Knight's Guard

Software Requirements Specification

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	09/13/14	Miguel Corona	Made changes to Section 1 and Section 2
v0.2	09/15/14	Miguel Corona	Event Table, Use case, Use case descriptions, Specific Requirements 3.1 Functional Requirements 3.3 Physical Environment 3.4 User and Human factor 3.5 Documentation Require 3.6 Data Requirements 3.7 Resource requirements
v0.3	09/16/14	Jonathan Bennett	Added details to Specific Requirements, assisted in modifying use case diagram
v0.4	09/18/14	Miguel Corona	3.4 - 3.8 Details Added
v1.0	09/18/14	Jonathan Bennett	More details added to Section 3 (Specific Requirements), added Section 4 and definitions, finalized formatting.

Contents of this Document

Section 1: Introduction

Software to be Produced:

We are creating a multi-platform tower defense game, *Knight's Guard*. The game will have multiple levels for the user to play through. There will also be upgrades as the user progresses through the game.

Reference Documents:

- Concept of Operations
- Project Management Plan
- Test Plan

Applicable Standards

- Must follow standard design practices outlined for the Windows 7 and Android 2.3+ platforms, as determined by the operating system manufacturer. This includes the visual appearance, its functionality and packaging of compiled code.

Definitions, Acronyms, and Abbreviations

- GameMaker Studio: GameMaker is a video game creation platform, which uses the Game Maker Language (GML) as its primary scripting language. It allows for cross-platform deployment, which makes it suitable for the game to be released simultaneously on Microsoft Windows and Android platforms.
- GML: this stands for Game Maker Language, the programming language for GameMaker. The syntax for GML is similar to C, C++, and Javascript. It is interpreted similarly to Java's "JustInTime" compilation technique.
- HTML: HyperText Markup Language is the standard language used to create web sites. It instructs the web browser of what text and layout to choose to the viewer.
- CSS: Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a file written in a markup language, such as HTML. CSS is most commonly used with HTML to control the layout of a web site.
- PHP: PHP Hypertext Preprocessor (a recursive acronym) is a popular open source scripting language, which is suited for web development and web services.
- FPS: Frames Per Second. This is a measure of how many individual frames (images) are shown per second to the viewer. A higher frames per second creates a smoother and more lifelike experience.

Section 2: Product Overview

Assumptions:

- We are assuming that the tower defense game will be able to run on a normally configured Windows PC or Android device.
- We assume that the intended users will have some experience with the game type "Tower Defense." If not, they can read a basic introduction to the game and how it is played in the user manual.
- We assume that our game engine will properly export to multiple platforms, as it has been advertised to do. We are confident in this assumption as thousands of other games have successfully used these features of the engine.

Stakeholders:

