

Sprint 1 Report

*Team Report*

*Group: Team B*

*Project: Crossing Streams*

*Course: Cosc 470 - Software Engineering*

*Submitted by: Billy Spelchan for Team B (Nov 1st)*

Website: <http://10.1.144.90> (need vpn access)

Game Server: <http://10.1.144.91> (need vpn access)

# Project Summary:

We are creating a Multiplayer Online Rogue-like Top-Down Shooter (MORTDS) with the game play style based off of games such as Binding of Isaac, Enter the Gungeon, and Gauntlet. The key distinguishing feature of this project is the multiplayer features.

The project can be broken into five basic problem areas: the web front-end, web back-end, database, game client, and game server.

The web front-end, aka website, would be where users of the game would login or register, learn about the game, download the game client, view news, participate in forums, and possibly participate in tournaments. This would have to be easy to navigate and a pleasant experience for the user as it has an impact on the game experience as a whole.

The web back-end would be the databases and software necessary to run the website. It has to be secure as player registration information will be kept on the database.

The game client would have to be able to also log into the site as it would share login information. As the game is multiplayer, it is important that all players are using the same version of the client so patching facilities would have to be built into the client. The client would have to communicate with the game server to get map information as well as know the location of other players when they are in the same room. Basic text chat would probably have to be in the game.

The game server would handle the world and positions of the players and items within the world. As the game would have procedurally generated dungeons, a large amount of effort will be required to generate these in such a way that the game preserves a consistent difficulty curve. The server would also have to track the player's behaviour so that it would be possible to block players for bad behaviour. Some cheat detection, such as making sure actions are within normal ranges, would also be a nice option to prevent cheating. While the game server would run on our server, for better performance it is likely that we will also want the ability for clients to act as a server for LAN games, though the LAN version would not have any of the cheat monitoring otherwise hackers would be able to make undetectable cheat bots.

What we are trying to accomplish with this project is creating a game that fits the niche of other similar style games, but introduces a true multiplayer aspect to the game. Currently all of the other games of this genre only work as single-player games. When the system is done a user will be able to download our client, update to the most recent build of the client and play the game. The game flow consists of clearing rooms of enemies in a dungeon floor, and descending the dungeon; killing bosses on every floor until the final floor has been reached. Along the way players will be able to upgrade their players with random power-ups, and the more they player plays the game the more power-ups they will unlock for future runs through the dungeon. Below is a happy-path diagram of how the user will go from first installation to winning the game.



# Problem statements:

Here is our initial “final” version of the problem statement:

The client wants to create a Multiplayer Online Rogue-like Top-Down Shooter (MORTDS) that leverages concepts from games such as Binding of Isaac, Enter the Gungeon and Gauntlet. This project’s timeline consists of two semesters. The first semesters will consist of proof of concept and the second will be focused on adding and refining game features such as procedural map generation. The game should be distributable to clients in an online environment.

The problem involves the creation of a set of interconnected systems that combine to form our game experience. The team will be challenged by the interlinking of the different technologies, organizing group tasks, and finding relevant assets for the project. A second problem will be in discovering what makes the genre compelling and making sure our game builds upon the compelling elements while adding multi-player features. Our constraints will be time, communication, scheduling/availability and the difficulty curve of learning the utilized technology.

When you combine the team enthusiasm for the project with our client's observation that many Isaac players want such a game, there is certainly potential with the project. Looking at the numbers for Binding of Isaac, (http://store.steampowered.com/app/113200/ accessed September 23) The game has been given 37,882 positive reviews while only having 1,508 negative reviews with 39,235 steam purchases. Enter the Gungeon (http://store.steampowered.com/app/311690/ last accessed September 23) had 5,413 positive reviews and 625 negative reviews with 5709 Steam purchases. These numbers indicate that there is a potential market for such a game and that a well created game would be able to recoup its development costs.

The users of this game platform should be able to interact with other users in a variety of ways. The backend should be robust and expandable to leverage the unique integration of the platform allowing for future development. Administrators should have the ability to review player behavior and make corrective actions to maintain the overall user experience.

