T-404-LOKA, Final Project

GUMMS

Bjarmi Árdal Bergsteinsson
Finnbogi Darri Guðmundsson
Guðný Björk Gunnarsdóttir
Óskar Ögri Birgisson
Þorgeir Auðunn Karlsson

2013
BSc Computer Science

Instructor: Birgir Kaldal Kristmannsson
Examiner: Sigrún Eva Ármannsdóttir

Contents

1 Introduction

GUMMS (Generic Universal Massive Multiplayer System) is a set of game rules and tools, which allow creation of game worlds quickly and inexpensively. GUMMS covers online, strategy and roleplaying game elements.

Today's game development and publishing is becoming increasingly expensive and time consuming. The need for a set of game rules that are designed with reusability and cost savings in mind is apparent. The mantra "content is king" is often sung and GUMMS will allow developers more time on content and less on redesigning and problem solving. With GUMMS the end-user benefits by getting to experience different stories and worlds rather than having to learn a new set of rules every time they play.

The problem is big and the solution cannot be found within the time frame given for this project. The goal is therefore to obtain funding to fully develop GUMMS.

What is being introduced here is the first corner piece of a much bigger puzzle. The team has built the turn-based combat subsystem for GUMMS. This is just a small taste of things to come that should whet the appetite and interest of investors and our future fan base. It showcases what online strategy roleplaying games could and should be. The team has created a workable alpha-demo of this subsystem and skinned it to be a multiplayer arena game with a simple goal: choose your team, enter the field of battle and kill the enemy to win. Investors and fans alike are able to see what the general gist of the idea is through the alpha-demo prism.

2 The Project

The project goal was to create a working, aesthetically pleasing and playable alpha-demo prototype to exhibit the game system. In this prototype the user gets the feeling of how the game mechanics work and what online strategy roleplaying combat is about. It gives the team's business developers the ammo to sell the idea to investors. In addition, it works as a platform for the technical and gaming development team to work on their ideas.

As stated before, GUMMS is a set of rules and tools for creating games. What it is not, is a graphical engine. The team feels that the problem of in-game graphics has been solved sufficiently well and opts to use a third party engine for their work.

The game will run on Windows XP, Windows 7 and Windows 8. Future releases will include Mac and Linux, as well as some or all of the mobile platforms.

2.1 The Team and Facilities

The team consists of five students studying Computer Science at Reykjavík University (RU). They are:

Bjarmi Árdal Bergsteinsson	bjarmi11@ru.is
Finnbogi Darri Guðmundsson	finnbogi11@ru.is
Guðný Björk Gunnarsdóttir	gudnyg11@ru.is
Óskar Ögri Birgisson	oskarb11@ru.is
Þorgeir Auðunn Karlsson	thorgeirk11@ru.is

The project is intellectual property owned by the team. The team has not yet established a legal entity nor do they have premises to work from and consequently the facilities were provided by RU.

2.2 The Game

In the game the player takes charge of a team. The team consists of one to three characters selected from one of four races. The team is placed in an arena where it is pitched against another team which is put together by another player. The game can be played in two modes: hot-seat mode (two players play on one computer), or network mode (two players over a network connection). In both modes the goal is the same; destroy the other team before it destroys you.

2.2.1 Feature list

The following is a list of the main features for the project. In the chapter *The Rule Set* the features are explained in further detail.

1. Initiative System

   -Characters' initiative attributes determines the order of each turn.

   -Actions cost pips and each character has a limited supply of pips to spend during its turn

2. Turn-based Combat System

   -Ranged Combat

   -Melee Combat

   -Spell Casting

   -Total freedom in placement and rotation in the arena (vector-based, not grid-based)

3. Server-side Checking

   -"Uncheatable" server state, using a new technology in how a game state is kept on the server

   Server sends information to the clients on a "need to know" basis.

4. Character Options

   -Ten main attributes

   -Secondary attributes derived from Main Attributes to flesh out the character

   -Skill-based character creation system

5. Intuitive User Interface (UI)

   -Right click Graphical User Interface (GUI) with contextual actions for the player to choose from

2.2.2 The Rule Set

Two conventions to simplify the user experience are used in this rule set:

1. A higher number is always better
2. No such thing as negative bonus or positive penalties (some games do this, it confuses the player).

Team Selection
A player's team can be selected from one of four races: Humans, Robots, Skeletons or Teddy Bears. Each race has its own set of specialties - e.g. Skeletons are more skilled in ranged combat and wear heavy armor, while Robots have lighter armor and are more mobile. Each race has a leader with higher skills, attributes and a magical weapon, as well as a magician that has access to magical actions.

The Map
One of the things that make this project different from other turn based strategy and roleplaying games is that the map is not gridded. Most games divide the playing field into squares, like a chess board. GUMMS uses a points and vector approach.

Characters
A character is a collection of attributes, skills, actions, traits, and equipment. A character works as the conduit for the player's will inside the game. All actions the player performs inside the game are done through a character.

Unlike most strategy and roleplaying games, GUMMS uses a skill-based character system as opposed to a class-based one. Instead of characters being of a certain class (such as Warrior or Mage) the system allows you to mix and match what skills the character has. This allows players to create whatever character they want from the components of the system. This feature is not shown properly in the version of the sub-system presented, as the character progression system was not implemented due to time constraints. However the basics for the system are included.

Attributes
Attributes describe a character's innate capabilities and are used to evaluate the strengths and weaknesses in broad terms. There are ten main attributes. There is the Pip Count that determines how many actions a character can take during its turn. The other nine come in three categories: mental, physical, and social. Each category has Agility, Health, and Strength attributes, which can be seen in Table 1. Secondary attributes are calculated from the main attributes. These include Initiative, Movement, and Hit Points for each of the three categories.

Table 1: Main Attributes

Attribute	Mental	Physical	Social
Strength	Mental Strength (MS) *"Willpower"*	Physical Strength (PS) *"Strength"*	Social Strength (SS) *"Force of Character"*
Agility	Mental Agility (MA) *"Intelligence"*	Physical Agility (PA) *"Dexterity"*	Social Agility (SA) *"Charisma"*
Health	Mental Health (MH) *"Sanity"*	Physical Health (PH) *"Health"*	Social Health (SH) *"Self-esteem"*
Pips			

Skills
Skills describe a particular knowledge. Each skill is represented by a number (the skill level). Skills, in conjunction with equipment, dictate what actions the character can take. Having the right skill, the right equipment, and the right target, opens up an action that the character can perform on that target.

Actions
There are two types of actions: voluntary and involuntary. A voluntary action is something that the player decides to do, whereas an involuntary action is a reaction to something that happens

to the character when it is not its turn. An example of a voluntary action is attacking and moving, while an involuntary action is dodging or parrying an incoming attack. In order to take an involuntary action the character must have pips left to pay for the action.

