folder system within VirtualBox instead of a source control server. Although this initially seemed as though it would cause problems, in practice this worked extremely well. Mentioned earlier in this chapter, the Ubuntu virtual image I was using became corrupted part-way through development. If a source control server were used and the changes had not been committed everything would have been lost; however as the source code was stored on the host machine everything was saved.

In terms of using a virtual machine itself, if the game needed to be tested on a difficult machine configuration it was very easy to copy the virtual machine and modify the copy's physical properties and test the game that way. The downside to this, however, was that a virtual machine is never a true representative of a real machine (Stack Exchange, 2011). Before release, the game would have to be tested on a physical machine.

4.3.2.2 IDE Contrasts

For the XNA on Windows/Xbox 360 Visual Studio was the IDE of choice, however for MonoGame on Linux MonoDevelop had to be used as Visual Studio is only available for Windows (Microsoft, 2010).

MonoDevelop and Visual Studio have a lot of similarities, which aided a lot in the cross-platform development. Although the UI and file management were very similar (Delimarsky, 2010), the key similarities were the compilations methods. Both IDEs handle build configurations in the same manner; therefore I was able to continue using the same pre-processor statement to handle platform dependent code (Microsoft, n.d.).

The problem with using two different IDEs for development was the having to use different solution files for each build. It is possible to use different build configurations for each platform (Microsoft, n.d.), or even use one big project file with several different solutions: this is how XNA compiles its Window and Xbox 360 builds (Microsoft, n.d.). The difficulty here occurs when you have to share source code. In this case it was not a problem as the pipeline used shared folders from the host machine to the virtual machine; however when using completely independent machines this is not possible and source control is your only solution. This introduces its own issue, in that you have to re-setup the project solution every time you checkout the source to a new machine.

4.3.2.3 Platform Dependent vs. Independent

When it comes to cross-platform development you always have to use platform dependent code, however the question being posed is when the use of platform independent code impacts performance, how far do you go?

The specific example used was the issue of multi-threaded rendering. When using XNA you can make graphics calls from any thread (Stack Exchange, 2012), however with OpenTK all graphics calls must be made from the main thread (Stack Exchange, 2012). The way around this is to queue operations to be executed on the main thread at a later point: however should this be performed only on the OpenTK dependent build and not the XNA dependent build?

The solution used was to queue all operations regardless of the build, however a different way of approaching this could have been to execute those operations immediately for the XNA build. Although this may have been more efficient in terms of multi-threading, however this may have caused issues at a later date because when you make the call to queue the operation the programmer expects that call to happen later and not immediately. This scenario is unlikely, however it is a platform dependent bug that would only occur during testing on Windows or the XBox 360.

Conclusion

Chapter 5

5.1 Chapter Aim

5.1.1 Purpose

This chapter will provide a conclusion for entire project; outline the successes, the failures and the reasons for both. This will include a critical analysis of the employment of MonoGame, its relevance to the port and the problems incurred. Furthermore this chapter will analyse the development process as a whole and give recommendations for further development, as well as cross-platform development and the usage of middleware.

5.1.2 Project Aims

The projects aims, as outlined in chapter one, were first: to explore the difficulties of porting existing games by using the process of porting Space Salvager from Windows to Linux, using MonoGame as a middleware replacement for XNA. Second, it is to investigate the appropriateness of MonoGame as a middleware for porting to Linux. This chapter will go on to expand and resolve these aims by analysing the objectives and how the project fulfilled them.

5.2 Objectives Analysis

5.2.1 Port Space Salvager to MonoGame

5.2.1.1 Recap

This objective was very straightforward: link the existing source code into a new solution using MonoGame for Linux. Resolve any compiler errors and log for analysis later.

5.2.1.2 Result

There were no complications in the initial move to MonoGame, however when attempting to build the game using the most recent source code complications arose. The MonoDevelop compiler threw a number of *internal compiler errors* in regards to calling almost any constructor for a class within the *Microsoft.Xna.Framework.Graphics17* namespace. These issues were resolved by removing all references to MonoGame, except the ones that were actually being used.

5.2.1.3 Analysis

The issue seemed to revolve around the solution being setup using the default MonoGame template, which came from the installed MonoGame 3 package. As the solution was trying to use the locally compiled source code there were some conflictions.

It is a reasonable assumption to make that, considering the number of issues and changes with the OpenTK library being used, there were a large number of changes going on with any class in MonoGame that uses the OpenTK library. This would account for the difference in graphics classes and conflictions when attempting to compile them.

The conclusion here is, when the source code is available for a middleware, either use the packaged installation provided or the source code: not both. Furthermore, when using the source code stay with one revision for as long as possible. If you must update the source code be sure to know the changes between them and allocate time accordingly if necessary.

5.2.2 Patch Missing Components

5.2.1.1 Recap

The tasks for this objective were to locate any missing components from MonoGame which the game relies on; then work around them or patch them with a platform specific solution.

5.2.1.2 Result

Using Mono as a replacement for .NET produced two main errors: first, Mono does not include the *ManagementObjectSearcher*class, secondly the *SpinLock* class does not function correctly. Both of these were resolved very easily by replacing both with similar classes. The

17 This is from the MonoGame framework, not the XNA framework. Although they use the same namespace, this is for cross-compatibility reasons.

former was replaced with a function that returns the number of physical cores, instead of the number of logical cores; the latter was replaced by another mutex called *ReaderWriterLockSlim.* While using the beta version of MonoGame 3 there were a number of issues, these were mostly resolved after it came out of beta. The remaining issues were no support for the *Video* class and a missing method on the *Texture2D* class.

The latter issue was easily worked around by using a different overload, then getting more data back than required and simply locating the desired data. This was more costly, but as it'sonly called during initialisation and loading, the difference was negligible.

The former, however, was more complicated. This was never fixed, but instead worked around. The classes introduced wrapped around the existing *Video* and *VideoPlayer* for XNA, but for MonoGame implemented basic functionality that allowed development to continue with negligible impact on the games performance.

