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
...	...	...

SE 3XA3: Development Plan
Mari0

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

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.

1 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. 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.

2 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.

3 Team Member Roles

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

4 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.

5 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.

As a proof of concept, we will demonstrate a simple level design, with it being a small area and containing simple walls and platforms, a moveable character and working portal mechanics, to prove that the difficulties can be overcome.

6 Technology

We will be using the Unity game engine to handle game physics and will be writing the rest of the game with C#. The IDE we will be using is MonoDevelop which is a Unity integrated IDE and we will be writing our own unit tests.

7 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 camel case format.

8 Project Schedule

Click here for the Gantt Chart.

9 Project Review