(NOT ready to start) Review and compare your original problem statement from the beginning of the project against your final releases – this will have to wait until the end of the semester (end of semester)

Are there any changes to the desired deliverables? (modifications, exclusions, additions)

Sprint 1:

No changes this sprint (Needs update each sprint)

Sprint 2:

Sprint 3:

# Team Profile:

Team roles. Daniel is our unity expert. Ben is Go and web framework experienced. Billy is web developer and game developer. Corey is our database expert. Marc has user interface design experience. All members hope to gain knowledge of procedural generation, and working on larger teams.

Roles and responsibility:

Corey: Product Owner (is contact person between team and client), Developer.

Ben: Scrum Master when James is not available, Developer.

Billy: Assigned leader. Developer.

Marc: Developer.

Daniel : Developer.

# 

# 

# Development Process:

We are using the Agile Scrum process to complete this project. The Agile Scrum process relies on a self-organizing, cross-functional team, which we believe to be. Using the Scrum methodology, our project is comprised of a series of sprints, timeboxed to 2-weeks per sprint over the course of this project.

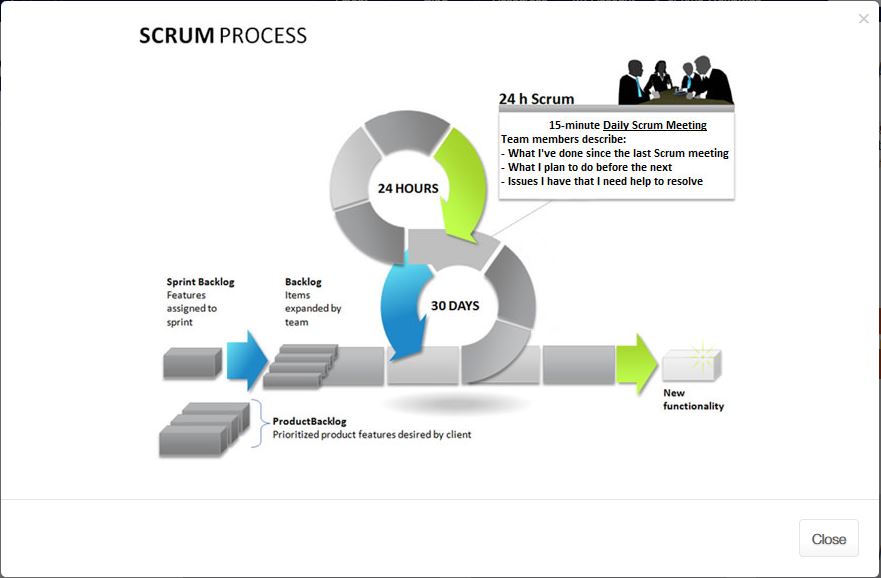
"Scrum methodology advocates for a planning meeting at the start of the sprint, where team members figure out how many items they can commit to, and then create a sprint backlog – a list of the tasks to perform during the sprint."

We have several Scrum meetings throughout the week (Tuesdays, Wednesdays, Thursdays and Saturdays), during this time, we discuss what has been done, what will be done next and any problems that we face.

"The Scrum model sees daily scrums as a way to synchronize the work of team members as they discuss the work of the sprint."

At the end of a sprint, we will conduct a sprint review and retrospective, where we demonstrate the project functionality and reflect on the sprint that has ended while identifying ways to improve.

We hope that using the Agile Scrum process, we will increase the quality of deliverables, adapt better to change and be more in control of the project as a whole.