The probability of an action being successful is calculated using an equation that considers attributes, skills, equipment, and the type of the action. Each action costs a certain amount of pips to perform, allowing the player to strategize around pip spending.

Once an action is performed, the game calculates the level of success (LOS) using the above mentioned equation. The LOS is then used to calculate the effect of the action on the playing pieces. Some actions are contested actions, i.e. the target of the action has a chance of contesting the effect. The system determines the LOS of the opposing action and uses that to augment the LOS of the contested action.

Combat
A combat turn is a unit of time inside the game. During a turn each character performs a set of actions. The order in which characters take actions during a turn is decided by the initiative attribute of the characters. The system allows spells that increase or decrease a character's initiative score so that it can move in the turn order.

Attack actions come in two types: melee and ranged. A melee attack is an attack with a hand-held weapon onto a target that is within reach. A ranged attack is an attack with a ranged weapon, e.g. bows and throwing knives, that is within the weapon's range. Both types of attacks are modified by the difference in size between the attacker and the defender, making it harder to hit smaller targets. Ranged attacks are modified by distance between the target and the performer. A weapon's bulk rating is used to further adjust the chance of a hit if the target is within the minimum range or reach of a weapon. All attack actions dictate how they can be defended against. If the target has an involuntary defense action that matches the attack, he gets to perform that action. If more than one defensive action is possible, the game randomly selects one defense action. The game quickly makes a comparison between the LOS of the attack and the LOS of the defense in order to decide who won. If the defender scores higher the attack was a failure. Otherwise, the defender's level of success is used to augment the attack action LOS.

Doing Damage
If an attack is successful the weapons' damage output is matched with the action's to calculate the total damage. Damage can be done to any of the attributes of a character. The LOS of the attack is used to modify the damage output. Furthermore, some weapons (especially magic ones) will have damage outputs that are always calculated, even if it does not match with the damage output of the attack action.

If the target is wearing armor, it can resist some or all of the damage output. If damage gets through the armor, the target receives a damage trait.

Traits
Traits are modifiers to a character's state and come in three types: permanent, ticking, and ticking-permanent. Permanent traits are traits that go onto a character and stay until an action

gets performed to take them away, e.g. wounds stay on the character until it is healed. Ticking traits go onto a character for a set amount of time and then disappear on their own accord, such as a spell that gives the character increased physical strength for three turns. Ticking-permanent traits stack over time and are not removed. They must be removed by an action like permanent traits. For example, if a character gets a ticking-permanent trait for 5 Physical Hit Points (PHP) over three turns, he would lose 5 points of PHP for the next three turns and at the end have a total of 15 PHP permanent damage. Healing can stop ticking-permanent traits before they reach their full potential, e.g. stop bleeding by applying first aid.

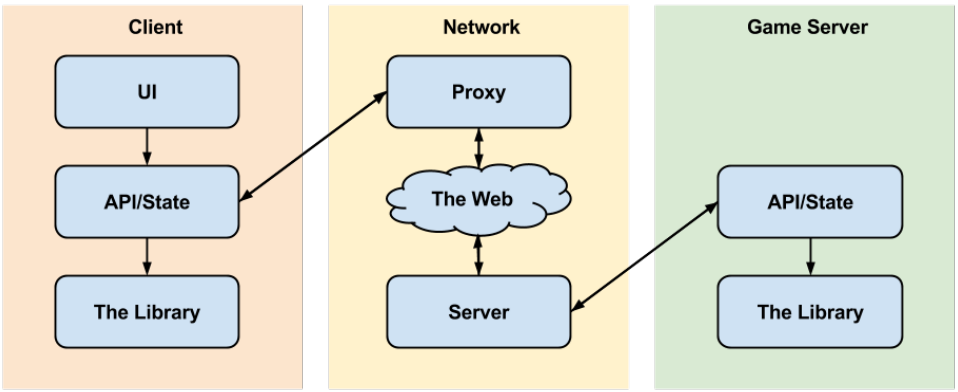
Hit Points
Hit points describe the character's damage resistance to certain types of damage. There are three types of hit points in GUMMS: mental, physical and social. Losing all hit points of a given type has an in-game effect. If a character loses all his Physical Hit Points, he dies. If a character loses all Mental Hit Points, he becomes insane (acts like he's dead). Social Hit Point damage causes the character to lose morale and his skill levels go down.

Magic
Magic actions are a way of passing traits onto targets without the use of an attack. There are two types of magic; buff and debuff. Buffing is when a magician casts a spell on a target for a positive effect, e.g. healing or giving more strength to a team member. Debuffing is the opposite, e.g. lowering an enemy's Physical Agility so he is less likely to hit during combat and moves slower. Each debuffing magic action has a set of attributes that are used to limit or stop the effect of the magic. The magic user must roll a contested skill roll versus the character's attributes. Each spell has a Mental Hit Point (MHP) cost attached to it, so a magician must pay MHP to cast spells.

2.3 System Architecture

The system is first split into three layers: client, network, and game server. The client is then divided into three subparts: UI, API/State, and Library. The game server has two parts: API/State and Library. The two layers are connected by the network layer that consists of the proxy and server. The code for the library is the same in the client and the game server.

Figure 1: Layers



2.4 The Library

The Library is a set of generic game rules and can be thought of as the rule book for the game. The library is unaware of the state of the game at any point, but is called by either the client or server State/API to handle actions inside the game.

2.5 Client

The client is the layer that the player interacts with. Depending on the mode of play, the client either manages the game state by itself (hot-seat mode) or uses the proxy in the network layer to communicate with the game server for updates to the game state.

2.5.1 UI Layer

This is the presentation layer that implements the user interface for the game. The layer communicates with the rules and game state through the client side API/State. It creates requests to the API/State and reports back any changes to the player once the request has been handled.

2.5.2 The API/State Layer

The function of the API/State on the client is twofold. First, it takes care of the game state in hot-seat games. Second, it handles request/reply cycles from and to the UI by using the proxy from the network layer.

2.6 Game Server

In a multiplayer game, the game server takes care of the game state, and handles request and replies from the client.

2.6.1 The API

The API layer encapsulates the game rules and game state. It can be considered as a service to the game world. It handles all requests by the clients by using the server part of the network layer.

2.7 The Network Layer

The original idea for the network layer was to use SignalR[1] between the server and the client. Problems arose with SignalR and Unity, as Unity does not allow anything above .NET3.5. The solution was to complicate the network layer by introducing a proxy in-between the network and the UI.

