**(Game Title Here and mention it is the GDD/TDD)**

Word Count: 4420

Alexander McRoberts, Kyle O’Donnell

Contents

[Game Design Document: 2](#_Toc497815040)

[Game Outline: 2](#_Toc497815041)

[Initial Team Roles: 2](#_Toc497815042)

[Revised Team Roles: 3](#_Toc497815043)

[Graphical Information: 3](#_Toc497815044)

[Gameplay and Mechanics: 5](#_Toc497815045)

[Powers: 6](#_Toc497815046)

[Story: 7](#_Toc497815047)

[Narrative: 7](#_Toc497815048)

[Technical Design Document: 7](#_Toc497815049)

[Control Scheme: 7](#_Toc497815050)

[Target Audience: 7](#_Toc497815051)

[Scope (Level Design): 8](#_Toc497815052)

[UI: 8](#_Toc497815053)

[Milestones: 8](#_Toc497815054)

[Requirements: 8](#_Toc497815055)

[Development Methodology: 9](#_Toc497815056)

[Testing: 11](#_Toc497815057)

[Quality Assurance: 13](#_Toc497815058)

[Resources: 14](#_Toc497815059)

[Work Done: 14](#_Toc497815060)

[Information Sources Used: 17](#_Toc497815061)

# Game Design Document:

# Game Outline:

To add: A game overview, summary, theme and to justify them

Overview: The game that we plan to make is a fairly lighthearted 3D platformer game focusing around our protagonist (….) someone with the power to absorb powers from defeated foes much like Megaman from the Capcom franchise of the same name. The game would focus on this element as the protagonist wanders about the in game world seeking to (….) via utilizing the assorted powers the player can obtain.

The game’s main selling point is the ability to gain powers based on the enemy you just defeated and using them to progress in the game.

The theme of the game as it were would be a pretty cheerful one as all we are really aiming for is for the player to enjoy themselves with the powers and world offered rather than making things deep and philosophical. This design choice was selected both because we do not really have the time, and skills, required to make something truly deep and interesting thematically and because we at least have some experience with trying to make fun games in the past.

# Initial Team Roles:

Initially the team established consisted of three team members and the work was allocated as appropriate. Though it is important to note that every team member will be doing some of everything anyway.

* Alexander McRoberts: Documentation Lead + Team Leader
* Kyle O’Donnell: Lead Programmer
* Jordan Reilly: Lead Asset Creator

**Justifications**:

Kyle was assigned the lead programmer role primarily because he has either the most, or the second most amount of experience with the Unity engine which is what we will be using to make the game. In addition to this he already had some experience making platformers in Unity as seen when he provided the base of one during the second week.

Jordan was assigned the role of asset creator largely on request as he, apparently had a fair bit of modelling and animation experience.

Alexander received his primary role, the role of documentation lead, because of his prior experience making Game design documents in other circumstances whether for college or in his free time. In addition in was usually the highest rated work handed in when he submitted work in college suggesting he was better at documentation than other tasks like coding or animation. His secondary role was given to him more because of attendance than anything else as he had been the most consistent member in by that point meaning he’d be around to interact with all of the members whether they are in or not.

# Revised Team Roles:

Due to complications with certain team members, namely Jordan Reilly transferring to another course his role of asset creator had to be re-distributed amongst the two remaining team members. For the most part both Alexander and Kyle kept their previous assigned tasks but were given new work to do as well leading to the roles to be as follows.

**Alexander McRoberts**: Documentation Lead, Team Leader, Assed Creator (concept art and basic modelling)

**Kyle O’Donnell**: Lead Programmer, Asset Creator (modelling and animation)

**Justification:** The roles were established like this as, considering he was doing documentation anyway, making Alexander design the concept art seems sensible enough. He was given only basic modeling duties because of his inexperience with animation.

