

Sprint 1 Report

*Team Report*

*Group: Team B*

*Project: Crossing Streams*

*Course: Cosc 470 - Software Engineering*

*Report version 1.1*

*Submitted by: Billy Spelchan for Team B (Nov 9th)*

Contents

[1. Project Vision: 3](#_Toc466456209)

[2. Problem statements: 4](#_Toc466456210)

[2.1 Review and compare your original problem statement 4](#_Toc466456211)

[2.1 Are there any changes to the desired deliverables? 5](#_Toc466456212)

[3. Team Members 6](#_Toc466456213)

[4. Development Process 8](#_Toc466456214)

[4.1 Description of the Agile Scrum process 8](#_Toc466456215)

[4.1.1 Product Backlog (See below - Not quite ready yet) 8](#_Toc466456216)

[4.1.2 Sprint Planning (See below - Not quite ready yet) 8](#_Toc466456217)

[4.1.3 Sprint Backlog (See below - Not quite ready yet) 8](#_Toc466456218)

[4.1.4 Sprints (See below - Not quite ready yet) 9](#_Toc466456219)

[4.1.5 Scrum (See below - Not quite ready yet) 9](#_Toc466456220)

[4.1.6 Demos (See below - Not quite ready yet) 9](#_Toc466456221)

[4.2 Project development and management platform (JIRA) 9](#_Toc466456222)

[4.3 Preliminary Communication Method 9](#_Toc466456223)

[5. Glossary of terms: 11](#_Toc466456224)

[6. Results 13](#_Toc466456225)

[6.1 Requirement gathering and specifications: 13](#_Toc466456226)

[6.1.1 Functional requirement / Specification: 15](#_Toc466456227)

[6.1.2 Non Functional requirement / specification 16](#_Toc466456228)

[6.1.3 Specific Requirements 17](#_Toc466456229)

[6.2 Engineering Execution 19](#_Toc466456230)

[6.2.1 Team Software Integration Plan (best for continuous integration) 19](#_Toc466456231)

[6.2.2 Builds (procedure of code integration and building the final product, version control, software change control) 19](#_Toc466456232)

[6.2.3 Installation (procedures, user manual or user guide, source code or repository) 20](#_Toc466456233)

[6.3 Project development results 20](#_Toc466456234)

[6.3.1 Sprint planning / roadmap summary 20](#_Toc466456235)

[6.3.2 Burn down chart history from Sprint 0 to final 20](#_Toc466456236)

[6.3.3 Sprint velocity and hours 23](#_Toc466456237)

[6.3.4 Risk Tracking from Sprint 0 to final 24](#_Toc466456238)

[6.3.5 Summary of Retrospectives 24](#_Toc466456239)

[6.3.6 Summary of Scrum Meetings 25](#_Toc466456240)

[7. (NOT ready to start) Maintenance 27](#_Toc466456241)

[7.1 (NOT ready to start) Source Code locations (git repository) 27](#_Toc466456242)

[7.2 (NOT ready to start) Build Script and Setup 27](#_Toc466456243)

[8. Conclusion 28](#_Toc466456244)

[8.1 Summary of your project results 28](#_Toc466456245)

[Sprint 1 - The Crossing 28](#_Toc466456246)

[(sprint 2 and 3 NOT ready to start) 28](#_Toc466456247)

[8.2 Suggested list of improvements 28](#_Toc466456248)

[Sprint 1 - The Crossing 28](#_Toc466456249)

[8.3 Lessons learned 28](#_Toc466456250)

[Sprint 1 - The Crossing 28](#_Toc466456251)

[Sprint 2 - The Streams 29](#_Toc466456252)

[Sprint 3 - The Crossing Streams 29](#_Toc466456253)

[9. References 30](#_Toc466456254)

FINAL DOCUMENTATION

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

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

TOTAL = 91 + 5 (bonus for the improvement ) =96 %

# 1. 2/2 Project Vision:

The overall objective of this project is to develop and design a comprehensive game platform and the logical software that allows users to seamlessly connect with other players, databases, and websites. The Crossing streams game is an online solution that allows users to play over the internet with each other, upload their stats to a website and communicate either in game or online.