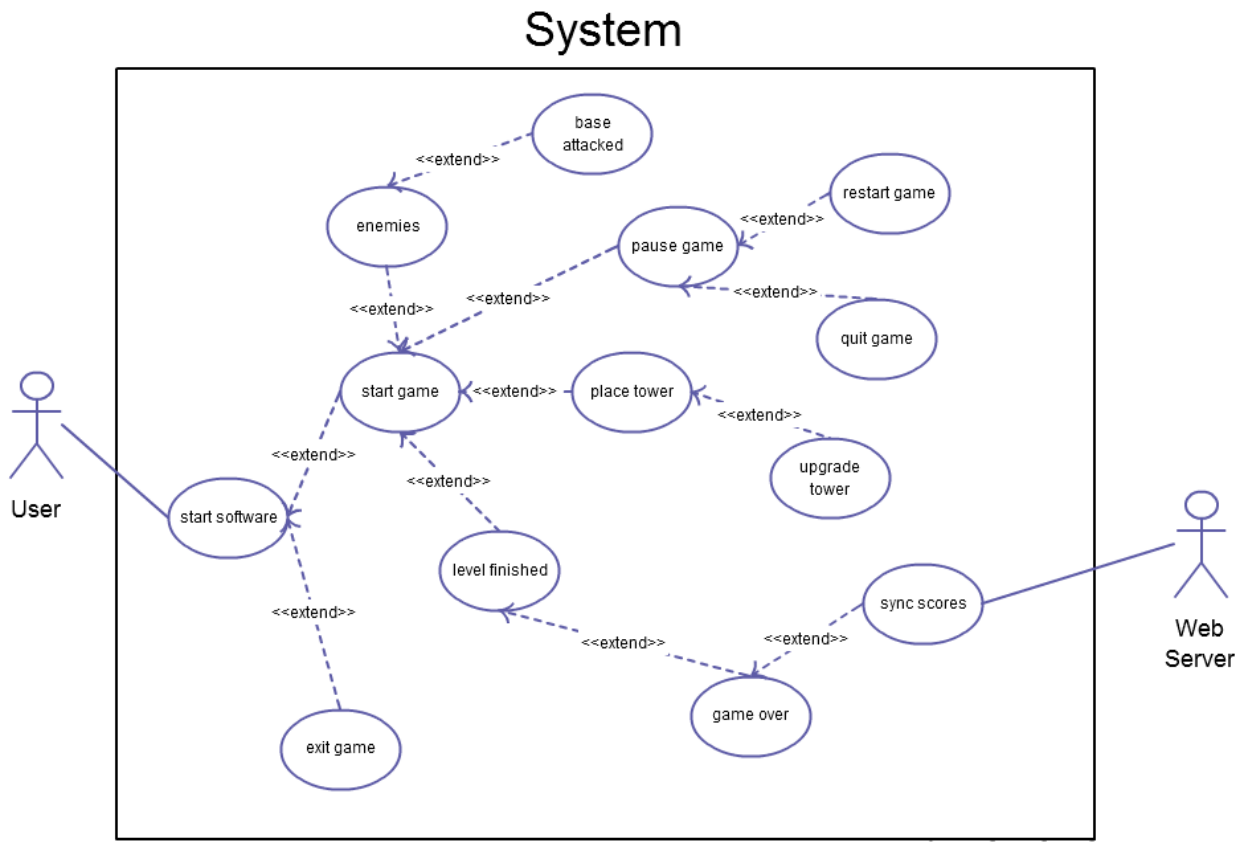
- Client - This is the company/individual we are developing the software for. In this case, the professor/TA is the primary client.
- Software Engineers - These are the people who create the tower defense game software.

- Users - The people who will be playing the application. This can include the Client, the Software Engineers, and other students or the general public.

Event Table:

Event Name	**External Stimuli**	**External Responses**	**Internal data and state**
Software Activation	Software Activation	Open User Interface	Load the game and prepare for start
Place Tower	User places tower in game	Place a Tower in the user specified location	Create specified tower
Start Game	User clicks “start game”	Game starts	Initiate game
Level over	Current level completed	Go to next level	Load next level prepare to start
End of game	Game completed	Display credits	Load credits and possible save score
Upgrade tower	User clicks tower and selects upgrade	Display possible upgrades	Wait for user to choose an upgrade, determine if user can upgrade
Restart game	User clicks “restart game”	Game restarts	Game wiped and starts over
Pause game	User presses “pause game”	Game pauses	Game pauses, current state maintained/saved
Base attacked	User sees Tower being attacked	Grade lowered	Grade lowered
Quit game	User clicks “Exit game”	Game closes	Exit program
Sync high score	Users clicks “Sync score with leaderboard”	Game syncs and displays success/fail message	Score saved to online leaderboard
Spawn enemy	Start of a new game/level/wave	New enemy created	New enemy created and stored along with the current enemies

Use Case Diagram:



Use Case Descriptions:

- Start Software: When the user starts the game program, loads dashboard to start or quit game.
- Exit Game: When the user opts to exit and close the software.
- Start Game: When the user opts to play the game.
- Place Tower: When the user decides to place a defense tower on the map to destroy enemies.
- Upgrade Tower: When the user wants to upgrade a tower already placed on map and has enough currency/experience.
- Level Finished: When the user has completed the current level and successfully defended the base.
- Game Over: When the user has completed every level and accomplished the goal of defending the base.
- Pause Game: When the user wants to temporarily halt the game.
- Restart Game: When the user has paused the game and wants to end the current game and start from the beginning.
- Base Attacked: When a generated enemy has made it past the tower defenses and damages the base building.

