

University of St. Andrews



A Survival Game

by

Tom Clark

Supervisors:

Colin Allison

Tristan Henderson

17/04/2007

Abstract

The aim of this project was to research and show how the expanding field of computer game design and implementation is related to Computer Science concepts and techniques. I did this by creating a game; demonstrating how games can be designed and built using Software Engineering techniques. At the end of this report I state my opinion on whether or not using Software Engineering techniques in game development is useful.

After a context survey of work in the area I decided the game was to be based around the survival of a particular event. Due to time constraints this game did not have to be developed to the level of a commercial game or involve many modern game technologies such as multiplayer. Since the game was not initially intended to be a commercial game that needed to make money I was free to experiment with new and unusual ideas. The game was designed and in this report there are explanations and descriptions of the ideas that were used.

The game was tested to ensure it met its original objectives and worked correctly. Usability testing was also performed by a small-group of game testers who were given versions of the game and provided their opinions on it. I give an evaluation of whether this was helpful.

The User Manual was required to be short and not required reading to play the game. The success or failure of all stages of the project is described.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 15, 302 words long.

In submitting this project report to the University of St. Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Tom Clark

Contents

1. INTRODUCTION.....	5
1.1 THE CREATED GAME	5
1.2 HOW THE PROBLEM WAS SOLVED	6
1.3 SUCCESS IN SOLVING THE PROBLEM.....	7
2. PROJECT DETAILS.....	11
2.1 MAIN IDEAS OF DESIGN	11
2.1.1 <i>Allow the player to do what they want.</i>	11
2.1.2 <i>Why zombies</i>	11
2.1.3 <i>No guns</i>	12
2.1.4 <i>Allow players to make mistakes</i>	12
2.1.5 <i>Top down view</i>	13
2.2 UNUSUAL DESIGN FEATURES	13
2.2.1 <i>Fear</i>	13
2.2.2 <i>Graphics</i>	14
2.2.3 <i>Stat icons</i>	14
2.2.4 <i>Dual purpose wood</i>	15
2.2.5 <i>Stat bars</i>	16
2.2.6 <i>Safe and unsafe areas</i>	16
2.2.7 <i>Progressive learning</i>	17
2.2.8 <i>No examine function for items</i>	17
2.3 NOVEL FEATURES	18
2.3.1 <i>In-game help</i>	18
2.3.2 <i>Levels as areas</i>	18
2.3.3 <i>The minimap</i>	18
2.4 SPECIAL ALGORITHMS	19
2.4.1 <i>AI of survivors</i>	19
2.4.2 <i>AI of Rot</i>	20
2.5 ODD IMPLEMENTATION DECISIONS	22
2.5.1 <i>Walls</i>	22
2.5.2 <i>Invisible zones</i>	23
2.5.3 <i>Picking up items</i>	23
2.5.4 <i>Error Handling</i>	24
2.6 OTHER IMPORTANT DECISIONS IN THE PROJECT	25
2.6.1 <i>The use of Game Maker</i>	25
2.6.2 <i>Involving separate game testers</i>	27
2.6.3 <i>The use of UML in game design</i>	28

3. EVALUATION AND CRITICAL APPRAISAL.....	29
3.1 EVALUATION WITH RESPECT TO THE MAIN PROJECT OBJECTIVES	30
3.1.1 <i>Computer game development as a branch of Computer Science</i>	30
3.1.2 <i>Applying Software Engineering techniques</i>	30
3.1.3 <i>Conclusion on the use of Software Engineering techniques</i>	31
3.1.4 <i>Originality</i>	31
3.1.3 <i>Required hardware</i>	31
3.2 EVALUATION OF THE CREATED GAME WITH RESPECT TO ITS OBJECTIVES	32
3.2.1 <i>Allowing a user to control and move a character</i>	32
3.2.2 <i>Computer-controlled characters and their intelligence</i>	32
3.2.3 <i>In-game items</i>	32
3.2.4 <i>Education of survival techniques and complexity</i>	33
3.2.5 <i>Saving and loading a game</i>	33
3.2.6 <i>Endings</i>	34
3.2.7 <i>Story based upon survival</i>	34
3.2.8 <i>Involvement of strong language</i>	34
3.2.9 <i>Involvement of violence and gore</i>	35
3.3 HOW MY WORK COMPARES TO THAT DONE BY OTHERS	35
3.3.1 <i>Death</i>	35
3.3.2 <i>Digital Rights Management (DRM)</i>	35
3.3.3 <i>Length</i>	36
4. CONCLUSION	36
4.1 SUMMARY OF WORK DONE	36
4.2 STRENGTHS AND WEAKNESSES OF THE GAME	37
4.2.1 <i>Strengths</i>	37
4.2.2 <i>Weaknesses</i>	38
4.3 FUTURE DIRECTIONS	38
REFERENCES	40
APPENDICES	41
PROJECT SPECIFICATION AND PLAN	41
LITERATURE REVIEW AND SYSTEM DESIGN	57
CHANGES TO ORIGINAL DOCUMENTS	83
TESTING SUMMARY	85
USER MANUAL	90
MAINTENANCE DOCUMENT	104
ACKNOWLEDGEMENTS	108

1. Introduction

The purpose of this project was to research and show how the expanding field of computer games design and implementation is related to Computer Science concepts and techniques. A game was required to be created as an example, ensuring it was clear how the game relates to this field. The development of this game had to show how Software Engineering techniques can be applied to game creation. At the end of this project an opinion on whether or not these techniques are helpful is given.

In addition to the above there were also requirements which the developed game needed to meet that were defined after a context survey of similar work had been performed. In general the game had to be original yet fun to play and not require any special or high-quality hardware to run.

1.1 The created game

I chose to base the game on survival of a particular event. This was because I believe 'survival' to be a game genre and it is interesting that reference [2] does not include 'survival', only 'survival horror', implying 'survival' is a relatively unexplored genre. As such, it offered a suitable framework to work within. There are relatively few existing games in this genre to prejudice the style of the game to be created.

Given the restricted time frame the game did not need to be developed to a level that could be published. It did, however, need to be expandable into a full game or give ideas on which future games could be based.

Criteria the game had to satisfy:

- Whether the player-controlled character survived the event when the game was over had to be an absolute yes or no question, not requiring more than 2 different endings.
- The game was required to educate the player in the basic concepts of survival (for example, the need to eat and drink) but warn them that it is not a detailed or complete simulation of survival. This was not intended to heavily impact the entertainment value of the game.
- The game was to support the collection of in-game items that could then be used by the character controlled by the player.
- The game was required to be suitable for players with little knowledge of computers. To this end it had to be easy to use and understand without requiring a large User Manual to be read or any training to be given.

- Characters and any computer controlled enemies within the game were to move with some limited level of intelligence, for example, they could not pass through solid walls unless intended and all enemies had to be able to attack the player-controlled character.
- The game did not need to be able to support multiple players.
- It was a requirement that the game did not require any special hardware, a high-quality or a fast computer to be run.
- As users may not have time to play the whole game from start to finish a way to save and load a started game was needed, ensuring that the items the player had collected are retained.
- The game could not involve vivid scenes of violence or gore. Any scenes like this had to be stylised or kept to a minimum to encourage younger players.
- Strong language was not to be used.

1.2 How the problem was solved

This report aims to show how and to what extent the above problem was solved. Firstly, a "Project Specification and Plan" document was created listing the objectives of the project and game. It also listed in detail the requirements of the solution. This document is given as an appendix on page 41. The success of the solution at solving the problem with respect to the objectives is listed in the "Evaluation and critical appraisal" section on page 29. The "Project Specification and Plan" also included a context survey which listed and evaluated other work in some popular game genres. This context survey was limited to non-commercial games of a similar size to that which could be created in the time constraints of the project.

After this document was completed I created a "Literature review and system design" document. The first part of this document was a more complete survey and review of relevant material, including books and websites. This, like the context survey, was needed to ensure I was not just performing work that had already been done. The second part was the design of the game I would create. This included the design of a prototype with limited functionality and what would be done to expand that prototype into the final game. The "Literature review and system design" document is given as an appendix on page 57.

I also created a more detailed design document which is not included but can be downloaded from the project website [1]. This contained details that I thought unnecessary in the included design document such as screen layouts and level designs, for example.

Next I implemented the game using the Game Maker game development environment [3] that is described in the "Implementation Plan" of the "Project Specification and Plan". I included an evaluation of whether Game Maker was a good development environment to choose on page 25. I first implemented and tested a prototype version of the game to ensure I had the skills needed to complete at least some of the game and achieve some of its objectives. This prototype was demonstrated to supervisors and a presentation was given to explain it. The Microsoft PowerPoint slides used for this interim presentation can be downloaded from the project website [1]. The prototype game was expanded into the full game as designed.

Testing of the game was performed and a summary is given on page 85. Due to time constraints I was unable to perform as much testing as desired. At various stages of the project I provided a small number of game testers with a copy of the game which they could play and provide feedback on how the game was developing. More detail about testing and its success is given in the next section. I evaluate whether this proved useful or not on page 27.

A User Manual was written so users of the game could understand how to play and use all implemented features. This and all other documentation and project related material can be viewed electronically via the project website [1]. The User Manual is also given as an appendix on page 90. A Maintenance Document explains strategies to be used if errors are found after the game was released and how the game can be maintained. It is listed on page 104.

After this report was finalized and delivered the final version of the game was demonstrated and a final presentation given. This presentation gave an overview and summary of parts of this report for which Microsoft PowerPoint slides are given on the project website [1].

1.3 Success in solving the problem

The "Project Specification and Plan" document listed the objectives of the project and game in enough detail for them to be evaluated when finished. There were rather a lot of requirements for the project and game software but they were detailed and complete. Having many requirements made it difficult to ensure I had met all of them. Unfortunately, the plan of the project contained in this document was not very accurate. I found that many planned stages took longer than estimated. This could be because I had never created a computer game of my own design before or that I had very little experience of using the Game Maker development environment. No stage was delayed a catastrophically long time however, which was fortunate.

The "Literature review and system design" document was long but did contain a brief review of 10 books and websites related to game design and/or implementation. The actual design of the game was good enough to implement the game from start to finish. It included UML diagrams to aid understanding. This document was essential in monitoring everything that had been and still needed implementing in the game. However, it did not include any pseudo code or details of how features of the game would be implemented. Inheritance diagrams would have made the structure of the game easier to understand but I did not realise how much use I would make of inheritance until later in the project.

The implementation of a prototype version of the game was useful but, due to university coursework, it did not contain all of the features it was designed to include. It was useful in that it showed that I could implement at least part of the game. It also increased my confidence that I would actually get the game developed on time. I could perform some limited testing on this prototype to ensure that things designed were feasible and could be implemented. However, because this prototype did not include all intended features, I decided not to release it to testers for feedback when planned. Only when I had implemented these features did I release it as explained in the section "Involving separate game testers" on page 27. The feedback from testers was useful as they highlighted some errors I had overlooked or did not consider a problem. One example of a result of feedback is given in section "Picking up items" on page 23.

The prototype was expanded into the final version of the game. This was actually done in parallel with other stages of the project and took longer than initially planned. This is likely because I had never created a computer game of my own design before and I had very little experience of using the Game Maker development environment. Whether the final version met all its original objects or not is evaluated in section 3.2 on page 32. An almost complete final version was released to game testers who provided feedback on the game. This feedback was useful on how to develop the game further.

Not all testing performed is documented in the testing summary on page 85. As explained in the "Literature review and system design" document given as an appendix on page 57, there were two different types of testing; Software testing and Game testing. Software testing was essential in order to ensure the game met its objectives and requirements. It was infeasible, due to time constraints and lack of resources, to test the game on a variety of hardware and operating systems as would be done for a commercial game. This was defined to be outside the scope of the project in the initial stages of its development. Game testing was performed by a small group of game testers, a type of user identified in the "Project Specification and

Plan" document on page 43. This was achieved by releasing prototype versions of the game to the game testers at various stages throughout the project. The game testers were also given a questionnaire to complete and return to the designer. This allowed them to give feedback and their opinions on how the game was developing. All versions of this questionnaire are downloadable from the project website [1]. Game testing was a small-scale approximation to beta testing and usability testing. These are usually performed as part of commercial game development to evaluate a range of users' opinions of a game. I evaluate whether game testing proved useful or not on page 27.

The User Manual written for the game was an important area of the game design. It was intended that it should not be very large nor be required reading in order to play the game. On first inspection the User Manual does appear to be quite large but the majority of its length consists of pictures. I decided it was better in terms of understand-ability to have a document the user could easily relate to the game than a short and concise one containing few pictures. Some of this document is more required reading than initially hoped. For example, the section of the User Manual titled "Playing Hunger" is necessary to explain each stat and what they are used for. This is a problem that could not be solved as I cannot think of any feasible way to communicate this information in-game. I have attempted to minimise the required reading by splitting the manual into sections. It was intended that a user will only need to read the section they are having problems with and not the whole document. This also allows the user to lookup individual sections only when needed and get only the information they need.

Another way I tried to reduce the reliance on the User Manual was by making use of the in-game help facility provided by the Game Maker development environment. This displays a summarised version of User Manual whenever "F1" is pressed while the game is being played. However, the information in this summary had to be brief in order not to slow the game down and reduce gameplay into reading a document, so it cannot be used as a substitute for the actual User Manual.

Writing a Maintenance Document proved difficult as no one involved in the project was really sure what maintenance for a computer game is. Most software maintenance is involved with improving existing functionality and adding new features. These would not usually apply to a game because when a game is released it is considered a final product with any new features being reserved for an expansion pack or sequel. Also there is no on-line multiplayer element to the game so keeping any security algorithms updated for the latest attacks would not be an issue. This leaves:

- 1) Adaptive maintenance involved with ensuring the game runs correctly on new versions of operating systems or hardware.
- 2) Corrective maintenance to ensure any bugs discovered after the project has been released are fixed. This would be done by releasing patches and new versions of the game.

I have written a Maintenance Document but it will be limited when compared to other projects. More information about game maintenance is given in reference [15].

As the project demonstration and final presentation had not been given when this report was delivered I can not comment on their success.

In hindsight, the, potentially, most catastrophic problem encountered in the development of the project was related to Digital Rights Management as mentioned in the "Evaluation and critical appraisal" on page 35. Game Maker, the development environment used to create the game, comes in two versions; a registered version and a free non-registered version. The registered version costs £10 and gives access to more functions than the free version. For the majority of the project I was able to create the game using the free version. However, about half-way through development of the project I discovered a required feature could only be implemented with extra functions provided with the registered version. Normally this would have been easy to correct by simply registering Game Maker via the website. However, at the same time the creator of Game Maker merged with another company, YoYo games [4], and decided to release a new version of Game Maker, version 7.0. They also opted to change the registration system. Due to this change, registration of the previous version of Game Maker, the version I was using to develop the game, was disallowed until the new system was implemented. This would not be done until Game Maker 7.0 was released. I therefore had to wait without being able to complete the project. I also had the risk of the game created with version 6.1 not being compatible with version 7.0 and version 7.0 being delayed beyond the deadline of the project. These risks were both potentially catastrophic but were not included in the risk analysis section of the "Literature review and system design" document on page 79 simply because they were completely unexpected.

Fortunately, despite a delay, the new version was released in time and could be registered. The game was fully compatible with version 7.0 and development could be completed.

2. Project Details

This section summarises the achievements of the project and aspects of the project of particular interest.

2.1 Main ideas of design

A full design for the game can be found in Design section of the "Literature review and system design" document given on page 63. Also a more detailed design is given in the "Detailed Design" document available on the project website [1]. Here I will highlight and give reasons for the main design decisions.

2.1.1 Allow the player to do what they want

Different people have different things they want from a game. Some may want to just complete the game as quickly as possible, while others will want to investigate the story behind why the Rot are there. This decision was made to try and keep the game entertaining for a wide audience of users and not force them into a particular play style they dislike.

A good example of this is the inclusion of newspapers as items which expand the games story when used. These items can be collected at various points within the game world but there is no rule to do this. The game can be successfully completed without collecting any or all newspapers. This provides the user the opportunity to discover and investigate the games story but does not force it.

I realised in the early stages of the project that the inclusion of newspapers as items would significantly increase development time. Since delays to the project were unacceptable this feature had to be left as desirable functionality.

2.1.2 Why zombies

Zombies are very common in computer games today. It was therefore not an obvious choice for an original game. I chose to use zombies as enemies as I really wanted to show that there is more to zombies than how they are portrayed in most games. In most games they are just used as place-filler enemies, just to be shot and killed. I tried to show them as a general threat to people. Also the game is, or was intended to be, at least partially educational. Teaching the user that it was okay to kill other humans is to be avoided.

Zombies inspire horror. They are often grotesque, half-decayed humans. Naturally, this requires large amounts of gore. Since the game created was required to keep any scenes of

violence or gore to a minimum this presented a problem. I had multiple ideas of how to solve this problem. My first idea was to design and implement the game using stylized graphics. However, this did not work as explained in section 2.2.2 on page 14. The solution I eventually decided upon was to move away from the stereotypical image of a zombie and design something more humorous and far less threatening. The design had to be somewhat threatening as it needed to be immediately obvious to the player that the new design of zombie was still an enemy to be avoided. Whether or not this new design achieved these goals is largely decided by individual users.

Another reason was the event the player needed to survive could be more fictional than it could be if a more natural event was used. This meant the event could be an outbreak of zombies as opposed to a volcanic eruption or an earthquake for example. The event being easily recognised to be fictional reinforces the idea that the survival techniques used in the game are only basic guidelines and do not provide a complete guide to survival.

2.1.3 No guns

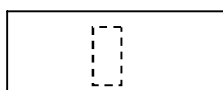
Most modern games involve guns, especially games involving zombies. Again this was a main idea to keep the game original and to minimise scenes of violence and gore as required. It also hopefully teaches users not to buy or use guns. I have stated that allowing the player to do what they want was a major design concept so I needed to include some form of weapon in the game. I decided to include only a single weapon that was simple to use. This was a simple wooden spear that could be rammed into enemies. Since the spear is wooden it breaks after a few collisions with enemies requiring the user to think before killing.

2.1.4 Allow players to make mistakes

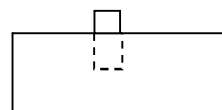
Items are relatively frequent and there are fire barrels scattered around the play area to help survival. Without these it is very difficult to complete the game. Items and fire barrels are not in random places so players (users) can form strategies after multiple plays of the game. To make it obvious which buildings contained a certain type of item, buildings would have special symbols on them to indicate the type of item they contained. This design choice was made early on in development but after more work on the project I discovered that having separate buildings to enter to collect items was a waste and would slow down gameplay. I decided that just representing buildings by walls and not switching and loading different game areas would be more efficient and still achieve the same purpose.

2.1.5 Top down view

I decided on a top-down view for the game with 2D (2-dimensional) sprites (pictures of objects for example, items). This was because the game covers a relatively large area and a town which is more difficult to depict with a side-on view. I could have used isometric sprites on a top-down background. I decided against this because it would be possible to lose sight of the playable character (the character controlled by the user) behind an object such as a wall. This problem would become worse if the character became partially visible and partially hidden by an object.



The playable character is completely hidden behind other objects.



The playable character is partially hidden behind an object and partially visible.

Isometric sprites would have made the game look almost 3-dimensional and more attractive but would be more difficult and time-consuming to implement.

2.2 Unusual design features

Since this game was never intended to be a marketable game I was free to experiment with unusual ideas. Commercial games *have* to make money so must use tried and tested ideas that users like and will pay money for. Commercial games often do contain unusual features but far less than are seen in non-commercial games. The following list gives a brief description and explanation of any unusual features in the game I developed.

2.2.1 Fear

I decided to include a Fear stat for the player. This slowly increases over time and receives a large increase when the playable character "sees" enemies. What is meant by "seeing" an enemy and how it was implemented is stated in the "Literature review and system design" document on page 71. At high levels of Fear the user will start to see hallucinations ('Fear effects'). A complete list of 'Fear effects' is given in the "Design" section of the "Literature review and system design" document on page 67. The occurrence of 'Fear effects' was implemented to add another aspect of learning to the game, in that people can't go on indefinitely without sleep. It was originally designed that 'Fear effects' become more frequent at higher-levels of Fear until the maximum level is reached. This was removed as it would have caused 'Fear effects' to become annoying and predictable. Instead, they would simply occur at random times beyond the threshold value of Fear. When Fear reached its maximum

level the playable character moves randomly and falls asleep where ever they are. 'Fear effects' affect the user's perception of the game more than actual gameplay. For example, many static enemies suddenly appear around the playable character and then quickly vanish. This poses no danger to the playable character as the enemies that appear are not real enemies and cannot attack them. This involves the user in the game more and hopefully makes them want to complete the game and escape the town where the game is set.

I included the fact that greater levels of Fear increase the movement speed of the character to add some balance or trade-off to the game. How much sanity are you prepared to sacrifice for movement speed? I also included a Speed stat in order to make it clear that the playable character's movement speed does increase. Without a visual representation of movement speed increasing at high-levels of Fear the user may miss or not understand the trade-off, therefore, ruining the sense of balance I tried to create.

