Knight's Guard

Project Management Plan

COP4331C, Fall, 2014

Team Name: Group 1

Team Members:

- Megan Postava
- Katie Jurek
- David Moore
- Miguel Corona
- William Adkins
- Jonathan Bennett

Modification history:

Version	Date	Who	Comment
v0.0	09/11/14	Jonathan Bennett	Imported template from previous classes
v0.1	09/11/14	Jonathan Bennett	Filled in Project Overview, Applicable Standards, Project Team Organization, Deliverables, Tools and Computing Environment
v0.2	09/13/14	David Moore	Edited Tools and Computer Environment, Filled in Configuration Management, Software Life Cycle Process
v0.3	09/14/14	David Moore	Filled in Quality Assurance, Risk Management
v0.4	09/16/14	Jonathan Bennett	Filled in Table of Work Packages, Time Estimates, and Assignments. Made changes to Software Life Cycle Process.
v1.0	09/16/14	David Moore, Jonathan Bennett	Version 1.0 complete.

Contents of this Document

Project Overview

In *Knight's Guard*, the player controls a set of towers which are used to defend the UCF Student Union. The towers can be placed on specific places of the UCF campus map. Multiple waves of enemies will come from various places on the map, and will try to destroy the player's towers and the Student Union. Once the player has survived a set amount of time, they will be given a grade based on how much damage the Student Union took and move on to the next level.

Reference Documents

- Concept of Operations

Applicable Standards

- **Coding Standard:** Code will all use a similar style to ensure ease of development and maintenance. All code will be commented, with each function given a header comment to describe its purpose and expected input/output. The Git system will be used for source code versioning, and each commit will be given a concise message to describe the changes that were made.
- **Document Standard:** Every team member is responsible for using proper grammar and spelling to the best of their ability. Team members may review and edit each other's work if needed, and open communication is encouraged. A professional style and word usage is expected. The Calibri 11-point font will be used, along with larger sizes needed for headings. Color may be used where deemed necessary or helpful.

- **Artifact Size Metric Standard:** The project will include milestones, which will be completed in phases (also known as “Agile sprints”), and can be used to judge how far along the project is to completion (which is to complete all milestones). There will be separate milestones for different parts of the application, such as coding and graphics, and for the overall project as well.

Project Team Organization

The team consists of six members: Megan Postava, Katie Jurek, David Moore, William Adkins, Miguel Corona, and Jonathan Bennett. For the project, each team member will participate in programming, artwork, testing, meetings, progress updates, and documentation (which includes updating their logs on the public team web site).

A few of the members will have specialized roles. Katie Jurek is the team’s expert on GameMaker programming and will oversee the code of the project. Megan Postava and Katie Jurek will be the team’s primary game artists. Jonathan Bennett will serve as the editor of technical documentation. Although these specialized roles exist to help utilize the individual talents of the team, each team member will participate in every area of the project.

Team members will be in daily communication through use of UCF webcourses, telephone, text messages and email. There is an official face-to-face team meeting on Friday every week, and more meetings may be scheduled if needed. During each meeting, the team will discuss individual progress and the team’s overall progress towards reaching their 2-week milestone. If a member needs a temporary absence for more than two days, such as due to an emergency, they will contact every member of the team to let them know.

Deliverables

Artifact	**Due Dates**
Meeting Minutes	Due within 48 hours after every scheduled face-to-face meeting. This is uploaded to the team’s public /files/ directory.
Individual Logs	Each team member shall update their individual progress at least once a week on Friday evening or earlier.
Concept of Operations	September 18th, 2014
Project Management Plan	September 18th, 2014
Software Requirements Specification	September 18th, 2014
Test Plan	September 18th, 2014
High-Level Design	October 23rd, 2014

Detailed Design	October 23rd, 2014
Test Results	November 25th, 2014
Build Instructions	November 25th, 2014
User's Manual	November 25th, 2014
Source Code	November 25th, 2014

Software Life Cycle Process

As the development of video games tends to have evolving needs over the course of the development cycle, the group has chosen the Agile development model to best suit our development needs. This model will allow us to keep our vision of the game in check with what is realistic for the scope of the project, the expectations of the professor and TAs, and allow us to develop the best product possible in the time frame we have allotted to us. It also enables us to regularly interact with our client(s), so that they may review new features and be involved with making the project as successful as possible.