Deliverables

* A standalone game that is both multiplayer (local/online) and single player
* A dynamic game server that connects game clients to allow for online gameplay
* A database that contains user data (user account, user stats, forum posts, etc)
* A basic website where players can register, pay and download the game
* Documentation which provides understanding of the software for future developers.

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.



# 2.(4/4) Problem statements:

Here is our initial version of the problem statement as used in the first sprint:

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.

## 2.1 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 sprint 2 (for comparison with second sprint problem statement revision)

## 

## 

## 2.1 Are there any changes to the desired deliverables?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Modifications | Exclusions | Additions |
| **Sprint 0** | Test sprint - No changes made | | |
| **Sprint 1** | No changes to deliverables made this time | | |
| **Sprint 2** | NOT ready to start |  |  |
| **Sprint 3** | NOT ready to start |  |  |

# 

# 

# 3. 3/3 Team Members

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.

**Benjamin Ward: *Scrum Master / Developer***

As a developer, the main responsibility I hold is maintaining the server infrastructure and designing the layout and execution of server-side net code that will allow for game clients to play together online. As a Scrum Master, I facilitate all scrum meetings (standup, sprint planning, review and retrospective) as well I keep track of what the other team Members are currently doing, providing assistance if any other developers run into roadblocks. I take notes at all the meetings and maintain the repository for documentation.

**Billy Spelchan: *Team Lead / Developer***

As a team lead, I will be communicating with James, handling the submission (and assembly if necessary) of the final document, trying to keep the team thinking about the larger goals of the project, managing our three strike policy, and acting as a devil’s advocate if other team members are not already taking on that role. As a developer I tend to be a jack of all trades so will probably have my fingers in everything. With my real-world experience, the key areas that I will be focusing on are the website and game development.

**Corey Frank: *Product Owner / Developer***

As a product owner, I will be in contact with our client Ben Heggie, I will keep him updated on our progress during development and will communicate any changes that the client wishes to implement. I am also in charge of setting up our meeting times and booking rooms for the group to meet. As a developer, I will be working on several parts of the system, I will be programming several game aspects including (but not limited to) level design, in game chat, the game patcher, as well as some web programming.

**Daniel Atkinson: *Developer***

As a developer, I will be focusing on the development of the game using my experience with game design, unity game engine, and C# programming language. I will ensure that our implementation of various game functions makes the best use of the unity game engine and conforms to best practices as far as is useful to us as a team. Because I have the most experience of the team members with game development and the use of unity I will be focusing primarily on some of the more technical and involved aspects of the game development including but not limited to enemy intelligence and game networking.

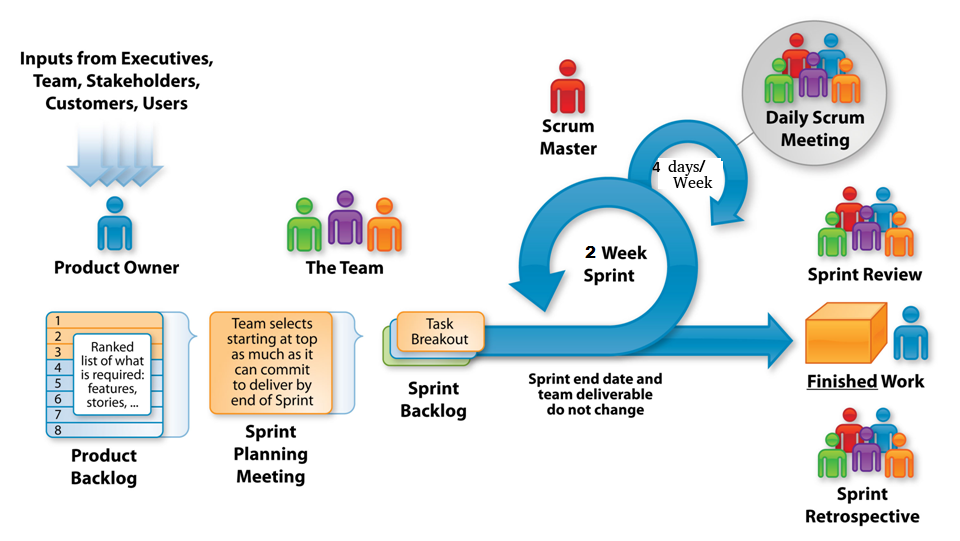
**Marc-Andrew Dunwell: *Developer***

As a developer, I will be focusing on learning and utilizing the Unity game engine to the best of my ability to develop several aspects that are core to gameplay, and with my experience in User Interface design, I will be ensuring that our game follows through with the standard that is expected to make playing easy, and simple for all users. I will be utilizing my experience in design and to keep the game attractive and following the theme that we have chosen.