*Source: https://www.cprime.com/resources/what-is-agile-what-is-scrum/*

System Architecture:

The development platform used consists of the following:

JIRA for handling the backlog and sprints

* <http://cs-oracle.okanagan.bc.ca:8088/>
* Board (CrossingStreams)

GitHub for hosting our repository

* Documentation (https://github.com/CoreyFrank/CrossingDocuments)
* Project (<https://github.com/CoreyFrank/CrossingStreams>)

Unity game engine

* Gameplay and game features

Jenkins ( NOT ready to start)

* Versioning and

<NOT ready to start - other tools used, such as Jenkins>

Our method of communication consists of 4 meetings a week as follows:

Tuesday 9:30 am - Weekly in person (Stand up) - To be changed

Wednesday 8:30 am - Weekly Lab class SCRUM and planning meeting for 3 hours

Thursday 6:30 pm, Evening SCRUM and additional planning

Saturday 1:00 pm - (Online) Skype meeting for SCRUM with additional planning

In addition to our meetings, we have set up a Slack channel (crossingstreams.slack.com) for our project as well as having email for all the members. Email should be sent to all members of the team even if they are not going to be participating in that particular aspect of the project in order to make sure that the team are all on the same page.

# 

# 

# Glossary of terms:

Agile - Refers to ta group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

Binding of Isaac – A popular game featuring procedurally generated levels and top-down shooting. While the theme of the game is controversial, the game play mechanics is the aspect of this game that we are focusing on.

C# - Microsoft’s alternative to the Java programming language. Like Java, it is an attempt to tame the C++ beast while adding garbage collection. It is the Scripting language used by Unity.

Continuous Integration - The process of regularly merging working copies from multiple developers into one up-to-date, comprehensive build.

Enter the Gungeon - A popular, top-down shooter wherein players journey through procedurally generated dungeons in search of a legendary gun. This game features a number of mechanics that we will be drawing inspiration from.

GoLang – Google’s attempt to tame the C++ beast while adding concurrency.

Gauntlet - A classic, top-down, hack and slash fantasy game noted for being one of the first multiplayer dungeon games. Featured four player multiplayer gaming on a single arcade machine.

Jenkins - An open source automation server that can handle build processes, continuous integration services, and testing. Jenkins conveniently can be submitted files or pull from a repository and run all services on a constant schedule and return a comprehensive result alerting developers of any issues.

Multiplayer – A game in which more than one player is playing in the same game world at the same time with the ability to interact with the other players who are playing the game.

Patcher - An add-on application that ensures that the client has the latest published version of the core application. If there is a newer version than the one installed, then it will download the new version for the user.

Permadeath or Permanent Death - When a player dies in the game and reaches a game over state then their progress is wiped or reset to some base value and they must start the game anew. There may be outside progress meters that remain but all match progress is reset.

PHP – PHP Hypertext Preprocessor. A server-side scripting language that merges HTML tags with a C-like programming language for generating dynamic HTML pages.

Procedural Generation – The technique where aspects of a game, such as the layout of the levels, is generated using a seed value resulting in vast replay-ability as each seed will in a different experience.

Rogue-Like - A game sub-genre defined by the key traits of procedurally generated dungeons, item collection, and permanent death.

Scrum - An Agile framework(subset of Agile) used to manage software development projects.

Slack - A private, free chat room style instant messaging service. Private team rooms can be created to facilitate team member communication and rooms can be broken into sub-rooms for more topic oriented discussion as well as one on one private messages between users.

Sprint - A regular, repeatable work cycle during which development/work is done and made ready to be reviewed.

Top-Down Shooter – A game with an overhead perspective where the predominant activity is shooting at enemies.