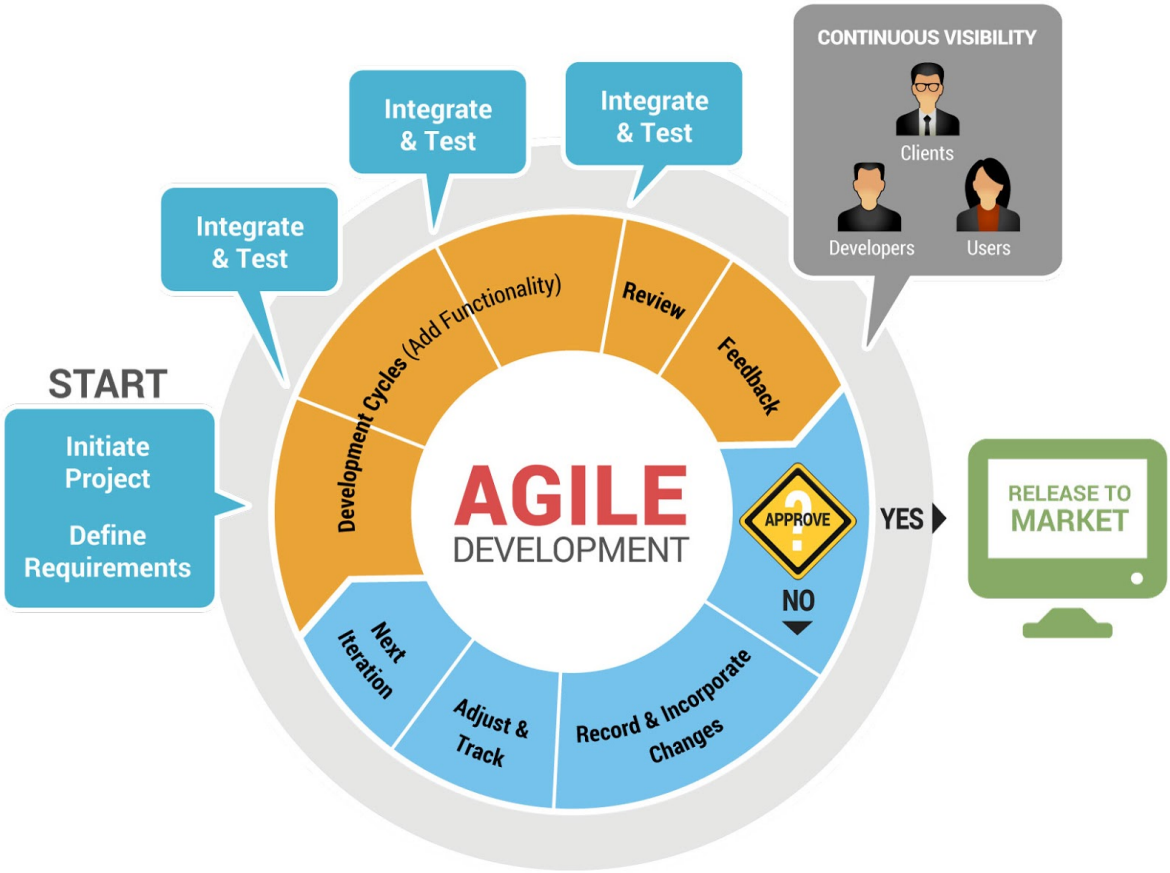

Image Credit: STA GROUP <http://www.stagrp.com/technology/application-solutions/>

Tools and Computing Environment

- Operating Systems
    - The team's personal computers consist of multiple operating systems, including Microsoft Windows 7, Mac OS X, and Linux. The project will be cross-platform for Microsoft Windows and Android operating system, as the engine is supported for both.

- Programming Languages
    - The programming languages used will be primarily GML for the video game. Other languages such as HTML, CSS, and Javascript may be used for the team web site. The GameMaker Studio IDE contains the compiler and will include any other libraries needed to run the game.

- Project Management
    - For project goal management, the team will be using the Trello service. This will allow us to be privy to what tasks require attention, who is attending to each task, and what tasks have been completed. Through this, each team member will have an accurate picture of the state of development at any given moment.

- Documentation and Source Code
    - For version control, the team will use Git and GitHub to host the code. For documentation and planning, the team will primarily use Google Docs, with the help of Microsoft Office if needed. To aid with tracking ideas and progress, the team will use the Webcourses discussion board and wiki.

Configuration Management

Version and change control will be managed using GitHub, while task management will be handled using Trello. Any time a change is committed to the repository, a comment will be added in reference to whatever goal in the flow the commit relates to. Commit conflicts will be resolved by the team members whose commits create conflict, though by using Trello and keeping tasks compartmentalized there shouldn't be much conflict between commits. Testing will be frequent and thorough to ensure that existing code remains functional.

Quality Assurance

By using an Agile development approach, we will be completing small chunks of work in short time frames. As we each complete our "sprints," each team member will be responsible for testing to ensure that each new "sprint" is functional as best as possible at each stage, as well as ensuring that previous stages have not stopped functioning as expected. If there are bugs that arise, they will be documented in Trello as well as an external Google Doc with more detail.