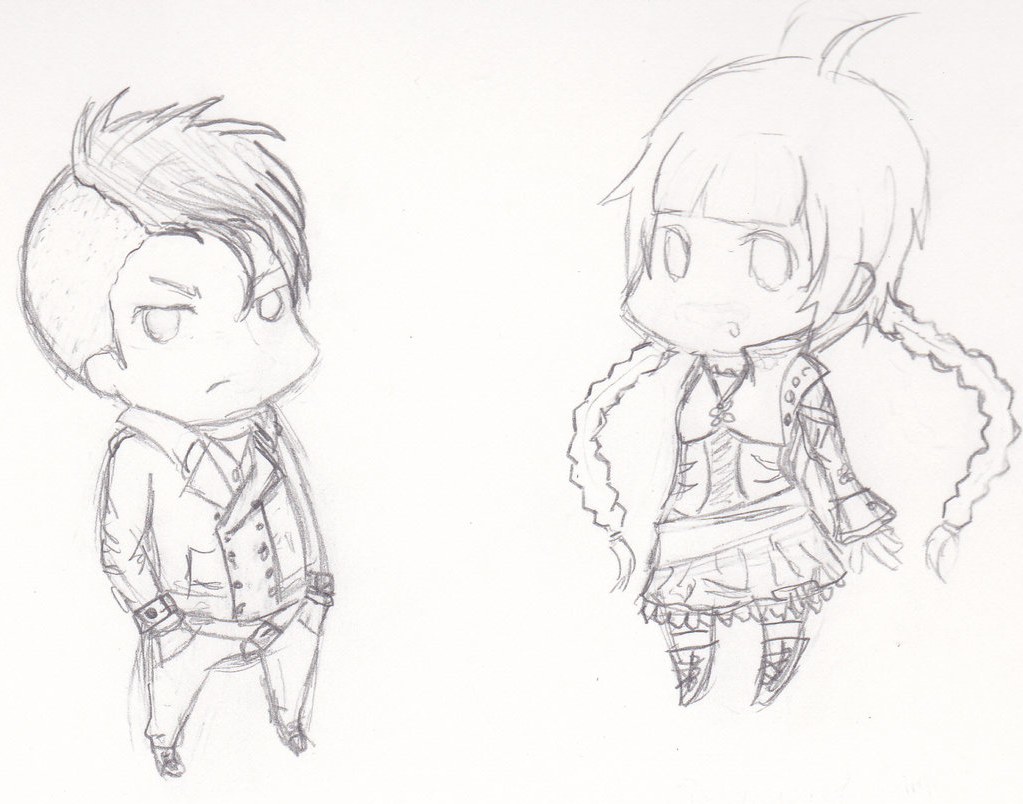
Kyle was left to also model and design some characters to tie into his programming work. This was deemed as reasonable because he seems more confident with animation and he’d have to do the rigging to match up with how he wants things to move in game anyway.

# Graphical Information:

To add: Concept art, models, information about the aesthetics, storyboards, asset list.

Graphical Style: The game’s overall art style was eventually decided to be a fairly ‘chibi’-like style. The reason this was decided upon was mostly because of it’s a fairly lighthearted non-serious style rather which suits a power-up intensive 3D platformer far more than a more realistic style such as the one seen in Naughty Dogs Uncharted franchise.

Another factor towards choosing this style was for modelling the player character and other enemies. Simply put, whilst the team might be able to model a realistic human in Maya it would take too much time which could negatively affect the game. On the other hand modelling a chibi character model takes far less time and is thus a more efficient and effective way to make models for the game.





The above images are the basic concept sketches of possible main character designs.

The models themselves will be modelled and textured in using Autodesk Maya before importing them into Unity for implementation.

# Gameplay and Mechanics:

To add: Information about the game, how it flows and other information such as AI.

For the most part the game will play like a standard 3D platformer. The protagonist can run around on a 3D plain jump and, in addition, can punch things to eventually defeat them. Upon defeating an enemy however the game’s main mechanic comes into play. Powers. Each defeated enemy drops a specific type of power which changes what the player can currently do. From fists to projectiles to time control the main selling point of the game would be all the powers available.