5.2.1.3 Analysis

The only issue which changed the game in any noticeable way was the lack of video player support for MonoGame. For Space Salvager, however, there is only one video played and this happens during the loading sequence, which minimised the impact. Although this conflicts with the definition of cross-platform development, as defined in the second chapter, this is negligible because this doesn't affect the gameplay.

5.2.3 Test and Debug

5.2.1.1 Recap

The concluding objective was to test the game on its new platform and framework, then debug any issues, resolve them accordingly and log them for analysis later. This was crucial to identifying the key issues in cross-platform development, as well as specifically porting.

5.2.1.2 Result

Although there were few complications from compiling the game on a new platform and framework, testing showed there were a number of issues with the port. Firstly the

GraphicsDevice behaved very differently, retaining all access to the render pipeline for the main thread only. This was resolved by adding all render operations to a queue, excluding rendering to the display, and executing them after the update call.

Secondly, non-Arial fonts would not render properly to the screen. Whenever any text using any *SpriteFont* which was not Arial was rendered to the screen nothing but blocks would appear.

Finally, OpenTKs window size was fixed at its default size. This default size was smaller than MonoGames default size and Space Salvagers. Although this was fixed later with the most recent source code, it re-introduced other graphics issues which caused the game to crash.

5.2.1.3 Analysis

Only the first of these three issues was resolved. This is down to the issues being related to the OpenTK library and not MonoGame itself. As there are many forks from the main OpenTK repository on GitHub it's difficult to know which fork to use. Although there is a MonoGame specific fork, when this library was used instead it introduced graphics access errors previously recorded when using the MonoGame 3 Beta library.

Until the OpenTK issues have been resolved, it's difficult to resolve these issues. There are ways of fixing the font rendering, by either implementing a new way of rendering fonts or using another piece of middleware to, either, compile the *SpiteFonts* in a more compatible manner or render them completely differently.

5.3 Summary

5.3.1 Outcome

5.3.1.1 Development Review

Given the nature of porting a game, the number of unknowns and continuing development of MonoGame; agile was definitely the right approach. When large issues occurred, such as the inability to change the window size or the single-threaded *GraphicsDevice* access; agile development allowed for the radical change in time allocation and re-structure.

The most notable issue that occurred was the enormous amount of time spent on resolving problems with OpenTK, as well as time spent attempting to resolve problems with the MonoGame 3 Beta library. The latter of these, however, could not have been foreseen as a change list for the full release was never available until release.

This does, however, highlight some of the key issues within porting a game from one platform to another. It is impossible to know the exact extent to which the code will have to be re-written and worked around. Although using a middleware can help reduce development time dramatically, there are still many unknowns. Agile development is the only methodology which can allow for the continual re-factoring of the development schedule.

5.3.1.2 Remaining Issues

There are still two outstanding issues within the game: window size and font rendering. However, because of these issues there are still large areas of the game that are untested. For example, both the station and player menus have not been tested; neither have the full extent of the game logic for the levels.

However, these areas have been tested on Windows and the XBox 360, and it is unlikely that any platform specific bugs have been introduced in these areas. In terms of the difference between MonoGame and XNA most of these features have been used and tested already during content loading and player storage loading. As unlikely as these bugs may be, the process of testing and debugging should also cycle through each platform, each set of platform specific bugs.

5.3.2 Future Development

5.3.2.1 MonoGame Still in Development

MonoGame is still under development, with the current release aiming to fulfill all the features XNA provides, before going on to expand even further with later releases. Taking into account the fact that MonoGame is still in development, it is still an excellent middleware for porting XNA based games to Linux.

Although there were many issues encountered during development, the speed at which MonoGame went from beta to release suggests these issues will be resolved within the next six months. Had MonoGame been used from the ground up there would have been far fewer issues encountered, however it is still a valid middleware to use for porting.

5.3.2.2 The Future of OpenTK

Most of the issues encountered in the project stemmed from the use of OpenTK. The change from beta to release resolved some of these, however there are still a number of issues remaining: most notably the inability to change the window size.

However, the future for OpenTK looks promising, with the MonoGame specific fork picking up support and the original developer re-appearing after a long absence. The future of OpenTK for MonoGame relies heavily on the inability to change window size being fixed. Being forced to run in a small window is a massive issue for a video game, without this fix there will be no future for MonoGame on Linux, let alone OpenTK.

The latest source code of MonoGame hasproven that this issue is resolvable, although temperamental as it introduces other problems. However, this is a step forward and shows great promise in the future.

5.3.2.3 Recommendations When Using Middleware

When it comes to using a middleware, and this applies to all forms of development, research is key. Knowing the exact extent of the middleware, what it can and cannot do, is crucial to its implementation with an existing game.

If further investigation had been done, in terms of how Space Salvager uses XNA and its counterpart functionality in MonoGame, development may've gone further; problems would have been anticipated and accounted for accordingly. Knowing the extent of the issues with OpenTK beforehand would have led to less time being spent on trying to resolve them and more time on working around issues to come back to at a later date.

Even so, middleware saves time and therefore money in development. Using MonoGame to port to Linux was much faster than creating a custom engine for running Space Salvager on Linux: even factoring in the issues introduced by MonoGame. When it comes to porting an existing game, middleware is the answer.

5.3.2.4 Recommendations When Developing Cross-Platform

Given all the issues discussed so far, it is clear that when intending to release on multiple platforms the best development strategy is to development cross-platform as early as possible. Cross-platform considerations should be made during the initial design and planning, as this will save time and money in the long run.

When it comes to tools for cross-platform development it is recommended to use an existing game engine. This saves a lot of time developing an engine and tools for the game, without taking into consideration the time saved for building for another platform. There are a number of game engines available which will cross-compile for multiple platforms, many of which are free.

However, when a middleware is being used for cross-platform development, it is even more important to plan this as early as possible. During this projects development there were fundamental changes made to the games underlying system, which have changed the efficiency and structure of the engine dramatically. For example, the way in which the game renders to memory is now performed as a deferral. These changes can impact on areas of the game that are unexpected and may only become apparent after the port has been released.

Lastly, test the game on all platforms being developed for, even when changes have been made that should only affect one platform. When the game is being tested the platforms should be cycled through until there are no bugs found on each platform consistently.