Risk Management

- We are all students with varying schedules and workloads. As such, a potential issue is the inability for a team member to be able to accomplish work at a reasonable pace. To ensure this does not happen or is not a detriment to the team in the case that it does, any work not completed at expected pace will be evaluated by the team on a regular basis.

- Our base idea for the game and the mechanics surrounding its play may not be fun. As any client commissioning a game would likely expect their game to be fun, we will be playtesting our game and tweaking mechanics as much as possible to ensure the game plays as well as possible, even if the final product does not match our initial expectations.
- In the event of data loss on any one person's machine, we have the project stored centrally on GitHub for data backup concerns. To avoid the loss of too much work, commits will be done frequently for small tasks to ensure the code base is as up-to-date as possible.
- Our tentatively planned scoreboard system may not be secure in preventing malicious players from spoofing high scores. We will be researching commonly used security mechanisms to prevent this from happening to the best of our abilities.

Table of Work Packages, Time Estimates, and Assignments

The below table explains the activities required to complete the project, the estimated time it will take to complete the activity, and the primary team members who will be involved and responsible for the task.

Note: Multiple activities will be worked on simultaneously, not sequentially. This means the three 8-week tasks listed in the table below will be worked on by the development team at the same time. This is to ensure the project is ready by the November 25, 2014 deadline.

All team members can and will be involved with all tasks, but certain tasks will be better suited to specific members of the team, due to their experience and/or talents. These members will oversee the task.

Activity	Estimated Time	Primarily Responsible
Planning: Create an internal game design plan that covers the basic functionality of the game: gameplay, artwork, sound	1 week	Katie, Jonathan
Planning: Create an internal game design plan to explain the enemies, their artificial intelligence & behavior, the player's defense abilities and behavior	1 week	Jonathan, David
Planning: Create an internal game artwork plan to explain the various pieces of art that will be needed	1 week	Katie, Megan
Artwork: Create the game map/levels	2 weeks	Katie, Megan
Artwork: Create the defensive abilities for the player (towers, powerups, weapons, items)	4 weeks	Katie, Megan
Artwork: Create the enemies to attack the player (includes animation, abilities)	4 weeks	Katie, Megan

Sound: Create the background music used for game menus and normal gameplay	2 weeks	Miguel, William
Sound: Create the sound effects for different items and actions in the game (powerups, attacks, etc.)	2 weeks	Miguel, William
Programming: Create the functionality of the game navigation. Browsing menus, selecting options.	8 weeks	Miguel, William
Programming: Create the functionality of the offensive abilities in the game. Towers, powerups, etc.	8 weeks	Katie, Jonathan
Programming: Create the functionality of the enemies in the game. Movement, artificial intelligence and path-finding, etc.	8 weeks	Megan, David
Programming: Create the functionality of a scoring system that can be synced to an online leaderboard.	2 weeks	David, Jonathan
Web Services: Setup the backend system to handle the online leaderboard feature	2 weeks	David, Jonathan

Note: If any artwork or sounds are not original work, the team members must ensure that they are allowed and properly licensed to use the assets in the game.

Technical Progress Metrics

As part of the Agile methodology, every task will be part of a "User Story", which is essentially a feature (or possibly a set of features). For example, "as a player, I want to start a new game" could be a user story, and this could include several development tasks such as creating the navigation menu, the first level, and a welcome message for the player to see.

Every two weeks, the team will meet to discuss and estimate which user stories (and their tasks) can be reasonably completed over the next two weeks. This two-week span is called a "sprint". After each sprint, we will evaluate how well we accomplished our tasks, and use this knowledge to improve our estimates for future sprints. Each user story is given a certain point value indicating its size and complexity, and over time we will be able to measure how many "points" the team is generally able to handle for each sprint.

This information will be collected in our meeting documentation. Individual tasks and user stories will be created as a board in the Trello software, so all developers can see who is working on each task and how it is progressing.

Plan for tracking, control, and reporting of progress

- The team will be in daily communication to report on progress and any issues that arise. Besides talking in person or through digital communications, each team member will also be tracking the

status of their tasks in the online Trello system. This allows every developer to know the current status of every task in the system with ease.

- There will be a weekly face-to-face meeting, where the team will display their progress to the rest of the team. These meetings will also be used to discuss new tasks and assign responsibilities to each member.
- Trello will also be used for bug tracking. There will be a Trello board specifically to report bugs and to track their progress. The Trello system enables the development team to discuss the bug, come up with a solution, and mark it as completed when it is fixed.
- When the team convenes after each sprint, a status report called the “meeting minutes” will be assembled consisting of goals both complete and incomplete. This will allow the team to track progress over time as well as give the client a way to be kept up to date on the project’s status.
- Individual activity logs for each team member will be publicly available on the group’s web site. The client is able to see each member’s weekly progress and notes, in addition to the team progress reports and meeting minutes.