Knight's Guard

Test 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	09/13/14	Katie Jurek	Filled out Overall Objective, Test Environment, Stopping Criteria, and Test Cases
v0.2	09/13/14	Jonathan Bennett	Added the Introduction and details to Test Environment, Stopping Criteria, and Test Cases.
v1.0	09/16/14	Jonathan Bennett	Added an additional Test Case and final formatting.

Contents of this Document

Introduction

The purpose of this document is to explain the details of the development team's test plan, which will be followed to aid in the creation of a polished and enjoyable video game that is devoid of errors.

Overall Objective for Software Test Activity

We intend to verify the stability of our game software in the environment we wish to use it in; that is, PC (Microsoft Windows 7 operating system) or eventually Android (version 2.3+). It must at the very least be able to work offline for a single player.

Reference Documents

- Concept of Operations
- Project Management Plan
- Software Requirements Specification

Description of Test Environment

The test environment is GameMaker: Studio on the developer's side, and Microsoft Windows 7 on the user's side. The testers include the developers (our development team) and volunteer users.

The volunteer users will not see the test environment; instead, they will run an .exe on a Windows 7 system that invokes the code made by the developers in the test environment. The game will be tested on a minimum of 10 different PCs, all using different hardware configurations and video display settings. This is to ensure the game is compatible with many different computers.

If the Android version of the game is finished on time, the developers and volunteer users will test the game on a variety of Android devices running different versions of the operating system. The hardware will consist of common smartphones, such as the Galaxy S5, along with tablets such as the Nook and Galaxy Tab.

Stopping Criteria

This is how we will handle testing the software:

- If errors are found during testing:
    - Each time a problem is found, we will stop and immediately attempt to fix that problem. The problem will be logged into a bug report. This will help prevent errors from carrying over into other areas of the project code, or being forgotten.
- If no errors are found during initial testing:
    - Testing must still be done even when no errors are found during development, because there is always the possibility that edge cases weren't considered or invoked in one test playthrough of the game.
    - If no errors are found, the development team will meet to discuss the possibility of more edge cases to take care of, and only if we cannot think of more will we move on from this part to continue with the next coding task.

- How do we define "good enough to deliver"?
    - The game must have met our expectations; that is, the game must run, be playable as a tower defense, and have damage, enemies, waves, and other game features working as expected. There must be no known errors that noticeably affect gameplay.
    - However, there may be minor cosmetic errors or errors where a workaround has been established. These errors will be logged and given a plan to fix. Since minor cosmetic issues do not affect the gameplay, they are not considered a blocker for release.

Description of Individual Test Cases

These initial set of test cases will cover the major features of the game. The project management is using the Agile methodology, which creates the likelihood of additional test cases being made over the span of the product development, as major and minor features are added and subject to possible changes.

Test Case 1: Starting the next wave

- **Test Objective:** When the player presses Space and the amount of enemies currently on the map is zero, then the next wave (or round) begins. Each round is divided into a number of waves.
- **Test Description:**
    a. There will be a variable that counts the number of enemies still on the map.
    b. The tester shall verify that this number is accurate, given the number of enemies on the screen.
    c. If this number is not zero, then pressing Space will do nothing.
- **Test Conditions:** See Test Environment; Windows 7 and a keyboard.
- **Expected Results:** The game correctly moves to the next round.

Test Case 2: Choosing the tower made available to add to the map

- **Test Objective:** Player will be able to build a certain tower based on the number of enemies killed so far.
- **Test Description:**
    a. There will be a variable that counts the number of enemies the player has killed with their towers.
    b. Every time this variable reaches a certain number, the player will be able to build one new tower in the spot of their choice
        - Except at the beginning, where the player will be allotted a few free towers to start their defense.
    c. The tester will play the entire level to verify that the towers only become available when they are supposed to (after the set amount of enemies have been killed).
- **Test Conditions:** See Test Environment.
- **Expected Results:** The game correctly calculates which tower the player can build based on the number of enemies killed.