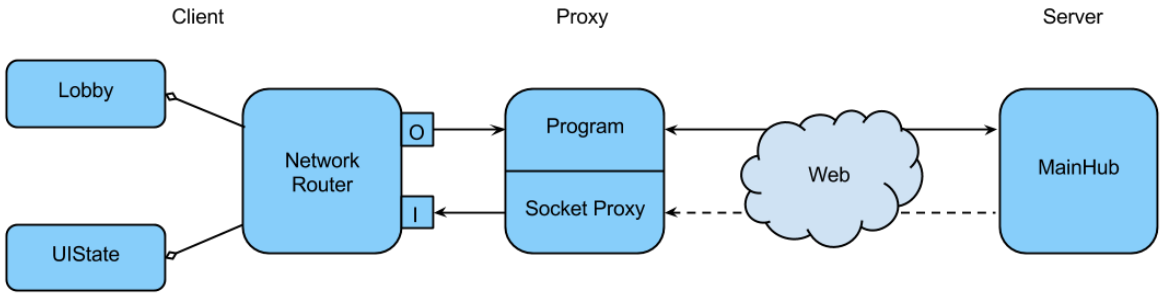
2.7.1 The Proxy

When a multiplayer game is started on a client it turns on the proxy and asks the server to setup a game. The API/State in the client communicates with the proxy through a socket connection. The proxy translates the message into a format the server can understand. The server passes the message onto the API/State in the game server for handling. The server then replies to the proxy, and the proxy translates the message back to the client side API/State.

2.7.2 Server

SignalR was chosen as the main architecture for the server hierarchy. SignalR is different from other solutions in that it is not HTML oriented. It does not parse out HTML or other web related content. The client uses HTTP to connect and authentication is done natively by SignalR. SignalR can call back to the client and carry out its requests.

Figure 2: Network Layer



3 Artifacts

The following artifacts are on the attached CD. None of the code is intended for public disclosure since development of the project will continue after the close of this assignment.

- **Final report:** This document, containing a detailed description of the project and its process.
- **Retrospectives:** A document containing the retrospectives for each release.
- **Project Journal:** A document containing all hours worked and the distribution over each release.
- **Risk Analysis:** A document containing the risk analysis for the project and how the risks were handled.
- **Source Code:** All of the code for the project.
- **System Manual:** A document that describes step-by-step how to set up the project and run it.
- **User Manual:** A document explaining how the game is played and describes what each operation is doing and its consequence.
- **User Testing Results:** A short report on how user testing was conducted and the results.

4 Project Management

4.1 Methodology

The team opted to use Kanban methodology for the project. Kanban was chosen because of its flexibility and it allowed easy slotting of schoolwork around the project. It helped that some team members had experience working with the Kanban methodology. The project would be broken into release cycles. During each cycle a story and its tasks would be in one of the following states: accepted, doing or done. Some XP Programming tenants were used such as pair programming and test driven development.

At project launch the team had a requirement catching session. Out of that session came 68 stories. These were listed into a prioritized project backlog. During the session a release plan was formulated and the stories that fit into the plan were placed into releases. There were seven releases planned, six full length releases and a shorter bug-fixing release at the conclusion of development. A code freeze was planned to occur during or after release six.

Before every release there was a release meeting where the release backlog was created, tasked, and time slotted. The team opted for a story point system for measuring work.

During release cycles the team had three standup meetings per week. During those meetings issues were raised, progress reported, and potential roadblocks brought up.

After each release there was a Post Mortem meeting where the team discussed how the release had gone. Each team member wrote down things he or she wanted more of, less of, or the same of. The team then voted on one thing to work on for the next release.

4.2 Roles

Although the project was very democratically run and everybody has a say in all decisions, there was a rough partitioning of roles, which were as follows:

- Bjarmi Árdal Bergsteinsson

  Graphics and GUI programming.

- Finnbogi Darri Guðmundsson

  Game mechanics programming.

- Guðný Björk Gunnarsdóttir

  Mathematics and administration.

- Óskar Ögri Birgisson

  Game logic and design. Product Owner.

- Þorgeir Auðunn Karlsson

  Technical leader.

Having said that, everyone did work on all parts of the game, these titles simply reflect who was accountable for certain aspects of the final product.

4.3 Tools

Unity3D was chosen as a graphic engine for the project as it met the project scope and aspirations. Additionally, the group had prior knowledge of its strengths and limitations. It is easy to use and gives professional results even though it is free.

The programming language of choice was C#. This went well with the graphic engine choice as Unity3D allows C# scripting. Visual Studio 2012 and 2013 were used as the integrated development environment.

Project management was done through Microsoft's Team Foundation Server (TFS). TFS integrates seamlessly with Visual Studio and allowed the team to project manage and version control through a single portal. It features a smart web UI that allows good overview of stories, tasks, burndown charts, etc. During the process the team was asked to hand in more detailed time management reports than TFS supplies out of the box, so time management to an Excel spreadsheet.

The web server is hosted on Microsoft's Azure system. It was chosen since it has a free student version and provides cloud services that have a 100% up time.

For graphics the team used Blender and GIMP. The Unity Asset store was scoured thoroughly to find free art assets. The team ended up buying a small pack of animations to use.

Dropbox and Google Drive were used for managing data. All technical reports were written in Latex. All calculations of hours worked and burndown charts were done in Excel. TeamViewer, TeamSpeak and Facebook were used for communication between team members.

4.4 Tests

The project was test driven. Most units that could be unit tested were.

4.4.1 Unit Tests

Much emphasis was placed on unit tests and primitive functionality was tested thoroughly. A total of 101 unit tests were written. Of those tests, 59 tested the library layer and 42 tested the API layer. Visual Studio allows for code coverage analysis of the tests. The library layer reaches 71.54% coverage and the API layer reaches 76.52%.

The build process automatically runs all unit tests and reports their outcome.

4.4.2 System Tests

Due to the complexity of various functionalities of the system it was important to test the efficiency and correctness of major actions in the system involving several components at a time. There were several tests made for the action process and for the movement of characters.

4.4.3 User Tests

User tests were carried out during release 7. During testing a developer sat down with a tester, gave him a list of tasks to perform, and recorded his reactions. At the end of the test the tester

was left to play with the product for as long as he wanted. He then got asked a few open ended questions.

The task list was as follows:

1. Start a multiplayer game
2. Move a character
3. Do a melee attack
4. Swap weapons
5. Do a ranged attack
6. Do Healing magic
7. Start a Hot-Seat game
8. Find Information on a character's Skills
9. Find information on a character's Equipment
10. Find information on current pips
11. Find Information on what type of armor you have

During each item the developer timed the task and asked questions at the end of it.

1. How was the experience?
2. What would you change?
3. What did you like?
4. Any other notes?

The open ended questions asked were:

1. What did you think about the look of the game
2. How did you like / dislike the controls of the game?
3. What are our next logical steps for the product?

Conclusions
Overall the user testing went very well and the testers had a positive experience with the product. The main comment from testers had to do with the GUI and how it could be improved. For example, all testers asked for a feature where you could see the maximum walking distance of the currently selected character. The range indicators for the weapons should be explained better as well.

A more detailed report can be found on the attached CD, the filename is *User Testing Results*.