Once I began implementing this feature I discovered that if the speed increase was directly related to the current value of the Speed stat this made the playable character move at uncontrollable speeds around the game world. I therefore implemented a separate speed increase value. This value was incremented by a small amount each time the Speed stat increased and was added to the base movement speed of the playable character each time they moved. This imposed a smaller maximum and minimum speed up than could be achieved without the special speed increase value.

2.2.2 Graphics

I originally designed for the graphics of the game to be a black and white 'pen and ink' style. This had the advantage of making the images used simpler and less time consuming to draw. I also chose this for some other reasons.

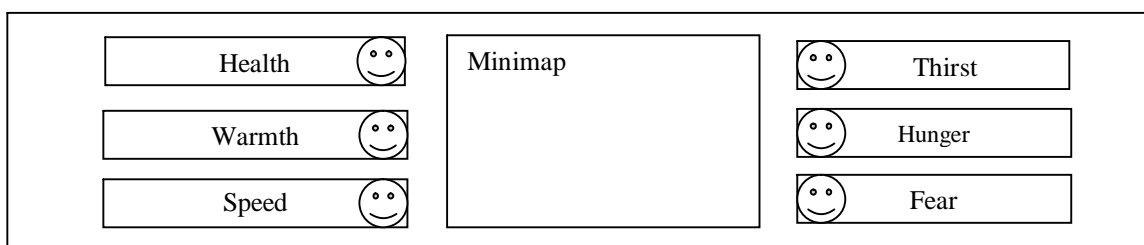
1. The style is original and has rarely been used in computer games before.
2. It 'masks' scenes of violence and gore by not having detailed or coloured images.
3. It makes available the use of random coloured objects as a 'Fear effect'.

Unfortunately this idea was not included in the final version of the design. This was because it was too difficult for the artist and, when implemented, would not have made the game as attractive to younger players as required.

2.2.3 Stat icons

As mentioned in the "Literature review and system design" document the user has to balance several stats in order to keep the playable character alive and complete the game. Some of the

stats need to be kept low and some need to be kept high. For example, the user will want the playable character's health to remain high while their thirst is kept low. While the need to maintain and balance several stats is quite unusual in games, it can be quite confusing if you have not read a User Manual. This is partly because it is unusual and therefore not familiar to the majority of users. I have tried to make this easier to understand by putting different types of stats on different sides of the screen. This means the stats to be kept low are located on one side of the screen and the stats to be kept high on the other. This aims to provide symmetry linking the two groups of stats but also showing they are different. A good improvement to this would be to add small good and bad icons to each end of a stat bar representing a stat. These icons would have to be consistent between stats to avoid confusing the player and would be placed at appropriate ends for stats that need to be kept high and stats that need to be kept low as shown below.



This would immediately show the player how to maintain different stats. Also since many stats can affect the playable character's health, icons would allow the player to easily and quickly recognise the stat which was causing their health to decrease.

2.2.4 Dual purpose wood

The in-game item Wood was given two uses. The first is to make a fire to increase the playable character's Warmth stat. If the Warmth stat reaches zero the playable character gets too cold, will start losing health and will eventually die if no action is taken. The second use of the Wood item is the ability to use it as a weapon which can kill enemies. Killing enemies will likely make it easier for the playable character to survive in some situations but once it is used as one it cannot be used as the other. Using all Wood items as weapons will make it harder for the user to maintain the playable character's Warmth as they cannot make fires. Using all Wood items to make fires will mean the user has no alternative but to avoid enemies. The user will have to decide which use is best for the current situation. This puts an element of 'choice' or strategy into the game. The game was required to educate users in the basic concepts of survival and I believe that making users think for themselves and form their own strategies is a good way to do this.

There is no reward for killing. This decision is intended to show users that there is no advantage to killing both inside and outside the game.

2.2.5 Stat bars

I could have added more realistic effects to the playable character's stats. For example, if the Warmth stat got too high the playable character would get hyperthermia causing their health to decrease. Effects like this may have increased the amount that could be learned from the game but I decided not to include them early on in the design of the game. Too many effects for the user to remember would decrease the entertainment value of the game. The Game Definition given in the "Project Specification and Plan" document on page 47 states that the learning experience provided by the game should not heavily impact its entertainment value.

2.2.6 Safe and unsafe areas

I designed that if a user chose the playable character to sleep outside then they have a higher chance of letting the playable character die and losing the game. If they chose the playable character to sleep inside a building there is less chance of this happening. There were two reasons for doing this:

The first was to add another element of strategy to the game to make the user think. I have already stated I believe that making users think for themselves is the best way to educate them. There are actually two levels to the choice made by the user. One is whether the playable character should sleep inside a building or outside. Hopefully the user will immediately recognise that sleeping inside a building is safer than sleeping outside. The other choice is whether to barricade that building. Buildings can only be barricaded using barricade items collected by the playable character. Barricade items are in limited supply and can only be used once. Sleeping in a barricaded building means the playable character has no risk of dying in their sleep. As most films and fictional books involving zombies suggest, a barricade can be broken by zombies. Therefore, the playable character sleeping in a barricaded building should really have a small risk of death. It is good game design to minimise the number of randomly occurring deaths, making the user feel that *they* could have done something differently and *they* were responsible for the death of the playable character. Consequently, sleeping in a barricaded building has no risk of death.

The second reason was that since the game is designed, in part to be an educational game it could not sensibly teach people that they could sleep anywhere.

2.2.7 Progressive learning

As well as enemies there are also computer controlled characters in the game called survivors. These are friendly towards the playable character and will not attack them. The playable character can "talk" to these survivors. Most survivors display text containing general survival hints that can be used by the playable character such as remembering to drink to stay healthy. A full list of what survivors can "say" is given in the "Detailed Design" document on the project website [1].

It was originally designed that some survivors would have special text to display the first time they are talked to. This special text would explain a survival tactic used in the game. For example, there will be a survivor who explains to the playable character, and therefore the user, how to sleep and what benefit sleeping gives them. This survivor would be placed early in the game. When the playable character then encounters and talks to that survivor the user will learn that the playable character needs to sleep in order to survive. This minimises the amount of information the user needs to remember before playing the game.

There are also survivors later in the game that explain more advanced tactics that are not needed in the first stages of it. This design decision aims to keep the user interested in the game by making them learn new skills throughout it instead of giving them all information at the start. It also may encourage the user to play the game again after completion so they can use the skills they learnt in later areas in earlier parts of the game.

The main tactics used in the game are given in the User Manual but using this method reinforces and demonstrates those tactics only when they are needed.

Sadly, due to time constraints and because it was not explicitly expressed as a requirement, the use of 'special text' was not implemented. It would be good to implement as a future development since I believe it would greatly increase the usability of the game.

2.2.8 No examine function for items

My original intention was to include an examine item function for each item. This would be accessed by the action menu of each item in the playable character's inventory. Invoking this function would have displayed a screen which informed the user what the item examined was for, for example what stat it increased. I later decided not to include this feature in the final design for two reasons. The first is that designing, implementing and testing a separate screen for each item adds a lot more work and development time to the project. The second is

involved with the cognitive dimension of secondary notation, particularly redundant recoding. Most, if not all, of what could usefully be put into an examine screen about an item would simply be a repeat of the information given in the User Manual for the game or what could be accessed via the in-game help. The only advantage of the examine function would be to give the user only the information they need when it is needed instead of requiring them to remember what every item can be used for.

It is worth noting that if more complex items for example, keys to specific buildings, had been used then the examine function would be required.

2.3 Novel features

This section lists and describes any novel features of the game that may be of interest. Most are not uncommon in modern games.

2.3.1 In-game help

There is a permanent message bar across the top of the screen stating how the user can get help containing a list of the game's controls. This help information is not the game's complete User Manual but only a summary of it. It aims to provide only the information that is of use when playing the game.

2.3.2 Levels as areas

Most games, particularly older ones, have a linear sequence of levels. There is commonly an exact sequence in which levels must be traversed. When a user completes level 1, for example, they start level 2 and cannot return to level 1 without starting a new game. Since the game I created relies on the user collecting items and using them efficiently an exact sequence of level transitions to follow would ruin the strategy element of the game. For example, a user may not need all items in a level, but realise that they may need more items than have been placed in the next level in the next level. Not allowing the user to return to the previous level would mean any uncollected items are wasted and would not allow the user to plan ahead. This may frustrate the user by not allowing them to do what they want and may cause them to stop playing the game.

2.3.3 The minimap

The minimap, which is located at the centre of the bottom of the screen, provides users with an aerial view of the entire game level. It shows the location of the playable character, all survivors, all items and all Rot. This allows the user to plan strategies about how best to

complete that level. For example, the playable character may be safe inside a barricaded building but require drink items to lower their Thirst stat. They can see a drink item on the minimap but they can also see there is a large number of Rot around the area where the drink item is. The user can then decide to look for another drink item further away from the Rot or wait until the Rot move away from the drink item.

A minimap is usually only included in strategy games where a user controls multiple units. I have briefly discussed strategy games in section 10 of the "Literature review and system design" document on page 62. I also evaluate an example strategy game in the context survey of the "Project Specification and Plan" document on page 44.

2.4 Special Algorithms

This section explains any special and complex algorithms used in the game. It describes the Artificial Intelligence (AI) given to computer controlled characters.

2.4.1 AI of survivors

In the "Literature review and system design" document there is a finite-state machine showing the AI of a survivor given on page 72. Survivors are friendly towards the player character and are not able to attack them. They can also not attack Rot, the only enemies in the game. Survivors required enough intelligence so that they did not walk through walls or other solid objects. As described in section 2.2.7 on page 17 the playable character can "talk" to survivors. This is an important method used to convey information to the user so I did not want survivors moving around the game world at random. To solve this, survivors could be forced to simply stay in one place and not move. This would have considerably simplified the required AI. I decided against this, however, as I wanted to show the different movement and behaviours of survivors compared to zombies (Rot). Apart from feeling more "natural" it would also make it immediately obvious for the user to determine friendly human characters from unfriendly enemy characters. The actual solution was to restrict survivors to only a limited area of movement as explained in the description of the finite-state machine in the "Literature review and system design" document.

While I attempted to capture part of natural human behaviour in the AI of survivors it is obvious that there are some limitations. For example, survivors will not "learn" routes and often collide with objects they have collided with before. Another limitation is the lack of diagonal movement. Sprites (images) for diagonal movement are very hard to draw accurately and because the player character cannot move diagonally I decided to force survivors to only

be able to move north, east, south and west. This makes for more "jerky", less smooth movement but removes the need for extra sprites and it would be bad design to make the user feel they cannot do something others are able to. The incorporation of diagonal movement may be considered for an improvement to the game in the future.

A further possible improvement would be to include and monitor basic survival stats for each survivor as well as the playable character. These stats may be limited to only Hunger and Thirst as documented in "Literature review and system design" document on page 66. There would then be a separate state for each high stat. When the survivor entered one of these states they would ignore the restriction on movement and move to try and find an item that lowered the high stat. If they found that item, the high stat would automatically be lowered and the survivor would return to their start location. There are problems with this idea.

- It would be very difficult and time consuming to implement successfully.
- Depending on how it was implemented, it may lower game performance as there are more stats to monitor and more state transitions.
- For the most part the behaviour would be hidden from the user as they will likely be concentrating on other areas for example, approaching Rot.
- Survivors would effectively "steal" items from the playable character since the items survivors "pick up" are destroyed. This will be more of a problem when the game is played for long periods of time. This concept may be more realistic of human behaviour but could cause the user to think of survivors more as competition than helpers which is not what was intended.

2.4.2 AI of Rot

In the "Literature review and system design" document there is a finite-state machine showing the AI of a Rot given on page 71. Rot act as enemies and attack the playable character whenever possible. However, they do need some intelligence so they can not move through walls or other solid objects. Since I have described how Rot move in each state and how they enter and exit those states in that section of the "Literature review and system design" document I will not describe them here. In summary they should behave approximately as the user would expect as often as possible, for example, Rot should act "naturally" and not move directly to the position of the playable character unless the playable character is in the Rot's line of sight.

This said, Rot are not normal humans, they are zombies and possess limited intelligence. They will not "learn" that if they repeatedly collide with a solid object they should find a

different route. They will change direction whenever they collide but they can and often will return to the same route again causing a further collision with the same object. This had the advantage that it made the AI simpler but also the disadvantage that, without a strict plan, it was difficult to determine if the AI was actually working correctly. For example, if a computer-controlled enemy hit a wall, moved away and shortly after moved back the way it had come to hit the same wall again it could easily be thought that there was a problem when this is actually the correct behaviour. This behaviour is perhaps better described by "artificial dumbness" than "artificial intelligence".

Another problem that I had was diagonal movement. An example of this is in the Chasing state when a Rot must move to the playable characters position providing they can see the playable character. The playable character will, at some point, be at a diagonal to the chasing Rot. This would therefore require the Rot to move diagonally but since sprites for diagonal movement are very hard to draw accurately and the player character cannot move diagonally I decided to force Rot to only be able to move north, east, south and west. This makes for more "jerky" less smooth movement but removes the need for extra sprites and it would be bad design to make the user feel they cannot do something others are able to. This is less of a factor than for the human survivors as they would be expected to move more "naturally". Support for diagonal movement would be a possible improvement of the game.

One feature that was only added later in the implementation of the project was the shading used when Rot change state. When a Rot performs the transition to the Chasing state they are shaded red to indicate this. They remain shaded red until they exit the Chasing state. This makes it clear to the user what danger the Rot poses to them. Another advantage is that it made the state-based AI of Rot much easier to test as I could tell exactly which state the Rot was in at all times. Coloured shading was used for changes in state in the game I evaluated in "Games featuring intelligent enemies" documented in the context survey of the "Project Specification and Plan" document on page 46. I originally discounted using coloured shading in the game developed as I thought it would make the game far less 'real' and cartoon like.

I originally intended for a different way for Rot to transition from the Chasing state to the Searching state. In the Chasing state a Rot moves towards the playable character. In the Searching state a Rot "knows" the playable character is nearby but cannot see them. The new state transition would be "Playable character out of sight or other survivor close by". This would make Rot immediately start chasing the computer-controlled survivor instead of the playable character. This would make the threat Rot pose and need to escape the town more obvious as Rot can be seen to want to kill all humans and not just the playable character. In

addition to being more realistic, it gives another way for the playable character to escape chasing Rot. This would be achieved by "leading" the chasing Rot into the path of survivors. This design was dropped from the final finite-state machine because of the partially educational nature of the game. It would be wrong to show users that they should condemn other humans to death in order to save themselves. Another reason is that it would have made the AI needed for survivors more complex. There would need to be ways in which the survivor could escape a chasing Rot.

2.5 Odd implementation decisions

This section explains any features of the game that were implemented in unusual ways.

2.5.1 Walls

To implement in-game walls I first obtained a sprite (image) for each section of the wall, for example, a sprite for corners and a sprite for vertical segments. Walls needed to be solid objects so the playable character cannot move through them but background sprites could not be made solid in Game Maker. In order to make walls solid I had to make every wall sprite an object. This led to a large amount of objects for the game to keep track of.

While testing the game I found it was possible for the playable character to become "stuck" on an edge of some of the walls. I did not think making the wall sprites more rounded would solve the problem so I decided on another solution. I would create one sprite which was a black square. I would then use this sprite to make a single wall object. This object would be solid but also invisible within the game. I deleted all of the old wall objects from the game and replaced them with the original non-solid wall sprites. I then placed the invisible wall object over the top of those sprites. When the game was played the playable character would collide with solid but invisible wall object and not pass through the wall sprite. The game would now have only one wall object to keep track of. I implemented and tested this approach but found many problems:

- The playable character could still become "stuck" on walls.
- To enable the single invisible wall object to work its sprite had to be larger than each original wall sprite which meant there were areas of the game that looked unblocked but the playable character could not move there.
- Creating buildings and connected sections of walls was harder because the wall sprites could not overlap.

I therefore reverted back to the original version of the game. Unable to think of a way to avoid the playable character getting "stuck" I implemented a method which is likely to free a

"stuck" character so play can continue. When the **Home** key on the keyboard is pressed the playable character will be move to a nearby position on the screen. This solution works in most cases and does not drastically affect the overall game.

By resizing the sprites of walls and the playable character, the problem of the playable character becoming "stuck" on walls seems to have been solved. The playable character getting "stuck" occurs almost randomly and so is very hard to test. Feedback from game testers indicates this problem occurs much less frequently than in previous versions of the game. I will keep the solution described in the above paragraph until I am sure the problem has been eliminated.

2.5.2 Invisible zones

In order to detect whether the playable character was inside a building I placed transparent, non-solid objects inside each building. At each step in the game (there are 30 steps every second) the location of the playable character is tested to determine if it is in the same position as any of the non-solid objects. This gives the immediate problem of a possible decrease in performance and "slow-down" in the game as it carries out this test almost continuously. I could avoid this by limiting the test to be performed only once every second, for example. However, because this is only a simple test with minimal complexity it is unlikely to cause a problem. In practice this assumption proves correct but that may just be due to the machines I have tested the game on.

Depending on the location of the playable character different text is displayed at the top of the game screen. If the playable character is in the same position as any of the non-solid objects inside buildings, this text displays "Unbarricaded". Whereas, if they are not the text displays "Outside". The location of this text never changes so provides the user with a consistent, always visible view of the current system state. Knowledge of the system state is crucial when allowing the playable character to sleep as it will inform the user of the risk of the playable character dieing. Associated with this text is the "Death risk" text. This text displays the associated risk, "High", "Low" or "None", with allowing the playable character to sleep in that system state or location.

2.5.3 Picking up items

As originally intended, the implementation of items uses solid objects to represent each item. For each item, I used the collision event between that item object and the playable character to determine when items are "picked-up". When an item is picked-up it is removed from the

level and stored in the playable character's inventory. This method worked correctly as long as there was an empty space in the inventory. If there was not an empty space the item would not be picked-up but remain in the level. Because a collision event was used and since every item was solid items became impassable to the playable character once their inventory was full. I thought this was not too much of a problem but the feedback from releasing the prototype version of the game to testers made it clear that it was.

Making the item objects non-solid did not solve the problem and only when the collision event was removed were the item objects always passable. I therefore, had to use some other method of picking up items. I considered just clicking with the mouse on items to pick them up but this would require a limitation on the items that could be picked-up to stop the user simply clicking on every item in the game view independent of where the playable character was. This would require the user to understand that limitation. Since a main design decision was to keep the game easy to use and understand the idea of limiting the items that could be picked-up was not used. I also considered a separate control to pick-up an item but this would have the same learn-ability problem.

I found that creating a script (a small program) that ran at the end of every step in the game solved the problem keeping the simplicity of colliding with items to pick them up. The script works by first assuming there is a collision and immediately exiting if that collision returned nothing, meaning there was no collision with an item. It will also exit immediately if the playable characters inventory is full. If however, there is an empty space in the inventory and an item has been collided with that item is added to the inventory. The item is removed from the level. The major drawback to using this script is inefficiency in that the script needs to be run at the end of every step. There are 30 steps every second so if the script itself was more complex it may lead to performance decreasing and "slow-down" in the game. In practice this "slow-down" does not occur but this may only be true on machines which the game has been tested.

2.5.4 Error Handling

Throughout the game it is possible that errors will occur. These maybe caused by differing hardware on which the game is executed or are unexpected software failures. The default used by the Game Maker development environment is to display all error messages when they occur.

It is desirable, in terms of software usability, not to give the user irrelevant information. Since the user cannot and should not attempt to correct errors themselves, displaying information about errors to them is useless. The method of hiding error messages from the user, provided by Game Maker, is to record them in a log file. This would allow the user to continue playing after the error had occurred. This is not safe and may cause the user to believe the game has not been developed well resulting in them never playing the game. This can be solved by instructing Game Maker to automatically exit the game as soon as an error occurs. This causes the game to immediately and unexpectedly exit should an error occur. There is no warning or explanation given to the user of what happened.

The solution I used was to record the information about an error in a log file and display a general error message. This general error message states only that an error has occurred and to please report it to the developer attaching the log file. The message does **not** contain any information about the error. The message contains only a single button which the user must press to exit the game. Users cannot continue a game which caused an error. This was intended to improve usability. In order to implement this it meant a test whether an error had occurred had to be performed every game step. There are 30 steps every second. I am not sure how the default method of error handling is implemented in Game Maker so the solution chosen may be more inefficient than the default. The test is only a single if statement so I do not believe it will cause performance problems. Testing has shown this belief to be true. I consider the sacrifice of efficiency for usability acceptable.

2.6 Other important decisions in the project

This section lists parts of the project that should be described that do not fall into any of the above categories.

2.6.1 The use of Game Maker

I decided to implement the game using the development software, Game Maker. This was described in the "Project Specification and Plan" document on page 53. An important point about how Game Maker works is the inheritance structure. This was omitted from the description because of the page limit applied to the "Project Specification and Plan" document. In Game Maker every entity, for example, the playable character, is represented by an object. Game Maker uses a single inheritance mechanism on the objects it creates. This means each object can only have a single parent object. Many objects can share the same parent object. This allows the common functions of an object or group of objects to be abstracted away and placed in a single parent object. The events specified in the parent object

are automatically copied to each child object of that parent, reducing the amount of repeated code.