5.4 In Closing

Although the game still has some outstanding issues on Linux, the project has been a success. It has successfully highlighted the difficulties of porting an existing game, as well as shown the appropriateness of MonoGame for cross-platform development.

As with most projects, there is room for improvement and expansion for later, however this project provides a good basis for understanding the difficulties of porting. This project has shown the successes of using MonoGame as a middleware, as well as showing the difficulties encountered and the failures. MonoGame may not be quite there yet, in terms of Linux development, however the project itself is still picking up momentum amongst developers. With XNA no longer being developed by Microsoft, more attention is being given to MonoGame and its dependencies. MonoGame is certainly a middleware to watch over the next six months.

Bibliography

Chapter 6

2 Person Games, 2012. *Space Salvager.* [Online]
Available at: http://www.2pgames.net/#!space-salvager
[Accessed 17 04 2013].

36creative, 2013. *Why Cross Platform Development is So Important.* [Online]
Available at: https://36creative.com/blog/mobile/1542/why-cross-platform-development-is-so-important
[Accessed 18 04 2013].

Andrews, J. & Baker, N., 2006. *XBOX 360 System Architecture.* [Online]
Available at: http://www.cis.upenn.edu/~milom/cis501-Fall08/papers/xbox-system.pdf
[Accessed 22 04 2013].

Angel, E., 2002. *OpenGL A Primer.* Boston: Addison-Wesley.

Anggriawan, E. A., 2013. *Go Native or Cross Platform?.* [Online]
Available at: http://ekospinach.blogspot.co.uk/2013/02/go-native-or-cross-platform.html
[Accessed 18 04 2013].

Anthony, S., 2012. *Valve: OpenGL is faster than DirectX - even on Windows.* [Online]
Available at: http://www.extremetech.com/gaming/133824-valve-opengl-is-faster-than-directx-even-on-windows
[Accessed 17 04 2013].

Arcuri, M., 2011. *The Future of Multiplatform.* [Online]
Available at: http://www.develop-online.net/features/1495/The-Future-of-Multiplatform
[Accessed 14 11 2012].

ARM Ltd., 2013. *ARM and Thumb-2 Instruction Set.* [Online]
Available at: http://infocenter.arm.com/help/topic/com.arm.doc.qrc0001m/QRC0001_UAL.pdf
[Accessed 22 04 2013].

ARM Ltd., 2013. *Smartphones.* [Online]
Available at: http://www.arm.com/markets/mobile/smartphones.php
[Accessed 22 04 2013].

BeRecursive, 2008. *Sprite Sheets in Xna: The Basics.* [Online]
Available at: http://www.berecursive.com/2008/c/sprite-sheets-in-xna-the-basics
[Accessed 24 04 2013].

BioSys, n.d. *Graphics APIs.* [Online]
Available at: http://www.cse.mrt.ac.lk/~dilumb/docs/BioSys/docs/pdf/graphicsapi.pdf
[Accessed 22 04 2013].

Blachford, N., 2005. *Cell Architecture Explained Version 2.* [Online]
Available at: http://www.blachford.info/computer/Cell/Cell0_v2.html
[Accessed 22 04 2013].

Code Project, 2012. *Case sensitive file names in Linux and not in Windows.* [Online]
Available at: http://www.codeproject.com/Questions/339068/Case-sensitive-file-names-in-Linux-and-not-in-Wind
[Accessed 17 04 2013].

Colbert, D., 2013. *Which is the superior mobile OS: iOS, Android, or Windows 8?.* [Online]
Available at: http://www.techrepublic.com/blog/tablets/which-is-the-superior-mobile-os-ios-android-or-windows-8/2684
[Accessed 18 04 2013].

Computer Hope, 2013. *Instruction Set.* [Online]
Available at: http://www.computerhope.com/jargon/i/instset.htm
[Accessed 22 04 2013].

Comte, Y., n.d. *XNA on Linux and Mac with.* [Online]
Available at: http://xnameetingpoint.weebly.com/monogameintro.html
[Accessed 23 04 2013].

Comte, Y., n.d. *XNA on Linux and Mac with.* [Online]
Available at: http://xnameetingpoint.weebly.com/monogameintroupdated.html
[Accessed 22 04 2013].

Craftwork Games, 2012. *Tutorial: BMFont rendering with MonoGame.* [Online]
Available at: http://www.craftworkgames.com/blog/tutorial-bmfont-rendering-with-monogame/
[Accessed 24 04 2013].

Cross-platformdevelopment.com, 2013. *Why cross platform?.* [Online]
Available at: http://cross-platformdevelopment.com/why-cross-platform/
[Accessed 18 04 2013].

Delimarsky, D., 2010. *A cross-platform story – Visual Studio solution in MonoDevelop.* [Online]
Available at: http://dotnet.dzone.com/articles/cross-platform-story-%E2%80%93-visual
[Accessed 23 04 2013].

DeLoura, M., 2011. *Game Engines and Middleware In The West.* s.l., s.n.

DeLoura, M. A., 2000. *Game Programming Gems.* Rockland: Charles River Media.

DeLoura, M. A., 2001. *Game Programming Gems 2.* Hingham: Charles River Media.

DiMaggio, L., 2008. *What is middleware? In plain English, please..* [Online]
Available at: http://magazine.redhat.com/2008/03/11/what-is-middleware-in-plain-english-please/
[Accessed 18 04 2013].

directhex, 2012. *MonoGame on Debian/Ubuntu - XNA development on Linux.* [Online]
Available at: http://www.youtube.com/watch?v=OFs319ECDEM
[Accessed 22 04 2013].

Edge, 2013. *The rise of Linux as a gaming platform.* [Online]
Available at: http://www.edge-online.com/features/the-rise-of-linux-as-a-gaming-platform/
[Accessed 18 04 2013].

Eisen, L. et al., 2007. *IBM POWER6.* [Online]
Available at: http://scc.ustc.edu.cn/zlsc/ibm_js22/200910/W020100308600541653665.pdf
[Accessed 22 04 2013].