5 The Progress

The initial release plan, with named releases, can be seen in Table 2. During development of the product, 44 of the initial 68 stories got finished. Early in the process it was clear that not all stories would be completed in time. A decision was made to ignore stories which applied to inventory, hit locations, missions, and experience progression, although these stories will be implemented in the future. The backlogs can be seen in Tables 4 and 5 in the appendix.

The team selected the stories for each release in unison and recorded their selection in TFS. More weight was put on releases 1, 4 and 5 since the features chosen for those releases were considered to be more important and/or time consuming than the others.

The retrospectives for each release can be viewed in more detail in the *Retrospective* document on the CD.

Table 2: Release Plan

Release	Name	Focus
1	Running Man	Movement, Setup
2	Hammersmashed Face	Attack System, Map
3	Thunderdome	Ranged and Melee Combat, Items
4	They Came From... Behind	Cover, Network Basics
5	The Great Divide	Two Player
6	It's All About The Team	Squad Selection
7	Setting The Stage	Game Mode

5.1 The Releases

In the following chapters you can see a release-by-release breakdown of the progress.

The Burndown charts consist of the work history over a specific period of time. The blue line corresponds to a guideline that is a linear decrease in the number of story points while the red line corresponds to the actual progress. The green bars show the hours that were worked each day. The x-axis shows the days and the y-axis the story points.

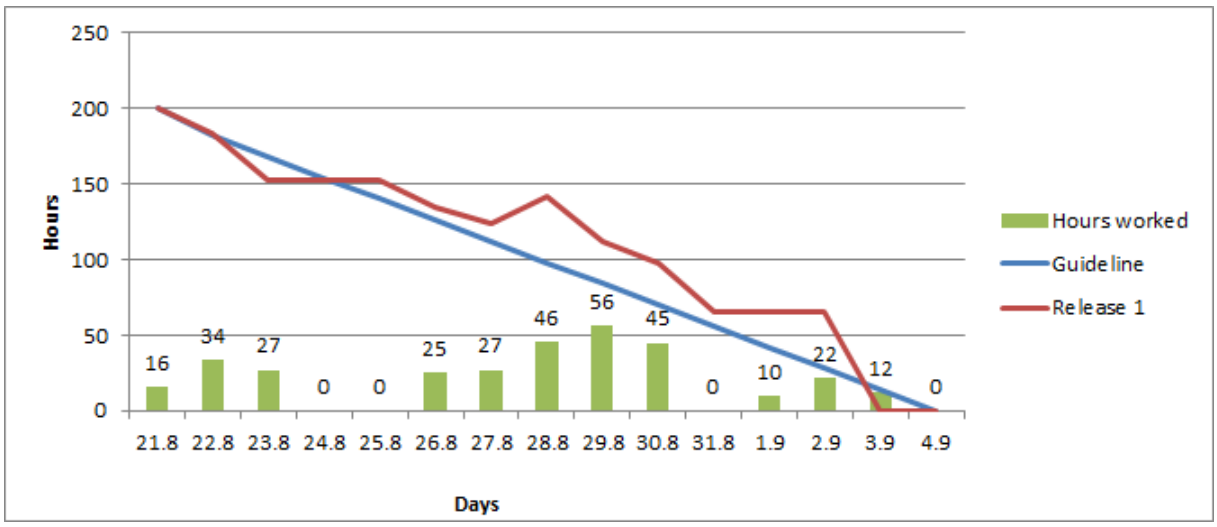
5.1.1 Release 1 - Running Man

During the first release the ground work for what was to come had to be laid down. The need for a good logging mechanism was addressed and the basics of the interaction with screen obects through Unity3D was sorted out. The main goal of the release was to see a guy walk on-screen. The stories assigned to this release were as follows:

1. Action Basics
2. Character Basics
3. Interact With Objects / Characters
4. Logging System
5. Map Basics
6. Moving the Character on the Map
7. Non-Tiled Combat Map / State
8. Priority Queue
9. Progress Report no. 1
10. Setup Project and Connect to Unity
11. Skip Action

The release was finished a few days ahead of schedule since it was arranged with respect to other school projects. During the release it came clear that there were no school projects scheduled for the first two weeks. Consequently there was a lot of free time for working on the project. The last remaining days were used for detailed code review, retrospective, and planning the next release.

Figure 3: Burndown Chart



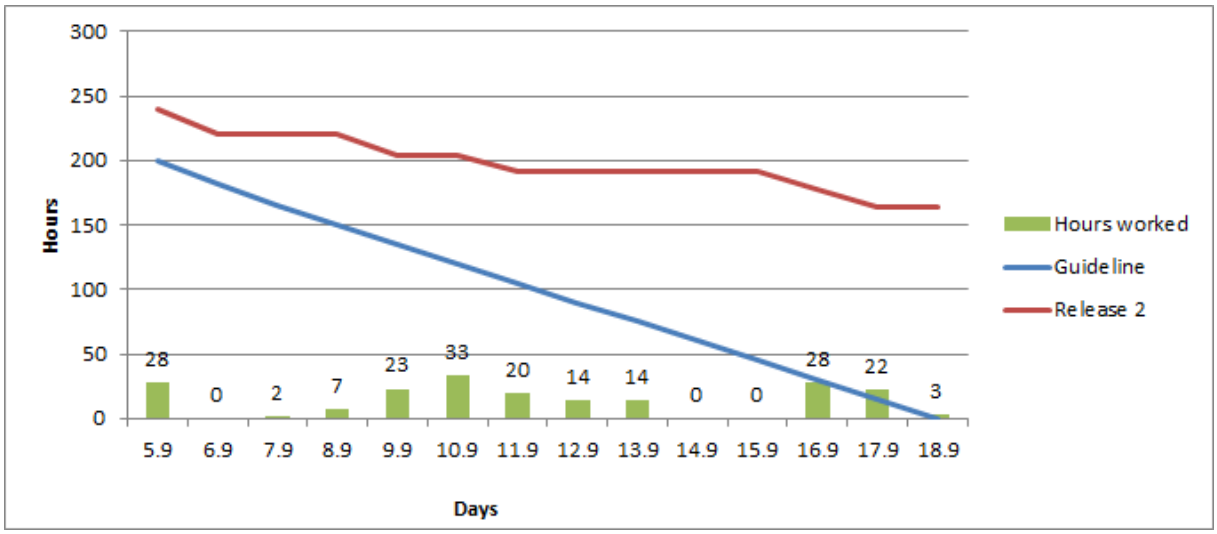
5.1.2 Release 2 - Hammersmashed Face

The release's main focus was to add combat and map functionality and a basic GUI. Taking the previous release's retrospective into consideration, Unity3D learning was added to the release so everyone could do basic things with the engine. Some team members had little or no experience in using unit tests, so that was added as well. The stories "Non-tiled Combat Map / State", "Map Basics" and "Skip Action" were to be continued in this release. The stories assigned to this release were as follows:

1. GUI for Character
2. Inspect Character
3. Learn Unit Testing
4. Learn Unity
5. Map Basics
6. Non-tiled Combat Map / State
7. Path finding
8. Skip Action
9. Terminate Action
10. Trait Basics

At the end of the release it was clear that the work hours had been underestimated. The main factors being overconfidence from the last release and a sharp increase in school projects. The features: "Terminating a Character" and "Trait Basics" were not completed and were moved over to the next release. A decision was made to push back the "Path Finding" feature until later in the process.