- Enemies: Once the user starts the game, enemies begin to generate semi-randomly and head toward the base.
- Quit Game: When the user opts to quit the current game.
- Sync Scores: The user can sync their high scores and see how it compares with the scores of other players from the online leaderboards.

Section 3: Specific Requirements

Each section listed below shall define the specific requirements of the system.

3.1 Functional Requirements

No: 4
Statement: The system shall allow the spawning of enemy units
Source: Client Requirement
Dependency: Requirement 2: the user must have started a new game
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game has been started, enemy units randomly generate
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 5
Statement: The system shall allow the user to place towers
Source: Client Requirement
Dependency: Requirement 2: the user must have started a new game
Conflicts: Towers cannot be placed where enemy units are currently located
Supporting Materials: None
Evaluation Method: This requirement is met when while the game has been started, the user can plac towers only on free pieces of the map
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 6
Statement: The system shall allow the user to upgrade towers
Source: Client Requirement
Dependency: Requirement 5: the user must have placed a tower already
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game has been started and a tower is on the map, the user is able to upgrade the tower.
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 7
Statement: The system shall allow the users base to take damage/be attacked
Source: Client Requirement
Dependency: Requirement 4: the system must allow the spawning of enemy units
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game has been started and a enemy is o the map, the the enemy is able to damage the base.
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 8
Statement: The system shall allow the user to pause the game
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game
Conflicts: None

Supporting Materials: None
Evaluation Method: This requirement is met when while the game is active, the user is able to pause the game and temporarily halt the playing process
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 9
Statement: The system shall allow the user to restart the game
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game, Requirement 8: the user must be able to pause the game
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game is active, the user is able to pause the game and temporarily halt the playing process. The user is then able to start the game from the very beginning as if they clicked the start game from the initial dashboard screen
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 10
Statement: The system shall allow the user the option to quit the current game
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game, Requirement 8: the user must be able to pause the game
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game is active, the user is able to pause the game and temporarily halt the playing process. The user is then able to quit the current game and go back to the main dashboard
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 11
Statement: The system shall allow the user to progress through levels
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game is active, the user is able to complete a level and more on to the next level.
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 12
Statement: The system shall allow the user to finish the game(Game Over)
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game, Requirement 11: the user must be able to beat each leve
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when while the game is active, the user is able play every level, and complete every level successfully.
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 15
Statement: Development of the system shall take place in an environment capable of coordinating wi a git repository
Source: Developer

Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met as long as the developers use
Revision History: Miguel Corona, 9/15/2014, Added Requirement

3.2 Interface Requirements

No: 1
Statement: The system shall allow the user to start the software
Source: Client Requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the game has reached the dashboard
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 2
Statement: The system shall allow the user to start the game
Source: Client Requirement
Dependency: Requirement 1: the user must have started the software
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the user is no longer on the dashboard. The user should be on the first “level”
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 3
Statement: The system shall allow the user to quit the software
Source: Client Requirement
Dependency: Requirement 1: the user must have started the software and be on the dashboard
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the user is able to exit and close the game from the starting dashboard
Revision History: Miguel Corona, 9/15/2014, Added Requirement

No: 14
Statement: The system shall allow the user to interact with the user interface
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software.
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the user is able to click/touch and initiate a new game. The game should respond and start within 1 second on average of tested devices, and take no longer than 5 seconds maximum on any capable device.
Revision History: ● Miguel Corona, 9/15/2014, added requirement ● Jonathan Bennett, 9/17/2014, added details