Epstein, Z., 2013. *A great visualization of Apple and Google's smartphone market dominance.* [Online]
Available at: http://bgr.com/2013/03/21/smartphone-market-share-chart-2005-2012-389100/
[Accessed 18 04 2013].

Galli, P., 2009. *The ECMA C# and CLI Standards.* [Online]
Available at: http://blogs.technet.com/b/port25/archive/2009/07/06/the-ecma-c-and-cli-standards.aspx
[Accessed 14 11 2012].

GamesIndustry International, 2011. *Money Games: The Middleware Opportunity.* [Online]
Available at: http://www.gamesindustry.biz/articles/2011-04-14-money-games-the-middleware-opportunity-article
[Accessed 18 04 2013].

Giant Bomb, n.d. *Porting.* [Online]
Available at: http://www.giantbomb.com/porting/3015-988/
[Accessed 18 04 2013].

GitHub, 2013. *GitHub.* [Online]
Available at: https://github.com/
[Accessed 17 04 2013].

Goodwin, S., 2005. *Cross-Platform Game Programming.* Hingham: Charles River Media.

Gordon, R., 2002. *Zen and the Art of Game Porting.* [Online]
Available at: http://www.pyrogon.com/about/diary/2_26_2002.php
[Accessed 14 11 2012].

Gregory, J., 2009. *Game Engine Architecture.* Wellesley: A K Peters.

Gropius, W., n.d. *The "von Neumann" Architecture.* [Online]
Available at: http://turing.cs.camosun.bc.ca/COMP112/notes/book/Chapter8.pdf
[Accessed 23 04 2013].

Grossman, A., 2003. *Postmortems from Game Developer.* San Fransisco: CMP.

Guy, R., 2009. *Painless Threading.* [Online]
Available at: http://android-developers.blogspot.co.uk/2009/05/painless-threading.html
[Accessed 18 04 2013].

Halfacree, G., 2012. *Valve confirms Steam for Linux.* [Online]
Available at: http://www.bit-tech.net/news/gaming/2012/07/17/valve-steam-linux/
[Accessed 18 04 2013].

Hansen, R. H., 2009. *Porting Games to Linux.* [Online]
Available at: http://www.hardware.no/artikler/ryan_c_gordon_and_michael_simms/68450/4
[Accessed 17 04 2013].

Harbour, J. S., 2012. *XNA Game Studio 4.0 for Xbox 360 Developers.* Boston: Course Technology PTR.

Hardwidge, B., 2011. *Carmack: Direct3D is now Better than OpenGL.* [Online]
Available at: http://www.bit-tech.net/news/gaming/2011/03/11/carmack-directx-better-opengl/
[Accessed 17 04 2013].

Hargreaves, S., 2010. *Reach vs. HiDef.* [Online]
Available at: http://blogs.msdn.com/b/shawnhar/archive/2010/03/12/reach-vs-hidef.aspx
[Accessed 24 04 2013].

Hochgurtel, B., 2003. *Cross-Platform Web Services Using C# & Java.* Hingham: Charles River Media.

Informatics Tech, n.d. *Linux vs Macintosh vs Windows (unbiased comparison).* [Online]
Available at: http://www.informatics-tech.com/linux-vs-mac-vs-windows-unbiased-comparison.html
[Accessed 22 04 2013].

Intel, n.d. *DXT Compression.* [Online]
Available at: http://software.intel.com/en-us/vcsource/samples/dxt-compression
[Accessed 18 04 2013].

Itterheim, S., 2012. *The Game Engine Dating Guide: How to Pick up an Engine for Single Developers.* [Online]
Available at: http://www.learn-cocos2d.com/2012/05/the-game-engine-dating-guide-how-to-find-the-right-engine-for-your-game/
[Accessed 17 04 2013].

Jackson, S., 2012. *XNA to MonoGame and beyond.* [Online]
Available at: http://darkgenesis.zenithmoon.com/xna-to-monogame-and-beyond/
[Accessed 22 04 2013].

jfo, 2005. *Why is STAThread required?.* [Online]
Available at: http://blogs.msdn.com/b/jfoscoding/archive/2005/04/07/406341.aspx
[Accessed 23 04 2013].

Julia, A., 2013. *Mobile Devices Hardware and Operating Systems.* [Online]
Available at: http://www.certshelp.com/blog/mobile-device-hardware-and-operating-systems/
[Accessed 18 04 2013].

JungHyun, H., Dong-Hyun, L., Hyunwoo, K. & Hoh, P. I., 2009. A Situtation Aware Cross-Platform Architecture for Ubiqutous Game. *Computing and Informatics,* Volume 28, pp. 619-633.

Karkar, H., 2011. *Overiew of Agile Methodology.* [Online]
Available at: http://www.slideshare.net/hareshkarkar/overview-of-agile-methodology
[Accessed 23 04 2013].

Kekalainen, F., 2007. *Game Development Middleware & OpenKODE.* s.l., s.n.

Khlud, C., 2012. *Why Mono Is Desirable for Linux.* [Online]
Available at:
http://www.phoronix.com/scan.php?page=article&item=mono_linux_desire&num=1
[Accessed 14 11 2012].

Kirmse, A., 2004. *Game Programming Gems 4.* Hingham: Charlies River Media.

Lee, J., 2012. *7 Key Differences Between Windows & Linux You Should Know About Before Switching.* [Online]
Available at: http://www.makeuseof.com/tag/7-key-differences-between-windows-and-linux-you-should-know-about-before-switching/
[Accessed 17 04 2013].

Lobão, A. S. & Hatton, E., 2003. *.NET Programming with DirectX 9.* Berkeley: Apress.

Louis, D., 2013. *Tales from the Dev Side: MonoGame is The One.* [Online]
Available at: http://indiegamerchick.com/2013/02/12/tales-from-the-dev-side-monogame-is-the-one/
[Accessed 22 04 2013].

Making Money With Android, 2012. *Multiple Platform Release Strategy.* [Online]
Available at: http://forums.makingmoneywithandroid.com/off-topic/629-multiple-platform-release-strategy.html
[Accessed 18 04 2013].