Test Case 3: Adding towers to the map

- **Test Objective:** Player can add towers to the map to defend their buildings.
- **Test Description:**
    a. Player will be able to add towers in a grid-based fashion.
    b. When the player hovers over a square that is free from obstructions, they will see a green highlight to show them they can place a tower there.
        - Tester shall observe an unobstructed square, and verify the highlight color is green.
    c. If they cannot place the tower there because there is an obstruction, the area will be red instead.
        - Tester shall observe an obstructed square, and verify the highlight color is red.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The player successfully adds a tower to the map that can help defend the player's buildings.

Test Case 4: Disallowing player from blocking the enemy's path with towers

- **Test Objective:** The player cannot place a tower in a spot that would block the enemy's path to the player's buildings.
- **Test Description:**
    a. Players can add towers to the map while enemies are moving through the map.
    b. At no point will a player be allowed to place a tower if placement of the tower in that spot would entirely block the enemy's path.
    c. This will be calculated by making sure the enemy's A* path can always find a route to the player's buildings.
    d. A message will be shown telling the player they cannot place the tower there.
    e. The tester shall try to block the enemy's path with a tower at multiple places.
    f. The tester will verify that the enemies are always able to find a route around the tower.
- **Test Conditions:** See Test Environment.
- **Expected Results:** If the player tries to place a tower in an invalid spot, a display message will be shown and they cannot build the tower there.

Test Case 5: Calculating enemy strength

- **Test Objective:** Enemies will become stronger depending on the round and wave number.
- **Test Description:**
    a. Each subsequent round of each subsequent wave will get gradually more difficult.
    b. This is done by increasing the hit point and speed variables of the enemies.
        - The tester shall verify that each enemy spawned has a higher hit point (health) and speed value.
    c. Some waves of enemies will be more powerful simply because there are more enemies spawned, and other waves will be stronger because the enemies actually get stronger.
    d. The increases in hit points and speed are calculated by a function of the round and wave numbers.
        - The tester shall verify that the calculated values match the expected values.
- **Test Conditions:** See Test Environment.

- **Expected Results:** The enemies will grow stronger over time.

Test Case 6: Spawning the enemies

- **Test Objective:** Enemies will spawn based on the wave number.
- **Test Description:**
    a. During each round, a certain number of enemies will spawn.
    b. The number spawned will be a function of the round number; at higher rounds, more enemies will be spawned.
    c. The tester shall verify that the number of enemies spawned matches the expected value based on the game design documents. The entire level(s) must be played to verify these numbers.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The enemies spawn at the correct location when each wave of each round begins.

Test Case 7: Controlling enemy spawn rate

- **Test Objective:** Enemies will spawn at a certain rate based on the type of enemy and the wave number.
- **Test Description:**
    a. When enemies are spawning, they must not spawn too quickly nor too slowly, for this would adversely affect the difficulty of the game.
    b. Instead, a middle-ground must be found: as the player reaches higher rounds and waves, the enemy's spawn rate will incrementally increase.
    c. This will also be dependent upon the type of enemy.
    d. The tester shall verify that the spawn rate of every enemy in the game matches what is specified in the design documents. Furthermore, the subjective opinions of testers will be collected to help balance the game for difficulty and enjoyment.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The enemies will spawn faster at higher rounds and slower at lower ones.

Test Case 8: Enemies move toward the player's buildings

- **Test Objective:** After spawning, the enemies will begin moving using A*.
- **Test Description:**
    a. During each round, after the enemies spawn, they will use the A* algorithm to move toward the player's buildings in order to destroy them.
    b. The tester shall verify that each enemy type on every level is moving in the expected way.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The enemies begin and continue moving toward the player's buildings.

Test Case 9: Recalculating enemy paths after a tower is placed

- **Test Objective:** After each tower is placed, enemies will recalculate the path they use to reach the player's buildings.
- **Test Description:**

a. Since the player is allowed to place towers while the game is in movement, the enemies must be able to react to these new tower placements.
b. They will do so by recalculating their path to the player's buildings each time a new tower is placed.
c. The tester shall verify that each enemy type on every level is moving in the expected way with every tower type that is available to be placed.

- **Test Conditions:** See Test Environment.
- **Expected Results:** The enemies correctly calculate a path to the player's buildings even when a new tower is placed.