3.3 Physical Environment Requirements

No: 16
Statement: The game shall run on the Windows 7 operating system.
Source: Client Requirement

Dependency: Requirement 1: the system must allow the user to open the software.
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the user is to install and run the software on a licensed version of the Windows 7 operating system. The game must look as intended by the developers and perform smoothly, as indicated by a framerate of at least 20 frames per second (and preferably 30+).
Revision History: Jonathan Bennett, 9/17/2014, added requirement

No: 17
Statement: The game shall run on the Android 2.3+ operating system.
Source: Client Requirement
Dependency: Requirement 1: the system must allow the user to open the software.
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the user is to install and run the software on an official version of the Android 2.3+ operating system. The game must look as intended by the developers, which means it scales well to different screen sizes of smartphones and tablets. Graphics should be clear and text should be readable. The gameplay must perform smoothly, as indicated by a framerate of at least 20 frames per second (and preferably 30+). The game controls must be capable of responding to touch input.
Revision History: Jonathan Bennett, 9/17/2014, added requirement

3.4 User and Human Factors Requirements

No: 19 (3.4.1)
Statement: The system shall provide the casual (average) player a fun experience.
Source: Developer requirement
Dependency: Requirement 1: the user must have started the software and be on the dashboard
Conflicts: None

Supporting Materials: None
Evaluation Method: This requirement is met when the game is not too difficult to complete.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 20 (3.4.2)
Statement: The system shall provide the determined player a fun experience.
Source: Developer requirement
Dependency: Requirement 1: the user must have started the software and be on the dashboard
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the games difficulty can be chosen.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 21 (3.4.3)
Statement: The system shall be easy to learn to use/play
Source: Developer requirement
Dependency: Requirement 1: the user must have started the software and be on the dashboard, User experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when an average user is able to learn the game in under few minutes. We will consider our volunteer beta testers to be an average user, as they will not be part of the game development.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

3.5 Documentation Requirements

No: 22 (3.5.1)

Statement: The documentation shall be readable and posted online.
Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when all group interactions and meeting notes are poste online. Notes should be easy to follow. Documentation shall be posted on the group web site and in the internal group documentation directory.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 23 (3.5.2)
Statement: The video game shall provide a guide on how to install and play the game.
Source: Client requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the game comes with a guide on how to install and play the game (the user manual). We will verify that the guide is helpful by having new users follow th guide and ensure they are able to play the game successfully.
Revision History: Jonathan Bennett, 9/18/2014, Added Requirement

3.6 Data Requirements

No: 24 (3.6.1)
Statement: The system shall kept track of the number of units spawned.
Source: Developer requirement
Dependency: Requirement 4(3.1.1)
Conflicts: None

Supporting Materials: None
Evaluation Method: This requirement is met when the game is capable of knowing exactly how many enemy units have been spawned in over a given interval. The tester will verify that the variable count correct.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 25 (3.6.2)
Statement: The system shall kept track of the damage done to the base.
Source: Developer requirement
Dependency: Requirement(s): 4(3.1.1), 7(3.1.4), 2(3.2.2)
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the game is capable of tracking the damage done t the base. The tester will verify that the variable count is correct, based on the tower and the enemy that is attacking it.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 26 (3.6.3)
Statement: The system shall kept track of the number of levels completed successfully.
Source: Developer requirement
Dependency: Requirement(s): 11(3.1.8)
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the game is able to determine when a level is completed successfully.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

3.7 Resource Requirements

No: 27 (3.7.1)
Statement: The system shall be created by individuals with at least basic knowledge using GameMake studios.
Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when at least one group member has experience using GameMaker, and/or when the developer has completed the basic introductory tutorials provided by the GameMaker community.
Revision History: ● Miguel Corona, 9/18/2014, Added Requirement ● Jonathan Bennett, 9/18/2014, added more details