# 4.(5/5) Development Process

## 4.1 Description of the Agile Scrum process

(that your team followed through the project development.)



*Source: http://www.etechpulse.com/2015/02/agile-scrum-process-sdlc.html*

### 4.1.1 Product Backlog (

PBL is an on-going prioritized list of user story) Should get this table from JIRA

The Product backlog is a prioritized list of work for the development team, containing short descriptions of the functionality that will be expected throughout the project. Our requirements obtained from the client provide the foundation for our current product backlog.

### 4.1.2 Sprint Planning ()

Sprint planning is a group effort involving every individual involved in the team (product owner, scrum master, and the entire development team). Our scrum master Ben runs the meeting; the product owner Corey describes the highest priority feature to the team, and all of the team defines work and effort by asking questions to turn the high-level user story into the detailed tasks of the sprint backlog. This method has been implemented in our project which has resulted in our Sprint 1 and Sprint 2 backlogs.

### 4.1.3 Sprint Backlog

The Sprint backlog is a list of tasks defined by the team that is to be completed during the Sprint. It is maintained by the product owner Corey, who sorted it by order of priority and developers pulled tasks by order or priority and their own individual expertise or interest. An example would be a developer that is interested in web server development would pull a task to do with server/client communication. Tasks are at points within the sprint added or removed from the backlog in response to risks occurring (for example, members of the group becoming sick), bugs popping up in the code base, or tasks that were estimated as low effort but found to be high in required work hours.

### 4.1.4 Sprints (

Sprints are timeboxed to 2-weeks per sprint over the course of this project. We have several Scrum meetings throughout the week during our Sprints (Tuesdays, Wednesdays, Thursdays and Saturdays). We would discuss what we planned on completing, and ensure that our tasks are completed before the end of the sprint.

### 4.1.5 Scrum (

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. After our Scrum meetings, we would continue working on our tasks which will be updated in the following scrum.

### 4.1.6 Demos (

At the end of each sprint a demo will be given on each successful feature completed where they are displayed to the project client.

## 4.2 Project development and management platform (JIRA)

The project team used Jira and Google Docs to document, develop user stories and sub-tasks. Git was used for version control. Furthermore we are using the following tools:

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>

## 4.3 Preliminary Communication Method

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

Tuesday 11:00 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 Slack and Skype (for scrum meetings) setup. Slack is an all in one embedded communication platform which allows for text, file sharing and other features available with plugins.

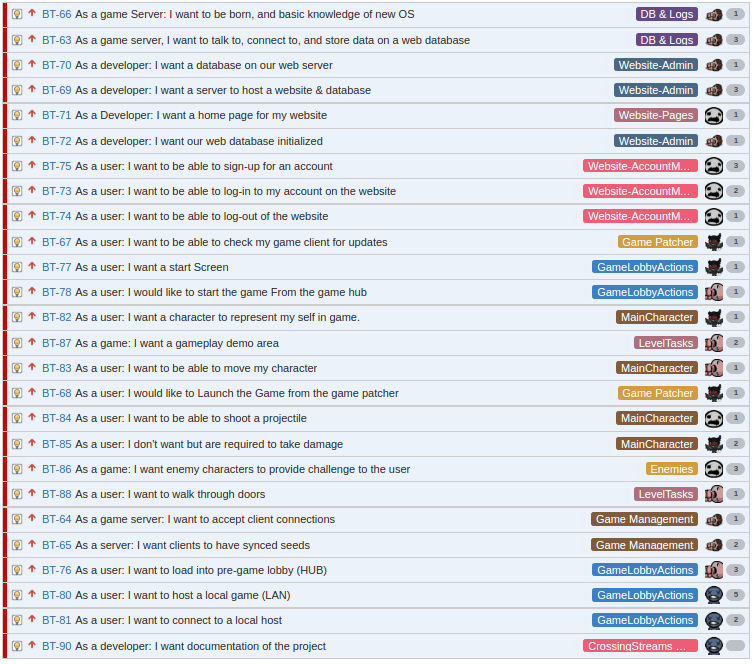
# 5. (3/3) Glossary of terms:

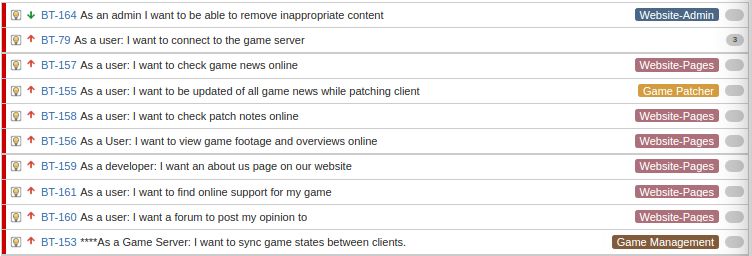
|  |  |
| --- | --- |
| **Agile** | A group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. |
| **Binding of Isaac** | A popular game featuring procedurally generated levels 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** | The application that is run in order to start the game. It 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 uses.r |
| **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. |

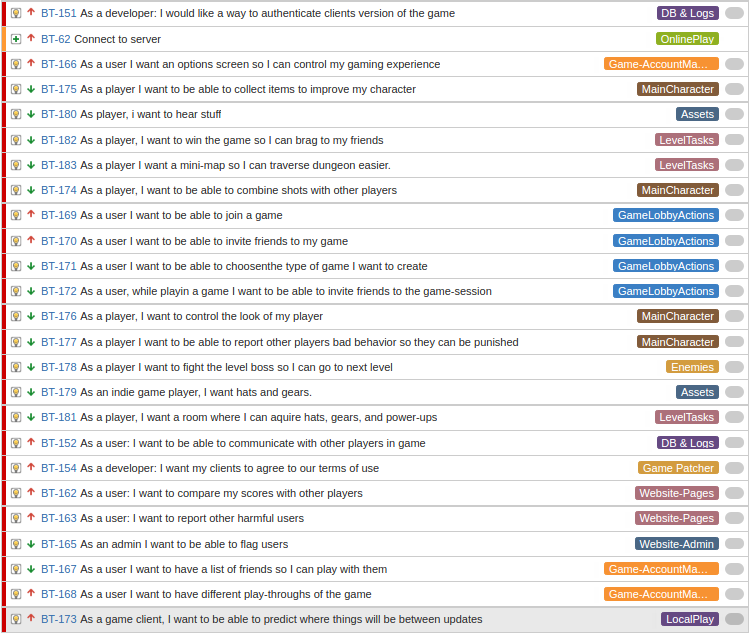
# 6. Results

## 6.1 Requirement gathering and specifications:

**Product Backlog**







### 6.1.1 7/10 Functional requirement / Specification:

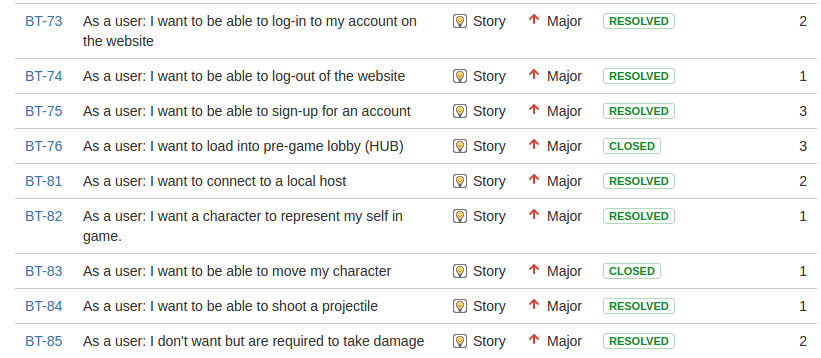
The Functional requirements are split into the sprints as followed:

Any catagorization to the following list of User Stories? This should be associated with your major components (or architectural layout) from your user story map..

**Sprint 1:**











**Sprint 2:**

**<NOT ready to start>**

**Sprint 3:**

**<NOT ready to start>**

### 

### 

### 6.1.2 10/10 Non Functional requirement / specification

Provided below is the system architecture that is used to help explain how the system’s technology works with each other.