An example of inheritance in the game created involves the playable character object. There are actually two playable character objects that, because of the abstraction provided by inheritance can be thought of as one. One object represents the character without a spear, which will lose health if they collide with an enemy, and the other represents the character with a spear, which does not lose health on collisions but kills the enemy. Both of these objects share the same functions. These shared functions are implemented only in the parent playable character object and can be performed by both child objects. In addition I only needed to test for the parent playable character object in if statements and the test would automatically be applied to each child object.

I have included the Game Maker source file of the game on the project website [1]. I have also included some of the scripts (small programs) used as text files. These and all other scripts are stored in the Game Maker source file.

In this section of the report I will evaluate the use of Game Maker as a development environment stating ways it helped and ways it hindered development of the game. Note that these are only concerned with version 7.0 of Game Maker and may not apply to later versions.

Ways it helped development

- A large advantage of using Game Maker was its community. This meant I could use pieces of code written by others and published on the Game Maker website in the game I created. Of course, I had to adhere to the copyright issues in using any such work but being able to reuse pre-written and pre-tested sections of code helped development greatly. Also tutorials, both about the use of Game Maker and about game design helped.
- An obvious advantage was the reduced implementation time required because of the use of inheritance as described above. I have not seen other free game development environments use inheritance. Also scripts, being reusable sections of code, reduced implementation time considerably.
- Game Maker handles the saving and loading of games at a high level. To perform these operations requires only one function call provided a suitable file name to save to or load from is given. This made it very easy to implement the save and load game features required and allow the user to save the current game and return to a previously saved

game. I originally thought I would need to override the default saving and loading functions in order to save all game data for example, collected items. With testing I found the provided functions did all I required, reducing development time significantly.

Ways it hindered development

- I originally designed for all menus and game screens involving multiple buttons to be controlled by both the keyboard and mouse. Unfortunately, I found this extremely difficult to implement in Game Maker. It was much easier and less time consuming to use one control mechanism and not both.
- The help text provided by Game Maker is complete in that it covers all standard functions and variable names that could be used within the game. The problem is that some of the descriptions of what functions do are very limited. Many standard functions do similar but slightly different things. For example, the functions `place_free()` and `place_empty()` do very similar things according to the help description and it is difficult to determine how they differ. In some cases the use of one may not achieve the desired output where the other does. This became a problem when implementing the code to place a fire in an empty location around the playable character when the Wood item is used.
- The one disadvantage with inheritance is that it does not appear to be able to be used in collision checking with a large number of child objects of different shapes. For example, it would have been helpful if all wall objects could have been grouped under a single wall parent object so collision with any wall would stop the playable character. When initially testing this I found it did not work correctly so I had to implement an event for collisions with each individual type of wall where necessary. While implementing other features of the program I found that it was possible to do this. Whether this was or was not possible was not clear from the start or in any documentation read.

2.6.2 Involving separate game testers

Throughout the project I distributed versions of the game to game testers, described as a type of user in the "Project Specification and Plan" document on page 43. There were only a small number of game testers. They would play the game and provide feedback making suggestions as to how it could be improved, things they did not like about the game and what, if any, errors had occurred. To help them do this a questionnaire was written and given to each game tester to complete. The results were taken into consideration when continuing development of the game. This form of testing was not designed to replace software testing which was necessary to ensure the game met its objectives and requirements. A summary of the feedback from game testers is shown on page 88 as an appendix.

As can be seen in the "Project Specification and Plan" document I initially planned for only two such tests. The first test would be performed after the early prototype version of the game was implemented whereas; the second would be performed on the almost completed final version.

Due to a loss of time because of university coursework, identified as a risk in the "Risk Analysis" on page 79, the release of the early prototype version of the game to game testers was delayed until after Christmas 2006. I had not had time to implement one part of the prototype I had planned. Unfortunately, this was a key part of the game so I decided that releasing the incomplete prototype for evaluation was a waste of time.

The release of the almost completed final version was delayed a week. The release was delayed as I wanted the saving and loading system available. In my opinion, saving and loading the game is relatively complex so I wanted to see how the game testers reacted to it. The majority of feedback from this version of the game would be considered for future development due to the time constraints of the project.

2.6.3 The use of UML in game design

To relate the design of the game to Computer Science I used the technique of including UML (Unified Modelling Language) diagrams. As explained in reference [5] the use of UML in game design is widely debated. In this section I will indicate any benefits and problems that UML gave. I will also state whether or not I recommend the use of UML in game design or not.

The main benefit of UML diagrams is that they are a good communication tool. They help the clients, who want the game to be created, understand in-game features and how they relate to each other. Part of the game may seem obvious to the game's designer who has had time to think about the game but may be difficult for a person not involved in the design to understand. Diagrams also become important in development of a game that involved a design team of more than one person. What one designer may understand another may not and a diagram is a useful way to convey ideas.

Diagrams are abstract dealing with only one part of the system. This makes diagrams clear and avoids unnecessary details but also causes problems. It is difficult to split a game into individual components that are not affected by any other part of the game. I would imagine that in modern games designs can become very complex and any diagrams would be too large

to be much use. An example of this in the game I developed is the design of items and stats as given in the "Literature review and system design" document on page 57. I intended to draw a single diagram of how each item was used and their effects. I realised however that a single diagram involving each item would be too large. Consequently, only the use of the more complex Wood item was shown in diagram form. The use of other items and their effects was only described in words.

Another difficulty, especially related to sequence diagrams and state transition diagrams is that there must be some pre-defined sequence in which actions are performed. In games this sequence is initially unknown. There are several different orders in which users can use in-game items for example. To consider any diagrams requiring a sequence to be useless in game design would be incorrect. Most games have a high-level sequence of game screens and level transitions. A flow control diagram showing this high-level sequence for the game created is given in the "Flow Control Diagram" document on the project website [1].

A lot of early games also had more low-level sequences to levels. As an example consider the PC game DOOM [6]. An example sequence for a level might be: Enter room, shoot enemies, collect yellow key, open yellow door. However, a sequence diagram could only rarely be created for the order that user must shoot the enemies in a room.

Particularly in the game I designed, I found some diagrams could be quite misleading without further information. For the 'Fear effects' I included a use-case diagram for how the playable character interacts with 'Fear effects' and the fear stat. Using the standard interpretation of this type of UML diagram it is clear that the playable character *uses* fear. That does not really make sense and could be interpreted in different ways. Another example of confusing diagrams is whether to keep the playable character and their inventory as a single entity or make them into two separate entities. Since the inventory is an attribute of the playable character and items within it can only be used by the playable character it may be natural to represent the playable character and their inventory as a single entity. However, items, when collected, are stored in the playable character's inventory. It does not make sense to describe collected items as being stored in the playable character.

3. Evaluation and critical appraisal

This section will give an analysis of the project and created game. I will evaluate the outcome of the project in relation to the main objectives identified in the "Project Specification and Plan" document. I will evaluate the game with respect to its objectives and state which

objectives have or have not been met. I will briefly compare the game created to some of the work done by others.

3.1 Evaluation with respect to the main project objectives

The main project objectives, given in italics, are taken from the main objectives section of the "Project Specification and Plan" document on page 42.

3.1.1 Computer game development as a branch of Computer Science

Research and explain the emergence of computer game design and implementation as a branch of Computer Science. Explain how this relates to the game that will be created giving terms of reference that relate its development to Computer Science.

I achieved this objective by reviewing several books and websites based on game design and implementation. It is quite surprising how many books there are on the subject of developing computer games and a lot of the ones reviewed use terms of reference that relate to Computer Science. For example, many books explaining how to implement artificial intelligence in games cover the A* algorithm which is used in the artificial intelligence branch of Computer Science. The reviews performed are documented in the "Literature review and system design" document on page 59. The reviews were concluded with a brief paragraph explaining how they relate to the game developed. Some books and websites gave very abstract and high-level ideas that could not easily be applied to a single game. I have not read all of the books reviewed so the reviews may not be a complete account of their content.

3.1.2 Applying Software Engineering techniques

Show how Software Engineering techniques can be applied to game creation.

This was achieved both with the review of books mentioned earlier and the use of UML (Unified Modelling Language) when designing the game. The game design is given in the "Literature review and system design" document on page 57. This document also includes a risk analysis and testing plan used in Software Engineering projects. It does not, however, include a fall-back plan. This is because a fall-back plan is not really necessary for a computer game. Reference [10] explains how game design can be and is used to teach Software Engineering.

3.1.3 Conclusion on the use of Software Engineering techniques

Give a conclusion stating whether or not the use of Software Engineering techniques was helpful in terms of game design.

I have stated my opinions on the use of UML in section 2.6.3 on page 28. However, these are only my opinions and reference [5] shows that the use of UML is still debated.

3.1.4 Originality

The game created must be original in some way in terms of gameplay, style of graphics or concept for example.

I believe the created game to be original. Through a context survey documented in the "Project Specification and Plan" I have shown that 'survival' is a relatively unexplored genre in computer games. 'Survival-horror' however, is not. I have discussed the reasons for and problems with using zombies for enemies in the game in section 2.1.2 on page 11. It was hoped that the involvement of zombies in the game added to its originality in the different way that zombies are used. Whether this is true or if zombies remove all sense of originality from the game can only be decided by the individual user of the game. I have highlighted any other original features throughout this report.

The game also uses techniques demonstrated in other games. For example, nearly all RPGs (Role Playing Games) use an inventory system allowing the collection and storage of items. Most items in RPGs are complex or powerful weapons and armour. By restricting the game created to simple items for example, food and drink, I aimed to re-enforce the underlying theme of survival. Hopefully, this makes the game slightly more original.

The need of the playable character to sleep, the use of Fear, 'Fear effects' and the lack of weapons are also quite original and uncommon features.

3.1.3 Required hardware

The game must not require any special or high-quality hardware to run.

Unfortunately, the game software could not be tested on a wide variety of systems so I cannot guarantee this objective has been met. The game does work as expected on all systems on which testing was performed. The minimum requirements need to install and run the game are listed on the project website [1].

3.2 Evaluation of the created game with respect to its objectives

The game objectives, given in italics, are taken from the "Project Specification and Plan" document on page 48.

3.2.1 Allowing a user to control and move a character

Allow a user to control a character on the screen and move about. The character should not be able to move through any "solid" objects within the game.

This has been met. The user controls the playable character and can move them about the screen via key presses on the keyboard. The playable character cannot move through walls, or anything else users would not expect them to be able to move through. It is worth noting here that the playable character often gets "stuck" on walls and is unable to move. In the first release of the game to the game testers, this problem alone was enough to stop many of them playing the game long enough to give useful feedback. The playable character getting "stuck" on walls was a re-occurring problem throughout the project as it happened randomly and was very difficult to test for. Judging from the feedback from game testers following the release of the almost complete game, I think I solved this problem. If, however, it does occur again and has not been solved it is a major disadvantage for the playability of the game.

3.2.2 Computer-controlled characters and their intelligence

Computer-controlled characters and enemies must move with some degree of intelligence and must not be able to move through "solid" objects.

This has been achieved as it is stated. A further description of the artificial intelligence of both types of computer-controlled character is given in section 2.4 on page 19.

3.2.3 In-game items

The player must be able to collect, store and use in-game items.

This has been achieved by giving the playable character an inventory. When the playable character moves over an in-game item it is collected. That item is removed from the current level of the game and placed in the playable character's inventory. Any item in the inventory can be used via action menus as described in the "Literature review and system design" document on page 69. The inventory was based on an example created by someone else as credited in the "Acknowledgements" appendix on page 108.

3.2.4 Education of survival techniques and complexity

*The game must allow the player to learn basic survival concepts without being overly complicated. It must make clear that these are only guidelines to help survival and the game does **not** provide a complete or detailed simulation.*

The complexity (in the non-technical sense) and understand-ability of the game can only be determined by the individual user. In an effort to make the survival concepts less complex I have removed certain features. For example, in reality if a human became too hot for long periods they may die of hyperthermia. In the game survival requires the user to keep the Warmth stat from becoming zero. Most users will therefore keep the stat as high as possible at all times. Introducing hyperthermia if the Warmth stat is too high would increase the educational value of the game at the sacrifice of playability. I consider playability to be more important. This is re-enforced by the statement "The game must educate the player ... [but] this should not heavily impact the entertainment value of the game" in the Game Definition of the "Project Specification and Plan" document on page 47.

At certain points in the game there is a large amount of information on the screen at one time. This may detract from the playability of the game by increasing complexity. Software development and Human-Computer Interaction has shown that is not user-friendly to have more than 7 ± 2 things on screen at one time. Disregarding in-game objects like enemies and items the amount of information shown is probably small enough to avoid confusion but if more was added then this would have to be taken into account.

In answer to the second part of the objective I have added a message to the main menu of the game that clearly states it does not provide a complete or detailed simulation of survival.

3.2.5 Saving and loading a game

There should be a way of saving game and returning to an already started game.

This was achieved in a sense. A way of saving a game and loading a previous game is given. The problem is that it is not very user-friendly. When the user wishes to save the game they must manually type in the name of the file to save to. If this file exists a warning is displayed. Whether this message is displayed is completely dependent on whether the user can remember or guess the names of save games they have already made. A secondary helper device could be used to store a list of save games but I would have preferred if this list could have been integrated into the game. Also a short description and/or screenshot of each saved

game may help the user determine if they do or do not want to overwrite a particular file. However, due to time constraints this was left as a desirable feature for future development.

Loading a previous game works in a similar way. Here the user is required to remember the exact name of the save game file and to know that it exists. This has similar problems as mentioned above and would definitely benefit from further development. Manually typing the name is neither robust nor reliable as users often make typing mistakes and spelling errors so a method where the user simply clicks on the name of the game they wish to load is desirable.

3.2.6 Endings

There should only be 2 different endings: Win and lose. This implies that the player can lose the game.

This objective has been met. If the playable character reaches the cable car then they win the game and the game complete screen is shown. If the playable character loses all their lives they lose the game and the game over screen is displayed.

3.2.7 Story based upon survival

The games story should be based around survival of some event.

This has been achieved in an unusual way. This objective may be thought to mean the survival in the aftermath of some natural event like a hurricane or volcanic eruption. While these may produce interesting concepts for games I decided a game where the playable character needed to survive an outbreak of zombies to be more entertaining in terms of gameplay.

In section 2.1.2 on page 11 I gave another reason for choosing an outbreak of zombies over a natural event. This was that it re-enforced that the survival concepts shown in the game are only general rules and not completely accurate.

3.2.8 Involvement of strong language

The game must not involve strong language.

This has been achieved and will hopefully encourage younger players to the game.

3.2.9 Involvement of violence and gore

Any depiction of violence or gore must be stylized.

This has been achieved and will hopefully encourage younger players to the game.

3.3 How my work compares to that done by others

This section describes any interesting differences to work done by others. These comparisons will mainly relate to the game created as part of the project and other games.

3.3.1 Death

In most games the character controlled by the user dies when they are killed by enemies. This gives the user some idea that the playable character might be killed during a battle. For example, if a user sees enemies swarming around the playable character they get a mental warning that the character may die. If the playable character dies too often the game is lost. Clearly, death is to be avoided. It is important to give an indication of death so as to avoid confusing the user and to make the user feel they should act quickly to save the character.

In Hunger, the game I created, this mental warning is harder to achieve. In most other games the playable character dies only when attacked by enemies. In Hunger the playable character can die simply because a stat is low or high. For example, if the Thirst stat is at its maximum value the playable character will start to lose health and eventually die when their health becomes zero. It therefore may not be obvious to the user the danger of being killed. In this way death can seem sudden as reported in feedback from the prototype version of the game. To try and solve this problem I implemented a near death warning system. This will flash a red screen and play a sound at regular intervals when the health of the playable character is low.

3.3.2 Digital Rights Management (DRM)

DRM is often used by publishers or copyright owners to control access to and usage of digital data or hardware [7]. DRM is used in many commercial computer games as a digital lock which stops people from running those games without a proper license [8]. This license is often a textual key that must be entered before the game can be run. There is a lot of debate as to whether DRM is needed. One example of DRM that is commonly overlooked is the restriction manufacturers place on games so they can only be run on one particular platform [9]. For example, a company may limit a game to only run on an Xbox 360 or PC. The game I created is only available to run on PCs using particular versions of Microsoft Windows

operating system. This is only because the Game Maker development environment enforces that restriction on all games created with it. Apart from this the game has no DRM. This is largely because the game involves the work of several different people. Agreeing on a particular form of DRM would be difficult. Also for a game of its size and the fact that it was not designed as a marketable game reduces the need for suitable DRM.

3.3.3 Length

Most RPGs (Role Playing Games), the genre that is closest to the game created, are very long. They may be divided into levels or provide a single large world for the playable character to explore. They rely on a user creating a character, sometimes through a detailed and time-consuming process, starting at a base experience level and then raising that level as they play the game. This is normally because of the length of the game. If a game takes a long time to complete it is important for the user to feel that the character *belongs* to them and that they are helping that character become more skilful every time they play the game.

The created game skips the character creation process entirely by offering only a single playable character that cannot be customised. This could be seen as an advantage as the game can be started quicker or a disadvantage because users may feel alienated from the character thus, meaning they do not want to save the playable character and help them escape the town. If users do not feel involved they are unlikely to play the entire game.

The game length could be ignored as just an implication of the restricted development time of the project but actually it was a design decision. There are not many games that can be completed in such a sort time. Again this can be seen as an advantage or a disadvantage depending on the user. Personally, I like games that can be completed quickly, but not too quickly. Other people like games taking weeks to complete.

4. Conclusion

In this section I will summarise the work of the project and show any strengths and weaknesses in it. I will also describe future directions in which the project and game could be taken.

4.1 Summary of work done

The problem, described in the introduction of this report, was solved as described in section 1.2 on page 6. The following is a list of the work done in the project.

- The description of the problem the project was intended to solve was documented. A way plan for achieving the solution was formed.
- A limited survey of other work in some popular game genres was performed to give ideas for the game to be created and ensure a very similar game had not already been made.
- Game design and implementation were linked to Computer Science concepts and techniques by reviewing a number of books and websites.
- A game was designed using Software Engineering techniques for example, UML diagrams, and was related to the above reviews.
- The game was implemented using Game Maker by first developing a prototype version and increasing that into the final version for submission.
- Software behaviour was tested to ensure the game performed as expected and met its objectives.
- The prototype version and almost complete version of the game were distributed to game testers, along with questionnaires. This allowed them to provide feedback on their opinions of the game and how they thought the game was progressing.
- A summary of feedback from testers and other tests performed was created and is given on page 85 as an appendix.
- Documentation was produced about all aspects of the project. Much of this occurred in parallel with other stages of the project.
- Two short presentations about the project were given and a demonstration of the finished game was given to the two project supervisors.

All documentation and source file for the game can be downloaded from the project website [1].

4.2 Strengths and weaknesses of the game

I have already described many points I feel have and have not worked well. The following have come to light after further use of the game or have been indicated by game testers.

4.2.1 Strengths

The entire process can be considered a major achievement or strength of the project. I have never created a game of my own design before so the majority of this project was a new experience. This is especially important because in the future I wish to work in the games industry. Having completed a Software Development Project in previous years [13], I assumed all computer games were designed in the same way as other software. Having researched this I have realised that not all games are designed this way and Software Engineering techniques are not always beneficial.

Like most games, many strengths and major achievements in terms of programming are hidden from the user. One such strength is the clipping mechanism used when dropping items or placing fires with the Wood item. This ensures fires and items cannot be placed underneath solid objects or outside rooms. It also ensures the playable character cannot "throw" dropped items over walls for example. I will not go into detail about how this works but essentially it checks each direction around the playable character and, if that direction is free of obstacles, provided it is inside the room, it places the item or fire there. In the case of placing fires they must always be placed in front of the playable character if possible. This provided another challenge but was overcome in a similar way.

4.2.2 Weaknesses

There is no immediate way to exit the game software during play. In order to exit the game the user must press Esc on the keyboard to open the in-game menu, click on the quit button, answer yes to a warning about a loss of unsaved data and, finally, click on the exit button when the main menu is displayed. This is a relatively long process so I considered implementing a feature to exit the game software immediately after a specific key was pressed. However, I think this would be a bad idea as it introduces the problem of users pressing the key accidentally and permanently losing their progress in the game.

The movement of the playable character is a minor problem. The user can move the playable character around with the directional keys on the keyboard. Some of the keys "override" the movement provided by another key but others do not. For example, if the user has the right key held down and presses the left key the playable character will continue moving right. If the user has the left key held down and presses the right key the playable character will immediately change direction to move right. Other games I have tested that have also been made with Game Maker have this problem suggesting it is inherited with Game Maker. However, I have seen games created with Game Maker that do not appear affected by this problem.

4.3 Future directions

In terms of the analysis of whether Software Engineering techniques are useful in game development I could evaluate game design documents used for commercial games. Unfortunately, it is very unlikely that a company that develops commercial games will make their design documents publicly available. This is likely because of competition from other game development companies and because many game designs are different, sometimes very

different, to the games they are developed into. There is one partial game design document which I have found. It is for the game DOOM [6] and is available from reference [11]. It was written in 1992 so is unlikely to be much use to competitor companies anymore.

In addition to the desirable functionality listed in the "Literature review and system design" document on page 76 I could also implement more levels in the game. This would be required if I wanted to make the game publicly available. With only three levels I think it is just too short and easy to complete. The majority of feedback from game testers also indicates the quickness in which the game can be completed is a problem.