The game would flow based on this with the player defeating one enemy and picking up their specific power only to defeat another enemy with it before gaining a different one and having to use that. This chain of gain, use, gain would continue until the level is completed or the player chooses not to collect another power.

Within each level assorted gems would litter the level which would act as a form of currency. After the player collects enough of these then they may buy items such as extra health from an in game store.

Powers:

As a major part of our game we went over several different ideas for which power-ups or powers the player should get. We ultimately only included a few of our original ideas with the discarded ideas listed below.

Discarded Powers:

Super Speed: Increased movement speed at the cost of other weapons. This was left out mostly out of personal preference as, while it would likely be easy to implement, it wouldn’t add anything to the game. It also wouldn’t allow the player to gain any more powers as they would lose all of their methods of attack rendering them incapable of beating an enemy for a new one. As a result the power was deemed too gimmicky and was dropped.

Flight: The power of flight allowing the player to move freely in the air. This was left out because of how pitifully easy it would be for it to break the game. Without having levels specifically built around it flight would allow the player to move more-or-less wherever they wanted skipping puzzles and enemies entirely as they move towards the goal.

Telekinesis: Initially was going to grant the ability to manipulate objects and creatures in the environment moving them around remotely. It was scrapped due to being far too complicated as it’d need a game all to itself to properly do such a mechanic justice and just putting it in would likely break several of the levels we designed.

The powers that we did choose to include are listed below.

Used Powers:

Fire Shot: Powered by a fire elemental this ranged attack deals damage and causes enemies to burn making them take extra damage as well.

Earth Shot: The force of a golem put into action. This takes the form of another attack dealing great damage in addition to reducing the targets defensive power.

Wind Shot: Granted by a spirit of the wind this is also a ranged attack that, while weaker than the others, pushes enemies back granting the player some breathing room.

Water Shot: The rage of the seas… or at least a mermaid. A blast of water with the second lowest damage in the game drenches the enemy slowing them down for a set time.

Super Strength: Brute force, plain and simple. A simple damage upgrade allows the player’s basic physical attack to deal the highest amount of damage in the game enabling them to tear through any enemy provided they’re close enough to connect.

Time Stop: Chronokinesis or the ability to manipulate time employed in one specific method. Using this deprives the player of a conventional attack instead granting them the power to freeze the flow of time for a set period. Doing so will stop all enemies and projectiles for a brief moment allowing the player to pass by unopposed.

# Story:

To add: Information about the game’s story

Initial Story Ideas: While we never intended for the story to be the major part of the game (not being very skilled writers) we threw around a couple of different ideas before settling on the final model only discarding them when they didn’t fit with some aspect of our game.

Biological Experimentation: This idea was initially put forward by Kyle as one of the first suggestions for the game’s story. It was focused around the idea of the protagonist, Nate Breckner, being involved in some human experimentation gone wrong. This was to be the source of the player’s ability to gain other powers. The player was to discover that the experiment had been intentionally sabotaged as part of some government conspiracy with the player discovering more about it as the game progressed.

While this did provide a justified reason for the protagonist’s powers it didn’t really fit with the chosen art style nor with the gameplay type either as ‘light hearted fun 3D platformer’ and ‘Government conspiracy’ do not really match. On top of this the plot sounded like something you’d see in a spy movie rather than a 3D platformer. Ultimately, not matching up with the game we were trying to make was the reason we didn’t go with this plot.

Final Story Idea: In the end what we went with was something a bit simpler and child friendly. The game is based around the protagonist (…) waking up in a strange land where enemies, upon being defeated, grant you powers. The ‘story’ of the game as it were would basically be trying to use the powers obtained to advance in and escape the world.