McGrath, R., 2013. *PSone Classics: Where We All Stand.* [Online]
Available at: http://blog.eu.playstation.com/2011/01/31/psone-classics-where-we-all-stand/
[Accessed 17 04 2013].

Microsoft, 2003. *C# Programming Language Future Features.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/aa289180(v=vs.71).aspx
[Accessed 22 04 2013].

Microsoft, 2005. */arch (Minimum CPU Architecture).* [Online]
Available at: http://msdn.microsoft.com/en-GB/library/7t5yh4fd(v=vs.80).aspx
[Accessed 22 04 2013].

Microsoft, 2006. *Keyboard/Mouse Input under XNA Framework.* [Online]
Available at: http://social.msdn.microsoft.com/Forums/en-US/xnaframework/thread/12a7a847-0c1b-436f-93eb-2d33add8449c
[Accessed 23 04 2013].

Microsoft, 2010. *Microsoft Visual Studio 2010 Ultimate Trial - Web Install.* [Online]
Available at: http://www.microsoft.com/en-us/download/details.aspx?id=12752
[Accessed 25 04 2013].

Microsoft, 2010. *Microsoft XNA Game Studio 4.0.* [Online]
Available at: http://www.microsoft.com/en-gb/download/details.aspx?id=23714#top
[Accessed 17 04 2013].

Microsoft, 2011. *DirectX End-User Runtime Web Installer.* [Online]
Available at: http://www.microsoft.com/en-us/download/details.aspx?id=35
[Accessed 17 04 2013].

Microsoft, n.d. *Adding New Platforms to an Existing Game.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/bb976061.aspx
[Accessed 25 04 2013].

Microsoft, n.d. *Compiling MSIL to Native Code.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/ht8ecch6(v=vs.71).aspx
[Accessed 22 04 2013].

Microsoft, n.d. *Cross-Platform Conditional Compilation Symbols.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/dd282469.aspx
[Accessed 25 04 2013].

Microsoft, n.d. *Environment.ProcessorCount Property.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/system.environment.processorcount.aspx
[Accessed 24 04 2013].

Microsoft, n.d. *ManualResetEventSlim Class.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/system.threading.manualreseteventslim.aspx
[Accessed 24 04 2013].

Microsoft, n.d. *Monitor Class.* [Online]
Available at: http://msdn.microsoft.com/en-

us/library/System.Threading.Monitor%28v=vs.100%29.aspx
[Accessed 19 11 2012].

Microsoft, n.d. *Monitor.Enter Method (Object).* [Online]
Available at: http://msdn.microsoft.com/en-us/library/de0542zz.aspx
[Accessed 19 11 2012].

Microsoft, n.d. *ReaderWriterLockSlim Class.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/bb300132(v=vs.100).aspx
[Accessed 24 04 2013].

Microsoft, n.d. *SpinLock Structure.* [Online]
Available at: http://msdn.microsoft.com/en-
us/library/System.Threading.SpinLock%28v=vs.100%29.aspx
[Accessed 19 11 2012].

Microsoft, n.d. *SpinLock.Enter Method.* [Online]
Available at: http://msdn.microsoft.com/en-
us/library/system.threading.spinlock.enter%28v=vs.100%29.aspx
[Accessed 19 11 2012].

Microsoft, n.d. *Sprite Font XML Schema Reference.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/bb447759.aspx
[Accessed 24 04 2013].

Microsoft, n.d. *SpriteBatch.DrawString Method.* [Online]
Available at: http://msdn.microsoft.com/en-
us/library/microsoft.xna.framework.graphics.spritebatch.drawstring.aspx
[Accessed 024 04 2013].

Microsoft, n.d. *Texture2D.GetData Method.* [Online]
Available at: http://msdn.microsoft.com/en-
us/library/microsoft.xna.framework.graphics.texture2d.getdata.aspx
[Accessed 24 04 2013].

Microsoft, n.d. *VideoPlayer Class.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.media.videoplayer.aspx
[Accessed 24 04 2013].

Microsoft, n.d. *What is Content?.* [Online]
Available at: http://msdn.microsoft.com/en-us/library/bb447756.aspx
[Accessed 18 04 2013].

MIPS Technologies Inc., 2003. *MIPS32™ Architecture For Programmers.* [Online]
Available at: http://www.cs.tau.ac.il/~afek/MipsInstructionSetReference.pdf
[Accessed 22 04 2013].

ModDB, 2013. *100 Most Popular Engines Today.* [Online]
Available at: http://www.moddb.com/engines/top
[Accessed 17 04 2013].

Mono, 2012. *.spritefont file not loaded.* [Online]
Available at: https://github.com/mono/MonoGame/issues/707
[Accessed 25 04 2013].

Mono, 2012. *GitHub How to load a Texture2D.* [Online]
Available at: https://github.com/mono/MonoGame/issues/581
[Accessed 18 04 2013].

Mono, 2012. *GitHub PreferredBackBufferWidth/Height has no effect on Linux.* [Online]
Available at: https://github.com/mono/MonoGame/issues/628#issuecomment-14331225
[Accessed 18 04 2013].

Mono, 2012. *GitHub Ray.Intersect(Plane) behaves dramatically different from the XNA version.* [Online]
Available at: https://github.com/mono/MonoGame/issues/799
[Accessed 23 04 2013].

Mono, 2012. *GitHub Texture2d GetData throws Not implemented exception.* [Online]
Available at: https://github.com/mono/MonoGame/issues/644
[Accessed 24 04 2013].

Mono, 2013. *DXT Compression Implemented.* [Online]
Available at: https://github.com/mono/MonoGame/pull/1273
[Accessed 18 04 2013].

Mono, 2013. *GitHub Home.* [Online]
Available at: https://github.com/mono/MonoGame/wiki
[Accessed 25 04 2013].

Mono, 2013. *GitHub mono/monogame.* [Online]
Available at: https://github.com/mono/MonoGame
[Accessed 17 04 2013].

Mono, 2013. *GitHub mono/opentk.* [Online]
Available at: https://github.com/mono/opentk
[Accessed 17 04 2013].