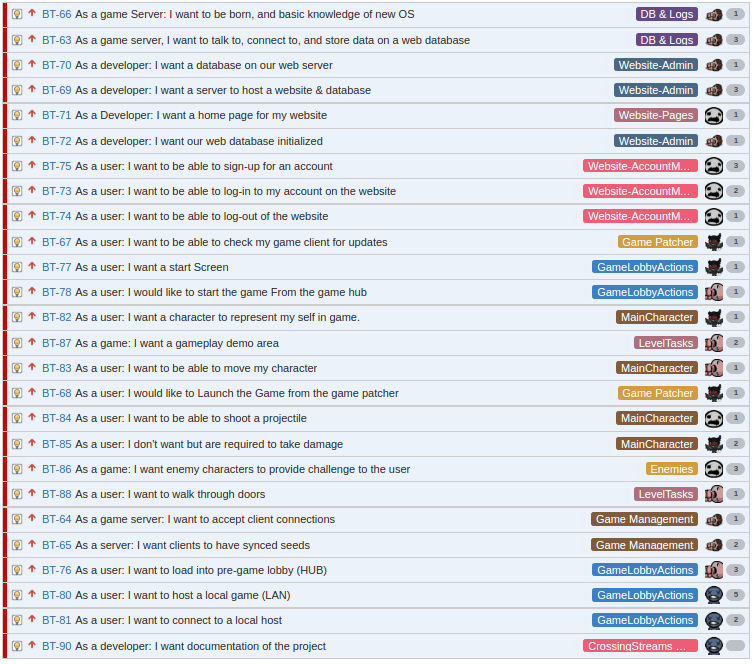
Unity – A popular game engine with no licensing fees until $100,000/year has been earned from the products produced with the product. The engine handles the basic functionality of the game allowing developers to focus on creating the content using C# as the scripting language.

# 

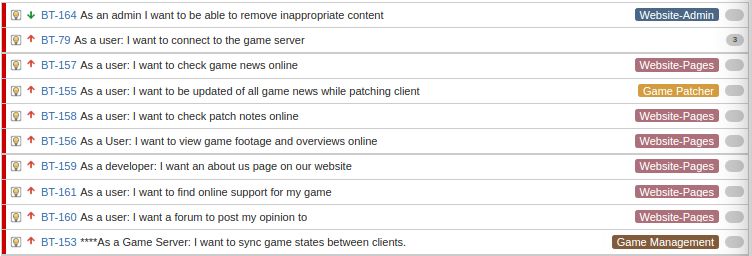
# Requirement gathering and specifications:

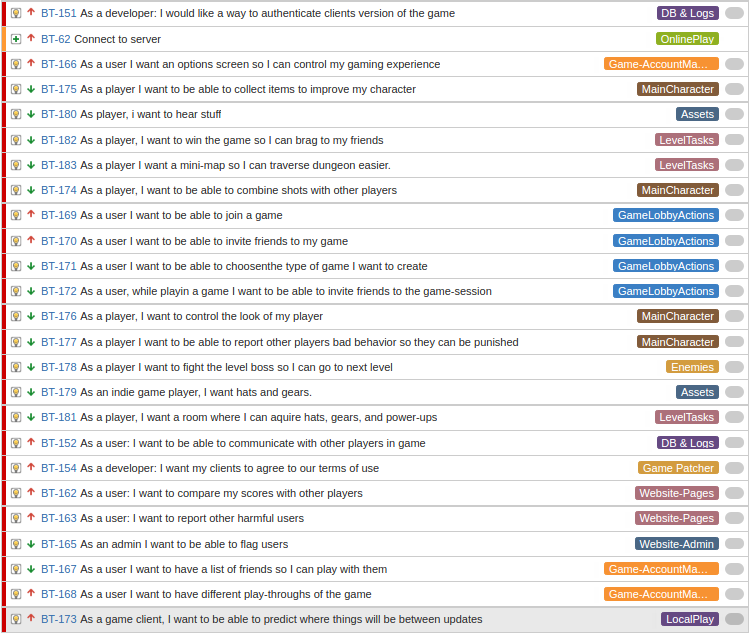
(Everyone)

(NOT ready to start- in progres) Product backlog (of the entire project)



The below stories have not been estimated yet.