It’s an Excuse Plot in every sense of the phrase and outside of the manual would likely not be mentioned. The reason this was chosen was because of the key fact that neither me nor Kyle is a skilled writer and thus, rather than try to write something deep and botch it up spectacularly we chose to go with something that was simple, effective and could be ignored instead figuring that this would allow greater emphasis to be placed on the gameplay.

# Narrative:

To add: information about the game’s narrative

# Technical Design Document:

# Control Scheme:

To add: Information about the games control scheme.

# Target Audience:

Considering that the game’s art style, genre and mechanics seem like something you would find in a lower rated cheerful game we decided it would be best to have our target demographic be children aged 7 and upwards. This is because we feel that the specific style, lack of plot and more whimsical nature of the game would appeal to them most.

As a result of the factors listed above we’d be aiming for a PEGI 7 rating.

Outside of the primary demographic listed above though there would be a periphery demographic as well. To be specific we’d be aiming at those who enjoyed the other 3D platformers such as Spyro the Dragon, Crash Bandicoot and any of the 3D Mario games. Based on the age of those franchises (Spyro: started in 1988, Crash Bandicoot: started in 1996, Mario: first 3D platformer released in 1996) this audience will either consist of adults or teenagers who learned about them through later games in the franchise. Hopefully the platforming and similar game style will be enough to attract such players.

# Scope (Level Design):

To add: Information about the games scope

The games size or scope would be a few world each with a few (say 3-4) levels each. Obviously for the prototype all of those levels will not be accessible and instead only the tutorial level and another couple of levels will be playable instead.

# UI:

To Add: Information about the UI and any maps in the game.

# Milestones:

To add: A list of dates that tasks are to be completed by.

* First Project hand in: 14/11/2017
* Final project hand in 1/12/2017

# Requirements:

To add: A list of functional and non-functional requirements for the game.

Functional Requirements:

* The game must have at least two powers that the player can gain access to.
* The game must allow for full 3D movement.
* The game must be playable using both controller and keyboard inputs.
* At least the protagonist must be a fully 3D rigged and rendered model.

Non-Functional Requirements:

* The projects initial draft must be finished by the 14th of November 2017.
* The projects final draft must be complete and handed in by the first of December 2017.
* The game must be a 3D platformer.
* The game must be created inside Unity.
* The game must follow all appropriate copyright laws.

# Development Methodology:

Before selecting the design approach we settled upon we examined several others though we discarded them for various reasons. Listed below are some of them.

Waterfall: the waterfall approach is a linear design approach that has the team go from one step from another in sequence without going back to any previous steps. The essential principles behind this were:

* Dividing the project into individual phases approached in sequence.
* Placing a major emphasis on points such as schedules, dates, and budgets and implementing the entire game at one time.
* Managing and extremely tight control over the project throughout its life using a large amount of documentation and formal reviews performed at the end of each stage.

The approaches strengths were as follows:

* It’s considered useful for less experience teams as it follows such a rigid straightforward schedule it’s hard to get lost.
* The extremely orderly manner in which the individual steps are performed, when combined with the frequent reviews can result in the developed software being of high quality.
* It’s easy to measure how the systems development is coming along.
* It also saves resources as everybody is following a strict schedule and knows what they’re doing.

As for the weaknesses it has:

* It is utterly inflexible and not particularly fast because of the slow step by step process and strict limitations on what could be done.
* The project rarely, if ever, moves back meaning that mistakes or changes can’t be easily fixed.
* It essentially needs everyone to know what they’re doing right at the start despite the fact that the client might not know what that is.
* It’s not uncommon to find mistakes, inconsistencies and unexpected problems when following this approach.
* The project cannot really be tested until it is entirely complete meaning errors and problems (such as it not working as desired) may show up and to be pretty difficult to fix. It’s also difficult to adjust to life-cycle changes and thus advantageous changes may not be applied as a result.
* Creating the documents and keeping them up to date is taxing and time consuming and such in depth documents may be difficult for clients or even for certain team members to read or appreciate.

