Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept. 29th – Rev0.1	Andrew, Kyriakos, Teodor	first draft
Dec. 6th – Rev1	Andrew, Kyriakos, Teodor	modified document to match current project. red text shows modifications.

SE 3XA3: Development Plan
Title of Project

Team #17, Test Icicles
Andrew Bennett 1319879
Kyriakos Kyprianou 400025691
Teodor Tomescu 400038361

1 Team Meeting Plan

To stay on track with the project and ensure completion of required topics by a set deadline, our team must meet up once a week for an hour. We want to make sure every member has a clear understanding of what is expected of them so these require mandatory participation by all team members. These meeting will take place every Tuesday from 5 PM to 6 PM on the second floor of the Mills library. In the meeting, we will delegate tasks to each member and ensure that they are fully informed on what will be expected of them. Member roles will be discussed in the Team Member roles section of our development plan however these are not necessarily rigid and can change during the software creation process.

An agenda will be needed during meetings to ensure efficient coordination for the team and to allow feedback from each member. We will draft a template to use as an agenda for each weekly meeting which will include the following topics: Feedback & Questions, Goals, Homework, Timetable, Meeting Review. In each meeting the project manager will keep track of what goes on in the meetings and ensure that every section of the agenda is appropriately filled out.

2 Team Communication Plan

Because not every member of our group owns a cell-phone we are currently using Facebook Chat for general communication on a day to day basis. We created a group with all three members in it and thus far it has been great for us to use as a non-technical means of communicating.

We will be using Unity's built in version control software as well as GIT over the course of this project. This is a great method of keeping track of changes are made to our code over time. It is especially useful since there are three of us all working on one project simultaneously and as we're able to keep a detailed report of every code contribution from each member, there is little that can go wrong on the repository. If ever we need to revert a whole commit we can do so as well. Git makes working on a project with multiple members effortless and its why we have decided to use it.

3 Team Member Roles

*Every team member is expected to contribute equally on the Unity build on top of their given role.

Andrew Bennett - Graphics, models, individual game features

The graphics and model expert will be responsible for overlooking everything pertaining to the graphics and physical modelling of our pinball game. Since he is the most knowledgeable in Unity game design he will also answer any questions pertaining to Unity.

Kyriakos Kyprianou- Project manager, individual game features

The project manager will oversee every meeting to ensure that our agenda is satisfied and note everything down in the template. On top of that, he will also organize any additional meetings the group needs to meet deadlines and clarify concerns. Everything will be then relayed to the remaining group members via the Facebook group chat.

Teodor Tomescu - Overall game manager, individual game features

The game manager will supervise the overall cohesiveness of the game and make sure that every members contributions are up to par with the initial goals that were outlined. He will make sure that members are on time with their work and judge the correctness of build features.

4 Git Workflow Plan

Deliberate and organized use of Git and Unity Version control features will be integral to our project's success. Our Git work flow will incorporate three major branches, Master, Development, Modelling, as well a number of more specific feature branches.

The Master branch will be kept up to date with fully functioning and completed feature sets. It will be our production branch.

The Development branch will be used to incorporate finished feature changes. ~~and will be the most up-to-date branch encompassing the project.~~

The Modelling branch will be used for updating the 3D models used in the game. While 3D modelling and game aesthetics are not the focus of our project, they will be updated over the course of the project. Finished changes made to the models or game visuals will be pushed to the Modelling branch, which will later be integrated into the Development branch.

Although Git is instrumental for the success of large-scale software projects, we have found that using Unity's on board version control helped simplify things as we had full Git functionality integrated in the Unity environment. This especially helps during sprints of rapid development as we can all stay up to date with changes very easily.

Tagging, Issues, and Milestones will also play an important role in our Git workflow.

Tags will be used to label every change made to the Master, Development, and Modelling branches. v1.0, v2.0, v3.0 etc will reflect major progress on the project and will be incremented after each push to the Master branch. Sub-versions (1.1, 1.2, 1.3 etc) will be incremented after each push to the Development and Modelling branches.

Issues will be used to keep track of bugs that need to be fixed, and likely will be the responsibility of the developer whose code introduced the bug.

Milestones will be used as a way to track feature requests and assign tasks to team members. While each team member is encouraged to work on features they would like to see in the project, Milestones will be used for the core features of the project that need to be included. In addition, Milestone 'due dates' will be used to ensure features are completed on time.