(NOT ready to start) Functional requirement / Specification:

(NOT ready to start) Non Functional requirement / specification <6.1.iii for long list of items>

Specific Requirements (interface, hardware, communication interface):

Server Stacks:

Server_Stack_Diagram.png

Web Server:

IP: 10.1.144.90

DNS: COSC470DCentOS-2016.cis.okanagan.bc.ca

OS: CentOS Linux release 7.2.1511 (Core)

Web Server software - Apache (http://10.1.144.90/)

/var/www/html (directory root for website)

- index.php is main landing page

Database - Postgresql (10.1.144.90:5432)

Web Panel - Webmin (http://10.1.144.90:10000/)

Game Application Server:

IP: 10.1.144.91

DNS: COSC470ACentOS-2016.cis.okanagan.bc.ca

OS: CentOS Linux release 7.2.1511 (Core)

Crossing Streams Game Server:

IP/Port: 10.1.144.91:25565

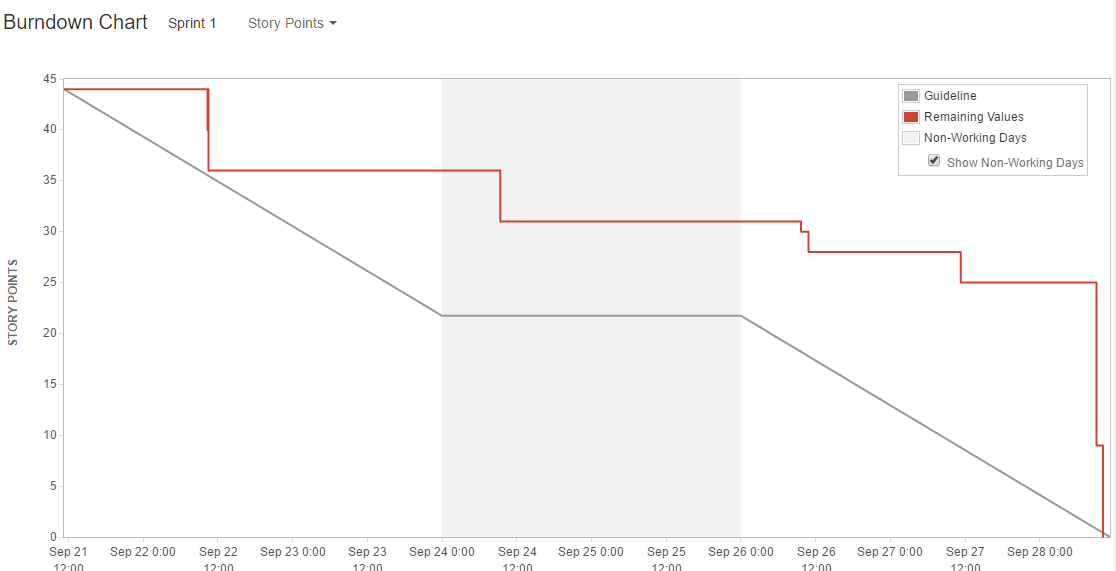
go version go1.7 linux/amd64

# 

# 

# Agile development planning

Sprint 0 - The Jira Tutorial (our data from JIRA is gone, we cannot see before our sprint 1)