In the end while this makes the approach useful for tasks that have clear end goals and unchanging requirements for our task, a game with requirements that will likely change and that has no definite end goal it is pretty useless. On top of this we will often need to go back to previous steps after completing some work such as updating the documentation with concept art or the UI section once those elements are designed and implemented so a linear design approach isn’t really applicable to our task.

Another approach we looked over was the Spiral approach. The spiral approach was a design approach that could use elements from the Waterfall, Prototype or incremental approaches with the name coming from the fact it goes around in several loops. The Key design principles for this approach were:

* Enabling risk avoidance by separating the project into several smaller sections that can be more easily changed.
* Having each individual stage follow the exact same steps: determine objectives, alternatives and limitations of the project, evaluate the alternatives found in the previous step and note and resolve any risks, develop and confirm a deliverable, go start planning the next iteration.
* Each cycle starts by identifying who the stakeholders (or clients) are and what requirements they have that need met.

As for the strengths of such an approach they were:

* It enables risks to be more easily avoided.
* As it can use a mix of several other development methodologies it can be used for a variety of different tasks.
* It allows for changes to be implemented without major issue later on in the project’s development.

Its disadvantages on the other hand were:

* It is usually expensive.
* The people involved in the process require a certain level of expertise and skill to properly use this design approach. This is at least partially because determining the right mix of Waterfall, Prototype and Incremental approaches to use for the specific project can be rather difficult.
* There are no real deadlines set meaning it could fail to meet a required completion date.
* It requires a huge amount of documentation which can be awkward and difficult to properly manage.

In the end, despite having some good points to it (avoiding risks, and allowing changes to be made later in the project) the downsides, most notably the fact that it requires experienced and skilled staff to carry out made it unfeasible and neither of us is particularly skilled in the required areas nor do we really know how to work out how much of each approach (Waterfall, Prototype and Incremental) to use.

In the end the design choice that we *did* choose was the one listed below.

Agile: Agile development is a name for an assortment of iterative, incremental design approaches. To be exact, it isn’t so much a strict base focused on following a step by step process instead being an adaptable people focused approach that relies upon several key principles known as the Agile Manifesto.

* Individuals and interactions over processes and tools.
* Functional software over detailed documentation.
* Customer collaboration over contract changing.
* Responding to change over following a plan.

The approach is more focused towards projects that don’t have specific goals or plans laid out for them as, if you did know that, then a waterfall approach or something similar might be better.

The strengths of the approach are:

* It allows changes to be performed even late into the project’s life cycle.
* It focuses on face to face communication allowing for better communication between team members.
* Working software is produced quickly.
* It has a high customer satisfaction rate.

Its disadvantages on the other hand are:

* It isn’t always easy to assess how much work will need put in for larger projects.
* If the customer isn’t clear as to what they want it’s easy for the project to go ‘off the rails’ as it were.
* It doesn’t focuses as much on required documentation and designing as it should.

The key reason we decided to use a version of Agile development was its adaptability. Compared to most professional plans the project doesn’t really have any clear requirements or end goals beyond ‘hand in a prototype within the time-frame’ so an adaptable approach is best for us as we likely *will* be modifying the document right up until the end of development.

We won’t be following all of key principles though, instead while we’ll take inspiration and influence from certain parts of the design model (such as individuals over processes, software over documentation, changes over a plan) there will be certain parts that we either can’t follow (customer collaboration over following a plan) or that we will not follow (not focusing on documentation). In regards as to why we won’t follow the not focusing on documentation stage it is because the project need a detailed document to be handed in at the end of it so not doing so would merely fail the project.

# Testing:

To add: Information about how the game will be tested along with technical test plans and filled out tests.

While the game will be tested constantly throughout its production with a variety offsets relating to its implementation the final block testing of everything that the game should be capable of can be carried out in one or more methods. Specifically there are two separate testing methods we considered for this Black box testing and white box testing.