To attempt to solve this problem and increase the length of the game I could simply add more levels, ordering them based on their difficulty to complete. Another solution would be to add episodes to the game. Each episode would be a similar length to the game and take place in different environments with similar but slightly different stories. For example, I could add another three levels to the game that were set in a desert environment. I could then decrease the amount of drink items available and increase the rate the playable character's Warmth stat increased. Adding new challenges to a game in this way is a good way to increase replayability [14]. These extra episodes could be downloaded later after the game has been released and accessed in a similar way to starting a new game. Episodic-content for games is highly debated but is proving to be popular in modern commercial games as explained in reference [12].

Other possible future developments would be:

- Correcting any errors that were found towards the end of the project which I did not have time to correct before the deadline.
- Expanding the saving and loading system as mentioned in section 3.2.5 on page 33.
- Implementing support for the deletion of saved games within the game software. Currently, the user is forced to exit the game and manually delete the file representing the save game from a directory.
- Adding a pop-up description of an item in the inventory when the mouse cursor hovers over it. The description would be as short as possible, containing only the item's name and effect. This would mean the user does not have to remember what each item looks like or what it is used for.

References

- [1] Title: Hunger - Project Website
Website URL: <http://uk.geocities.com/tdc1@btinternet.com/Hunger/Hunger.html>
- [2] Title: Wikipedia - Video game genres
Website URL: http://en.wikipedia.org/wiki/Computer_and_video_game_genres
- [3] Title: Game Maker by Mark Overmars
Website URL: <http://www.gamemaker.nl/>
- [4] Title: YoYo games
Website URL: <http://www.yoyogames.com/>
- [5] Title: GameDev.net - UML in game design?
Website URL:
http://www.gamedev.net/community/forums/topic.asp?topic_id=357428
- [6] Title: Wikipedia - Doom
Website URL: <http://en.wikipedia.org/wiki/Doom>
- [7] Title: Wikipedia - Digital Rights Management
Website URL: http://en.wikipedia.org/wiki/Digital_Rights_Management
- [8] Title: Valve, Steam and DRM
Website URL: <http://www.kuro5hin.org/story/2003/9/13/03945/7308>
- [9] Title: DRM in games
Website URL: <http://www.ploob.com/2006/07/07/drm-in-games>
- [10] Title: Teaching software engineering through game design
Website URL: <http://web.cs.wpi.edu/~claypool/papers/game-se/paper.pdf>
- [11] Title: 5 years of DOOM - The DOOM bible
Website URL: <http://5years.doomworld.com/doombible/>
- [12] Title: Why bother with episodic games?
Website URL: http://www.gamasutra.com/features/20070103/sanchez_01.shtml
- [13] Title: Xchange - A 2005-2006 software team project for the University of St. Andrews by team 1.
Website URL: <http://lab-cs3.dcs.st-and.ac.uk/~jh20051/Website/Xchange.html>
- [14] Title: IFWiki - Replayability
Website URL: <http://www.ifwiki.org/index.php/Replayability>
- [15] Title: Wikipedia - Game Programming
Website URL: http://en.wikipedia.org/wiki/Game_programming#Maintenance

Appendices

Project Specification and Plan

Contents

PROBLEM DEFINITION.....	42
MAIN OBJECTIVES	42
PROJECT BOUNDARIES.....	42
USERS	43
CONTEXT SURVEY	43
1. STRATEGY GAMES	44
2. ACTION GAMES.....	44
3. HORROR GAMES	45
4. GAMES AIMED AT YOUNGER PLAYERS	46
5. GAMES FEATURING INTELLIGENT ENEMIES	46
GAME DEFINITION.....	47
GAME OBJECTIVES.....	48
REQUIREMENTS SPECIFICATION	48
1. INTERFACE REQUIREMENTS	48
2. PERFORMANCE REQUIREMENTS	49
3. SECURITY REQUIREMENTS	49
4. OPERATIONAL REQUIREMENTS	50
5. DATA REQUIREMENTS	50
6. PHYSICAL ENVIRONMENT	51
7. OPERATIONAL ENVIRONMENT.....	51
8. FUNCTIONAL REQUIREMENTS	51
9. NON-FUNCTIONAL REQUIREMENTS AND CONSTRAINTS	52
IMPLEMENTATION PLAN	53
DELIVERABLES AND OTHER KEY DEADLINES.....	54
PROJECT MONITORING SHEET	54
SEMESTER 1.....	55
WEEKS AFTER EXAMS AND BEFORE SEMESTER 2.....	55
SEMESTER 2.....	55
REFERENCES	56
BACKGROUND BIBLIOGRAPHY	56

Problem Definition

"Gaming has blasted its way into the mainstream to become a multi-million dollar business, rivalling the film and record industries." - Quoted from reference [6]

The project will research and show how the expanding field of computer games design and implementation is related to Computer Science concepts and techniques. A game will be created as an example, ensuring it is clear how the game relates to this field. The development of the example game must show how Software Engineering techniques can be applied to game creation. At the end of this project an opinion on whether or not these techniques are helpful will be given.

In addition to the above there are also requirements which the developed game will meet as explained after a context survey of similar work has been performed. In general the game must be original yet fun to play and not require any special or high-quality hardware to run.

Main Objectives

The main objectives of the project are listed in order of priority.

1. Research and explain the emergence of computer game design and implementation as a branch of Computer Science. Explain how this relates to the game that will be created giving terms of reference that relate its development to Computer Science.
2. Show how Software Engineering techniques can be applied to game creation.
3. Give a conclusion stating whether or not the use of Software Engineering techniques was helpful in terms of game design.
4. The game created must be original in some way in terms of gameplay, style of graphics or concept, for example.
5. The game must not require any special or high-quality hardware to run.

Project Boundaries

This section lists what will **not** be addressed by the project.

1. Testing the developed game on many different types of computer with varying hardware and operating systems.
2. Illegal or unauthorized access to the game software.
3. External computer problems, such as viruses, preventing the game from running.

Due to time constraints it is not feasible to consider the following as part of the project. With more time they could be included.

1. Many modern game technologies for example, dynamic lighting, accurate in-game physics and animated 3D (3 dimensional) cutscenes.
2. Multi-player either co-operative or challenging.
3. Modification or map making tools allowing a user to modify the game.

Users

The people who will use the system are:

- The player
They will operate the game and control the playable character while it is being played. They will have limited knowledge of computers so the game and controls must be easy to understand and use.
- The client
They potentially represent the company who issued the problem and wish the game to be created. They are the potential distributor of the game on a non-commercial level or will adapt the game for marketing. They will distribute the game to the player or may be the player.
- Game testers
They will perform the same role as the player but will try to break or discover faults within the game. They may have more technical knowledge than the player.
- Game designer
They will be required to design, document and create the game. They will also be required to perform testing. They will be able to communicate with the game testers and, potentially, the client but not necessarily the player. Any maintenance to the project will be carried out by the game designer. Throughout the project the game designer will be able to release prototype versions of the game to game testers. Using these, and a questionnaire created by the designer, testers will give their opinions on the development of the game.

Context Survey

In this section I will evaluate other work in some popular game genres. It is sometimes difficult to split examples into these categories as they usually involve a slight mix of genres. Each example game was obtained from the reference with number given in [] after the name of the game. The references used are listed on page 56.

There are many different game genres. I consider only a few games and genres here, but there is a much more complete list of game genres, including descriptions of each, given in reference [5].

1. Strategy Games

Most strategy games involve a single user controlling multiple units, ranging from vehicles to infantry. The games are usually based around a high-level concept of war between factions, but can also be based around a small group of characters. The user must employ strategy to direct these characters to destroy the enemy team.

Example: Hovendall Tactics [1]

Good Points:

- Well made and simple main menu for game.
- Good interface to the game which always provides a way to get help.
- Being turn-based and using the mouse it is easy to control multiple characters at a time.
- Only information relative to the currently selected player is displayed on the interface to avoid clutter and avoid confusing the user.

Bad Points:

- No obvious user manual. Only available through game interface.
- User Manual long and complex without screenshots linking document to game.
- Have to click on (i) symbol for help which users might not recognise as a way to get help.

2. Action Games

Action is the broadest game genre. Any game that involves the user's quick re-actions can be considered an action game. The list of genres referred to above shows platform games as a separate genre, but I would include fast-paced platform games as action games. Platform games involve pressing keys at exactly the right time to overcome obstacles. They are often very simplistic, which is synonymous with action games.

Example: Sonic Zone [2]

Good Points:

- Easy to play without reading the user manual by using simple controls.
- Brightly coloured detailed graphics.
- Obvious objectives: To get to end of level and collect brightly coloured rings.

Bad Points:

- Heavily based on older Sonic the Hedgehog games.
- User manual does not contain pictures to link to game.
- No health so player forced to restart from beginning of level when hit by an enemy.

3. Horror Games

Horror games try to scare, horrify or shock the user in some way. This may be done by setting the game in a scary location, such as a haunted mansion, through clever use of sound or through visually disgusting enemies. Horror games often make use of elements seen in other genres, such as character development or item-collection in role-playing games.

Example: Cursed Undead [1]

Good Points:

- 3D graphics make the game look more attractive and realistic.
- Use of inventory to collect, store and use in-game items.
- Detailed game design document given with game. Contains pictures a screenshots to relate with game.
- Simple to use way of saving and loading a game by loading a game saved in one of 3 slots.
- Good interface showing all information related to player without cluttering screen.
- Can use mouse or keyboard to control playable character. It is good to have different control methods so the user can choose the one they prefer.

Bad Points:

- User manual is very short and does not contain pictures linking it to the game.
- Many scenes involving blood and gore.
- Can only be played above a set monitor resolution.
- Change in control method from keyboard to mouse when trying to load a game from "Load Game" menu. This may confuse the user.

4. Games aimed at younger players

This is not an individual genre but any game designed to appeal to younger players. They are usually simple games with unrealistic concepts. There are very few 'survival horror' games designed for young people.

Example: Ness' Christmas Journey [2]

Good points:

- It has a colourful and well written user manual. It is divided into sections corresponding to different elements of the game. For example, there is one section that shows and explains all the in-game items which the player can pick up. It is also short, readable and understandable.
- Objects within the game are brightly coloured and are not overdone with minor details.
- Simple main menu to begin the game containing only 4 buttons, including a button to get help.
- Good use of foreground and background images to make the game seem more realistic.
- Simplistic un-animated cutscene at beginning of game is like a comic book which relates well to younger gamers.

Bad points:

- Information to do with player (for example, health) is hard to see because of transparency.
- Seemingly unnecessary questions at the start of the game slow it down.

5. Games featuring intelligent enemies

This is not an individual genre as intelligent enemies are used in almost every type of game. For this survey intelligent enemies means computer-controlled enemies that cannot pass through solid objects and that react to the character controlled by the player by, for example chasing them.

Example: Pyramid Panic [3]

Good points:

- Short, understandable user manual.
- Enemies change colour using coloured shading when they are near or have detected the playable character, giving a visual warning to the player.
- Different ways to play game from the beginning requiring the user to make decisions.
- Objects within the game are well-drawn and brightly coloured.
- Enemies change starting position within the game world each time it is played adding replayability and providing the user with slightly different challenges each time a new game is begun.

Bad Points:

- User manual does not contain pictures linking it to the game.
- I could not get the way of saving a game to work.

Game Definition

I believe 'survival' to be a game genre and it is interesting that reference [5] does not include 'survival', only 'survival horror', which implies 'survival' is a relatively unexplored genre. As such, it offers a suitable framework to work within. The game will be based on survival of a particular event. There are relatively few existing example games in this genre to prejudice the style of the game to be created. Given the restricted time frame the game does not need to be developed to a level that could be published. It does, however, need to be expandable into a full game or give ideas on which future games can be based.

Whether the player-controlled character does survive the event when the game is over has to be an absolute yes or no question without requiring more than 2 different endings. The game must educate the player in the basic concepts of survival (for example, the need to eat and drink) but warn them that this is not a detailed or complete simulation of survival. This should not heavily impact the entertainment value of the game. The game should support the collection of in-game items that can then be used by the character the player controls. It must, however, be suitable for players with little knowledge of computers. To this end it must be easy to use and understand without requiring a large user manual to be read or any training to be given. Characters and any computer controlled enemies within the game must move with some limited level of intelligence, for example, they must not pass through solid walls in the game unless intended and all enemies must be able to attack the player-controlled character.

The game does not need to be able to support multiple players. As mentioned in the problem definition the game must not require any special hardware, a high-quality or a fast computer to be run. Users may not have time to play the whole game from start to finish so a way to save and load a started game should be given where the items the player has collected are retained.

The game should not involve vivid scenes of violence or gore. Any scenes like this must be stylised or kept to a minimum to encourage younger players. The game must therefore not involve strong language.

Game Objectives

The game requirements are listed in order of priority.

1. Allow a user to control a character on the screen and move about. The character should not be able to move through any "solid" objects within the game.
2. Computer-controlled characters and enemies must move with some degree of intelligence and must not be able to move through "solid" objects.
3. The player must be able to collect, store and use in-game items.
4. The game must allow the player to learn basic survival concepts without being overly complicated. It must make clear that these are only guidelines to help survival and the game does **not** provide a complete or detailed simulation of survival.
5. There should be a way of saving game and returning to an already started game.
6. There should only be 2 different endings: Win and lose. This implies that the player can lose the game.
7. The games story should be based around survival of some event.
8. The game must not involve strong language.
9. Any depiction of violence or gore must be stylized.

Requirements Specification

1. Interface Requirements

This details any requirements on how the game interacts with the user.

- 1.1 The interface must be easy to use, understandable and readable by users who do not necessarily have experience with using computers. Users can also not be given special training to use the interface.
- 1.2 The interface must be simple since the user must not be required to read a large User Manual in order to use it.

- 1.3 The interface must display a message stating that the game does not provide a detailed or complete guide to survival but only gives the basic concepts.
- 1.4 The main menu of the interface should allow the user to begin a new game, load a saved game, get help for playing the game and exit the game software.
- 1.5 The game screen itself must clearly show any data that is associated with the player for example, their number of remaining lives. This must be easily visible while the user controls the playable character.
- 1.6 The interface must display computer-controlled enemies, items and other in-game objects clearly.
- 1.7 When the player's character health reaches 0, they die. The interface must show this.
- 1.8 When the playable character reaches the end of the game the interface must show that the game has been completed.
- 1.9 The game screen may indicate a way to get help on playing the game.
- 1.10 The interface must provide a simple way to quit the game and return to the main menu mentioned above.
- 1.11 The game is controlled by the user pressing keys on a keyboard and clicking with the mouse.

2. Performance Requirements

This details performance requirements on the game for example, speed and response time.

- 2.1 The game must always be available for use while it is installed.
- 2.2 When the game software is run it should not take long for the game to load.
- 2.3 Loading a saved game should not take long.
- 2.4 When the user presses a key on the keyboard, clicks the mouse or activates a button on the screen it should cause a seemingly immediate response within the game.

3. Security Requirements

This lists any requirements that will be needed if problems occur in the project and other security concerns.

- 3.1 There may be a fall-back plan to provide a different solution to the problem or to explain features which can be left out to achieve a solution.
- 3.2 The game must not interfere with other software while it is running.

4. Operational Requirements

This specifies actions that are required to be performed by the game.

- 4.1 The game must allow the user to control an on-screen character (the playable character).
- 4.2 The game must not allow the playable character to disappear off the visible area of the screen while being controlled by the user.
- 4.3 The game should provide no more than 2 possible endings and clearly inform the user when an ending is reached.
- 4.4 The game must allow the playable character to collect and store in-game items. The user must be able to view the items they have collected.
- 4.5 The game must maintain information associated with the playable character's survival for example, thirst. The use of in-game items will effect that information. This will allow the user to learn the basic concepts of survival.
- 4.6 The game is required to provide some limited degree of intelligence for computer-controlled enemies and characters so, for example, they cannot walk through "solid" objects within the game and can identify and attack the playable character.
- 4.7 The game should provide a simple way of saving and loading previous games so they can be continued. The game may retain all items previously collected by the playable character when loaded.
- 4.8 The game must hide or stylize scenes of violence and gore.
- 4.9 The game must remove the playable character when their health becomes 0 or they die and reduce the number of remaining lives by one. The playable character is then repositioned in the game screen unless they have no more lives where the game will be over.
- 4.10 The game may be able to "Pause" or temporarily stop gameplay (ie. remove control from the player and stop all enemies) when instructed by the user. The game should then be able to restore gameplay when instructed by the user.

5. Data Requirements

This details memory requirements and the amount of data the game will handle.

- 5.1 The game will include only 1 playable character.
- 5.2 The game will contain several objects (for example, walls) and images.
- 5.3 The game will include several computer-controlled characters and enemies.
- 5.4 The game will include no more than 3 playable areas due to time constraints.

6. Physical Environment

This details the physical requirements of the game for example, required hardware.

- 6.1 The game must **not** require any special, fast or high-quality hardware in order to be played.
- 6.2 The game must run on 'standard' computers with a mouse, keyboard and monitor.

7. Operational Environment

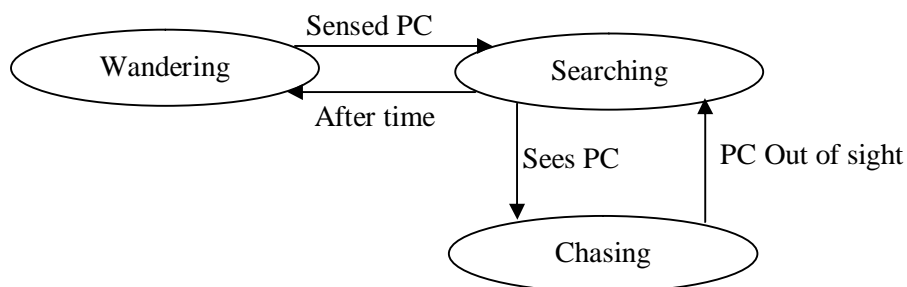
This details the non-physical requirements of the game for example, required software.

- 7.1 The game will run on any computer meeting the physical environment and using the latest version of the Microsoft Windows operating system.
- 7.2 The game may also run on different operating systems.
- 7.3 The game may **not** require any other special software to be played.

8. Functional Requirements

The game software must be able to do the following:

- 8.1 Load a menu screen allowing the user to begin a new game, load a previously saved game or exit the game. It may also provide a way to view high scores left by other users playing the game.
- 8.2 Allow the user to control the playable character on the screen without allowing the character to pass through solid objects in the game or disappear off the screen.
- 8.3 Maintain information associated with the playable character for example, thirst. This information will change while the game is being played so must be updated automatically. The information that will be maintained and displayed to the user is explained in the "Literature review and design" document.
- 8.4 Display computer controlled characters and enemies that have a limited degree of intelligence. These cannot move through solid objects and enemies should be able to identify and attack the playable character. The artificial intelligence of enemies should be based upon the following model.



PC = Playable character

The Wandering state is the default behaviour of the enemy. When the playable character gets close to an enemy they "sense" the character and change to the Searching state. Here the enemy moves in the general direction of the playable character but not necessarily towards them. When the enemy "sees" the playable character they switch to the Chasing state. Here they will move towards the playable character. If the playable character manages to evade the enemy and hides behind an obstacle where it can no longer be seen by the enemy the enemy will revert back to the Searching state and eventually the Wandering state if they do not see the playable character again in a fixed time. Variations of this model can be used for different computer controlled enemies.

- 8.5 Display in-game items that the playable character can collect. A way to store these items should be provided. The user must be able to view and use stored items.
- 8.6 Allow the user to quickly save a game to be continued later.
- 8.7 Allow the user to access the menu at any point while they are in control of the playable character.
- 8.8 Show a screen to indicate the user has completed the game and remove control of the playable character when they reach the end of the game.
- 8.9 When the playable character's health reaches 0 or they die they must be removed from the game and their number of lives must be reduced by 1. The playable character will be repositioned on the game screen unless they have no more lives where the screen indicating that the user has lost the game must be displayed and control removed from the playable character.

9. Non-Functional Requirements and Constraints

This lists other constraints on the project and software reliability.

- 9.1 The project is required to show and document the emergence of computer game design and implementation as a branch of Computer Science. This must be related to the game that will be created.

- 9.2 There is a time constraint on the project. It must be released, fully tested and documented by Tuesday the 17th of April 2007.
- 9.3 Other documents and logs need to be submitted to the client at various stages throughout the project and are listed in the "Deliverables and other key deadlines" section on page 54.
- 9.4 A meeting with the project supervisor must be held and logged each academic week of the project. These logs must be submitted to the client electronically by the end of that week.
- 9.5 The software of the project should be completed approximately a week before the deadline to allow it to be documented and tested.
- 9.6 The game should always be available for use while it is installed.
- 9.7 The game should run with a minimum amount of errors.
- 9.8 The game must be easy to use and understand by people with limited technical knowledge.

Implementation Plan

The game will be created using Game Maker (GM) which is explained in more detail at reference [4]. I chose GM because it is easy-to-use and I have used it in the past. It provides many features that are very useful for meeting the requirements listed above. For example, the embedded scripting language, GML (Game Maker Language), provides pre-written, tested and documented functions to implement artificial intelligence in games. It provides a GUI (Graphical User Interface), with drag-and-drop capabilities. It is object orientated. It works by registering objects with events and giving them actions to perform when these events occur. An example is when two objects collide and move off in separate directions. GM also provides a very easy way to define and create a game world in which objects can move around by allowing backgrounds to be placed within a game area or "room". While a game created using GM is being played the user can press 'F1' on the keyboard to automatically open a help file written by the developer. Another major advantage of Game Maker is that it has a lot of community support where I can find tutorials and ask for help if needed.