Mono, 2013. *GitHub Trouble setting the screen resolution.* [Online]
Available at: http://monogame.codeplex.com/discussions/433480
[Accessed 24 04 2013].

Mono, 2013. *GitHub Tutorials:prerequisites.* [Online]
Available at: https://github.com/mono/MonoGame/wiki/Tutorials%3Aprerequisites
[Accessed 18 04 2013].

Mono, 2013. *MonoGame Content Processing.* [Online]
Available at: https://github.com/mono/MonoGame/wiki/MonoGame-Content-Processing
[Accessed 18 04 2013].

Mono, 2013. *OpenTK Future.* [Online]
Available at: https://github.com/mono/MonoGame/issues/1528
[Accessed 22 04 2013].

MonoGame, 2009. *License.* [Online]
Available at: http://monogame.codeplex.com/license
[Accessed 18 04 2013].

MonoGame, 2012. *[Linux] Window has always the same size.* [Online]
Available at: http://monogame.codeplex.com/workitem/6966
[Accessed 22 04 2013].

MonoGame, 2012. *loading assets from a thread.* [Online]
Available at: http://monogame.codeplex.com/discussions/395896
[Accessed 25 04 2013].

MonoGame, 2013. *Downloads.* [Online]
Available at: http://www.monogame.net/downloads
[Accessed 17 04 2013].

MonoGame, 2013. *MonoGame 3.0.1.* [Online]
Available at: http://monogame.codeplex.com/releases/view/102870
[Accessed 23 04 2013].

MonoGame, 2013. *SpriteBatch.DrawString not working in metro.* [Online]
Available at: http://monogame.codeplex.com/discussions/434513
[Accessed 24 04 2013].

MonoGame, 2013. *Why different builds for every platform.* [Online]
Available at: http://monogame.codeplex.com/discussions/434232
[Accessed 22 04 2013].

MonoGame, n.d. *Discussion Topics for MonoGame - Write Once, Play Everywhere.* [Online]
Available at: http://monogame.codeplex.com/discussions
[Accessed 23 04 2013].

MonoGame, n.d. *Discussion Topics for MonoGame - Write Once, Play Everywhere.* [Online]
Available at: http://monogame.codeplex.com/discussions
[Accessed 25 04 2013].

MonoGame, n.d. *Main Page.* [Online]
Available at: http://monogame.codeplex.com/
[Accessed 19 11 2012].

MonoGame, n.d. *Platform Support Matrix.* [Online]
Available at:
http://monogame.codeplex.com/wikipage?title=Platform%20support%20matrix&referringTitle=Documentation
[Accessed 19 11 2012].

MonoGame, n.d. *Price.* [Online]
Available at: http://www.monogame.net/price
[Accessed 18 04 2013].

Moshovos, A., 2005. *Instruction Set Architecture and its Implications.* Toronto, University of Toronto.

Net Applications, 2013. *Desktop Operating System Market Share.* [Online]
Available at: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0
[Accessed 17 04 2013].

Net Applications, 2013. *Mobile/Tablet Operating System Market Share.* [Online]
Available at: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=1
[Accessed 18 04 2013].

Open Source Initiative, n.d. *GNU General Public License 3.0.* [Online]
Available at: http://opensource.org/licenses/GPL-3.0
[Accessed 19 11 2012].

Open Source Initiative, n.d. *MIT License.* [Online]
Available at: http://opensource.org/licenses/MIT
[Accessed 19 11 2012].

OpenTK, 2010. *OpenTK.* [Online]
Available at: http://www.opentk.com/
[Accessed 17 04 2013].

OpenTK, 2012. *Is OpenTK Dead?.* [Online]
Available at: http://www.opentk.com/node/3217
[Accessed 22 04 2013].

OpenTK, 2012. *OpenTK Keyboard Input Example.* [Online]
Available at: http://www.opentk.com/node/3071
[Accessed 23 04 2013].

Oracle, 2013. *Changelog for VirtualBox 4.2.* [Online]
Available at: https://www.virtualbox.org/wiki/Changelog
[Accessed 23 04 2013].

Oracle, 2013. *Understanding JIT Compilation and Optimizations.* [Online]
Available at:
http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/underst_jit.html
[Accessed 22 04 2013].

Oracle, 2013. *VirtualBox.* [Online]
Available at: https://www.virtualbox.org/
[Accessed 17 04 2013].

Oracle, n.d. *What is Java technology and why do I need it?.* [Online]
Available at: http://www.java.com/en/download/faq/whatis_java.xml
[Accessed 22 04 2013].

Pallister, K., 2005. *Game Programming Gems 5.* Hingham: Charles River Media.

Patrizio, A., 2013. *Why Intel can't seem to retire the x86.* [Online]
Available at: http://www.itworld.com/it-management/346559/why-intel-cant-seem-retire-x86
[Accessed 22 04 2013].

PC Mag, 2013. *Definition of: emulator.* [Online]
Available at: http://www.pcmag.com/encyclopedia/term/42579/emulator
[Accessed 17 04 2013].

Plunkett, L., 2012. *Gamers, Relax, Windows 8 is Fine (For Now).* [Online]
Available at: http://kotaku.com/5955658/gamers-relax-windows-8-is-fine-for-now
[Accessed 18 04 2013].

PsychoCats, 2012. *Installing Ubuntu inside Windows using VirtualBox.* [Online]
Available at: http://www.psychocats.net/ubuntu/virtualbox
[Accessed 17 04 2013].

QT Centre, 2011. *Maximal texture size in OpenGL?.* [Online]
Available at: http://www.qtcentre.org/threads/44797-Maximal-texture-size-in-OpenGL
[Accessed 24 04 2013].

Quandt, M., 2013. *Adding the Pipeline back to the MonoGame Content Pipeline.* [Online]
Available at: http://mquandt.com/blog/2013/02/adding-pipeline-to-monogame-content-pipeline/
[Accessed 24 04 2013].

RabbitCSV, 2013. *Installation on Ubuntu.* [Online]
Available at: http://wiki.rabbitvcs.org/wiki/install/ubuntu
[Accessed 17 04 2013].

Rabin, S., 2005. *Introduction to Game Development.* Hingham: Charles River Media.