Figure 4: Burndown Chart



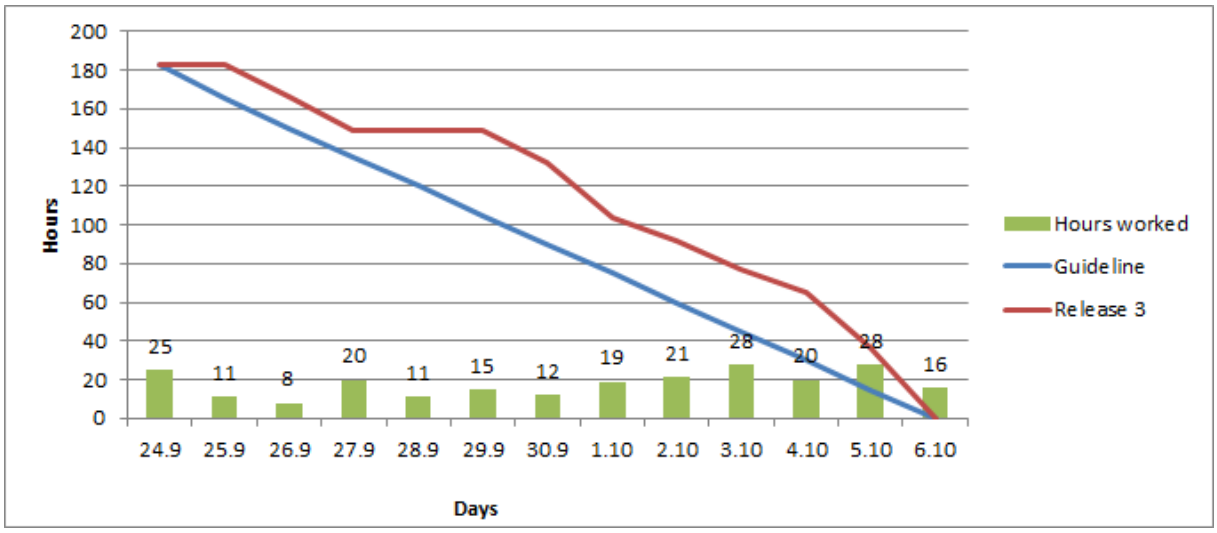
5.1.3 Release 3 - Thunderdome

The main focus for this release was on adding more combat actions and getting the trait system sorted out. Items were also added to the game: weapons and armors. This included implementing a damage system that connected items, characters, and actions to the trait system. There were two features from the previous release that were moved over to this release: "Terminate Character" and "Trait Basics". The stories assigned to this release were as follows:

1. Defense Action
2. GUI for Action
3. Item Basics
4. Melee Attack
5. Non-Combat Actions
6. Passive Traits
7. Ranged Attack
8. Terminate Action
9. Ticking Traits
10. Trait Basics

This release included a lot of refactoring; unused code was removed and a lot of cleanup was done in the UI layer. The action system was rewritten because of major changes to the attribute structure, including trait based manipulation of the attributes.

Figure 5: Burndown Chart



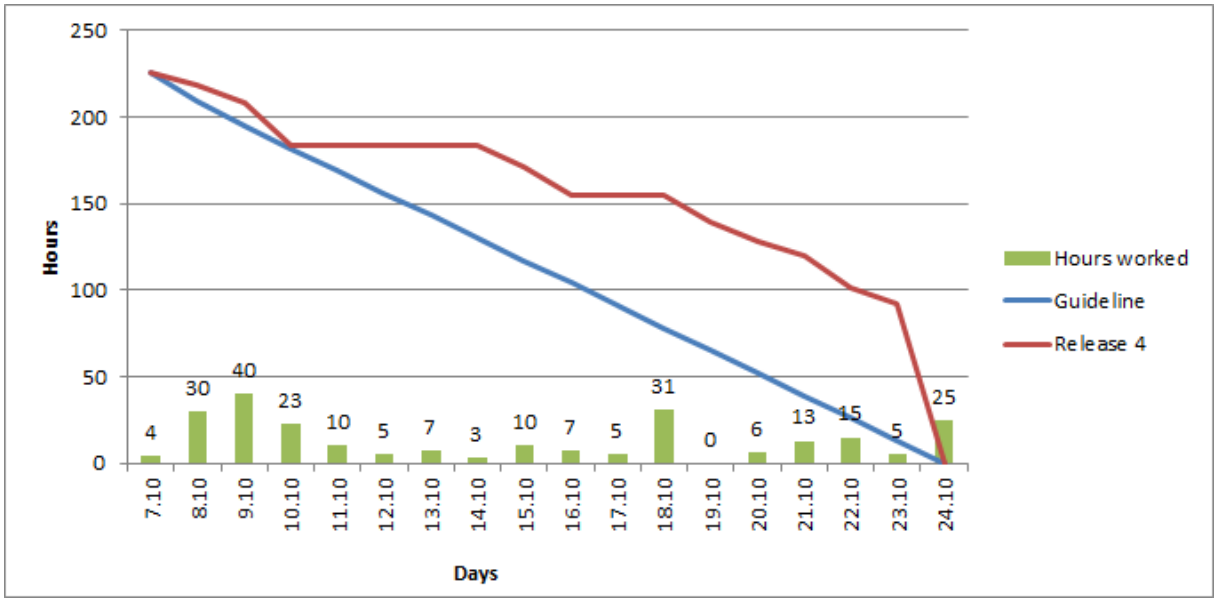
5.1.4 Release 4 - They Came From... Behind

The main focus of this release was to research how best to facilitate gameplay over the internet. The skill and pip systems were also added, making the character able to develop and gain skills in various fields. The stories assigned to this release were as follows:

1. Action Provider
2. Multiplayer Mode
3. Pip Action System
4. Progress Report no. 2
5. Skill Basics
6. System Cleanup

This release contained more workload than the others since the winter break was at the beginning of the release. A lot of research was done on Azure Server set-up which lowered the risk factor of the respective risk in the *Risk Analysis* document. A lot of effort was put into system cleanup and refactoring the UI. The release was extended by three days due to midterm exams and other school work getting in the way.