Fig. - System Architecture (will be updated)

The project can be broken into five basic systems. The data store, web server, patcher , game client, and game server (which has local and online versions).

The data store is our PostgreSQL database which is shared by the web server and the game server. User information is managed by the web server while the game server manages the statistics which the player generates.

The web server 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 patcher is what the user runs to start the game. It makes sure that the player has the latest version of the game client before launching the game client. As a multi-player game, it is important that the players of the game all have the same version of the client otherwise there could be issues due to slightly different ways that different versions of the game work.

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.

The server architecture is outlined in the specific requirements section and covers the datastore, website, and game server components. The patcher is a stand alone utility so has no underlying architecture that it is built upon.

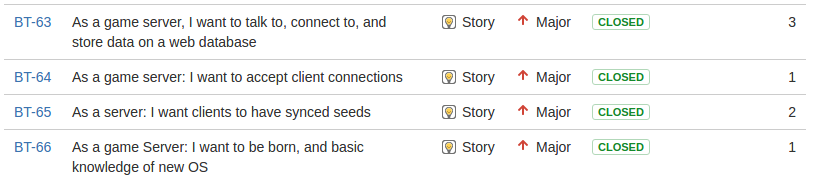


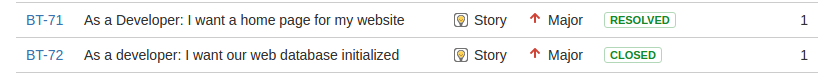
The game client is built on top of the Unity Game Engine which handles the graphics, physics, resource management, IO, GUI, and other low level tasks. It is a component based architecture so that game objects have behaviors attached to them to control them. The Unity Game Engine runs on a variety of operating systems and devices but for this project we are focused only on Windows, OS X, and Linux systems.

The Game Server, which can be local for LAN play or remote for internet play, will share the game state between the various game clients. The details behind this will be in sprint 2.

The Non Functional requirements are split into the sprints as followed:

**Sprint 1:**





**Sprint 2:**

**<NOT ready to start>**

**Sprint 3:**

**<NOT ready to start>**

### 6.1.3 Specific Requirements

(interface, hardware, communication interface):

The specific requirements are split into the sprints as followed:

**Sprint 1:**





**Sprint 2:**

**<NOT ready to start>**

**Sprint 3:**

**<NOT ready to start>**

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

## 

## 6.2 Engineering Execution

### 6.2.1 Team Software Integration Plan (best for continuous integration)

For the first sprint we opted to go with manual builds thinking that as our tasks were spread out enough over getting 5 separate components working together we should have very little issues so delaying the setup of Continuous Integration (CI) software could be delayed until the second sprint. This turned out to be a mistake as things did not go smoothly when it came to working with Unity, which four of the five developers had to do.

For the second sprint we are going to be setting up a Jenkins server. This CI software was chosen as it has support for Unity.

### 6.2.2 Builds (procedure of code integration and building the final product, version control, software change control)

For the first build, the software is manually built. The server software is compiled using go.

For Sprint 1 we did not have any Continuous Integration set up for building our Unity project, so manual building of the project was done. This requires setting up unity on your system.

Download unity from the Unity store (<https://store.unity.com/> ). We are using the personal edition version 5.4.2f2 which is free up to the first $100,000.

Install and set up your unity account. When you start unity, it will prompt you to create a new project. Select a name and location to store the project and name the project appropriately. Make sure that the project is set up as a 2D project. Note that 2D mode is not vital, but when you start in 2D mode the environment is set up for the creation of 2D games.

Make sure that meta files are visible so that you can use a repository such as git. This is done using Edit->Project Settings->Editor Settings. Set version control mode to visible meta files.

Use File->Save Project to make sure the project is created.

Copy the files from the git repository’s Unity/Asset directory into the assets directory within the project.

Refresh the assets folder in Unity (right click Assets folder in project folder, select refresh).

If the repository is up to date and no issues should be able to run. Keeping the repository up to date was one of our issues for Sprint 1 but we hope to get this under control for sprint 2.

For website pages, CI is simply the process of copying the pages to the web server. At the moment this is done manually, but it should be possible, if desired, to automate this process by pulling changes to the web pages from the repository and placing them in the html folder that contains the web site.