Rezaei, P., 2007. *A Performance Comparison of ReaderWriterLockSlim with ReaderWriterLock.* [Online]
Available at: http://blogs.msdn.com/b/pedram/archive/2007/10/07/a-performance-comparison-of-readerwriterlockslim-with-readerwriterlock.aspx
[Accessed 24 04 2013].

Riley, H. N., 1987. *The von Neumann Architecture of Computer Systems.* [Online]
Available at: http://www.csupomona.edu/~hnriley/www/VonN.html
[Accessed 23 04 2013].

Roberts, E., n.d. *risc vs. cisc.* [Online]
Available at: http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/risc/risccisc/
[Accessed 22 04 2013].

Roelofs, G., 2013. *Portable Network Graphics.* [Online]
Available at: http://www.libpng.org/pub/png/
[Accessed 24 04 2013].

Rouse, R., 2002. *Game Design Theory & Practice.* 2 ed. Plano: Wordware.

Routh, A., 2013. *Hands on: Ouya Review.* [Online]
Available at: http://www.techradar.com/reviews/gaming/games-consoles/ouya-review-1141503/review
[Accessed 18 04 2013].

Sala, T., 2012. *The mobile cross-platform development headache.* [Online]
Available at: http://indiedevstories.com/2012/03/14/the-mobile-cross-platform-development-headache/
[Accessed 18 04 2013].

Scribd, 2004. *Intermediate Languages.* [Online]
Available at: http://www.scribd.com/doc/16045582/Intermediate-Languages
[Accessed 22 04 2013].

Sherrod, A., 2005. *Texture Compression Techniques and Tips.* [Online]
Available at:
http://www.gamasutra.com/view/feature/2496/texture_compression_techniques_and_.php
[Accessed 18 04 2013].

Sikora, D. & Hattan, J., 2009. *Advanced Game Programming (a gamedev.net collection).* Boston: Course Technology.

Singh, S., Cheok, A. D. & Kiong, S. C., 2004. A Step Towards Anywhere Gaming. pp. 357-358.

SlimDX Group, 2012. *Download.* [Online]
Available at: http://slimdx.org/download.php
[Accessed 25 04 2013].

SlimDX Group, n.d. *SlimDX Features.* [Online]
Available at: http://slimdx.org/features.php
[Accessed 27 11 2012].

Sneddon, J.-E., 2011. *Linux Users Continue To Pay Most for the Humble Indie Bundle.* [Online]
Available at: http://www.omgubuntu.co.uk/2011/12/linux-users-continue-to-most-for-the-humble-indie-bundle/
[Accessed 17 04 2013].

Sonmez, J., 2013. *Cross Platform Game Development with MonoGame.* [Online]
Available at: http://pluralsight.com/training/courses/TableOfContents?courseName=monogame
[Accessed 25 04 2013].

Sony, 2010. *PS3 OS.* [Online]
Available at: http://community.us.playstation.com/t5/PlayStation-3/PS3-os/td-p/6521212/page/2
[Accessed 18 04 2013].

Sony, n.d. *Open Source Software used in PlayStation 3.* [Online]
Available at: http://www.scei.co.jp/ps3-license/index.html
[Accessed 18 04 2013].

Sousa, B. M. T. d., 2002. *Game Programming All In One.* Indianopolis: Premier.

Stack Exchange, 2009. *What is a mutex?.* [Online]
Available at: http://stackoverflow.com/questions/34524/what-is-a-mutex
[Accessed 18 04 2013].

Stack Exchange, 2009. *What is a race condition?.* [Online]
Available at: http://stackoverflow.com/questions/34510/what-is-a-race-condition
[Accessed 18 04 2013].

Stack Exchange, 2010. *What does [STAThread] do?.* [Online]
Available at: http://stackoverflow.com/questions/1361033/what-does-stathread-do
[Accessed 23 04 2013].

Stack Exchange, 2011. *Programming under virtual machine - pros and cons.* [Online]
Available at: http://stackoverflow.com/questions/3371784/programming-under-virtual-machine-

pros-and-cons
[Accessed 17 04 2013].

Stack Exchange, 2011. *What are the limitations of virtual machines?.* [Online]
Available at: http://superuser.com/questions/229105/what-are-the-limitations-of-virtual-machines
[Accessed 25 04 2013].

Stack Exchange, 2011. *Why do game developers prefer Windows?.* [Online]
Available at: http://programmers.stackexchange.com/questions/60544/why-do-game-developers-prefer-windows
[Accessed 17 04 2013].

Stack Exchange, 2011. *Why is it so difficult to emulate PS2 games in a PS3?.* [Online]
Available at: http://gaming.stackexchange.com/questions/17859/why-is-it-so-difficult-to-emulate-ps2-games-in-a-ps3
[Accessed 18 04 2013].

Stack Exchange, 2012. *11.10 vs 12.04 Which should I choose for stability? [closed].* [Online]
Available at: http://askubuntu.com/questions/128542/11-10-vs-12-04-which-should-i-choose-for-stability
[Accessed 17 04 2013].

Stack Exchange, 2012. *8 logical threads at 4 cores will at a maximum run 4 times faster in parallel?.* [Online]
Available at: http://stackoverflow.com/questions/10403201/8-logical-threads-at-4-cores-will-at-a-maximum-run-4-times-faster-in-parallel
[Accessed 24 04 2013].

Stack Exchange, 2012. *File permission issues with shared folders under Virtual Box (Ubuntu Guest, Windows Host).* [Online]
Available at: http://unix.stackexchange.com/questions/52667/file-permission-issues-with-shared-folders-under-virtual-box-ubuntu-guest-wind
[Accessed 23 04 2013].

Stack Exchange, 2012. *How to get number of CPU's logical cores/threads in C#?.* [Online]
Available at: http://stackoverflow.com/questions/13015794/how-to-get-number-of-cpus-logical-

cores-threads-in-c
[Accessed 24 04 2013].

Stack Exchange, 2012. *Is it possible to multi thread something that calls GPU?.* [Online]
Available at: http://stackoverflow.com/questions/11926438/is-it-possible-to-multi-thread-something-that-calls-gpu
[Accessed 24 04 2013].