Figure 6: Burndown Chart



5.1.5 Release 5 - The Great Divide

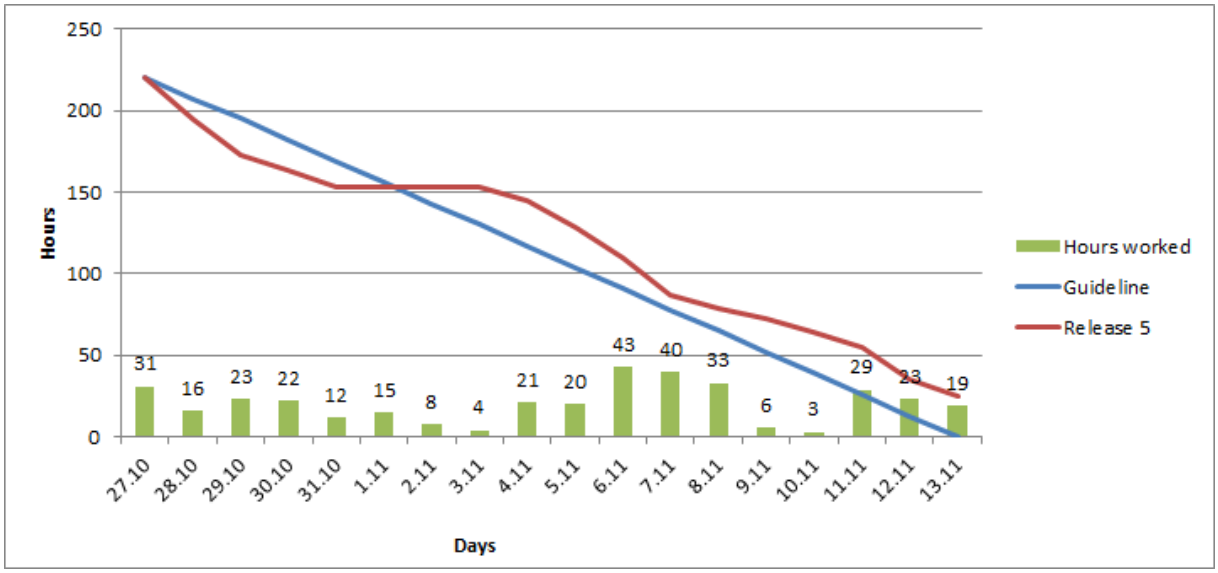
In the previous release a lot of research and preparation was done to be able to set up an Azure server. The emphasis for this release was to make multiplayer possible and to be able to place characters in cover. Since the project had become quite large and complicated it needed some refactoring. The stories "System Cleanup" and "Multiplayer Mode" were to be continued in this release. The stories assigned to this release were as follows:

1. Content Creation
2. Multiplayer Mode
3. Place Characters in Cover
4. System Cleanup

After trying the WebAPI, MVC5 and SignalR server architectures which are all based on HTTP transportation with Json as their data format, and weighing their pros and cons, SignalR was the most appropriate solution. However, because the Unity3D client uses a lower .NET version that does not support the SignalR client, a proxy was introduced to deliver messages between the client and the server.

The cover system proved difficult to implement. The team was unable to find the cause of an error and thus moved the story to the next release since this feature is critical to game play.

Figure 7: Burndown Chart



5.1.6 Release 6 - It's All About The Team

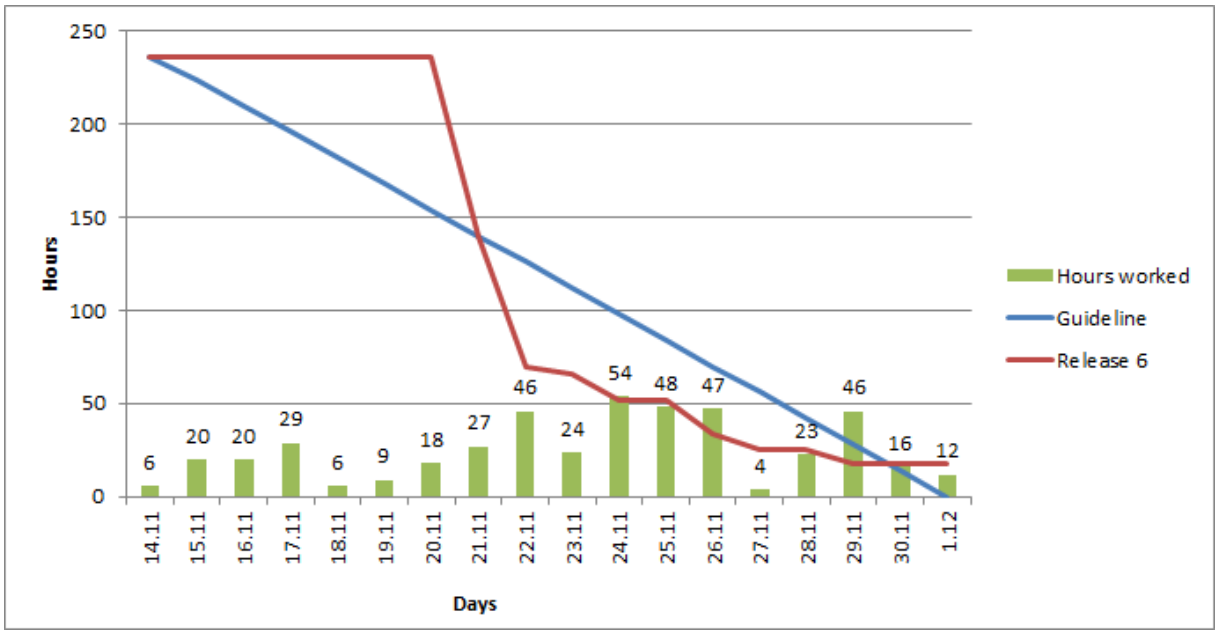
The main focus was to continue making the game more realistic by visualizing the map and populating it with obstacles. A lobby was added to the start of the game where the user could select a team and then start a game. Additionally, path finding, which was put on hold in release 2, was re-introduced. Work continued on the cover system and the error was resolved. The stories assigned to this release were as follows:

1. Heal a Character
2. Map Tools
3. Mission Map
4. Multiplayer Mode
5. Path Finding
6. Place Characters into Cover
7. Progress Report no. 3
8. Select Characters for Combat
9. Single Player Mode
10. Squad Selection
11. Team Factory

Since there was a very short period of time left for implementing any new features there would be a feature freeze in the next release, focusing mainly on visualization of the game and fixing minor issues.

For the first few days there wasn't much progress since path finding proved to be more difficult in implementation than expected.