A list of intended tests for the game is listed below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test | How the test is carried out | Expected Result | Actual Result | Comments |
| 001 | Player Movement: Left |  | The player moves to the left by a specified amount. |  |  |
| 002 | Player Movement: Right |  | The player moves to the right by a specified amount. |  |  |
| 003 | Player Movement: Forward |  | The player moves forward by a specified amount. |  |  |
| 004 | Player Movement: Back |  | The player moves backwards by a specified amount. |  |  |
| 005 | Player Jump |  | The player jumps up by a specific value. |  |  |
| 006 | Gravity |  | The player falls for a bit before hitting the ground. |  |  |
| 007 | Camera Tracking |  | The camera follows the player. |  |  |
| 008 | Basic Attack |  | The player goes through the attack animation. |  |  |
| 009 | Enemy vs Basic attack |  | The enemy either takes damage or is defeated. |  |  |
| 010 | Power Collection: Super Strength |  | The super strength power is collected. |  |  |
| 011 | Power Collection: Fire |  | The fire power is collected. |  |  |
| 012 | Power Collection: Water |  | The water power is collected. |  |  |
| 013 | Power Collection: Earth |  | The earth power is collected. |  |  |
| 014 | Power Collection: Wind |  | The wind power is collected. |  |  |
| 015 | Shot: Fire |  | A fire projectile is created and fired. |  |  |
| 016 | Shot: Water |  | A water projectile is created and fired. |  |  |
| 018 | Shot: Earth |  | An earth projectile is created and fired. |  |  |
| 019 | Shot: Wind |  | A wind projectile is created and fired. |  |  |
| 020 | Enemy vs Super Strength |  | The enemy takes more damage. |  |  |
| 021 | Enemy vs Fire Shot |  |  |  |  |
| 022 | Enemy vs Water Shot |  |  |  |  |
| 023 | Enemy vs Earth Shot |  |  |  |  |
| 024 | Enemy vs Wind Shot |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

A completed version of the test plan is shown below:

# Quality Assurance:

To add: Information about how we’ll handle quality assurance.

# Resources:

**Software:** In order to properly make the game a collection of various programs or software will be required either to implement the game or to produce some of the assets for the game. The programs we suspect we’ll need will be listed below.

|  |  |
| --- | --- |
| Software | Price |
| Unity | Free |
| Microsoft Word | Already owned |
| Microsoft Powerpoint | Already owned |
| Autodesk Maya 2018 | Available for free for students |
| Paint.net | Free |
| Paint | Free |
| Audacity | Free |
| Adobe Photoshop | Already Owned |

As can be seen above all of the software listed is either free for everyone, free for students or are already possessed by one or both team members. This means that none of the team members need to waste money purchasing the products to work on the task.

**Hardware:** Hardware is also required in order to produce the game as several pieces are either mandatory for getting the computer to function (such as a keyboard or monitor) or useful for producing part of the game or documentation.

|  |  |
| --- | --- |
| Hardware | Price |
| Monitor | Owned already |
| Laptop | Owned already |
| Mouse | Owned already |
| Keyboard | Owned already |
| Graphics Tablet | Owned by 1 member already (no longer needed) |
| USB Memory Stick | Already Owned |
| Scanner | Already Owned |
| Desktop Computer | Already Owned |

As can be seen above all of the necessary hardware is already possessed by at least one of the team members thus reducing any required costs to 0 barring accidents.

Extra Notes: it is worth mentioning that most of the hardware and software listed above *is* available inside the university campus thus allowing for work to be performed there. It is somewhat more limited though as some labs, and as a result, software may not be available if a class is in session at the time.

# Work Done:

Week 1:

* As the induction week nothing was done

Week 2:

* The task and teams were established.
* Contact information was shared within the group.

Week 3:

* Kyle completed a very basic base to use for the game.
* Alexander created a github location for the project.
* Alexander created a base for the GDD.
* The group as a whole decided we would be working on a 3D platformer.