### 6.2.3 Installation (procedures, user manual or user guide, source code or repository)

The user obtains the software by going to the web site. Once they have created an account they can download the patcher which is used to download, update, and run the game.

From a developer’s perspective, once the servers have been set up with the suite of software we are using, run the server software, and copy the website pages to the appropriate directory so apache can serve up the pages. The patcher grabs the latest version of the game client from a repository so a copy of the latest version of the build will have to be placed there.

## 6.315/20 Project development results

How did the team establish the following roadmap? Look for the user story map results to support the follow results. The map should also show the user story priority adjustments based on feedbacks from (previous sprint results, stakeholder)

### 6.3.1 Sprint planning / roadmap summary

|  |  |
| --- | --- |
| **Sprint 0** | **The Jira Tutorial**  Test user stories were associated with this sprint. Members of the team went through a Jira Tutorial and added stories, tasks, epics, estimations, etc and logged hours into open tasks which were then set to done once completed. All members contributed and the tutorial is considered to be a resounding success as our utilization of Jira has gone seamlessly. |
| **Sprint 1** | **The Streams**  The system being developed is made up of a number of interconnected components which at the end of this sprint are going to be working at the most basic level possible. The client will be able to download the patcher from the game site, update to the current (first) build of the game, and run through a very simplified version of the game with just rudimentary server connections. |
| **Sprint 2** | **The Crossing**  NOT READY TO START |
| **Sprint 3** | **The Crossing of Streams**  NOT READY TO START |

### 6.3.2 Burn down chart history from Sprint 0 to final

|  |  |
| --- | --- |
| **Sprint 0** | **The Jira tutorial**    The burndown chart shows that there is inconsistency. The first half shows that not much work was logged for Wed-Fri, someone was a keener on the weekend and in the last half we started to log hours. In the end it’s shown that much of the work wasn’t being logged till the very end. All in all, the takeaway is we should be more consistent in logging hours. |
| **Sprint 1** | **The Streams**  burndown.PNG  The burndown for sprint 1, (at 9:26PM tuesday Nov.1)  We *are off to a slow start due to starting a day late as well the tasks assigned to my other team members require a considerable amount of infrastructure setup before they can start. This was resolved over the course of the first weekend.*  We finished most of the tasks (37 out of 44 story points) in the overall sprint, 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. These reasons lead to the remaining time estimate not being as close to the bottom.    The Burndown chart of our completed story points reflects a more accurate representation of how many tasks we had yet to complete. From the chart, there are 10 outstanding user story points, how would that affect your next sprint? |
| **Sprint 2** | **The Crossing**  NOT READY TO START |
| **Sprint 3** | **The Crossing of Streams**  NOT READY TO START |

### 

### 

### 6.3.3 3/3 Sprint velocity and hours

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Hours Estimated** | **Hours Completed** | **Story Points estimated** | **Story Points completed** |
| **Sprint 0** | 99w 4d 7h 10m | 99w 4d 7h 10m | 44 | 44 |
| **Comment:**  This was a test sprint. The values entered above are not realistic or accurate at all. Some of the user stories and tasks that were estimated were using experimental values to determine what Jira could handle which is reflected in the estimated hours. | | | |
| **Sprint 1** | 1w 1d 4h 30m | 6d 3h 30m | 44 | 37 |
| **Comment:**  Our sprint Velocity for sprint 1 can be seen in the chart below, it shows we committed to 44 story points and we completed 37 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. | | | |
| **Sprint 2** | 2w 4h 15m | - current sprint | 54 | - current sprint |
| **Comment:**  NOT READY TO START | | | |
| **Sprint 3** |  |  |  |  |
| **Comment:**  NOT READY TO START | | | |
| **Velocity Chart** |  | | | |

### 6.3.4 (5) Risk Tracking from Sprint 0 to final

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability of Risk** | **Size of Loss (Days)** | **Risk Exposure (Days)** |
| Team member falls ill or otherwise unavailable. | 40% | 5 | 2 |
| Repository unavailable (github goes down) | 5% | 1 | .05 |
| The school’s Servers go down and we can’t work push to production servers or test online infrastructure. | 20% | 5 | 1 |
|  |  |  |  |
|  |  |  |  |
| Exposure: | | |  |