The main and only disadvantage is that GM does not provide an easy way to produce software listings or UML (Unified Modelling Language) diagrams. This is mainly because it is not reliant on a single long program file containing code. Instead the code is split between objects each containing only the functions they require.

Deliverables and other key deadlines

The project must be heavily documented. Some of these documents will only be used internally within the project (a milestone) but others will be required to be given to the client (a deliverable). This section lists those documents and any other deadlines set by the client.

All presentations are 10 minutes long, allowing 5 minutes for questions. Slides for these presentations are to be created using Microsoft PowerPoint. Each demonstration is 30 minutes long. A more detailed list of deliverables can be found in reference [7].

Type	For	Date
Deliverable	Description and Objectives document (Desc.)	8/10/06
Deliverable	Specification and Plan document (Spec.)	29/10/06
Deliverable	Literature review and system design document (Design)	13/11/06
Milestone	Detailed design document for game (Detailed design)	20/11/06
Deadline	Prototype software demonstration.	15/12/06
Deliverable	Interim Presentation and PowerPoint slides (Interim)	15/12/06
Deliverable	Revised Specification, Design and Plan documents (Revise docs.)	09/02/07
Deliverable	Complete first draft of project report (Report draft)	16/03/07
Milestone	Completed and tested software	30/03/07
Deliverable	Final project report, software and documentation including software listings (Final)	17/04/07
Deadline	Project Demonstration (Demo)	20/04/07
Deliverable	Final Presentation and PowerPoint slides (Talk)	20/04/07

I have not included the logs of supervisor meetings to be delivered to the client each week to avoid clutter.

Project Monitoring Sheet

The main tasks of the project, when they are scheduled to be completed and when they must be submitted are shown in the tables below. There are weekly meetings with my supervisor to discuss the project. I have not included these on the tables to avoid clutter. I have not scheduled much time to write the project report which is a large document. I will work on this for a part of most weeks where possible.

I will use a sort of depth-first approach to the way the game is created. For example, I will implement and test the game first to ensure basic gameplay and then create the main menu to allow access to the game around that.

Black bars indicate the time (in person-weeks) to spend on each task. A vertical grey bar indicates a deadline, deliverable or milestone. The final column in each table is used to show if the tasks are completed.

Semester 1

	2	3	4	5	6	7	8	9	10	11	12	
Desc.	█											✓
Spec		█	█	█								✓
Design				█	█	█						✓
Detailed design				█	█	█						✓
Implement prot. game							█	█	█			✓
Implement prot. menu									█			✓
Prototype testing									█			✓
Give game to testers for feedback									█			✓
Interim										█	█	✓

Weeks after exams and before semester 2

	1	2	
Revise docs.	█		✓
Create test data for final game		█	✓
Implement game			█

Semester 2

	1	2	3	4	5	6	7	8	9	
Revise docs.	█									✓
Implement game		█	█	█						✓
Complete menu				█	█					✓
Testing summary				█	█					✓
Give game to testers for feedback						█				✓
Report draft						█	█			✓
Final								█	█	✓
Talk									█	✓
Demo									█	✓

I have not scheduled for the spring vacation between the 24th of March 2007 and 8th of April 2007 as this is to be used as "catch-up" time for anything which is not complete.

References

- [1] Title: Universiteit Utrecht - Game Design
Website URL: <http://www.cs.uu.nl/docs/vakken/gds/games.html>
- [2] Title: Game Maker by Mark Overmars - Games Showcase: Standalone games
Website URL: http://www.gamemaker.nl/games_exe.html
- [3] Title: The Game Maker's Apprentice - Game development for beginners:
Companion CD
Authors: Jacob Habgood and Mark Overmars
Publisher: Apress
Publication Year: 2006
- [4] Title: Game Maker by Mark Overmars
Website URL: <http://www.gamemaker.nl>
- [5] Title: Wikipedia - Video game genres
Website URL: http://en.wikipedia.org/wiki/Computer_and_video_game_genres
- [6] Title: BBC NEWS | Technology - Gaming comes of age
Website URL: <http://news.bbc.co.uk/2/hi/technology/2583697.stm>
- [7] Title: CS4099 (major) / CS4098 (minor) Software Project
Author: James McKinna, University of St. Andrews.
Publication Year: 2006

Background Bibliography

- 1. Title: Software Engineering Lecture notes - Software Engineering Management
Author: Ishbel Duncan, University of St. Andrews.
Publication Year: 2005
- 2. Title: Xchange - A 2005-2006 software team project for the University of St. Andrews by team 1.
Website URL: <http://lab-cs3.dcs.st-and.ac.uk/~jh20051/Website/Xchange.html>

Literature review and system design

Contents

LITERATURE REVIEW	59
1. BETTER GAME CHARACTERS BY DESIGN: A PSYCHOLOGICAL APPROACH [1, 2].....	59
2. RULES OF PLAY: GAME DESIGN FUNDAMENTALS [3]	59
3. PROGRAMMING BELIEVABLE CHARACTERS FOR COMPUTER GAMES [4].....	60
4. AI GAME PROGRAMMING WISDOM [5, 6]	60
5. PROGRAMMING GAME AI BY EXAMPLE [7].....	61
6. GAME DESIGN COMPLETE [8]	61
7. THE GAME MAKER'S APPRENTICE: GAME DEVELOPMENT FOR BEGINNERS [10].....	61
8. GAME STUDIES: WHERE THE ACTION IS [11]	62
9. GENDER AND COMPUTER GAMES: EXPLORING FEMALES' DISLIKES [12].....	62
10. GAME RESEARCH - STRATEGY [13]	62
DESIGN	63
PART 1: THE FINAL VERSION OF THE GAME.....	63
1.1 <i>Overview</i>	63
1.2 <i>Gameplay design</i>	64
1.2.1 Game summary and main objective.....	64
1.2.2 Sleeping.....	65
1.2.3 Stats.....	66
1.2.4 'Fear effects'	67
1.2.5 In-game items.....	67
1.2.6 Using items and inventory.....	69
1.2.7 Being attacked.....	70
1.3 <i>Artificial Intelligence design</i>	70
1.3.1 AI of Rot.....	70
1.3.2 AI of survivors	71
1.4 <i>None gameplay screens and interface design</i>	73
1.4.1 Main Menu.....	73
1.4.2 In-game Menu.....	73
1.4.3 Game over screen	74
1.4.4 Game complete screen.....	74
1.4.5 Conversation screen.....	74
1.4.6 Credits screen.....	75
1.4.7 Saving and loading	75
1.5 <i>Level design</i>	75
1.6 <i>Media design</i>	76
PART 2: THE PROTOTYPE.....	76

PART 3: DESIRABLE FUNCTIONALITY 76

PART 4: HOW TESTING WILL BE PERFORMED 78

 4.1 *Software testing* 78

 4.2 *Game testing*..... 79

PART 5: RISK ANALYSIS 79

 5.1 *Risk analysis table*..... 79

 5.2 *Alternative development environment: RPG Maker XP [14]* 80

REFERENCES 80

BACKGROUND BIBLIOGRAPHY 81

Literature review

This section will include a review of several computer game development books and articles. Their usefulness with respect to the game I will develop is also evaluated.

1. Better Game Characters by Design: A Psychological Approach [1, 2]

This book deals with character design within games. It discusses how real-world people react to certain events and explains how these reactions can be incorporated into games. It focuses more on realistic human behaviour and psychology than the technical side of computer game creation or Computer Science. A large part of the book is devoted to animation styles and how to portray the emotion of in-game characters, in particular characters body language and movement in relation to other characters. All techniques discussed in the book are reinforced by examples. There are even interviews with expert game character designers and researchers pushing the boundaries of the social behaviour of game characters.

In general the book seems to focus more on large-scale games for marketing and not any game I could develop in the limited time frame of this project. It may be a useful book for future work or in computer graphics and animation courses.

2. Rules of Play: Game Design Fundamentals [3]

Described as "...a monumental examination of the emerging field of games design..." by The Guardian in November 2003. The book adopts a unified model for all types of games be it board games, sports, computer games, etc. The book has been reported to have been used by some professional game creators. It defines many core features of games such as design and interactivity. It is heavily based on game design theory as an emerging field that is being seen increasingly more as a serious profession. With 672 pages and small print it is a large book that may be difficult to understand by novices to games development. It covers every aspect of computer game use from a story telling medium to a communication method. It is divided into 4 sections with a different game design document written by experts in the field, at the end of each.

The book was published in 2004 so may be starting to show its age but it is one of the only books available that focuses on game design theory without focus on practice. It is a good book to have for anyone considering games design as a profession. However, with 672 pages, it presumably goes into too much detail than the example game to be created would need.

3. Programming Believable Characters for Computer Games [4]

This book is aimed at game programmers rather than game designers but therefore does involve Computer Science more than the other books reviewed. It focuses more on enemies in games but does explain artificial intelligence (AI) well using familiar techniques like finite state machines. While not attempting to push forward any boundaries it does give programmers tried and tested ways of creating believable AI in games.

The book focuses on the step-by-step creation of 3-dimensional (3D) animated characters with relation to common AI techniques used in Computer Science. These techniques include path-finding, rule-based systems, goal-orientated planning and decision trees. These can be applied to AI in general and not just to game programming.

Since it is focused more on 3D characters it does not apply much to the game I can create in the restricted time scale of this project.

4. AI Game Programming Wisdom [5, 6]

This book contains a collection of formal articles by experienced authors concerning useful techniques that can help game developers avoid reinventing the wheel. There is a large section of the book devoted to machine learning believing that the next major development of Computer Science and computer games will be to create a machine that learns. Complex AI techniques are broken down into easily understandable sections and most are accompanied by program code or pseudo-code. It includes descriptions of A* path-finding which is often used in commercial games. Some of the examples can be hard to follow without a good programming background. As the articles are written by professionals in the game industry they often explain decisions made and problems encountered in game development projects. Many articles make use of UML diagrams and C++ but a detailed understanding of C++ syntax is not required. The book focuses a lot more on AI development for PCs and does not take the restricted resources available to consoles or handheld computers into account.

It is definitely not for beginners to computer science but when you have some experience and understanding of computer systems it becomes a good resource for taking common computer science techniques and applying them to games. From the review it goes into more detail than needed for the AI I will need to create for the game.

5. Programming Game AI by Example [7]

The book includes many AI techniques used in Computer Science and computer games. Explanations of concepts are entertaining to keep the reader interested and helpfully depicted with diagrams and illustrations. It discusses Finite State Machines for state based AI and Messaging so objects within the game can send messages to each other to co-ordinate attacks for example. Scripting, fuzzy logic, goal-oriented AI using states and path tree search are also covered. The book also covers path-finding in large detail with many descriptive diagrams. A plus side for the game I am to develop is that all the examples in the book are based upon 2-dimensional games. That said the techniques discussed can still be applied to 3-dimensional games. The book does assume the reader is familiar with C++ and STL (Standard Template Library), which I am not.

This book may be of use in the project if the AI of computer-controlled characters is developed further.

6. Game Design Complete [8]

This book understands that most games are developed under constraints, be it hardware limitations, marketing issues, time, money or sales potential. The book tries to make you think of these as more of an opportunity to create exciting and original games rather than a problem. It covers many aspects of game design including designing for licenses when a game is based on a film for example and how to design games for technology that has significant limitations for example, limited memory or limited display size. It also covers controversial design ideas such as dealing with strange and challenging environments such as Mars or the arctic. The book features many useful design techniques and modern approaches. It even includes a section on Disaster Management for when things go wrong in game development.

Overall, I think this is a very good book for game design which is a very important part of game development. After all, if a game isn't designed well it won't sell well and may be difficult to implement. It may be of use for the game I will develop.

7. The Game Maker's Apprentice: Game Development for Beginners [10]

This book is a must for all beginning game developers. It takes you through all the stages of game development from having no idea for a game to implementing state-based AI using Finite State Machines. It is written by both former computer scientists and current professionals in the games industry.

Since I will be using Game Maker to create the game this book is essential. There is a website with very useful tutorials that can be downloaded and a list of other recommended game design books.

8. Game studies: Where the action is [11]

This website contains links to many academic papers analysing computer game design. In particular "Formal models and game design" describes how games are thought of as systems which are designed using mathematical models. Many of those models are theoretical including state-transitions, mappings and logic but there are also more practical techniques related to computer science. The article also describes why formal methods are important when designing a game in the same way that they can be important for designing any computer software.

Most of the mathematical models it gives appear far too complex for the game I will create in this project. In my opinion I would need to consider using these formal methods if I was creating a marketable game. That said they do have the disadvantage of requiring special knowledge to use in the same way that formal methods used in any other computer projects require. Also the use of formal methods in design would, most likely slow the entire project which I can not afford.

9. Gender and Computer Games: Exploring Females' Dislikes [12]

This article shows results of studies into the gender of game players and describes how these relate to game design. It provides many statistics, facts and figures. In general it shows that women play games a lot less than men and when they do they play for a shorter time. The main reason for this is the lack of meaningful social interaction and the violent and sexual content of most games. It also suggests that women gamers do not like the competitive elements of games.

I have seen many articles on the debate about the gender of players. The game I will develop will not involve much violence since this is a requirement. There may not be much competition in the game I will create, but I think competition adds to the enjoyment of a game. Many game design books support this argument.

10. Game Research - Strategy [13]

This article focuses on strategy games. These are games in which you control several characters called units. The article describes how strategy games started as turn-based games

where the player moves and gives orders to their units in one turn. The computer or other player then takes their turn, moving units and constructing buildings. Play continues in this fashion with a break between each turn. Strategy games have evolved into "real-time" games where there is no concept of turns and every action takes place almost immediately. Every player, including computer controlled players, play (for example, ordering units to move) at the same time so a useful analogy is to think of each turn in a turn-based game being joined to form one overall turn.

It would be more difficult to design a strategy game that met all the main objectives given in the "Project specification and Plan". I do, however, need to keep strategy in mind during the design stage of the project. Since the game is required to educate the player in the basic concepts of survival via the collection and use of in-game items it will need some elements of strategy and planning for success. I believe making players think for themselves and form their own strategies is a good way to educate them.

Design

In this section I will design the game that will be created. The game definition, objectives and requirements given in the "Project Specification and Plan" document will be taken into account. I will adopt a Software Engineering process for the design including the use of UML diagrams. The design will be split into five parts. The first part is the design for the final version of the game. This will endeavour to meet all the game objectives and main objectives of the project. The second part will be the design for the prototype system that will meet some but not all the stated objectives. Part 3 includes desired functionality which should be considered optional and may or may not be included in the final version of the game. Part 4 summarises how the game created will be tested. Part 5 includes a risk analysis and a brief description of an alternative game development environment.

Part 1: The final version of the game

1.1 Overview

The game will have the title Hunger and run on a PC. It is a game that focuses mainly on survival with limited reliance on fighting enemies or horror. It provides some limited educational content and aims to show the basic concepts of survival for example, the need to eat and drink. There will be enemies but they are mainly enemies to be avoided instead of fighting directly. As described in later sections the game provides the user with choices as to how to play while keeping focus on the main objective of the game as stated in section 1.2.1.

The game requires the user to think about when to use in-game items for maximum effect in order to survive. The majority of the game is 2-dimensional with a top-down perspective.

The game can be divided into five main areas of design.

1.2 Gameplay design

This is the design of the actual game that is played. In general all gameplay will be 2-dimensional and viewable only from the top down.

1.2.1 Game summary and main objective

The game will be set in a town called "Fondville" which consists of an island connected to the main city via two bridges and a cable car. The island is divided into 3 main areas as described in the "Level design" section. The two bridges are inaccessible so the player (user) must guide the playable character from the southern end of the island to the cable car in the north in order to complete the game. To make things more difficult the town has been overrun by zombies called 'Rot'. These will act as enemies and attack the playable character. They are entirely controlled by the computer. There will also be computer controlled characters called survivors which the playable character will meet at various stages throughout the game. These will provide the user with survival tips and tell them how to play the game successfully.

In order to educate the user in the basic concepts of survival, various stats will be maintained as described in section 1.2.3. The playable character will be able to move over and pick-up in-game items which are then stored in the inventory described in section 1.2.6 on page 69. These items can then be used to increase or decrease the maintained stats. The user must therefore manage the stats of the playable character while trying to progress and not be killed by the Rot.

The playable character will be able to move around the game world by key presses on the keyboard. They will never be able to move out of viewable area of the game world or through solid objects.

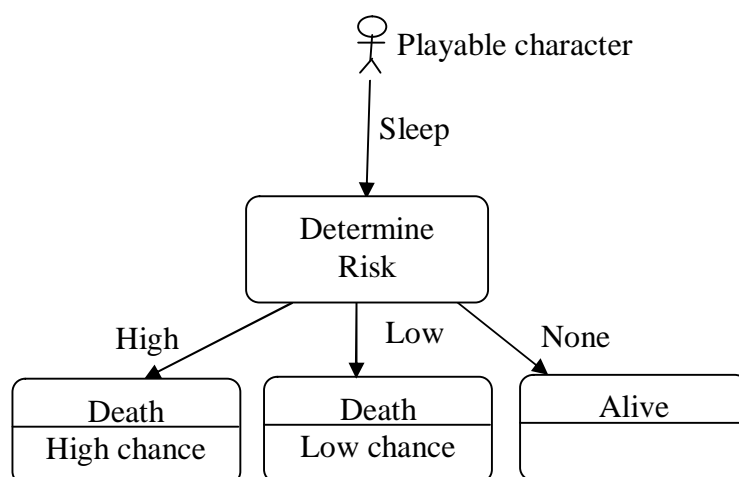
Information about the game story, the text that is or can be displayed by survivors and other details are given in the "Detailed Design" document on the project website [15]. A high-level flow control diagram of the game is given in the "Flow Control Diagram" document on the project website [15].

1.2.2 Sleeping

During gameplay the user can instruct the playable character to sleep. Sleeping is used to affect the "Fear" stat described in the next section. Sleeping will be indicated with a static screen and will temporarily stop gameplay. There is a progress bar to indicate how long until the sleep cycle ends. When the current sleep cycle ends the playable character "wakes up" and gameplay is restored to normal.

Since the game world has become inhabited by Rot obviously the playable character should not be able to sleep anywhere with no risk of being killed. There are 3 categories of place where the playable character can sleep. It is safest to sleep inside barricaded buildings. Buildings are initially unbarricaded but can be barricaded using the barricade item described in section 1.2.5 on page 67.

<u>Where character sleeps</u>	<u>Risk</u>	<u>Description</u>
Outside	High	Random chance of death. High chance.
Inside a unbarricaded building	Low	Random chance of death. Low chance.
Inside a barricaded building	None	No chance of death.



A summarised statechart diagram of Sleeping

When the playable character dies as a result of sleeping a dialog box is displayed informing the user of the death. When the user clicks the button labelled 'OK' the playable character is removed from the game screen and one live is lost. After a short time the character reappears at another position on the screen.

1.2.3 Stats

These will be shown on the game screen at all times when the user has control of the playable character. The stats will be represented using horizontal bars that increase or decrease with their numeric value. The following is a list of all stats.

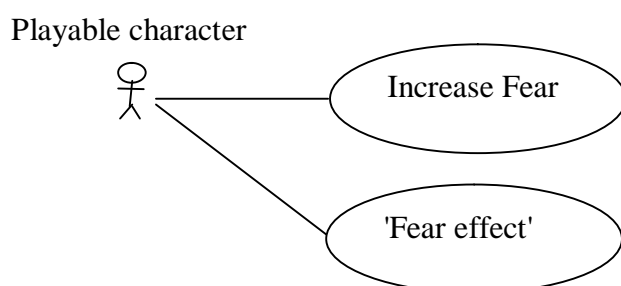
<u>Stat</u>	<u>Want to keep</u>	<u>Description</u>
Health	High	Decreases when playable character collides with Rot or if 0 of an indicated stat. Increases by using bandage or medkit item. When becomes 0 playable character dies. The playable character is removed from the game screen and 1 live is lost. After a short time they reappear at another position on the screen. If no more lives Game Over is displayed.
Warmth	High	Decreases over time when playable character not near fire. Increases when playable character is near a fire. When becomes 0 health slowly decreases.
Thirst	Low	Increases over time. Decreases by using drink item. When maximum value is reached health rapidly decreases.
Hunger	Low	Increases over time. Decreases by using food item. When maximum value is reached health slowly decreases.
Fear	Low	Increases slowly over time. Large increase if playable character "sees" Rot (explained in "Detailed Design" document on the project website [15]). When maximum value is reached the playable character moves around randomly at a high speed and after a short time sleeps where ever they are. The playable character needs to sleep to reset this stat to 0. Has the advantage of slightly increased movement speed at higher levels but the disadvantage of 'Fear effects' (described in next section).
Speed	High	Stays at constant level until Fear stat increases above a certain value then starts to increase over time making the playable character move faster.

To make it clear which stats the user should maintain at a high level and which should be at a low level they will be divided into two categories. Each category will be shown on a different half of the game window to provide symmetry.