Stack Exchange, 2012. *Multithreaded Rendering in OpenGL.* [Online]
Available at: http://stackoverflow.com/questions/11097170/multithreaded-rendering-on-opengl
[Accessed 18 04 2013].

Stack Exchange, 2012. *Performance/architectural implications of calling SpriteBatch.Begin/End in many different places?.* [Online]
Available at: http://gamedev.stackexchange.com/questions/24298/performance-architectural-implications-of-calling-spritebatch-begin-end-in-many
[Accessed 24 04 2013].

Stack Exchange, 2012. *Should I use Game Engines to learn to make 3D games?.* [Online]
Available at: http://gamedev.stackexchange.com/questions/22262/should-i-use-game-engines-to-learn-to-make-3d-games
[Accessed 17 04 2013].

Stack Exchangfe, 2013. *monogame play video.* [Online]
Available at: http://stackoverflow.com/questions/15549973/monogame-play-video
[Accessed 24 04 2013].

Taylor, C., 2011. *How Xamarin Gave Mono Life After Novell.* [Online]
Available at: http://gigaom.com/2011/12/12/xamarin-mono/
[Accessed 14 11 2012].

Timothy, 2013. *Steam For Linux: A Respectable Showing.* [Online]
Available at: http://linux.slashdot.org/story/13/03/02/0840225/steam-for-linux-a-respectable-showing
[Accessed 18 04 2013].

Tong, C., 2011. *Remote Debugging in Visual Studio: Squashing Bugs in their Native Environment.* [Online]
Available at: https://www.simple-talk.com/dotnet/visual-studio/remote-debugging-in-visual-studio-squashing-bugs-in-their-native-environment/
[Accessed 22 04 2013].

Transgaming, 2013. *Cider.* [Online]
Available at: http://transgaming.com/cider
[Accessed 17 04 2013].

Treglia, D., 2002. *Game Programming Gems 3.* Hingham: Charles River Media.

Tsakiris, G., 2008. *VirtualBox: access Windows-host shared folders from Ubuntu-guest.* [Online]
Available at: http://www.giannistsakiris.com/index.php/2008/04/09/virtualbox-access-windows-host-shared-folders-from-ubuntu-guest/
[Accessed 17 04 2013].

Ubuntu, 2008. *Virtual Machines.* [Online]
Available at: http://ubuntuforums.org/showthread.php?t=778437&highlight=virtual+machines
[Accessed 17 04 2013].

Ubuntu, 2012. *file sharing between ubuntu 11.10 and windows 7.* [Online]
Available at: http://ubuntuforums.org/showthread.php?t=1944492
[Accessed 17 04 2013].

Ubuntu, 2013. *Releases.* [Online]
Available at: https://wiki.ubuntu.com/Releases
[Accessed 17 04 2013].

Unity, 2013. *Multiplatform.* [Online]
Available at: http://unity3d.com/unity/multiplatform/
[Accessed 17 04 2013].

Valve, n.d. *Porting Source to Linux.* [Online]
Available at:
https://developer.nvidia.com/sites/default/files/akamai/gamedev/docs/Porting%20Source%20to%

20Linux.pdf
[Accessed 17 04 2013].

Vaughan-Nichols, S. J., 2011. *Is Mono Dead? Is Novell Dying?.* [Online]
Available at: http://www.zdnet.com/blog/open-source/is-mono-dead-is-novell-dying/8821
[Accessed 14 11 2012].

VisualSVN, n.d. *HELP // VisualSVN Server certificate key usage violation in Subversion clients built against.* [Online]
Available at: http://www.visualsvn.com/support/topic/00056/
[Accessed 17 04 2013].

Wallen, J., 2000. *The Linux files: The differences between Linux and Windows files.* [Online]
Available at: http://www.techrepublic.com/article/the-linux-files-the-differences-between-linux-and-windows-files/5033390
[Accessed 17 04 2013].

Whitaker, R. B., n.d. *Drawing Text With SpriteFonts.* [Online]
Available at: http://rbwhitaker.wikidot.com/drawing-text-with-spritefonts
[Accessed 24 04 2013].

Wikipedia, 2009. *Console Wars.* [Online]
Available at: http://en.wikipedia.org/wiki/Console_wars
[Accessed 18 04 2013].

Wikipedia, 2010. *Cross-platform.* [Online]
Available at: http://en.wikipedia.org/wiki/Cross-platform
[Accessed 18 04 2013].

Wikipedia, 2013. *Comparison of OpenGL and Direct3D.* [Online]
Available at: http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D#Direct3D
[Accessed 17 04 2013].

Wikipedia, 2013. *DirectX.* [Online]
Available at: http://en.wikipedia.org/wiki/DirectX
[Accessed 17 04 2013].

Wikipedia, 2013. *Filesystem Hierarchy Standard.* [Online]
Available at: http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
[Accessed 17 04 2013].

Wine HQ, 2013. *Wine HQ.* [Online]
Available at: http://www.winehq.org/
[Accessed 17 04 2013].

Xamarin, 2010. *Adding linked files to a project?.* [Online]
Available at: http://mono.1490590.n4.nabble.com/Adding-linked-files-to-a-project-td2019997.html
[Accessed 22 04 2013].

Xamarin, n.d. *About.* [Online]
Available at: http://www.mono-project.com/About
[Accessed 14 11 2012].

Xamarin, n.d. *FAQ: Licensing.* [Online]
Available at: http://www.mono-project.com/FAQ:_Licensing
[Accessed 19 11 2012].

Xamarin, n.d. *MonoDevelop.* [Online]
Available at: http://monodevelop.com/
[Accessed 17 04 2013].

Xamarin, n.d. *What's new in MonoDevelop 2.4.* [Online]
Available at: http://monodevelop.com/download/what%27s_new_in_monodevelop_2.4
[Accessed 17 04 2013].

Young, G., 2012. *SpinLock.* [Online]
Available at: http://mono.1490590.n4.nabble.com/SpinLock-td4657203.html
[Accessed 24 04 2013].