Figure 8: Burndown Chart



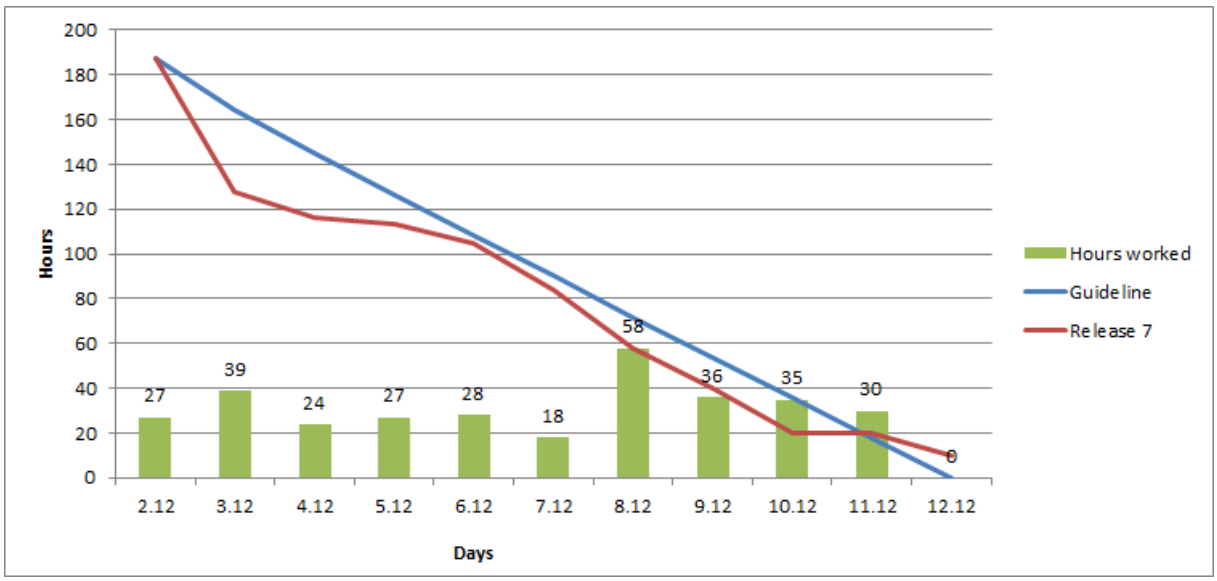
5.1.7 Release 7 - Setting The Stage

For this release there was a feature freeze, focusing only on fixing and improving features that had not been fully implemented, including the path finding from the previous release. There were only two stories for this release, which were as follows:

- Finishing Touches
- Path Finding

At the beginning of the release all members sat down together and listed the things that needed fixing. The list can be found in the *Retrospective* document. This list was then set as tasks for the story "Finishing Touches". It included improving the UI and the Lobby, finishing the remaining animations for actions, and camera configurations, to name a few. Finally, all documents for the final hand-in were to be finished.

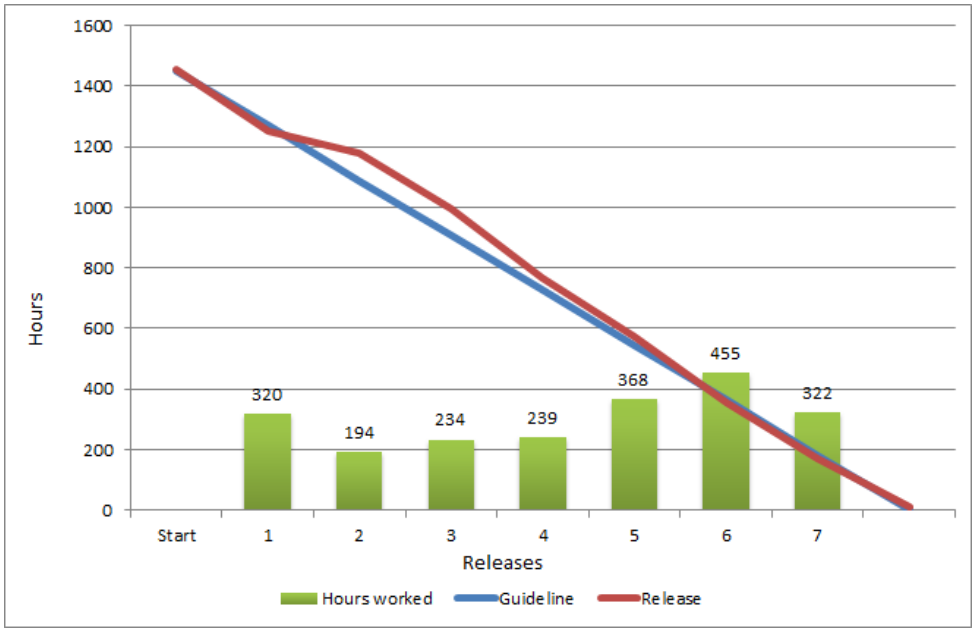
Figure 9: Burndown Chart



5.2 Release Summary

Figure 10 below shows the overview of the overall process. As can be seen the curve is almost linear. All main features that were considered to be most important were finished. Although it would have been nice to implement more features, time was limited and thus features that were considered unessential had to be ruled out.

Figure 10: Release Burndown



Hours spent on the entire project can be viewed in Table 3 below. A more detailed overview of hours worked for each release can be seen in the *Time Journal* document on the CD where hours worked by each member was summarized for each release.

Table 3: Time Estimation: All Releases

Name	Estimated Work Hours	Actual Work Hours	Total
Release 1	250	320	160%
Release 2	200	194	97%
Release 3	200	234	117%
Release 4	225	239	120%
Release 5	225	368	184%
Release 6	200	455	228%
Release 7	200	322	161%
Total	1500	2120	141%

6 Summary

Overall work went well despite the fact that the scope of the project was extensive. The main reason the project went as well as it did was the team's enthusiasm and good planning. The fact that all team members have been friends and worked closely together over the past two years had a great impact on the success of the project as well.

Although some tasks had been over- or underestimated, it had little effect on the final result. There were a few setbacks along the way regarding path finding and the cover system, but they were eventually solved. The lesson to learn from that is not to panic even if a solution does not present itself right away, tenacity wins the day.

The team learned a lot during the process and gained confidence in their abilities to deliver a product like this. It also became apparent that it takes more than pure programming skills to finish a project like this. None of us could have done this alone, it was a team effort all the way.

If there is anything the team would have done differently, it was probably to spend more time researching solutions before diving in. With the timeframe of the project, this proved impossible but the team looks forward to having more time in the future. Perhaps scaling the project differently would have helped but the team is happy with the end product as it is.

The team would like to thank the University of Reykjavík for its assistance as well as Microsoft and Unity3D for their contribution in software and services. Hjalti Magnússon, our "landlord", should be mentioned as well. Thank you for your support, without the teamwould have been homeless.

7 Future Vision

The goal of the project was to develop a pre-alpha of the GUMMS system to present to investors. This project will serve as the launching pad for a game studio that produces the entire GUMMS system. GUMMS will be developed by King Coyote Studios (KCS). KCS consists of a team of software developers and business leaders who are currently formalizing the company.

KCS is a gaming studio that specializes in online, strategy, and role-playing games. Our vision is to change the industry through a new approach to the way games are developed and how players interact, and through a dynamic system that promotes direct involvement in development with infinite possibilities.

Our mission is to put gameplay first. Our team is dedicated to ensuring that all of our games are fun and engaging. We are not in the business of benchmarking hardware, we are in the business of building awesome games.