1.2.4 'Fear effects'

'Fear effects' are experienced as the Fear stat increases beyond a certain value. They only occur at high-levels of Fear so as never to become annoying and predictable. There are many 'Fear effects' and when one is to occur (they occur at random past the threshold value) it will be chosen at random from the following list:

- Static Rot suddenly appear near playable character and then disappear again.
- The user sees items which are not there. When the playable character gets close to them they disappear. They also disappear after sleeping.
- The screen suddenly goes black for a few seconds and then returns to normal. While the screen is black the game is stopped.
- A large picture of a Rot is displayed on the screen for a few seconds. A sound file plays and then the screen returns to normal. While the picture is displayed the game is stopped.



A use-case diagram of 'Fear effects'

1.2.5 In-game items

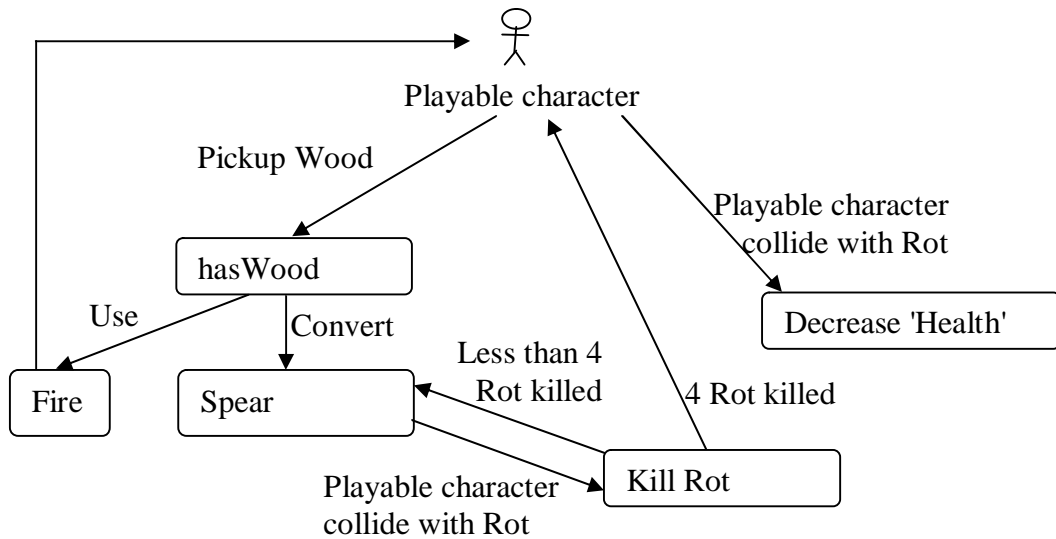
All items can be collected by the playable character in the game. To do this they simply move over of the item. Collected items can be viewed and used from the inventory designed in section 1.2.6 on page 69. When items are used none, one or more of the stats described in section 1.2.3 are altered.

<u>Item</u>	<u>Effect when used</u>	<u>Description</u>
Food	Decrease hunger	-
Drink	Decrease thirst	-
Medkit	Increase health a large amount	-
Bandage	Increase health a small amount	-

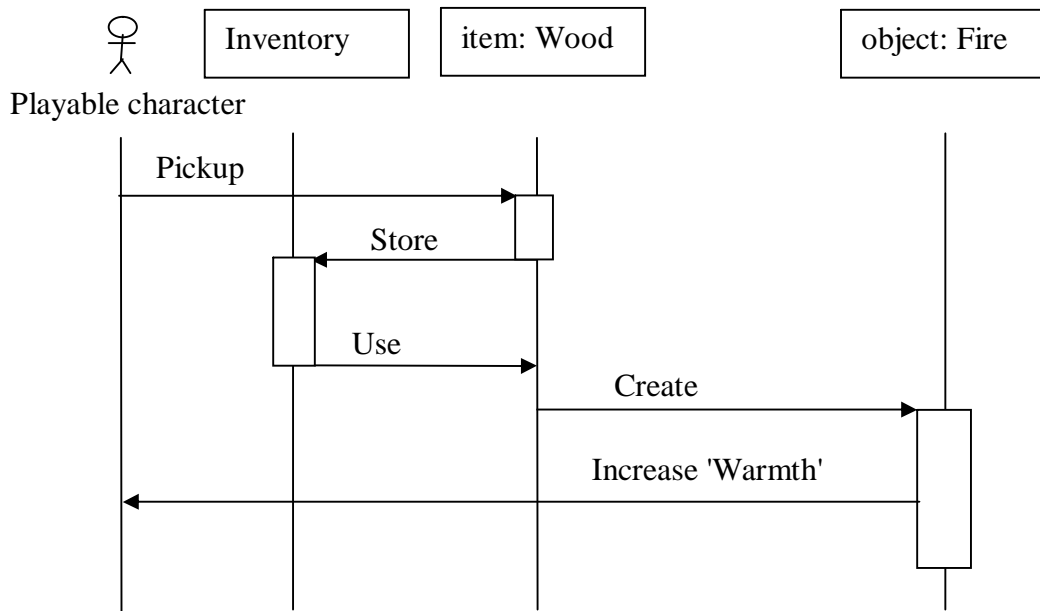
Barricade	Stops Rot entering buildings	Use of this item will make the current building that the playable character is in inaccessible to the Rot and therefore safe. Safe areas are desirable for sleeping. The playable character is not affected by used barricades and can move through them. Used barricades will be removed from the game after 4 sleeps.
Wood	Variable	<p>This is a dual purpose item with a different use depending on the purpose selected by the user.</p> <ul style="list-style-type: none"> • In its default state wood can be used to make a fire near to the playable character. This will increase their Warmth stat provided the playable character stays close to the fire. The playable character, enemies and other characters will be able to move through fires. Used fires will be destroyed after a set period of time. • If the user decides to convert the wood into a spear they can use it as a weapon to kill Rot. When the user uses a spear the graphic representing the playable character will change and they can then simply move into Rot to kill them. Once killed, Rot are removed from the game. After 4 Rot have been hit by the spear the spear will break and the playable character will revert to normal with the spear removed. Once the wood has been converted to a spear it cannot be converted back to use as wood.

In addition to wood being used to create fires and increase the playable characters “Warmth” stat there are also fire barrels. Like fires made with wood, fire barrels only increase the Warmth stat if the playable character is within a limited radius of them. Barrels are not moveable or collectable game objects.

Below is a state diagram for the more complex wood item.



An expansion of the path leading to the Fire state as a sequence diagram:



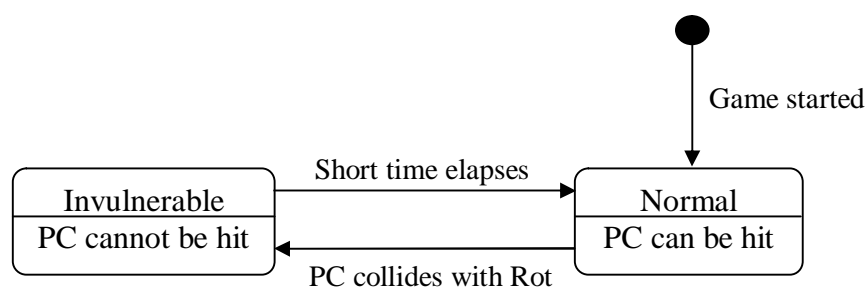
1.2.6 Using items and inventory

The game will allow the playable character to collect and store items. Collected items are viewable from the inventory screen. The inventory can be viewed at anytime while the user has control of the playable character. Gameplay will not be stopped while the inventory is open and the inventory screen will not occupy the entire game window. A diagram showing the layout of the inventory screen is shown in the "Detailed Design" document on the project website [15].

Items in the inventory can be used or dropped by selecting options from an action menu specific to the type of item clicked on. The "Detailed Design" document gives a table of the options in each items action menu. When items have been used they are removed from the inventory and cannot be used again. Dropped items are removed from inventory and placed at a random location near the playable character. All dropped items can be collected again and used as normal.

1.2.7 Being attacked

As mentioned before the enemies in the game are Rot and when the playable character collides with a Rot their health decreases. Upon this collision the playable character is also moved back slightly and made invulnerable for a short time. While invulnerable the playable character cannot be attacked and the graphic representing them is made partially transparent. This is summarized in the below statechart diagram.



PC = Playable character

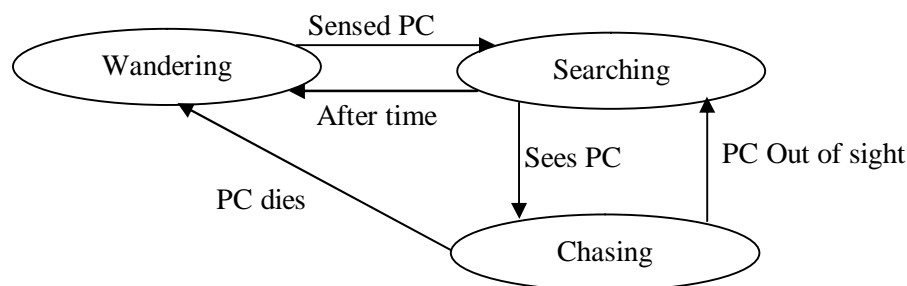
● = Initial state

1.3 Artificial Intelligence design

As mentioned in the "Gameplay design" section there will be two types of computer controlled entities: Rot and survivors. Neither will be able to move through solid objects such as walls. This rule will not be represented on diagrams to avoid clutter.

1.3.1 AI of Rot

Rot will act as enemies within the game and attack the playable character. The following finite-state machine describes their behaviour.



PC = Playable character

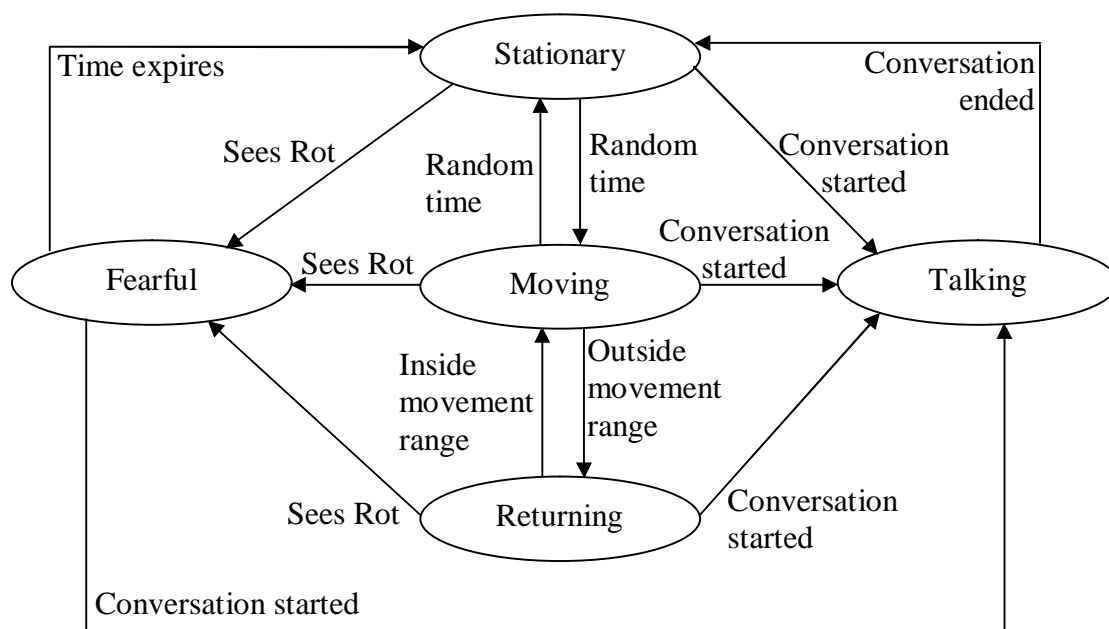
Initially Rot are in the Wandering state. Here they will move around at random with a slow movement speed. If the playable character moves too close to a Rot, that Rot will "sense" them and switch to the Searching state. Here they will move in the general direction of the playable character but not necessarily towards them. They will receive a slight increase to their movement speed. If a Rot in the Searching state then "sees" the playable character they will switch to the Chasing state and move towards the playable character. They will receive another slight increase to their movement speed. To "see" the playable character a line drawn from the centre of the Rot to the centre of the playable character must not be broken by solid objects, including other Rot. The graphic representing the Rot will be shaded red to indicate they are in the Chasing state.

If the playable character hides behind an obstacle and can no longer be seen by the Rot the Rot will switch back to the Searching state. Also if the playable character moves far away from the Rot (out of their visible range) the Rot will switch back to the Searching state. This will remove the red shading and start a timer. Once that timer has reached 0 the Rot will switch to the Wandering state as long as the playable character has not been seen by the Rot.

If the playable character dies while a Rot is in the Chasing state that Rot will revert back to the Wandering state. The red shading will be removed from the Rot.

1.3.2 AI of survivors

Survivors will act as friendly characters and advisors to the playable character. The following finite-state machine describes their behaviour.



The initial state for all survivors is Stationary. In this state they will not normally move. After a random time they will switch to the Moving state. In this state the survivor will walk in a random direction. When the survivor has walked far from their start location (original position in game world) they will switch to the Returning state and move towards their start location. When the survivor is close to their start location (within their normal movement range) they revert back to the Moving state. In this way they will never move far from their start location. After a second longer random time they will switch back to the Stationary state.

If the playable character is within a short distance of a survivor in any state except the Talking state, and the user presses a particular key on the keyboard it will start a conversation. This immediately transfers the survivor to the Talking state. Here they will not move but will turn to face the playable character. They will then choose a random statement from a list of possible statements to display in the "Conversation screen" designed in section 1.4.5 on page 74. If this was the first time that survivor was talked to they will display a special statement if they have one. Once a conversation has ended the survivor will switch to the Stationary state. The user can end a conversation by pressing the "End conversation" key on the keyboard.

If the survivor sees a Rot nearby the survivor will switch to the Fearful state. Here they move away from the Rot at an increased speed. This could occur in any state but is ignored in the Talking state. After a time, the survivor will stop and return to the Stationary state. If they can still see the Rot they will return to the Fearful state and move again. If they can't see the Rot

they will move back to their start location. When they are at that position they return to the Moving state.

1.4 None gameplay screens and interface design

This section covers the design of screens where the user does not control the playable character in the game for example, the main menu and end game screens. All none gameplay screens except the game complete screen use the mouse to select and activate buttons.

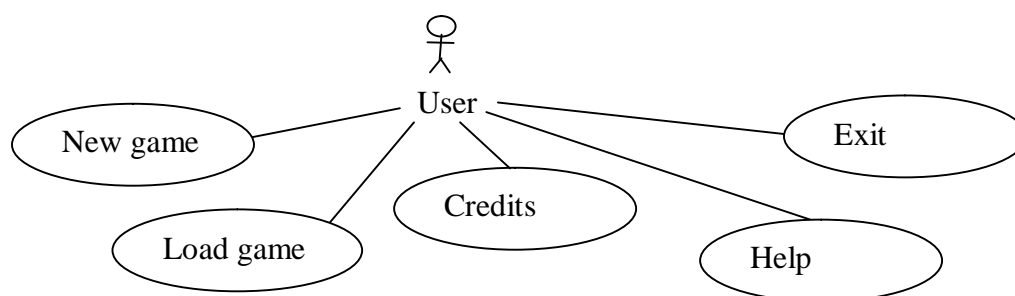
1.4.1 Main Menu

This will allow users access to the actual game and initiate gameplay. It will contain several buttons that can be clicked by the user to perform various actions. The buttons and associated actions are:

<u>Button name</u>	<u>Action</u>
New game	Begin a new game.
Load game	Load a previously saved game.
Credits	View the names of people involved in the project.
Help	Display the help information for the game.
Exit	Exit the game software and return to the operating system.

There will be a logo showing the title of the game and some text stating that the game does not provide a detailed or complete survival guide and only illustrates basic concepts.

A diagram showing the layout of the main menu is shown in the "Detailed Design" document on the project website.



Use-case diagram of the main menu

1.4.2 In-game Menu

There will also be a sub-menu that can be accessed while the game is being played. It will contain the buttons:

<u>Button name</u>	<u>Action</u>
Continue game	Close this menu and return to the game being played.
Save game	Save the current game.
Load game	Load a previously saved game.
Help	Display the help information for the game.
Quit to main menu	Quit the current game and return to the main menu.

A diagram showing the layout of the sub-menu is shown in the "Detailed Design" document on the project website [15]. Gameplay must be temporarily stopped while this menu is viewed and continued when the "Continue game" button is pressed. If development time for this part is short the in-game menu can be implemented as the main menu with some disabled buttons.

1.4.3 Game over screen

This will be displayed when the number of lives is equal to 0. It will contain the words "Game Over" in large text and provide 3 buttons:

<u>Button name</u>	<u>Action</u>
Return to main menu	Redisplay the main menu.
Load game	Load a previously saved game.
Exit	Exit the game software and return to the operating system.

A diagram showing the layout of the game over is shown in the "Detailed Design" document on the project website [15].

1.4.4 Game complete screen

This will be displayed when the playable character reaches the cable car at the end of the game. After 1 or 2 minutes this screen disappears and the main menu is displayed.

1.4.5 Conversation screen

This will be displayed when the playable character starts a conversation with a survivor. This screen will therefore be viewable while the game is being played. It will not occupy the whole of the game screen. The game will stop while this screen is being viewed and resume when the user presses the "End Conversation" key on the keyboard. If the conversation is larger than will fit on one conversation screen the user will have to press a key on the keyboard to view the remaining text on another screen. A diagram showing the layout of the conversation screen is shown in the "Detailed Design" document on the project website [15]. If time is short the conversation screens can be replaced by default Game Maker message boxes.

1.4.6 Credits screen

This will display the names of all the people involved in the project and their role within it. The names will scroll upwards. The screen will also include a button to return to and redisplay the main menu.

1.4.7 Saving and loading

At many points in the game the user can save their progress through it by clicking the save game button. The user is prompted for a name for the save. All game progress is saved in `<name>.sav` files where `<name>` is the name the user input. If `<name>` is the empty string or if it contains any of `< > \ / ? " * : |` it is said to be invalid and a new name is asked for.

If a file with `<name>` already exists a warning is displayed asking the user if they wish to overwrite that save game. If they do not a new name is asked for. If they do the game is saved and a conformation message displayed.

Loading of save games is done in a similar way. When a load game button is clicked the user is prompted for the name of the save game to load. In the same way as saving the game, this name cannot be invalid. If a saved game with the input name does not exist an error is given and no save game is loaded.

When saving the game or loading a save game, if the input name is CANCEL the prompt asking for a name disappears and no game is saved or loaded.

1.5 Level design

The game will be divided into 3 levels or "areas". Each area will usually be too big to fit in a single window. Views (provided by Game Maker) will be used on each area that follows the playable character as they move about that area. In each area there will also be a view that represents a smaller map or mini-map of that area. The mini-map will show the location of the playable character, location of obstacles, location of enemies, location of survivors and the location and type of in-game items.

Each area will depict a different part of the virtual town where the game is located. The area the playable character starts in (level 1) will represent the sub-urban village area of the game. That will connect to the thinner tunnel area (level 2). Finally, there will be a large area representing a more urban landscape (level 3). This last area will contain more enemies and

obstacles than the first two to make the game progressively more difficult as the user advances through it. The third area will contain a stylized picture of a cable car which the playable character can move into to complete the game.

Diagrams of level layouts can be downloaded from the project webpage [15]. A diagram of level 1 is given in the file "Level1.bmp". A diagram of level 2 is given in the file "Level2.bmp". A diagram of level 3 is given in the file "Level3.bmp". The designs shown in these files are **not** an exact match to level layouts used in the final version of the game.

1.6 Media design

This section covers the pictures used to represent in-game objects and any sound effects required by the game. Since the actual graphics and sound are not an important part of this project I will only list and briefly describe them in the "Detailed Design" document available on the project website [15]. If time is short some of these may be omitted or simplified.

Part 2: The prototype

This will:

- Allow the user to control the playable character on the screen via key presses. This character will not be able to move outside the viewable area of the game world.
- Contain **only** a single level which the character can move around in and collect items.
- Include all stats specified above that change dynamically.
- Store collected items and allow all those items to be used as specified above. Note this does **not** include 'Fear effects'.
- Include a main menu screen with all buttons. Many buttons will **not** yet have any functionality.

Note that any pictures and sounds used in the prototype may not accurately represent the final version of the game.

Part 3: Desirable functionality

These are extra functions that could be performed in order of most desirable. They are not required and therefore may not be implemented.

1. The inclusion of the newspaper item:

<u>Item</u>	<u>Effect when used</u>	<u>Description</u>
Newspaper	Discover game story	Use of this item will show a part of the games story. There will be a maximum of 4 newspapers and they will not be removed from inventory when used. Since there are only 4 in total there is not the problem of them taking up too much inventory space. The text they will show when used is given in the "Detailed Design" document on the project website [15].