No: 28 (3.7.2)
Statement: The system shall be created by individuals with experience coding.
Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the group members have experience coding. As th entire development team is composed of senior computer science students who have passed the UCF foundation exam, this requirement is automatically met.
Revision History: ● Miguel Corona, 9/18/2014, Added Requirement ● Jonathan Bennett, 9/18/2014, added more details

No: 29 (3.7.3)

Statement: The system shall be created using GameMaker Studios
Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when GameMaker Studio is used to create the system. Each developer shall have a copy of GameMaker Studio on their individual machine.
Revision History: ● Miguel Corona, 9/18/2014, Added Requirement ● Jonathan Bennett, 9/18/2014, added more details

3.8 Security Requirements

No: 18 (3.8.1)
Statement: The system shall prevent the malicious change of scores
Source: developer
Dependency: Requirement 1: the system must allow the user to open the software, Requirement 2: the system must all the user to start a game, Requirement 11: the user must be able to beat each leve
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when the developers have ensured that the score keepin system is secure by testing for common security exploits and by following standard secure coding practices. A user should not be able to edit anyone's score, including their own, except by normal play of the game as intended.
Revision History: ● Miguel Corona, 9/15/2014, added requirement ● Jonathan Bennett, 9/17/2014, added details

No: 30 (3.8.2)
Statement: The system shall prevent users from changing system settings.

Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when users are unable to change the games internal settings. The testing team will attempt to modify these settings in an unauthorized way.
Revision History: ● Miguel Corona, 9/18/2014, Added Requirement ● Jonathan Bennett, 9/18/2014, added more details

3.9 Quality Assurance Requirements

No: 31 (3.9.1)
Statement: The system shall be easy to navigate and readable.
Source: Developer requirement
Dependency: Developer experience
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met when an average user is able to navigate through the games menus/settings without any difficulty. The text should be readable on screens of all sizes, since the game will be playable on smartphones, tablets and desktops/laptops.
Revision History: Miguel Corona, 9/18/2014, Added Requirement

No: 32 (3.9.2)
Statement: The video game must play smoothly.
Source: Developer requirement
Dependency: Developer experience
Conflicts: None

Supporting Materials: None
Evaluation Method: This requirement is met the video game runs at a minimum of 20 frames per second on the typical Windows 7 or Android 2.3+ platform. The ideal goal is to reach 30 frames per second or higher on average.
Revision History: Jonathan Bennett, 9/18/2014, Added Requirement

No: 33 (3.9.3)
Statement: The video game must be well tested.
Source: Client requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met the video game has been tested thoroughly by the development team and beta testers. A “thorough” test means that every function and unit (tower, enemy, etc.) has been tested and verified to be working correctly by at least two people.
Revision History: Jonathan Bennett, 9/18/2014, Added Requirement

No: 34 (3.9.4)
Statement: The video game input should feel responsive.
Source: Client requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met the input delay of the keyboard/mouse/touch screen is less than 0.1 seconds to respond. This means that when a user clicks their mouse or taps their screen, the user should be able to visually see a change on screen in less than 0.1 seconds on average, with a maximum time of 0.5 seconds.
Revision History: Jonathan Bennett, 9/18/2014, Added Requirement

No: 35 (3.9.5)
Statement: The project must meet the client's expectations.
Source: Client requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: This requirement is met the client approves of the progress and completion of the project. To help ensure its success, the development team is using the Agile methodology and will be communicating regularly (at least once every two weeks) with the client to demonstrate progress of the project.
Revision History: Jonathan Bennett, 9/18/2014, Added Requirement

Section 4: Supporting Material

Attached below is the proposal for the video game, which includes details on the platform and gameplay. Meeting notes are also attached which explain some of the ideas and features that will be in the game.

Additional details for the project and its management can be found in the "Project Management Plan" and "Concept of Operations" documents.

Group 1 Proposal

Knight's Guard

Product 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.