Our future vision is quickly becoming a reality as our first seed investment is coming early 2014. The company will be formalized in March 2014 and move into an office space on April 1st 2014. The work presented here will form the basis of a market ready release within 24 months.

We already have an established network of companies and people that have committed to assisting KCS. Work on securing a publishing and/or distribution deal is in progress with a world renowned company.

What was built during these three months forms only a small subpart of the GUMMS Turn Based Combat module and an even smaller part of the whole GUMMS concept. We will spend the next semester at Reykjavik University working on getting more features into the combat system.

Development of other modules will also begin. Objectives have been set to implement the Character Progression System and some of the 4X (explore, expand, exploit, and exterminate) game elements (Headquarters mode, where players can manage their in-game empire). Additionally, the team wants to developed a prototype of the item design and equipment building subsystems.

Through this project we realized a sudden need for concept art and improved graphics, animations, and sounds. As such we are currently searching for both a technical, and a concept artist.

The end products we are aiming for is a set of games, each skinned to a market appropriate theme (i.e. zombies, sci-fi, fantasy). In each of these games the player will take control of a team of characters, which he can manage in his style of play. If you prefer to be a builder, you build things for other people. If you fancy yourself a business guru, you find your corner of the market and exploit it. If you prefer combat, you train your characters and battle other players and non-player entities for fame and fortune. If you like diplomacy, you start your own in-game guild and make all the big decisions. The possibilities are endless. If you desire a new theme, you can move to another world, taking with you some of the things you've put into the first world, never having to start afresh again!

References

[1] Microsoft. Introduction to signalr, http://www.asp.net/signalr/overview/signalr-20/getting-started-with-signalr-20/introduction-to-signalr, 2013.

Appendices

A Product Backlog

Table 4: Product Backlog: Completed Stories

Title	State	Iteration Path	Closed Date
Action Basics	Done	Release 1	3.9.2013 17:51
Character Basics	Done	Release 1	3.9.2013 17:52
Interact With Objects / Characters	Done	Release 1	3.9.2013 18:05
Logging System	Done	Release 1	3.9.2013 17:53
Move the Character on the Map	Done	Release 1	3.9.2013 17:54
Priority Queue	Done	Release 1	3.9.2013 17:52
Progress Report no. 1	Done	Release 1	3.9.2013 17:50
Setup Project and Connect to Unity	Done	Release 1	29.8.2013 12:07
GUI for Character	Done	Release 2	22.9.2013 17:59
Inspect Character	Done	Release 2	22.9.2013 23:44
Learn Unit Testing	Done	Release 2	22.9.2013 17:59
Learn Unity	Done	Release 2	11.9.2013 16:57
Map Basics	Done	Release 2	22.9.2013 17:59
Non-Tiled Combat Map / State	Done	Release2	22.11.2013 05:01
Skip Action	Done	Release 2	22.9.2013 17:52
Unarmed Attack	Done	Release 2	22.9.2013 17:59
Defense Action	Done	Release 3	30.9.2013 21:04
GUI for Action	Done	Release 3	30.9.2013 17:25
Item Basics	Done	Release 3	4.10.2013 15:19
Melee Attack	Done	Release 3	25.9.2013 16:34
Non-Combat Actions	Done	Release 3	29.9.2013 04:31
Passive Traits	Done	Release 3	6.10.2013 16:38
Ranged Attack	Done	Release 3	30.9.2013 18:47
Terminate Action	Done	Release 3	23.9.2013 14:58
Ticking Traits	Done	Release 3	6.10.2013 16:36
Trait Basics	Done	Release 3	4.10.2013 17:42
Action Provider	Done	Release 4	24.10.2013 16:09
Pip Action System	Done	Release 4	24.10.2013 22:06
Progress Report no. 2	Done	Release 4	21.10.2013 19:17
Skill Basics	Done	Release 4	24.10.2013 15:45
Content Creation	Done	Release 5	13.11.2013 14:18
System Cleanup	Done	Release 5	13.11.2013 14:19
Heal a Character	Done	Release 6	27.11.2013 23:07
Map Tools	Done	Release 6	26.11.2013 03:56
Mission Map	Done	Release 6	23.11.2013 18:21
Multi Player Mode	Done	Release 6	1.12.2013 14:13
Place Characters Into Cover	Done	Release 6	21.11.2013 21:19
Progress Report no. 3	Done	Release 6	27.11.2013 01:28
Select Characters For Combat	Done	Release 6	14.11.2013 10:09
Single Player Mode	Done	Release 6	21.11.2013 22:57
Squad Selection	Done	Release 6	26.11.2013 21:46
Team Factory	Done	Release 6	22.11.2013 01:33
Finishing Touches	Done	Release 7	12.11.2013 08:17
Path Finding	Done	Release 7	10.12.2013 17:08

Table 5: Product Backlog: Uncompleted Stories

Title	State	Iteration Path	Closed Date
Area of Effect Attacks			
Capture the Flag			
Change Stance			
Create and Terminate Items			
Create Missions Procedurally			
Decide The Amount of Experience Characters Receive			
Destructible Environment			
Discard Items			
Drop Items			
Experience Points			
Experience Status			
Group Inventory			
Inventory Basics			
Item Generation Suite			
Keep the Fort			
Loot Basics			
Loot Bodies / Containers / Map Locations			
Minimap			
Mission Generation Suite			
RTS Like Movement For Units Until Combat			
Rush The Defense			
Select a Hit Location For An Attack			
Skill Specialties			
Spend Experience Points			

B Bug List

1. Action bar disappears in multiplayer
2. Animations for Arrows/ Missiles
3. Animations for defense
4. Animations for defensive stance
5. Animations for hits
6. Animations for magic
7. Balance Teams
8. Balance Weapons
9. Be able to zoom out more
10. Connect the armor to characters
11. Documentation
12. Free camera rotation
13. Fix queue so it handles initiative changes
14. HP can exceed the maximum
15. Lobby - Improve character sheet
16. Lobby - Improve Flow
17. Lobby - Logo
18. Lobby - Proxy client starts when game starts
19. Lobby - Report that user is in queue for a game
20. Lobby - Show server status
21. Lobby - Tool Tips
22. Magic actions need to work
23. Mental HP loss has no effect
24. Ranged attacks are not working
25. Reload after ranged attack
26. Skill list looks funky
27. Sync problem during multiplayer
28. Trait panel doesn't update when it's open
29. UI - Equipment panel needs to display all equipment
30. UI - Flexible display for pips
31. UI - Floating text needs more content
32. UI - Name of characters over icon in action bar
33. UI - Remove Scrolling from team overview
34. UI - Scale UI with resolution
35. UI - Tool tips
36. UI - Waiting for turn message
37. UI - Walking distance
38. UI - Weapon range
39. Weapon bulk needs ATH
40. Weapon range does not upgrade on weapon switch
41. 'You lost' message when you win