### 6.3.5 Summary of Retrospectives

|  |  |
| --- | --- |
| **Sprint 0** | This was a test sprint that was utilized to help members of the group understand the Jira tool that will be used. |
| **Sprint 1** | 1. **Continue:**    1. Good vision of what we wanted for the end of sprint 1 (Theme)       1. Good grasp of scope    2. Communication between members in the last week went surprisingly well.    3. All committed to the project and the tasks assigned.    4. Trust (everybody did what they said they would do) 2. **Stop:**    1. Integration (we didn’t do it and suffered for it)       1. Pushing to master (more frequently) when you finish the story.          1. Uml diagram (minimal at least)          2. Clear test plan (unit test) and test plan to describe how to do the testing.       2. Automatically build/test/deploy    2. Lateness to meetings.    3. Acceptance criteria could use work    4. User stories could be better defined (more specific) 3. **Start:**    1. Set up Jenkins CI server (get up and running and document how it works)    2. More diagrams to explain architecture    3. Final documentation and presentation start now.    4. Presentation needs more slides       1. Task breakdown for each user story per team member |
| **Sprint 2** | NOT READY TO START |
| **Sprint 3** | NOT READY TO START |

### 6.3.6 Summary of Scrum Meetings

|  |  |
| --- | --- |
| **Sprint 0** | **The Jira tutorial**  *No Scrums were held for this sprint tutorial.* |
| **Sprint 1** | **The Streams**  **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 - <sick>  Dan - hooking up servers (launching from client)  ***What we're working on (for Wednesday):***  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 3** | **The Crossing of Streams**  NOT READY TO START |

# 

# 

# 7. (NOT ready to start) Maintenance

## 7.1 (NOT ready to start) Source Code locations (git repository)

**Documentation:** https://github.com/CoreyFrank/CrossingDocuments

**Server Application:** https://github.com/CoreyFrank/CrossingServer  
 **Web and Game Application:** https://github.com/CoreyFrank/CrossingStreams

## 7.2 (NOT ready to start) Build Script and Setup

7.3 (NOT ready to start) User Manuals / Guides

## 

# 8. (12/12)Conclusion

## 8.1 Summary of your project results

### Sprint 1 - The Crossing

We were able to complete most of the stories in the first sprint and as a result have a foundation for building off of in future sprints. The servers have been set up hosting our database, web-server and game-server. The game server can accept connections and share the game seed with the players. The website allows the creation and validation of users with registered users who are logged in able to download the patcher. The patcher will grab the latest version of the game from the repository but launching the game once updated was not implemented before the end of this sprint. The game runs with the ability to transition between rooms. Player’s can move and shoot in different directions, with monsters having different behaviors and attributes.

Our client seemed happy with the team’s demonstration but would like to see multiple players playing the game as that is, for him, one of the biggest selling points of this project. OK While the team does agree that multiplayer is important, this required the necessary infrastructure to be in place first, which was the goal of this first sprint. OK

### (sprint 2 and 3 NOT ready to start)

## 8.2 Suggested list of improvements

### Sprint 1 - The Crossing

The three key things that we need to focus on are CI, diagrams, and test plans. OK CI was clearly an issue that we ran into as not seeing the overall changes that a contribution has to the build clearly was detrimental. OK Documenting and creating simple diagrams of the work that we have completed is going to have to be a bigger priority so that other members of the team can get up to speed on what other members are doing especially as tasks start having larger overlap. Finally, having a formalized test procedure for the stories we have completed would be a good idea to make sure that we can always verify that what we have done is not causing other problems and that the program is working the way as intended

## 8.3 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. OK 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. OK 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. OK The project we are creating is very complicated so we are still working out stories that will need to be added to the backlog for the next eight sprints. Getting the task break down structure down helps in establishing the acceptance criteria as well… While this is and isn’t a bad thing, as we are working out details that will affect the future of this project, 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 aiming to get 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 something that must be planned for otherwise it could lead to major issues in future sprints. It this (multiple players) is a high risk item (sounds like it is), this should started in the coming sprint..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 ??? not sure what this means. 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.

### Sprint 2 - The Streams

### Sprint 3 - The Crossing Streams

# 

# 

# 9. 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 (http://php.net)

Golang (https://golang.org/)

Unity (https://unity3d.com)