Test Case 10: Calculating the strength of the towers' projectiles

- **Test Objective:** The towers' projectiles' strength will be dependent upon the type of tower built.
- **Test Description:**
    a. Each individual tower has a certain type of projectile.
    b. Some projectiles are faster than others, some are larger, and still others are more accurate.
        - These values will be chosen depending on the type of tower.
    c. The tester shall verify that each projectile's attributes are set correctly, as according to the design documents.
- **Test Conditions:** See Test Environment.
- **Expected Results:** Each tower has a unique projectile with its own strengths and weaknesses.

Test Case 11: Spawning the towers' projectiles

- **Test Objective:** Player's towers will shoot projectiles at enemies when they come close enough.
- **Test Description:**
    a. Each tower the player has placed will shoot projectiles at the enemies. These projectiles will increase in strength based on the type of tower that was built.
    b. There will be a set interval between the tower shooting projectiles where it does not shoot another projectile.
    c. The tester shall verify that this interval is correct for every tower type, as detailed in the design documents.
- **Test Conditions:** See Test Environment.
- **Expected Results:** Each tower, when an enemy comes within its attack range, will shoot projectiles as defense.

Test Case 12: Projectiles damage enemies if hit

- **Test Objective:** Enemies will be damaged upon contact with projectiles.
- **Test Description:**
    a. Player's towers will shoot projectiles aimed at the enemies.
    b. If the projectiles hit them, the enemies will briefly turn a different color and have their hit point variable reduced.
    c. The tester shall verify that the enemy's health is lowered whenever hit by a projectile, and that it is lowered by the correct amount according to the tower that shot the projectile.

- **Test Conditions:** See Test Environment.
- **Expected Results:** Enemies will have fewer hit points after being hit by a projectile than before.

Test Case 13: Destroy the projectiles if they go off-screen

- **Test Objective:** Destroy unnecessary projectiles.
- **Test Description:**
    a. When a tower shoots a projectile, that projectile will disappear if it hits and damages an enemy.
    b. If it does not, and moves off-screen instead, then it can be removed to avoid bogging down performance.
    c. The tester shall verify that the projectiles disappear when moved off screen, and that the performance of the game plays as expected.
- **Test Conditions:** See Test Environment.
- **Expected Results:** Unnecessary projectiles will be removed from memory.

Test Case 14: Winning the game

- **Test Objective:** Player wins the game.
- **Test Description:**
    a. If the round and wave numbers are above a certain value and the player has not yet lost, then there are no more rounds or waves left and the player has won; a winning message will be displayed.
    b. The tester shall verify that the correct winning message is displayed after they finish the level.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The player wins the game and can exit.

Test Case 15: Losing the game

- **Test Objective:** Player loses the game.
- **Test Description:**
    a. If all of the player's buildings have been destroyed, or the player's base (the Student Union) is destroyed, then a losing message will be displayed and the player will be able to restart the game.
    b. The losing condition will be based on a variable that counts the number of buildings the player has, and if it reaches zero, this condition is triggered.
    c. The tester shall verify the correct losing message is displayed when the appropriate buildings are destroyed.
    d. The tester shall verify that the variable condition is being triggered correctly.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The player loses the game and can choose to restart.

Test Case 16: Graphics display correctly

- **Test Objective:** Verify that the graphics display correctly on multiple video cards and computer configurations.
- **Test Description:**

    a. On each test computer, every menu and level needs to be played and visually inspected to ensure every piece of game artwork (map, towers, enemies, etc.) are displayed correctly.
    b. A correct display means that each piece is shown using the expected colors, animation, size, frame-rate (smoothness of gameplay), and resolution.
    c. The tester(s) shall verify that each aspect of the graphics is displaying correctly.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The graphics of the video game display consistently across all machines.

Test Case 17: Able to sync leaderboard score

- **Test Objective:** Give the player the ability to sync their score to the leaderboard server.
- **Test Description:**
    a. In order to increase competition among human players, a player can save their game score to an online leaderboard, where their name and score will be shown to other players.
    b. The tester shall verify that their end game score is correctly synced to the server. This will require checking both the game and server log to ensure the data is synced correctly on both ends.
- **Test Conditions:** See Test Environment.
- **Expected Results:** The game and web server will contain the same data for the leaderboards.