Week 4:

* Jordan left the group to transfer to another course. This resulted in team roles being reallocated.
* We had a talk about how to re-allocate team roles along with what kind of style we wanted the game to follow.
* Kyle improved the basic movement of the main character in the base made for the game.
* Alexander filled out the team roles section, some of the resources section, part of the milestones section and part of the design approach section within the Game Design Document.
* Alexander uploaded an unfinished GDD to github.

Week 5:

* Alexander worked on some concept art for some of the game’s enemies.
* Kyle designed some concept art for the protagonist.
* Alexander worked on the design document some more filling part of the story, art, and game outline.
* Kyle worked on the games implementation some more.
* Kyle completed the main character’s model.
* Alexander uploaded an updated design document to github.

Week 6:

* Alexander added information to the GDD namely to the Target audience, testing and QA, story and Resources sections.
* Kyle continued producing the frame of the game in Unity.
* Kyle started rigging the main character’s model.
* Kyle added some work to the GDD specifically he added information to the resources section.

Week 7:

* Alexander added more information to the GDD’s scope and requirements sections.
* Alexander added more information to the GDD’s design approach section.

Week 8:

* Alexander added more to the GDD’s Design approach section.
* Alexander added information to the GDD’s resources section.
* Alexander created a 3D asset (a Gem) for use in the game.
* Alexander re-worked the referencing style to match what was expected for UWS.
* Kyle handed over an example of the current game state to Alexander.
* Alexander designed the template for the presentation in PowerPoint.

Week 9:

* Alexander updated the GDD’s Design approach section.
* Alexander updated the GDD’s Testing and QA section.
* Alexander split parts of the GDD into different section.
* Alexander reorganized the GD’s layout.
* Alexander created the model for an enemy in the game.

# Information Sources Used:

**Websites Accessed:**

**Known Author:**

Penna Sparrow. (2017) Spiral Model : Advantages and Disadvantages [Online] Available: <http://www.ianswer4u.com/2011/12/spiral-model-advantages-and.html#axzz4wPdB45M3> [Accessed 31/10/2017]

Luke Plunkett. (2016) The Art of Uncharted 4 [Online] Available: <https://kotaku.com/the-art-of-uncharted-4-1777213424> [Accessed: 03/10/2017]

Lizzy Nyanko. (2015) What is Chibi? [Online] Available: <https://honeysanime.com/what-is-chibi-definition-meaning/> [Accessed 03/10/2017]

**Unknown Author:**

How To Write Your First Game Design Document. (2014) [Online] Available: <https://www.nyfa.edu/student-resources/how-to-write-a-game-design-document/> [Accessed 26/06/2017]

PEGI. (2017) [Online] Available: <http://www.pegi.info/en/index/> [Accessed 10/10/2017]

Excuse Plot. (2017) [Online] Available: <http://tvtropes.org/pmwiki/pmwiki.php/Main/ExcusePlot> [Accessed 10/10/2017]

Agile Development – Advantages, Disadvantages and when to use it?. (2017) [Online] Available: <http://www.360logica.com/blog/agile-development-advantages-disadvantages-and-when-to-use-it/> [Accessed 07/11/2017]

**Games Used:**

Crash Bandicoot. (1996) [Video Game, PlayStation, E] Santa Monica: Naughty Dog

Super Mario 64. (1996) [Video Game, Nintendo 64, 3+] Kyoto: Nintendo

Mega Man. (1987) [Video Game, NES, E] Chūō-ku, Osaka: Capcom

Uncharted: Drake’s Fortune. (2007) [Video Game, PlayStation 3, 16+) Santa Monica: Naughty Dog

Spyro the Dragon. (1998) [Video Game, Playstation, E] Burbank, California: Insomniac Games

**Books Accessed:**

**Appendices:**

[**http://moodlearchive2017.uws.ac.uk/pluginfile.php/493736/mod\_resource/content/1/CoRE/index.htm**](http://moodlearchive2017.uws.ac.uk/pluginfile.php/493736/mod_resource/content/1/CoRE/index.htm)