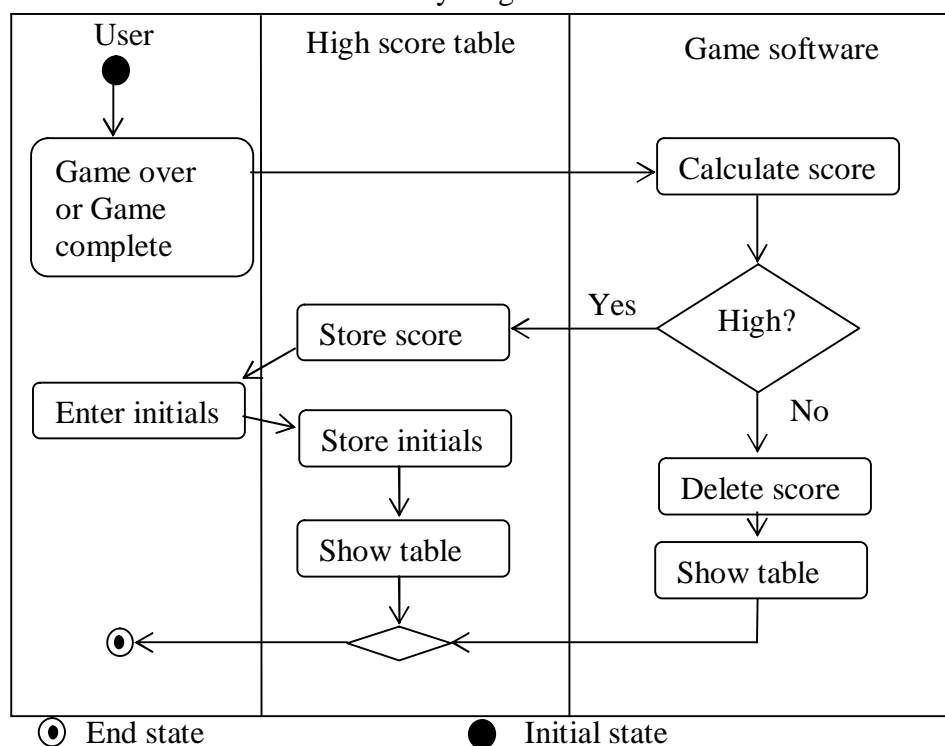
2. Via the in-game menu the user can access a map of the town to check their progress through the game. This will require an extra button:

<u>Button name</u>	<u>Action</u>
View Map	Display the map of "Fondville" and highlight the area the playable character is in.

3. A procedure to automatically detect the resolution of the monitor displaying the game and resize the game window accordingly. This would be done between the activation of the game software and the main menu being displayed.
4. Giving the playable character a score that is 0 at the beginning of a new game. It would increase by collecting newspapers and completing tasks set by survivors. The score would be maintained when the game is saved and loaded. At the Game Over and Game Complete screens the score would be automatically compared to other scores and stored if high. If a score was stored the user is able to input their initials. There would also need to be an extra button on the main menu:

<u>Button name</u>	<u>Action</u>
View High Scores	Display the top 15 highest scores and the initials entered by the user who made them. Scores are ranked from 1-15 with 1 being the highest.

An activity diagram of this



5. The inclusion of the following additional 'Fear effects'.
 - All Rot in the current game view move towards the playable character independent of obstacles for a short time. For example, Rot move through walls while effect is activated.
 - Game window goes blurry but eventually becomes clear again.
 - All graphics become greyscale for a limited time before returning to normal.
6. An Options screen that can be accessed by clicking a button on the main menu. This would contain a slider to set the game difficulty and menus that allowed the game controls to be changed. If desirable function 3 is not met then there could also be a control to manually set the resolution of the game.
7. Allow survivors to be killed by Rot.

Part 4: How testing will be performed

The created game will be tested in 2 different ways.

4.1 Software testing

The game software will be tested to ensure it meets all the requirements listed in the "Project Specification and Plan" document. This will be done by testing components in the sections designed above. For example, once the stats have been implemented they will be tested before the implementation of the in-game items is started. Sections already tested will need to be

tested again when other sections that affect them are implemented. For example, it will be tested that all in-game items correctly affecting only the stats they are designed to. When the final version of the game is complete it will be tested for any errors that were not detected and fixed in earlier stages.

4.2 Game testing

As briefly mentioned in the "Project Specification and Plan" the game created will be given to game testers at various points during development. There will only be a small number of game testers. They will provide feedback on the development of the game and complete a questionnaire, created by the game designer to assess a range of people's reactions to the game. All versions of this questionnaire used in the project can be found on the project website [15]. The feedback and answers to the questions will be taken into consideration when continuing development of the game. If the results of game testing indicate any new features should be added to the game these will be considered desirable functionality and may or may not be incorporated in the final version of the game. Game testing will not affect the prototype of the game.

A summary of the testing performed and feedback from game testers will be produced.

Part 5: Risk Analysis

This section analyses the risks involved in the project and gives a brief description of an alternative game development environment which can be used to create the game if needed.

5.1 Risk analysis table

<u>Risk</u>	<u>Risk Probability</u>	<u>Recovery Plan</u>
Loss of time due to university coursework or other reasons.	High	Implement only core functionality. Delay or cancel releasing of prototype game software to testers.
Planned stages taking longer than estimated.	Medium	Implement only core functionality.
Unreliable code.	Low	Perform frequent testing and rewrite code if needed.
Game Maker development environment not capable of implementing all requirements of the game.	Low	Re-implement game using different development environment and re-draft documentation.

5.2 Alternative development environment: RPG Maker XP [14]

RPG Maker XP is another game development environment which, like Game Maker, only creates games to be run on the Windows operating system. Also like Game Maker it is event-driven, registering objects with events and performing some action when the event occurs. The main difference between this and Game Maker is that it only creates RPGs (Role-Playing Games). This is okay for the game I will create as it is similar to a short RPG. More information can be found at reference [14]. I now list the advantages and disadvantages of RPG Maker XP in relation to Game Maker.

<u>Advantages</u>	<u>Disadvantages</u>
<ul style="list-style-type: none"> • Many professional sprites, sounds and images provided with environment. • Isometric sprites and layering of objects gives created games a 3-dimensional look. • Pre-programmed way of entering and exiting buildings with separate interiors. • Support for encryption and decryption of game files to increase security. • Easy to implement conversations with computer controlled characters. 	<ul style="list-style-type: none"> • Limited to 30-day trial version and then costs \$60 for full version. Game Maker is free for non-commercial use. • No official tutorials on software use or game design. • Stats for character creation are overcomplicated for game I will create. For example, it will not make use of a characters strength or dexterity. • Unclear how to add new functionality for example, an inventory. • No built-in debug feature.

References

- [1] Title: Book Excerpt - Better Game Characters By Design
Website URL: http://www.gamasutra.com/features/20060602/isbister_01.shtml
- [2] Title: Better Game Characters By Design
Website URL: <http://friendlymedia.sbrl.rpi.edu/bettergamecharacters/>
- [3] Title: Rules of Play - Game Design Fundamentals
Website URL: http://www.amazon.co.uk/Rules-Play-Game-Design-Fundamentals/dp/0262240459/sr=1-1/qid=1161366626/ref=sr_1_1/026-6976647-0885206?ie=UTF8&s=books
- [4] Title: Drama princess - Tale of tales develops an autonomous character for real-time 3D - Programming Believeable Characters For Computer Games
Website URL: <http://www.tale-of-tales.com/DramaPrincess/wp/?p=17>

- [5] Title: Review of AI Game Programming Wisdom
Website URL: <http://www.omlettesoft.com/document.php3?did=108>
- [6] Title: IGDA Forums
Website URL: <http://www.igda.org/Forums/showthread.php?threadid=1534>
- [7] Title: GMAN-GAMES - Programming game AI by example
Website URL: <http://greggman.com/edit/editheadlines/2005-02-12.htm>
- [8] Title: Paraglyph Press - Game Design Complete
Website URL: <http://www.paraglyphpress.com/pr12062005.php>
- [9] Title: ibs.it Internet Bookshop - Game Design Complete
Website URL:
<http://www.internetbookshop.it/ame/ser/serdsp.asp?shop=2293&e=1933097000>
- [10] Title: The Game Makers Apprentice - Game Development for Beginners
Authors: Jacob Habgood and Mark Overmars
Publisher: Apress
Publication Year: 2006
- [11] Title: Game Studies - Where the action is
Website URL: <http://www.gamestudies.org/0501/>
- [12] Title: Gender and Computer Games - Exploring Females' Dislikes
Website URL: <http://jcmc.indiana.edu/vol11/issue4/hartmann.html>
- [13] Title: Game Research - Strategy
Website URL: http://game-research.com/?page_id=33
- [14] Title: RPG Maker XP
Website URL: http://www.enterbrain.co.jp/tkool/RPG_XP/eng/index.html
- [15] Title: Hunger - Project Website
Website URL: <http://uk.geocities.com/tdc1@btinternet.com/Hunger/Hunger.html>

Background Bibliography

1. Title: CS4099 (major) / CS4098 (minor) Software Project
Author: James McKinna, University of St. Andrews.
Publication Year: 2006
2. Title: Tutorial - What is a good game?
Author: Mark Overmars
Publisher: Mark Overmars
Publication Year: 2004.

3. Title: Software Engineering - Seventh Edition
Author: Ian Sommerville
Publisher: PEARSON - Addison Wesley
Publication Year: 2004
4. Title: Practical UML - A hands on introduction for developers
Website URL: <http://dn.codegear.com/article/31863>

Changes to original documents

Project Specification and Plan

6/11/2006	For requirement describing what the AI of enemies must be based on (Requirement 8.4) the following changes were made: <ul style="list-style-type: none"> • Phrase <i>finite-state machine</i> is replaced with <i>model</i> as clients may not understand finite-state machine. • Base model only applies to enemies and not characters. • Description of base model changed to clearly relate to diagram.
6/11/2006	Contents page changed to be consistent with other documents.
20/11/2006	References updated to clearly relate to text and Background Bibliography section created. Contents page updated.
31/01/2007	Added statement of plan for spring vacation.
20/02/2007	Added that development of game must show standard software engineering techniques. Problem definition re-worded.
27/02/2007	Complete re-formatting. Project requirements and objectives separated from game requirements as recommended by supervisor. Contents page updated.
12/03/2007	Short description of games in context survey added. Quote about gaming industry being big business put at top of problem definition.
13/03/2007	More explanation of choice of game genre. Paragraph with example educational game removed. Re-formatted and contents page updated.

Literature review and system design

20/11/2006	References updated to clearly relate to text and Background Bibliography section created. Contents page updated.
20/11/2006	Inclusion of part 4 detailing how the game will be tested. Contents page updated.
24/11/2006	Adding statement of how menus and other none gameplay screens are controlled.
22/01/2007	Included risk analysis table. Contents page updated.
24/01/2007	Added statement that all collected items can be dropped and collected again later.

25/01/2007	Added description of credits screen. Added reference to high-level flow control diagram. Contents page updated.
26/01/2007	Included a description of what happens when the playable character dies as a result of sleeping.
30/01/2007	Added "If time is short the conversation screens can be replaced by default Game Maker message boxes."
31/01/2007	Included short description of alternative game development environment and compared to Game Maker. Changed some diagrams and words to keep document consistent. Updated contents page.
11/02/2007	Added to description of finite-state machine of Rot that if player moves out of visible radius of Rot then Rot switches back to searching state. Rot switches back to wandering state when playable character is killed.
17/02/2007	Change finite-state machine of survivors to switch back to the stationary state from the moving state after a random period of time.
19/02/2007	'Fear effects' will not get more frequent at higher-levels of fear. Contents page updated.
22/02/2007	Complete change of finite-state machine for survivors.
08/03/2007	Added to 'Fear effects' that after sleeping all remaining items that are not actually there are destroyed.
23/03/2007	Included section "Being attacked" about what happens when the playable character collides with a Rot. Also added "Saving and loading" section explaining how the user saves the game. Contents page updated.
30/03/2007	Increase of Fear stat by seeing Rot referred to "Detailed Design" document. I thought design of how it is done too low-level for the submitted design document.
05/04/2007	'Fear effect' that was to increase speed of all Rot and survivors removed as caused AI not to work correctly when implemented. Also it affected the game more than just the user. 'Fear effects' were intended only to affect the user psychologically.

Testing summary

Contents

SOFTWARE TESTING	85
1. TESTING ROT AI.....	85
2. TESTING SURVIVOR AI.....	86
3. TESTING ITEM PLACEMENT AND CLIPPING.....	87
4. OTHER TESTS	87
GAME TESTING	88
1. FEEDBACK FROM PROTOTYPE VERSION OF GAME.....	88
2. FEEDBACK FROM ALMOST COMPLETE VERSION OF GAME.....	89

Software testing

The following tests are only a very small proportion of the tests performed.

1. Testing Rot AI

<u>Test</u>	<u>Reason</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Satisfied</u>
2 Rot placed in level without survivors present.	Rot can move around the game level without passing through solid objects eg. walls.	Rot move anywhere around the game level without passing through solid objects. They walk in random directions.	Rot move anywhere around the game level without passing through solid objects. They walk in random directions.	Yes
Yellow circles drawn around Rot to show their "sensing" range. Playable character moved into circle. After a short time playable character moved out of circle.	1. When playable character enters the "sensing" circle of a Rot that Rot transfers to the Searching state. 2. When the playable character leaves the circle for a time the Rot switches back to the Wandering state.	1. Rot move as for above test until the playable character enters the circle. When playable character inside circle Rot begins to move slightly faster. They randomly move towards or away from the playable character but do not pass through solid objects. 2. Playable character leaves circle for a time. Rot starts	1. Rot move as for above test until the playable character enters the circle. When playable character inside circle Rot begins to move slightly faster. They randomly move towards or away from the playable character but do not pass through solid objects. 2. Playable character leaves circle for a time. Rot starts moving slowly again	Yes

		moving slowly again and in random directions. Rot never moves through solid objects.	and in random directions. Rot never moves through solid objects.	
--	--	--	--	--

2. Testing survivor AI

<u>Test</u>	<u>Reason</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Satisfied</u>
2 survivors in the Stationary state placed in level without Rot present. Transition to Returning state disabled.	Survivors can move around the game level without passing through solid objects eg. walls. With the Returning state disabled survivors can move anywhere in the level.	Survivors move anywhere around the level without passing through solid objects. They walk in random directions and stop at random intervals.	Survivors move anywhere around the level without passing through solid objects. They walk in random directions and stop at random intervals.	Yes
Transition to Returning state enabled. Additional survivor placed in level. Yellow rectangles drawn to show survivors normal movement range.	Survivors can change from the Moving state to the Returning state when far from their start location. When inside their normal movement range survivors move as in previous test.	As for previous test plus survivors do not move far outside normal movement range.	As for previous test plus survivors do not move far outside normal movement range.	Yes
1 Rot placed in level.	Survivors can change from any state to the Fearful state. They should move at a much greater speed away from the Rot and do not pass through solid objects. Eventually they stop and if they can no longer see the Rot they return to their normal movement range.	As for above tests plus when a survivor sees the Rot the survivor changes to the Fearful state. They move at a much greater speed away from the Rot and never pass through solid objects. After a time they stop and if they can no longer see the Rot they return to their normal movement range. Here they move with the same results as above tests.	As for above tests plus when a survivor sees the Rot the survivor changes to the Fearful state. They move at a much greater speed away from the Rot and never pass through solid objects. After a time they stop and if they can no longer see the Rot they return to their normal movement range. Here they move with the same results as above tests.	Yes

3. Testing item placement and clipping

<u>Test</u>	<u>Reason</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Satisfied</u>
Wood item collected. Playable character positioned with a wall on their right side and another above them. Playable character faces to the right. Wood item used to place fire.	Fires cannot be placed under wall objects and must be placed at the first space around the playable character that does not contain a wall. Fires must not appear behind walls.	Fire placed below the playable character and not far away from them. Fire does not appear behind any walls.	Fire placed below the playable character and not far away from them. Fire does not appear behind any walls.	Yes
Wood item collected. Playable character positioned with wall on their left side and the edge of the level below them. Playable character faces downwards. Wood item used to place fire.	Fires cannot be placed under wall objects or outside the level. Fires must be placed at the first space around the playable character that does not contain a wall and is inside the level. Fires must not appear behind walls.	Fire placed to right of the playable character and not far away from them. Fire does not appear behind any walls.	Fire placed to right of the playable character and not far away from them. Fire does not appear behind any walls.	Yes
Items collected. Playable character positioned in bottom left corner of level touching left and bottom edge. Playable character facing bottom edge. 2 Items dropped.	Items dropped are removed from inventory and appear on the ground at first available space around playable character. Items must not be dropped outside the level.	The 2 items dropped are removed from inventory and appear on the ground above the playable character and not far away from them. Both items appear inside the level. Both items are dropped in the same place.	The 2 items dropped are removed from inventory and appear on the ground above the playable character and not far away from them. Both items appear inside the level. Both items are dropped in the same place.	Yes

4. Other tests

<u>Test</u>	<u>Reason</u>	<u>Expected Result</u>	<u>Actual Result</u>	<u>Satisfied</u>
Items collected. Playable character killed by Rot with inventory open.	User cannot use items or open inventory when playable character dead.	Inventory automatically closed when playable character dies. Any open item action menu is closed.	Inventory automatically closed when playable character dies. Any open item action menu is closed.	Yes

8 items collected. All stats altered. Game saved. Game exited and restarted. Game loaded.	User can save the game and load it to continue playing. Saving and loading a game does not change stat values or remove collected items from the inventory.	When game loaded: - All items collected before save are located in inventory. - All stats set to values before save. - Game continues as before save with all characters in the same positions as before save.	When game loaded: - All items collected before save are located in inventory. - All stats set to values before save. - Game continues as before save with all characters in the same positions as before save.	Yes
Game save started. Name: the?game."	Save game names can not contain invalid characters.	Error message output stating the?game." is invalid. New name asked for when OK clicked.	Error message output stating the?game." is invalid. New name asked for when OK clicked.	Yes
2 barricade items collected. 1 barricade placed (barricade 1). Sleep. When wake, place second barricade (barricade 2) in same level. Sleep.	Barricades decay over time when the playable character is in the same level as them. Barricades should decay independently of each other.	Yellow barricade appears when placed. After sleeping barricade does not appear to have changed. 2 nd yellow barricade appears when placed. After next sleep barricade 1 changed to green. No change to barricade 2.	Yellow barricade appears when placed. After sleeping barricade does not appear to have changed. 2 nd yellow barricade appears when placed. After next sleep barricade 1 changed to green. No change to barricade 2.	Yes

Game testing

The following section lists sample responses to questions on the provided feedback questionnaires. All responses are written by game testers whose name is given at the bottom of the response. Both questionnaires and all feedback from game testers are available on the project website.

1. Feedback from prototype version of game

Did you find any errors with the game, for example, you got stuck on an object and could not "unstick" yourself, the game did not display properly etc.?

Yes, I got stuck between a number of items and couldn't find a way to release the player. This was quite frustrating. - *Kristoffer Marc Getchell*

2. Feedback from almost complete version of game

Did you press the HELP button or F1 at anytime to access the in-game help? Did this solve any problems you had? If it did not solve the problem did you look at the User Manual? Did you know where to find the user manual?

I used the F1 help quite a lot. It was occasionally useful when I wanted to grasp basic game concepts (I didn't find the user manual). It was good for this task, but would have been better if it explained what specific inventory icons were for, as this wasn't clear. - *Angus Macdonald*

Did you understand the aim of the game and how to get from level to level?

Not really, I wasn't sure if the aim was just to stay alive, keeping in mind all of the variables such as fear and thirst etc, or to kill the Rot, or to try and save the princess. Maybe it was just to avoid those annoying other characters that you can't kill for fun. A clear indicator (maybe on the mini-map) should indicate the aim of each level, and where the end is. "Going up" isn't really that well defined. - *Alex Lupu*

How often did you find yourself going back to read the user manual? Did you even need to read it at all? Approximately how much time did you spend reading after playing the game?

I don't think it was necessary to read the manual, it is pretty obvious what you're supposed to do. It is nice for the storyline though! - *Christopher Jackson*

If you ever used the save game and/or load game features did you understand how to use them?

I did understand how to use the save and load features, however it was somewhat difficult to use as there was no way to browse for a saves game. If the name was forgotten, how would I be able to find my previous progress? Having a file browse window would greatly improve this! - *Kristoffer Marc Getchell*

User Manual

Contents

1. INTRODUCTION.....	91
STORY.....	91
2. GETTING STARTED	91
INSTALLING HUNGER	91
UNINSTALLING HUNGER	91
RUNNING HUNGER.....	92
3. CONTROLS.....	92
4. MAIN MENU.....	93
5. STARTING A NEW GAME.....	93
6. PLAYING HUNGER.....	94
7. INVENTORY.....	96
8. ITEMS.....	97
9. ENEMIES	98
10. OTHER SURVIVORS.....	98
11. SLEEPING.....	99
12. PAUSING THE GAME AND THE IN-GAME MENU	100
13. SAVING A GAME.....	100
14. THE GAME OVER SCREEN.....	101
15. EXITING THE GAME DURING GAMEPLAY.....	101
16. RESUMING A SAVED GAME.....	101
17. TAKING SCREENSHOTS.....	102
18. DELETING A SAVED GAME.....	102
19. VIEWING THE GAME CREDITS.....	102
20. GETTING HELP.....	102
21. THE HUNGER WEBSITE.....	103
22. KNOWN PROBLEMS.....	103

1. Introduction

Thank you for downloading Hunger developed by Tom Clark as a computer science major software project. Other people involved in the project, including my supervisor, are listed in the credits section of the game. This manual will tell you everything you need to know to play the game. I hope you enjoy playing the game as much as I enjoyed making it!