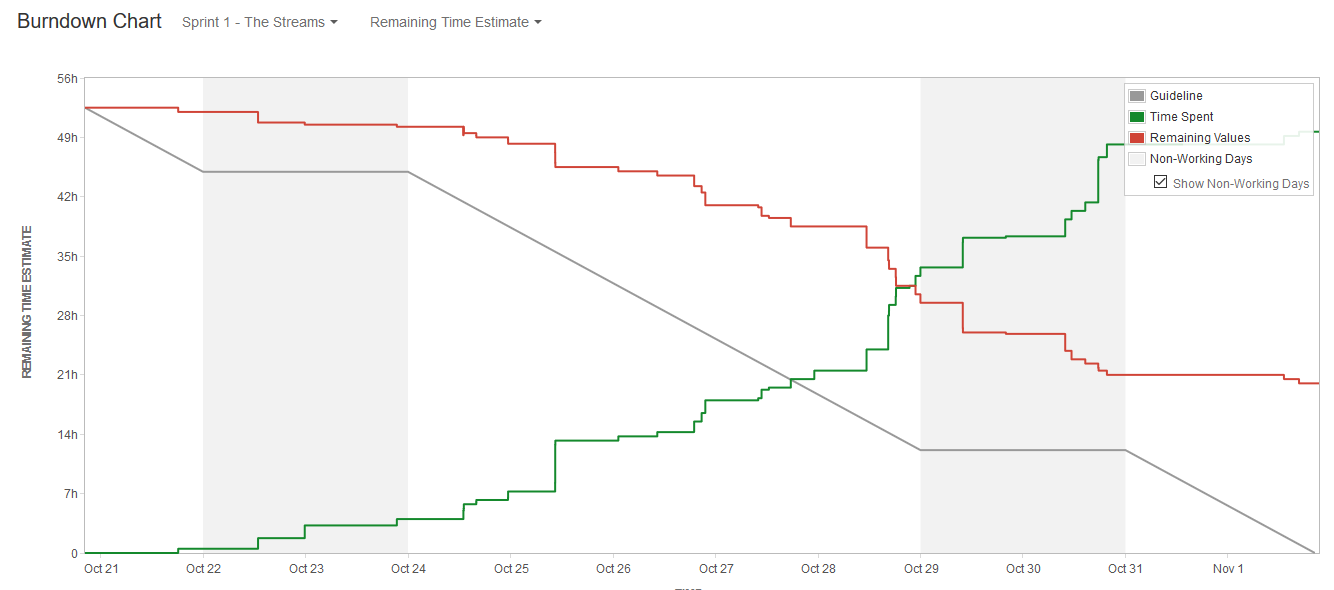


Sprint 1 - The Streams

(NOT ready to start) (sprint roadmap and planning for each sprint)

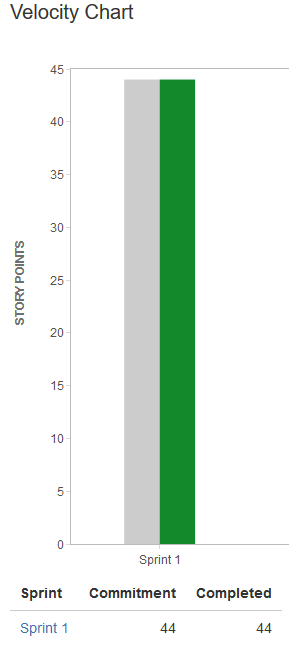
Burn down chart history from Sprint 0 to final with lessons learned from each sprint

Below is our burndown for sprint 1, (at 9:26PM tuesday Nov.1) As far a burndowns go yes it doesn’t show we finished, but some tasks in JIRA were not defined properly, and there were some underestimates made for some tasks, as well as time not logged while working with research and tutorials.



Sprint velocity and hours. With summaries of user points and hours per sprint

Our sprint Velocity for sprint 1 can be seen in the chart below, it shows we committed to 44 story points and we completed 44 story points, though there is still more to do that hasn’t been captured in JIRA, For the most part our user story estimations have been quite accurate. Though we need to refine our criteria more.



Risk tracking table for each sprint: Can be found in our project documentation, under our risk management plan.

(NOT ready to start) Summary of Retrospectives from each sprint (not done yet, haven’t finished and haven’t received client feedback, perhaps next time the client demo meetings & retrospective could be done before the deadline for the documentation of our work.)

SCRUM meeting notes for each sprint (at least 2 per sprint)

Saturday Oct. 29 Meeting

What we've done:

Corey: Patcher work, not finished (will upload tonight)

Was going to do JIRA Tasks (JIRA not working)

Marc: Committed code, from unity

Daniel: Front end work for client LAN

Billy: Finished AI components (haven't moved to done, JIRA issues)

Started website, (cannot get onto OC VPN)

Ben: Worked on setting up game servers, how to send and get data

(not here, was moving boxes, excused)

What we're working on (for tuesday)

Corey: Finish Patcher

Somewhat of a game demo for tuesday

JIRA tasks (Checking to see if it updates)

Marc: Starting on documents (team lead)