Features

- Real-time action Tower Defense game type.
- The game takes place on the UCF campus map. The player's goal is to defend the Student Union against attacks. The player is able to build defensive towers around the map which have a variety of weapons and abilities.
- The player is given a grade based on wow well they defends the base. Like university, the best grade possible is an A, while the worst is a F.
- CPU A.I. controlled enemies include characters based on germs (represents student illness), emergencies, and tough assignments. These enemies will either attack the towers, or focus on attacking the Student Union.
- If the player is feeling overwhelmed by enemies, they can restart the game by taking a Withdrawal. The player can beat the level by surviving a set amount of time in the game, known as a Semester.
- With successful play, the player can will become more powerful, gain new abilities and towers to help fight off the enemies. Each tower type has its own special ability to help fight off enemies.

Platform(s)

- **Microsoft Windows 7+,** a popular operating system for home users and businesses. The game will first be tested and made playable for the Windows platform, with the intention of making it playable for mobile as well:
    - **Android**, a mobile operating system, developed by Google and based on the Linux kernel. The game is intended to work on both mobile phones and tablets running the Android operating system. We plan to support version 2.3+.

Language & Game Engine

- **GameMaker: Studio**: version 1.2+. GameMaker is a video game creation platform, which uses the Game Maker Language (GML) as its primary scripting language. It allows for cross-platform deployment, which makes it suitable for the game to be released simultaneously on Microsoft Windows and Android platforms.
- **GML:** the programming language for GameMaker. The syntax for GML is similar to C, C++, and Javascript. It is interpreted similarly to Java's "Just-In-Time" compilation technique.

COP4331C – Group 1
September 5, 2014
Meeting Notes

- Finalization of Prior Data, Decision of what program to use.
- List of things to look into:
    - GM: Source Control - *Professional Version Only*
    - Unity's 2D system
    - GM: UI quality
- Game Data
    - Vectorized UCF map - add details if we want to stylize, and use that as our terrain
    - -No player - automated turrets + 'skill based' click to use weapons (more powerful, one per 'base')
    - Create multiple skill weapons in different themes.
    - How many weapons per building? We know one skill based, but how many regular?
    - Minimal versus Ideal version of game
    - Create one auto weapon for minimal, but more than that for ideal. Same towers for each building, but the skill weapon changes per base.
    - Currency of some sort - Knightcash - powerup called 'financial aid' that will add more cash - what about an enemy that is like unsubsidized loan and every time it shoots the base you lose money instead of health.
    - EXP meter for unlocking things, currency to purchase.
    - Both EXP and money come from enemy death.
    - Bonuses for fast level completion? Would require additional testing for deciding the balance of that feature. [Ideal Game]
    - AoE towers, single target, long range, short range, status effect like slow, freeze, poison. - towers that just do status effect, or effect AND damage?
    - Double-triple-etc kill could provide multipliers/benefits - on death event starts a timer that tracks how many are killed in that time.
    - Enemy types - long range, short range. Flying/Swimming units that have different terrain rules?
    - Have a moat around the student union? Final line of defense?
    - Enemy damage types: melee, elemental?
    - Enemies themselves: Sickness, Assignment, Parking ticket, Unsubsidized Loan, Family Emergency
    - Focus on non-touch first, but keep the UI so that it can easily port to touch.
    - Balance types:
        - Assignments are slow but heavy damage, melee range only. High health. We could do different levels of assignments as well for scaling. Could also do like midterm/final bosses that come out halfway through a round/at the end of the round. Or start them all at the same time and change their speeds. Targets base.

    - Sicknesses - ranged or melee, not as much damage but status effects like fire slower or are poisoned. Targets towers.
    - Emergencies - ranged.
    - Unsub Loan - don't do any actual damage, but take away money. Parking ticket falls in this category (flying type ticket?) Melee range. Automated defenses won't take care of it, have to use skill shots. (not many of them per round)
- Towers rotatable, field of view, can change during play but that's all you can do to influence their workings, but they can be upgraded to have larger cones. 360 instead?
- Weapons centralized to buildings? or placeable around the path?