Table 1: Revision History

| **Date** | **Developer(s)** | **Change** |
|---|---|---|
| 2016-09-30 | Jose Ballesteros | Added Coding Style and created Gantt Chart to be pointed to |
| 2016-09-30 | David Hobson | Added Team Meeting Plan, Team Communication Plan, Team Member Roles |
| 2016-09-30 | Jeff Pineda | Added Git Workflow Plan, Proof of Concept Demonstration, Technology |
| 2016-10-21 | Jeff Pineda | Modified Proof of Concept Demonstration secton to accurately represent what is/was demonstrated, and updated Technology section to include new IDE being used and the selection of a testing framework. |
| 2016-12-01 | David Hobson | Updated different sections based off comments. Added Abstract. Added roles to team meeting plan. Added Training and Education to Git Workflow Plan. Added more specific requirements to proof of concept. Added Doxygen for C# documentation. |
| ... | ... | ... |

# SE 3XA3: Development Plan
# Mari0

Team 9, Ninetendo
David Hobson hobsondd
Jose Miguel Ballesteros ballesjm
Jeff Pineda pinedaj

## 1 Abstract

This program is a game that is similar to Mari0 and will be created in Unity. Mari0 is a crossover of Super Mario Bros. and Portal. Super Mario Bros. is a platforming that has a character (Mario) running to the right side of the screen avoiding enemies and jumping around obstacles. Portal is a puzzle game where a character has a "portal gun" where they can place up to 2 portals. These portals are connected and allow the player to move through them, going trough one portal makes the character come out the other side. These portals are added to the main Super Mario Bros. game and allow the character to interact with them in the same way you can in Portal.

## 2 Team Meeting Plan

Our team will be meeting at the regular lab times to quickly discuss changes as well as upcoming deadlines for the project. Our lab times are Wednesday from 8:30 - 10:20 and Friday 2:30 - 4:20. The majority of each meeting will be focused on the closest deadline to be met, discussing each members' schedules and using that information to create a work plan to meet the next deliverable. The rest of the meeting time will be used to discuss changes/issues that need to be addressed in regards to previous deliverables, and to talk about future deliverables. If there are other times that we need to meet, we will discuss using the Facebook chat to decide on a time where all team members are available.

David Hobson is the project manager and therefore has the role of facilitating the meetings and making sure that everything is running smoothly. Jeff Pineda is responsible for making sure that documents are reviewed each meeting to go over any errors and things we can improve on. Jose Miguel Ballesteros has the role of making sure that the implementation is following the documentation and our plans.

# 3 Team Communication Plan

The communication will mainly happen through a Facebook chat that all team members have access to. Everyone apart of the chat can communicate quickly and effectively about milestones, changes, and meetings. Any shared files will be pushed to the repository on GitLab that everyone in the group has access to. E-mail may be used as well to send certain files or messages but this will be used rarely.

# 4 Team Member Roles

**Project Leader and Developer:** David Hobson
**Developer and Graphic Designer:** Jose Miguel Ballesteros
**Developer and Documentation Manager:** Jeff Pineda

# 5 Git Workflow Plan

Since we all have limited experience with Git, we will be utilizing a centralized workflow in the early stages of the project. However, as we get more comfortable with Git and when the need for multiple people to work on one file at the same time becomes more frequent, we will move to a featured branch workflow.

Furthermore, it is highly recommended that new developers that are later to the project to finish the Git training that is available through the 3XA3 Lab 3. There is no need for education within Unity, however, having programming experience in C++, C#, or Java is a great asset. In addition, having a good understanding of object orientated programming is essential.

# 6 Proof of Concept Demonstration Plan

The most difficult part of the implementation of Mari0 will be coding the 'portals' and their mechanics. Two portals can be placed anywhere within line of sight of the main character and on any surface. These two portals are connected, with any projectile or character entering through one portal exits out of the other portal, maintaining their physical properties, such as acceleration and velocity, they had when first entering.

Testing whether or not player inputs lead to correct gameplay/visual outputs will be relatively simple, as this can be handled by writing unit tests. Testing the other aspects of the game such as puzzle difficulty, control feel and responsiveness, and enjoyability of the game will be more difficult. Testing these will need to be done through beta tests and player survey feedback.

For the proof of concept demonstration, we will demonstrate a simple level design to show working character physics, working character animations, and most of the portal mechanics. Character physics include the ability to move left, right, and jump in a basic level environment. The character also should react to different obstacles that will be presented in the demo, such as the ground and walls. The portal mechanics to be demonstrated are the property that two portals are connected, and the player will maintain the physical properties they had when entering the first portal upon exiting the second portal. The ability to place portals on any surface will not be implemented by the time of the proof of concept demonstration. The character should easily be able to walk through different pairs of portals showing off how the physics is maintained. For example, two portals being place on the ground and the user should be able to jump into one and the jump physics should continue through into the second portal.

# 7 Technology

We will be using the Unity game engine to handle game physics and will be writing the rest of the game with C#. Members of the group will either use MonoDevelop, a Unity integrated IDE, or Visual Studio for coding. We will be writing our own unit tests. We will be using NUnit as our unit-testing framework. Our C# code will be documented using Doxygen. This will be able to show the different modules that are used within the program along with the methods that are used.

# 8 Coding Style

We will be using Microsoft's coding convention for C# with a couple alterations.

- Open curly braces will follow the expression?s parameters and the closing brace will be inline with the expression as shown in figure 1 below.

```
public A(int n){
    count = n;
}
```

- All constants will be named in capital letters
- Method names will be named in upper camel case format.

# 9 Project Schedule

Click here for the Gantt Chart.

## 10 Project Review