5 Proof of Concept Demonstration Plan

5.1 Technologies

Because our project is a 3D game we have decided to build it using Unity, a third party game engine. Some of the reasons for choosing Unity are as follows:

1. Unity is a free software.

2. Unity is a professional quality game engine. It has been thoroughly tested and is robust and reliable enough to meet all of our needs, including integration of 3D modelling and code, built in testing, and multiple export formats.

3. There are an abundance of resources available to assist us in using Unity. Unity has thorough, clear documentation, and because Unity is such a popular game engine, there are numerous third party guides and tutorials covering all aspects of Unity.

5.2 Modules + MVP

The core modules necessary for our game include:

1. A pinball 'playing field' consisting of a solid floor, walls, point-scoring obstacles, and 'bumpers', and containing a 'pinball'.

2. Realistic game physics which includes simulated gravity, acceleration of the pinball, and collisions between the playing field and the pinball.

3. A scoring system that keeps track of how many points the player has accumulated during the game.

4. Game-Ready and Game-Over states. The Game-Ready state should have a menu with 'Start Game'~~and 'View High Scores' options~~and 'Exit Game'. The Game Over state should display the player's score, and a menu button to return back to the main menu.

5. A flipper module to control the flippers and assign sufficient power to their action.

6. A spring module to enable the player to power up and launch the ball into play.

7. A module to return the ball back to its starting position once the current ball hit the bottom of the playing field.

The minimum viable product for our project will include a basic 3D model of the pinball, playing field, and bumpers. Gameplay of the MVP will include a pinball that falls due to the effects of gravity and collides with the playing field obstacles including the bumpers. The bumpers should be functional and responsive to player input.

5.3 Risks

The risks for our project are estimated to be manageable.

Basic game elements such as models and physics should be relatively straightforward to implement, however one concern is that the game physics (gravity, collisions, acceleration) could feel unrealistic to players. Menus, game states, and a scoring system are also estimated to be low-risk.

Testing is an area that will be of some difficulty. Testing of the game code itself can be done through custom testing suites, where we will manually check for edge cases on every variable and function within the code.
However, due to the graphical nature of a game, there are a number of bugs that could arise from graphics elements including the 3D models. Fortunately, Unity provides testing tools that will help alleviate this concern.
Unity itself is very simple to install and provides a lot of useful functionality for game developers. Unity's testing tools will be of great use during this project. Likewise, Unity's convenience exporting tools will allow our game to be extremely portable without needing to rewrite our code to suit different software environments.

6 Technology

The engine that we will be using for our game is Unity 3D. We will use C# in Unity to create scripts and bring the physical model of the game to life. In order to ensure the desired functionality of our game we will be using Unitys included test framework called Unity Test Tools. This framework is a package that gives developers all the necessary tools require for creating and executing automated tests inside the Unity editor. Using Unity Test Tools, we will be able to create unit tests and integration tests to ensure our code is up to our standards.

To create documentation for all of our C# scripts we will be using a Unity editor plugin that allows the creation of automatically generated documentation via Doxygen. This plugin is made by a private user however it has great reviews and can be easily installed onto Unity. For more information about the plugin you can visit http://www.jacobpennock.com/Blog/unity-automatic-documentation-generation-an-editor-plugin/.

7 Coding Style

In our project a coding style that all team members are familiar with will be implemented. Some elements of this include:

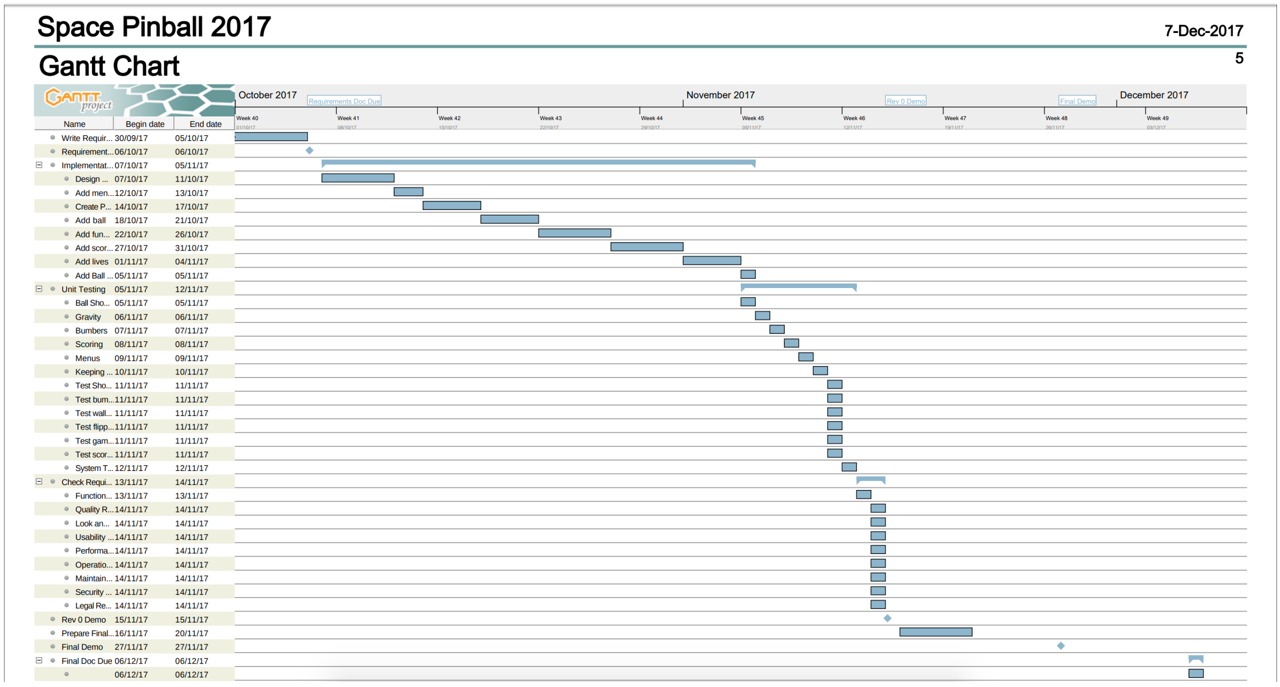
1. Brace style will be the java variant of K&R, with braces on the same line as function and class declarations, as well as control statements.

2. Tabs will be used rather than spaces for indentation

3. In general, line length should be kept under 80 characters.

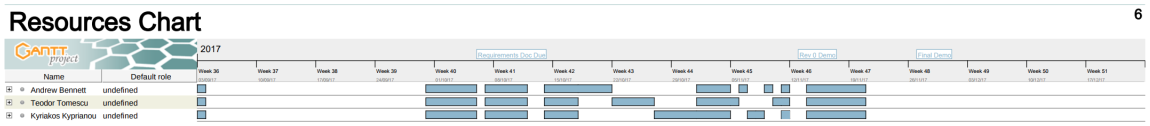
4. Variable names will be kept sensible and will use camel case rather than snake case or pascal case.

8 Project Schedule



Name	Begin date	End date
Write Requir...	30/09/17	05/10/17
Requirement...	06/10/17	06/10/17
Implementat...	07/10/17	05/11/17
Design ...	07/10/17	11/10/17
Add men...	12/10/17	13/10/17
Create P...	14/10/17	17/10/17
Add ball	18/10/17	21/10/17
Add fun...	22/10/17	26/10/17
Add scor...	27/10/17	31/10/17
Add lives	01/11/17	04/11/17
Add Ball ...	05/11/17	05/11/17
Unit Testing	05/11/17	12/11/17
Ball Sho...	05/11/17	05/11/17
Gravity	06/11/17	06/11/17
Bumbers	07/11/17	07/11/17
Scoring	08/11/17	08/11/17
Menus	09/11/17	09/11/17
Keeping ...	10/11/17	10/11/17
Test Sho...	11/11/17	11/11/17
Test bum...	11/11/17	11/11/17
Test wall...	11/11/17	11/11/17
Test flipp...	11/11/17	11/11/17
Test gam...	11/11/17	11/11/17
Test scor...	11/11/17	11/11/17
System T...	12/11/17	12/11/17
Check Requi...	13/11/17	14/11/17
Function...	13/11/17	13/11/17
Quality R...	14/11/17	14/11/17
Look an...	14/11/17	14/11/17
Usability ...	14/11/17	14/11/17
Performa...	14/11/17	14/11/17
Operatio...	14/11/17	14/11/17
Maintain...	14/11/17	14/11/17
Security ...	14/11/17	14/11/17
Legal Re...	14/11/17	14/11/17
Rev 0 Demo	15/11/17	15/11/17
Prepare Final...	16/11/17	20/11/17
Final Demo	27/11/17	27/11/17
Final Doc Due	06/12/17	06/12/17
	06/12/17	06/12/17

Figure 1: Gantt Chart



Name	Default role
Andrew Bennett	undefined
Teodor Tomescu	undefined
Kyriakos Kyprianou	undefined

Figure 2: Resources Chart

9 Project Review