**<Your Project Name Here>**

**Concept of Operations**

**COP 4331, Spring 2016**

Team Name: <Create a team name, or use your group number>

Team Members:

* Greg Kelso
* Mark Boutwell
* Joel Gardyasz

Modification history:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 01/28/16 | Greg | Template |
| v1.0 | 02/3/16 | Greg | Operational Features & Implementation |
| v1.0 | <date here> | <who> | <put comment to summarize the changes made in this version> |
| ... |  |  |  |

Contents of this Document

The Current System

The Proposed System

* Needs
* Users and Modes of Operation
* Operational Scenarios
* Operational Features
* Expected Impacts
* Analysis

**The Current System**

<Include a brief description of the current system. This is most applicable if your system builds on another or performs a task similar to other systems.>

<If there are no other systems that are similar to the one you are creating, indicate that and briefly describe what your system will do. First ensure that you have performed an exhaustive search for similar software.>

<1 or 2 paragraphs.>

**The Proposed System: Needs**

<Describe why a new or modified system is necessary. What will your system provide that the current system does not? Consider situations where yours is easier to use, cheaper, more accessible, or provides more or less features. If your system is a new system, why will people need it?>

<1 paragraph to 1 page.>

**The Proposed System: Users and Modes of Operation**

<Briefly describe each class of user and each mode of operation for the proposed system.>

<Will you have more than one class of users? For mobile productions, consider free version users and pay version users. How will they differ? Consider low level users who may only use a few features of your product and high level users who may want to take full advantage of your product.>

<What are the modes of operation? These are the states that your system can be in.>

<1 – 3 sentences per user and mode>

**The Proposed System: Operational Scenarios**

<Describe the major operational scenarios for the proposed system. What will people use your system to do? Consider each feature that is relevant to your system.>

<Include typical scenarios and a few atypical scenarios (errors, high risk situations, etc.) How will your system handle faults? These may be incorrect inputs, loss of internet connection, system crash, etc.>

<1 paragraph per scenario>

**The Proposed System: Operational Features**

Must Have:

1. An environment for the player to move around in and interact with.
2. A player that the user controls, and contains multiple actions controlled with keyboard input.
3. Enemies for the player to fight.
4. A health system for both the player and enemies, in order to add engaging gameplay mechanics.
5. Objects for the player to interact with (could be currency to collect, or weapons to pick up and use).
6. Easily understandable explanation of the user controls, gameplay mechanics, and an objective for the player to work towards.

Would Like to Have:

1. An in-depth tutorial explaining controls, mechanics, etc. explained in #6 above.
2. A fully-fledged dungeon, similar to The Legend of Zelda on the NES, complete with increasing difficulty, power-ups, and a boss at the end.
3. Music and sound effects.

**The Proposed System: Implementation**

The video game will be developed using Unity2D. The content will be exported as a desktop standalone usable with Windows, OSX, and Linux. In addition to Unity's built-in tools, all scripting will be done in C#. The reason we chose Unity2D is because we all have some experience with the tool, and we believe it will be easier to get a impressive video game with the current time constraints by using 2D over 3D. We will be using C# because in our opinion it is better suited to handle the scale of a Unity project as opposed to Javascript.

The development will probably have a rough start as we get a feel for the technical requirements of the project, combined with the individual strengths and weaknesses we all bring to the table. Because of this, the only learning curve we will have to deal with is how we effectively communicate and work together on the same project, since we all have experience with Unity and C#. Our main limitation will be with version control. Unity offers version control for its paid users (which we aren’t), and from we’ve heard, there tend to be a lot of merge conflicts when people use Git on Unity projects. We will be sticking with Git for now and hopefully not run into too many issues later in the project.

There are a lot of alternatives we could have considered instead of Unity2D. Unity3D was an option, but we believed it would be too much work for one semester. MonoGame, PyGame, and Unreal are also other game engines with a lot of documentation and support online. However once we realized that we all have experience with Unity, we decided to stick with what we knew. In addition to our personal experience with Unity, it is our opinion that it is better documented than other options at the moment, which will make developing the video game very easy.