Template for MS Powerpoint

Daniel: POST and GET request

Documentation

Billy: Working on website, getting working on local machine, but needs VPN access

Ben: (Not here, but he is done almost all his tasks)

Issues:

Marc: JIRA

Billy: Cannot access OC - VPN

All: JIRA not working

Tuesday, Nov. 1 Meeting:

What we've done:

Corey - Patcher, working in unity, documentation

Ben - created game server and client server, request vpn access, helping team members commit,

working with Dan on how game server API works

Billy - website, download, all except password hash, final report

Marc - ---

Dan - hooking up servers (launching from client)

What we're working on (for tonight/tomorrow):

Corey - Demo, documentation & reports

Ben - individual report, team report

Billy - password hashing in php

Marc - individual/team report, powerpoint

Dan - individual report, team reports

Issues:

JIRA offline (weekend)

Billy - No VPN access to college

Ben - Moving on the weekend

Sprint 2 - The Crossing

(NOT ready to start) (sprint roadmap and planning for each sprint)

(NOT ready to start) Burn down chart history from Sprint 0 to final with lessons learned from each sprint

(NOT ready to start) Sprint velocity and hours. With summaries of user points and hours per sprint

(NOT ready to start) Risk tracking table for each sprint

(NOT ready to start) Summary of Retrospectives from each sprint

(NOT ready to start) SCRUM meeting notes for each sprint (at least 2 per sprint)

Sprint 3 - The Crossing of Streams

(NOT ready to start) (sprint roadmap and planning for each sprint)

(NOT ready to start) Burn down chart history from Sprint 0 to final with lessons learned from each sprint

(NOT ready to start) Sprint velocity and hours. With summaries of user points and hours per sprint

(NOT ready to start) Risk tracking table for each sprint

(NOT ready to start) Summary of Retrospectives from each sprint

(NOT ready to start) SCRUM meeting notes for each sprint (at least 2 per sprint)

# 

# 

# Conclusion

(What happened to document only what is necessary??????

(NOT ready to start) Summary of your project results

(NOT ready to start) Lessons learned

## Sprint 1 - The Crossing

We had trouble with some tasks that had vague or ill-defined acceptance criteria that made it hard to judge when a task was completed or what was required to finish any given task. Some of our acceptance criteria also conflicted with what the task itself was requesting and either should have been applied to a whole new task or the task should have been rewritten to better define what was expected of the developer. We will take this knowledge of trying and struggling with acceptance criteria and better inspect that the criteria that we apply to tasks in sprint two are quantifiably defined, clear, and match up with the tasks they are assigned to.

Because in part due to poorly defined tasks and partly due to trying to plan ahead for tasks in future sprints a considerable amount of time was spent working on things that weren’t tasks. While this is and isn’t a bad thing, because all of the tasks for this sprint were finished, that time could have been better spent on current tasks and not on work that wasn’t in this sprint. For future sprints we will spend more time defining the issue in JIRA and hopefully stay closer to the work that needs to be done and not unrelated work. Additionally checking in more often with JIRA and using the comments section will hopefully keep developers on task and let others keep up with committed work and provide guidance if necessary.

The networking issue could be larger and more complicated than we had anticipated. This is not necessarily an issue but something that must be planned for. Some tasks or stories might need to be reprioritized or moved to a sooner sprint than might have otherwise been intended. Additionally we will use this found knowledge to distribute tasks differently and take it into account for our planning sessions. New stories or tasks might be needed for investigation or different implementation methods but this will be discussed at our sprint planning and backlog grooming meetings.

Unity physics is not consistent between different platforms. We need determinism for our project so may need to write our own physics engine. As our needs are simple, this may not be as bad as it sounds but will have to be something we consider.

# 

# 

References

Binding of Isaac, (<http://store.steampowered.com/> app/113200/ accessed September 23)

*Enter the Gungeon* (<http://store.steampowered.com/> app/311690/ last accessed September 23)

PHP (php.net)

Golang (https://golang.org/)