Story

Fondville is a peaceful place. It consists of an island just off the coast from the continent and main city. Few, if any, distinguishing features, flooding and job shortage have lead to its sharp decline. Today the population of Fondville is mere thousands, most of which reside in the main city. However, it is not the main city but the island to the west, where our story begins...

On the night of the 5th of June 2009 the Rot appeared. Where they came from is not known. What they are is not known. Why they are here is not known. All that is known is that they will stop at nothing to kill living people. In an effort to control the situation the government has ordered the 2 bridges connecting the island to the main city destroyed. The cable car in the Northern district of the island was left intact so any survivors could escape the abandoned island, based on the assumption that the Rot are to stupid to use it.

2. Getting started

Installing Hunger

Hunger can only be downloaded as a zipped file. You will need to double click the zipped file to open it and extract the installer before it can be run.

To install Hunger simply double click on the installer called "Install.exe" and follow the on-screen instructions.


Uninstalling Hunger

To uninstall the game:

1. Open the Start menu
2. Navigate to Programs
3. Expand the folder labelled Hunger
4. Select the option Uninstall.

Running Hunger

Once you have successfully installed Hunger navigate to the folder where it was installed. By default Hunger is installed in the Program Files folder on your hard disk.

Double click the Hunger icon  to start playing. The installer will automatically create a shortcut to this on the desktop.

You can also run the game from the Start menu. To do this:

1. Open the Start menu
2. Navigate to Programs
3. Expand the folder labelled Hunger
4. Select the option Play Hunger.

Also available in the Start menu is this User Manual in Microsoft Word and PDF format.

3. Controls

Action	Key
Movement	
Move upwards	W or ↑
Move downwards	S or ↓
Move left	A or ←
Move right	D or →
"Unstick" character	Home
Sleep	R
Conversations	
Start/Continue/End conversation	E
Immediately end conversation and open in-game menu	Esc
Item use and Inventory	
Open or close inventory	I or F
Use item	Left-click on item
Others	
Open in-game menu	Esc
Open in game help	F1
Use button	Left-click on button
Take screenshot	F12
Toggle between windowed mode and fullscreen mode	F4

4. Main Menu



When you have started the game and it has loaded, the Main Menu will be displayed. This lets you start a new game (see section 5), load a saved game (see section 16 on page 101), get help on what to do next and about playing the game (see section 20 on page 102), view the credits of the game and who was involved in making it (see section 19 on page 102) and exit the game, returning to the operating system.

5. Starting a new game

From the Main Menu click the NEW GAME button with the left mouse button to start a new game of Hunger. The first area, Southern Fondville, is loaded and you are given control of the female skater in the middle of the screen. From here you must survive the mindless Rot, control your fear and keep your body healthy until you reach the cable car to safety in the third area, Northern Fondville. To progress through the first two areas look for roads leading north. Good luck!

6. Playing Hunger



This is you, the playable character.

HEALTH: [full green bar]


This is your health bar. If health gets low the screen will flash red periodically. If this gets to 0 you will die so try to keep it full. Your health will go down when you:

1. Collide with a Rot
2. Have a full thirst bar.
3. Have a full hunger bar.
4. Have an empty warmth bar.

To increase health, use a medkit or bandage. A solid green bar shows you are at maximum health and cannot increase it further.

WARMTH: [full orange bar]

This is your warmth bar. Warmth slowly goes down over time and if this gets to 0 you will start to lose health. Use a wood item to place a fire to increase it. You can also increase

warmth by standing near a fire barrel 

SPEED: 

This is your speed bar. Your speed will never decrease but once your fear bar (described later) has reached a high level your speed will slowly increase until the speed bar area is completely filled. The playable character will begin moving faster as well.

THIRST: 

This is your thirst bar. Your thirst will steadily increase until the bar area is full. When this happens your health will go down rapidly so always keep thirst as low as possible. Drink a drink item to decrease thirst. Note that thirst cannot decrease below the level shown.

HUNGER: 

This is your hunger bar. Your hunger will slowly increase over time until the bar area is full. When this happens your health will start to go down. You should keep hunger low to remain healthy. Eat a food item to decrease hunger. Note that hunger cannot decrease below the level shown.


FEAR: 

This is your fear bar. This slowly increases over time. It will also increase if you see Rot (see section 9 on page 98). If your fear reaches a high level you will start to hallucinate. If your fear fills the bar area with a solid yellow bar you will lose all control, move around randomly for a short time and immediately fall asleep wherever you end up. The only way to decrease fear is by sleeping but you do not want to sleep too often as you will not get the speed increase. Can you sacrifice your sanity for higher movement speed?

LIVES: 

This shows how many lives you have left. Each time you die, by your health reaching 0 or if you are killed in your sleep, you lose a life. You have only three lives and which can never be replenished if lost. If you lose your last life (the last skater icon disappears) you will lose the game.

LOCATION: **OUTSIDE** 


This shows your current location in that area of Fondville. If you are not inside a building the location text will display OUTSIDE. If you are inside a building (for example, ) the

location text will change to UNBARRICADED. If you are inside a building and that building has been barricaded (see barricade item on page 97) the location text will change to BARRICADED.

DEATH RISK: HIGH

This represents the likelihood of your character dieing during her sleep. The text will change depending on the location you are in with OUTSIDE having a HIGH risk, UNBARRICADED having a LOW risk and BARRICADED having no risk or NONE. See section 11 on page 99 for more about sleeping.

7. Inventory










Throughout the game you will need to pick up items to stand any chance of survival. To pick up items simply move over them. If there is room in your inventory they will be picked and stored in your inventory. When your inventory becomes full a sound will play and  is displayed in the top-right of the screen. You must use or drop items before you can pick up any more.

Press I or F on the keyboard to open your inventory and display the cursor. You can still move your character while the inventory is open and the game is **not** paused. Left-click with the mouse cursor over an item to open its action menu. The action menu allows the clicked on item to be used or dropped by pointing to the desired action and left-clicking on it. All items are deleted when used. When items are dropped they are removed from inventory and placed near your character. You can re-collect dropped items later by moving over them. Some items have more options in their action menus. To close an items action menu click the CLOSE action in the menu or left-click on another item.

You can also left-click and hold down the mouse button to drag items around the inventory and change which slot they are stored in. Press I again to close the inventory. When your character dies the inventory will immediately be closed if it is open.

All collected items are retained when your character loses a life so you will not have to collect them again.

8. Items

Drink		Drink to decrease thirst
Food		Eat to decrease hunger
Medkit		Use to give a large increase to health
Bandage		Use to give a small increase to health
Barricade		Place to barricade a building so Rot cannot enter. Barricades can only be placed on spaces marked  by  . If a barricade cannot be placed a sound will play and the item will not be used. If the barricade was placed successfully the space will change to  which Rot cannot pass.
Wood		2 uses: <ul style="list-style-type: none"> • Place Fire to place a fire nearby and increase warmth. Warmth will only increase while near a placed fire. Placed fires will eventually burnout. If a fire cannot be placed a sound will play and the item will not be used. • Use as spear to convert the wood into a wooden spear and use it. When you collide with Rot now you will kill them. The spear will break after hitting 4 Rot. Wood used as a spear cannot be placed as a fire.

9. Enemies

Apart from simple survival, the only enemies you will face are Rot.




Very little is known about Rot. What are they? Is "Rot" their true name? Where do they come from? Why do they relentlessly attack all living creatures? These are a few of the questions that need answering. From their movements Rot are thought to be stupid and slow but when approached they will move towards their prey until they can no longer see it or they kill it. For this reason Rot should be avoided wherever possible, making sure to keep a safe distance. It has been discovered that if Rot see prey they turn red and immediately give chase. As soon as they lose sight of their prey they lose the red colour. The reason for this colour change is unknown.

Rot are also capable of striking fear in the hearts of nearby people. If you get close to a Rot, providing you can see them, they darken and cause your fear to increase. Fortunately, darkened Rot cannot cause additional fear after this increase. When you move out of sight, darkened Rot will become normal again enabling them to cause fear.

If you collide with a Rot they will hurt you, unless you have a spear equipped as described for the wood item on page 97. Your health will decrease and the playable character will become partially invisible for a few seconds. While partially invisible you can still move normally but cannot be hit. The Rot will continue to chase you so it is advisable to find somewhere to hide until they give up.

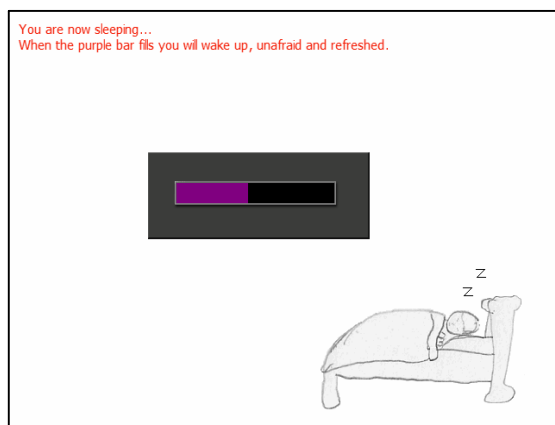
10. Other survivors

On your journey across the island you will encounter other humans  trying to survive the Rot. You can talk to these survivors to get survival advice.

When you get close to a survivor a yellow circle will appear around them. Press E on the keyboard to start a conversation. To continue the conversation press E again. If the survivor has no more to say the conversation will be ended when you pressed E.

If you want you can end a conversation immediately and open the in-game menu (see section 12 on page 100) by pressing Esc.

11. Sleeping



Pressing R on the keyboard will make the playable character sleep. Sleeping is important for your survival as it is the only way to reduce fear. You must choose where you sleep wisely as not everywhere is safe. The risk of being killed in your sleep is displayed in the DEATH RISK at the top of the screen (see page 96). Your current location is displayed in the LOCATION (see


page 95). Changing location from outside to inside a building will lower the risk of being killed in your sleep. To ensure there is no chance of being killed during your sleep you must barricade the building. See the barricade item on page 97 for how to do this.


When a building is barricaded the LOCATION will change to BARRICADED and you can sleep without risk of death. Note that some buildings require more than one barricade item to be placed before they become barricaded. Such buildings have multiple spaces marked by




While sleeping you have no control of the playable character. Control will be restored when the purple bar fills the black area.

If you die during sleep a message will be displayed to explain this. Click OK to return to the game but with 1 life lost.

Provided you do not die during sleep, any placed barricades  in that area will decay once you wake. They decay in stages depending on the number of times you have slept. After one

sleep they remain unchanged and strong. After two sleeps they change to . After three

sleeps they become weak and change to . When you wake from a fourth sleep any weak barricades collapse and will no longer keep out Rot. A sound will indicate one or more barricades have collapsed.

12. Pausing the game and the in-game menu

By pressing Esc on the keyboard during gameplay and while sleeping, you can pause the game. This will display the in-game menu as shown below.



To unpause the game and continue playing simply press the Esc key again or left-click on the CONTINUE GAME button. This menu also lets you save the current game (see section 13 on page 100), get help on what to do next and about playing the game (see section 20 on page 102), load a saved game (see section 16 on page 101), and quit the current game.

Clicking on QUIT TO MAIN MENU will display a question asking you to confirm if you want to quit or not. If you click YES you will quit the current game and return to the main menu (see section 4 on page 93). Any unsaved progress will be lost. If you clicked NO you will be returned to the in-game menu from where you can continue the game.

13. Saving a game

You can save the current game to continue later by following these steps.

1. Press Esc on the keyboard to open the in-game menu.
2. Click SAVE GAME.
3. Enter a name for the saved game. If you do not wish to save the game type CANCEL.
4. If the name is invalid a message will be displayed. Click OK to repeat step 3 and enter a new name.
5. If a save game with that name already exists you will be prompted whether or not you wish to overwrite it. If it did not exist go to step 7.
6. If you click YES go to step 7. If you click NO you will have to repeat from step 3 entering a new name.
7. The game has been saved and a conformation message will be displayed. The message displays the name game has been saved as. **Do not forget this name as you will need it to resume that game.** Click OK to continue.

14. The game over screen



When all your lives are lost, either from your health reaching 0 or dying during sleep, you have lost the game. The game over screen will be displayed. This lets you return to the main menu of the game (see section 4 on page 93), load a saved game (see section 16 on page 101) and exit the game. You cannot continue a game ending in game over. To continue playing you must click RETURN

TO MAIN MENU and then start a new game from the first area by clicking NEW GAME.

15. Exiting the game during gameplay

To exit the game during gameplay and return to the operating system follow these steps:

1. Open the in-game menu by pressing Esc on the keyboard.
2. Click QUIT TO MAIN MENU.
3. Click YES. If you click NO you will return to the in-game menu without exiting the game.
4. On the main menu click the EXIT button.

If you are playing the game in windowed mode you can also exit the game by clicking the X in the top-right corner. Click YES to exit or NO to continue playing.

16. Resuming a saved game

To continue a game you have saved, follow these steps from the main menu:

1. Click the LOAD GAME button.
2. Enter the name of the saved game you wish to load. If you do not wish to load a game type CANCEL.
3. Click OK.
4. If no save game with the entered name exists a message will be displayed and no game will be loaded. Click OK to return to the main menu.
5. If a save game with the name you entered exists in the "Save_games" folder, that game will be loaded. After clicking OK on a conformation message the in-game menu is shown (see section 12 on page 100). From here you can click CONTINUE GAME to continue playing the loaded game.

Alternatively, if you are already playing a game you can load a previous game by pressing Esc on the keyboard to open the in-game menu. Click LOAD GAME and follow the stages described above.

17. Taking screenshots

Press F12 at any point in the game to take a screenshot. A small picture of a camera will appear for a short time near the top right corner of the screen to show that the screenshot has been taken successfully. Screenshots are saved in bitmap files in the "Screenshots" folder contained in the folder Hunger was installed. If the "Screenshots" folder is accidentally deleted screenshots it will be re-created the next time the game is run. You can take as many screenshots as you like.

18. Deleting a saved game

All saved games are saved in the "Save_games" folder contained in the folder Hunger was installed. By default Hunger is installed in the Program Files folder on your hard disk. Simply find the file named

`<name>.sav`

where `<name>` is replaced by the name of the save game you want to delete, and delete it.

19. Viewing the game credits

To view who a list of people involved in creating Hunger click the CREDITS button on the main menu. To return to the main menu (described in section 4 on page 93) click the RETURN TO MAIN MENU button or press Esc on the keyboard.

20. Getting help

Clicking on a HELP button or pressing F1 while playing the game will open a help window giving a summary of this document. While the help window is open the game is paused. To continue playing close the window by pressing the X button in the top right corner of the window. If you still have questions you can e-mail them to the game's developer at tdc1@btinternet.com

21. The Hunger website

For more information about the game and how it was produced, please visit the Hunger website at: <http://uk.geocities.com/tdc1@btinternet.com/Hunger/Hunger.html>

22. Known problems

During a game it is possible for your character to become stuck on wall objects. If this happens press the **Home** key on the keyboard. This should nudge your character off the object but if you find you are still stuck change direction by pressing one of the movement controls and try again. Keep changing direction and pressing the **Home** key until you can move about freely.

If this does not work after several attempts you will have to exit the game and restart it. Any progress you have made will be lost.

Maintenance Document

Contents

OVERVIEW	104
CORRECTIVE MAINTENANCE.....	104
ADAPTIVE MAINTENANCE.....	105
COMMENTARY.....	106
REFERENCES	107

Overview

This document explains how maintenance to the project should be performed. There are two types of maintenance to consider:

- 1) Corrective maintenance - Ensuring any bugs discovered after the game has been released are fixed. This is done by releasing patches and new versions of the game.
- 2) Adaptive maintenance - Ensuring the game runs correctly on new versions of operating systems or hardware.

The "Commentary" section gives the standard used for commenting source code.

Corrective Maintenance

Corrective maintenance - Ensuring any bugs discovered after the game has been released are fixed. This is done by releasing patches and new versions of the game.

The game was created using the development environment Game Maker [1]. Irrespective of the environment used it is almost impossible to create a game without errors and bugs that are only detected after it was released. Users will often report these errors to the game developer. Changes must be made to the game software in order to eliminate the error. If a user did report an error, it is important that they give the developer a copy of the **game_errors** file in the folder Hunger was installed. This will contain a record of the text message that was generated by the error. This message will give the section of code that caused the error. All code used in the game software is contained in individual objects or in scripts. Objects have been labelled with the prefix **obj_** and scripts with the prefix **scr_** so they can be easily identified. The section contained in the error message will contain one of these labels.

Once the section of code has been identified it is often useful to insert **show_message** function calls into the program. These stop the game running until the OK button is clicked. Identifying which messages do display and which do not can be a good indication of exactly where the error occurs. Note that a line number recorded in the error message will only indicate where the error was detected and not necessarily where the error occurred.

By running the game in debug mode the values in global and instance variables can be determined. The times they change value can also be seen. This is done by entering the name of the required variable in the watch list. Using the debug feature simple expressions can be evaluated during run-time and their result displayed. For example, entering the expression **instance_exists(obj_rot)** will display 1 if there are any Rot objects present in that level and 0 if there are none. For more information about how to use debug mode see reference [1].

Once the error has been fixed the software can be re-released as a patch [3]. The easiest way to do this would be to make sure the new version of the game will overwrite the previous version when it is installed. The new version is distributed and replaces the previous version.

Adaptive Maintenance

Adaptive maintenance - Ensuring the game runs correctly on new versions of operating systems or hardware.

The game was created using the development environment Game Maker [1]. This is documented to produce games that are portable to any hardware and any version of the Microsoft Windows operating system. However, it is possible that some types of hardware or versions of operating system may stop the game executing as expected. The problem here is that there are so many different types of hardware it is almost impossible to ensure the game runs correctly on each one. The most feasible way to test whether the game works on a particular set of hardware is to release it to users. The same applies for operating systems.

If users find any errors with the game they will hopefully report these errors to the game developer. The developer can repeat the procedure leading to the error on different hardware and on the operating system the user was running. If this produced an error the problem is likely a bug in the games software (see "Corrective Maintenance"). If, however, no error was produced the problem is caused by the different hardware or operating system.

Assuming the reported error was a problem caused by different hardware or a different operating system there are three ways the problem might be solved.

1. The most drastic option is to access the Game Maker website [1] and check the latest version of the Game Maker software. If the latest version is the same as the version used to create the game this option will not work. If the latest version of Game Maker is different, download that version. The majority of versions of Game Maker are backwards compatible with games made in older versions. Update the game, or part of it which may be causing the problem, with any new functions if available. Test the game to ensure it works as before the update. If this reveals no problems the new version of the game can be released.
2. Carry out more testing to narrow down the part of the game software in which the problem is caused. Once the general area of the problem is known the Game Maker community can be asked if they have had similar problems. This can be done by posting the section of code on the Game Maker forum [2]. Copyright over the section of code posted would need to be stated to protect it from being illegally copied and used. If this did not solve the problem, the creator of Game Maker can be e-mailed as there may be a bug in Game Maker.
3. Many standard functions in Game Maker have equivalents that perform the same task in different ways. Carry out more testing on the game to narrow down the part of the game software in which the problem is caused. Once the area of the problem is known these equivalent functions can be used in place of functions that could be causing the problem.

Commentary

The source code has been commented to explain any difficult sections. Any code written after the game is released should also be commented in this way. The date the code was changed, a brief summary of the change or changes made and the problem those changes solve should be placed at the top of the section of code as a comment.

If the task or tasks performed by a script is changed, its new task must be summarised and placed at the top of the script as a comment. Any new scripts written must also be commented in this way.

References

- [1] Title: Game Maker by Mark Overmars
Website URL: <http://www.gamemaker.nl/>
- [2] Title: Game Maker Community (Powered by Invision Power Board)
Website URL: <http://forums.gamemaker.nl/>
- [3] Title: Wikipedia - Patch (computing)
Website URL: [http://en.wikipedia.org/wiki/Patch_\(computing\)](http://en.wikipedia.org/wiki/Patch_(computing))

Acknowledgements

During this project I have made use of work done by others. I have used their work with their permission and in agreement with the copyright they have set.

1. Some of the sprites, pictures and other images were drawn by Kat Davis.
2. Version 6.1 and version 7.0 of the Game Maker development environment used to create the game were developed by Mark Overmars and YoYo games. Version 7.0 can be downloaded from: <http://www.gamemaker.nl/>
3. The Game Maker implementation of the player's inventory was adapted from the example by Nick Koza (tahnok100).
His example can be found at: <http://www.tahnokgames.com/>
4. The music and midi files used in the game were created by Mathew Duguay of Zer0mus Midi co.
The website of Zer0mus Midi co. is located at: <http://www.freewebs.com/zer0mus/>
5. The background for the conversation screen and the code to create the textbox for conversations were created by J-Factor.
The Textbox 2.0 example used can be downloaded at:
<http://forums.gamemaker.nl/index.php?showtopic=130501>
6. Feedback on the development of the game was provided by several game testers.
These are: Kristoffer Marc Getchell, J Ross Nicoll, Angus Macdonald, Alex Lupu, Christopher Jackson