Testing Document
TOWER OF ARKADY
Scott Richards, David Warthen, Nic Ward, Asher Yusim, Samantha Steinberg, Sara Turner
November 25, 2014

SECTION 1. INTRODUCTION

Tower of Arkady is a top-down puzzle game with a focus on speed. The player arrives at the tower and learns of a hostage on the top floor in need of rescue. The game becomes progressively more difficult as the player travels from one floor to another, moving through mazes, extinguishing fires and dodging bullets before diffusing a bomb to rescue the hostage.

This game is for entertainment purposes only and is specifically designed for the Game Expo on December 11. We expect the audience to consist primarily of college-aged students with a small amount of time available to play the game. As such, Tower of Arkady was designed to be short, challenging and enjoyable.

SECTION 2. OVERVIEW OF THE GAME

The game begins as the player arrives at the Tower of Arkady and learns of a hostage in need of rescue. Before entering the tower, the player is briefed on game movement and objectives, and a list displays high scores. The timer will begin counting down when the player enters the tower.

Tower of Arkady is a graphics game with dramatic music and varying sound effects as the player moves through the game and interacts with objects. A pause feature is excluded from the game as a deliberate design chose so players must move quickly through the game to earn a spot in the high scores. As an added incentive to avoid game hazards, including fire and bullets, additional time will be removed from the game clock each time the player collides with a game hazard. Additionally, time will be removed to the game clock each time an incorrect wire is cut while diffusing the bomb. The game ends, and the timer stops, only when the player diffuses the bomb at the top of the tower. High scores are earned by players who complete the game with the largest amount of time left on the play clock.

SECTION 3. TOP LEVEL FAILURE MODES

Describe the various ways that a user might cause the program to malfunction. For example, if the user needs to type specific commands at some part of the program, you should test to see what happens if the user types an incorrect command. You should determine whether the user can damage something or become frustrated by making any type of error. You should list the types of errors and the way that the program should deal with them. After you identify these top level errors, you should devise tests that will help you uncover situations where your program does not handle

Users could possibly cause the game to malfunction by skipping mechanics and just running towards doors, by trying to use/pick up items they are not supposed to. Also what could happen if the user traps himself, will he be able to just go to the end game screen or have to wait for the timer. Also the user could cause errors by trying to run back through levels or trying to get out of boundaries of the wall.. The function to put out fires could cause a malfunction when used on tiles that aren't on fire. Out of bounds errors, mechanics errors and item errors can break the game.

SECTION 4. MODULES AND TESTING

Describe the modules that your game uses. This could include such things as graphics.py, Ren'Py, Pygame or other modules in the Python libraries. For modules such as these that have extensive user communities, you don't need to do additional testing. You can just identify them and note where you can find information about errors in these packages. If you are using modules that you have written yourself or that you obtained from other people that are not widely tested, be sure to describe the testing that you will do before you include the module in your program.

We will be using the modules Pygame, PyMap, and sqlite3. But also writing our own for Character, Gameplay Window, Interactive Objects, and so forth. Our own modules will be tested by the methods that I mention later. PyMap is widely used and well tested and I can find information on it from https://code.google.com/p/pymap/. Pygame has their website http://www.pygame.org/news.html. And information on Sqlite3 can be found at http://www.sqlite.org/testing.html.

For the modules written by us I will test using the following methods:

Version 1 will be tested for being able to walk through the walls, walking back through to the other stages of Tower of Arkady Finished Dec. 2 (any additional changes they make will be tested after fixed for all tests)

These will be tested by:

- Using WASD keys to move into walls
- Press all keyboard functions to see if they effect any errors
- Move back through the door at the start of the floor

Version 2 will be trying to use items on interior walls, trying to move through all the obstacles from different directions. Dec. 2

- Move next to wall
- Activate any items I have on the interior walls
- Move to obstacles
- Attempt to move through the obstacles
- Try out all keyboard functions [all keys on my keyboard. To see if anything triggers something]

Version 3 will be tested for being able to completing a level then moving on and trying to go back, also what would happen if the user just runs straight to the next floor, ignoring the objects, and the key(s). Dec. 2

- Complete the levels
- Try to move back to old levels

- Avoid mechanics and press keyboard functions
- Try to open the doors without the key

Version 4 will be tested for being able to move objects into exterior and interior walls, pushing the move-able objects over other mechanics, using items on the move-able objects and the walls. Also if there are multiple bombs on one level trying to make the bombs blow each other up, what happens if the bomb blows up with you right next to it. Dec. 5

- Move objects into unmovable obstacles
- Try to move objects over bombs, fire, etc.
- Try to activate items on move-able objects.
- If multiple bombs are on the same level place them next to each other and trigger them
- Try to blow myself up with bombs

Version 5 will be tested if I trap myself what will happen, I will try to test if I get the same time twice which one goes up higher on the leader board. I will also try to test what happens if I never do anything and just let the clock run. Dec. 5

- Trap myself in move-able objects, and obstacles.
- Play the game through twice with the exact same time and see how the high score system reacts
- Stand in mechanics to cause the time to just keep going up
- Sit inactive

SECTION 5. TOP LEVEL TESTING

Indicate the testing that you will do to make sure that your game is working correctly. Indicate whether you will have testers other than yourself. It is always a good idea to get as many different people as possible to try to break your game so that you can make the game as dependable as possible.

I will be testing each version of the game after it is finished so that I can get back to the coders as quick as possible about any errors that need to be adjusted. I will also see if I can make the game lag, and what will happen if its done. Also test if I can find a way to pause it. Also when I get to the completed playable version of the game I will use some of my friends that play a lot of games that have experience playing beta's.

SECTION 6. IMPLEMENTATION PLAN AND TIMELINE

Give a schedule for testing. Make sure that you allocate enough time and effort to do a good job. Include time for fixing errors that you will find, since you will almost certainly find errors of all sorts.

TIMELINE FOR BUILDING THE

GAME (for your reference)			
Version	Capabilities	Time Required	Completion Date
1	Five floors that the sprite can walk through.	6 hours	Dec. 1
2	Five floors with objects that the sprite must walk around and the addition of interior walls.	6 hours	Dec. 1
3	Implementation of key and locked door to move to another floor.	6 hours	Dec. 1
4	Implementation of mechanics, including bombs and moveable objects.	10 hours	Dec. 4
5	Implementation of the game clock and high scores.	3 hours	Dec. 4
6	Refinement and debugging for dress rehearsal.	4 hours	Dec. 8
7	Final revisions for Game Expo.	4 hours	Dec. 10

Version 1 will be posted by Dec 1st, so I will try to get my preliminary tests completed the next day so that I can get back to them as quickly as possible.

Version 1 will be tested for being able to walk through the walls, walking back through to the other stages of Tower of Arkady Finished Dec. 2 (any additional changes they make will be tested after fixed for all tests)

Version 2 will be trying to use items on interior walls, trying to move through all the obstacles from different directions. Dec. 2

Version 3 will be tested for being able to completing a level then moving on and trying to go back, also what would happen if the user just runs straight to the next floor, ignoring the objects, and the key(s). Dec. 2

Version 4 will be tested for being able to move objects into exterior and interior walls, pushing them over other mechanics, using items on the move-able objects and the walls. Also if there are multiple bombs on one level trying to make the bombs blow each other up, what happens if the bomb blows up with you right next to it. Dec. 5

Version 5 will be tested if I trap myself what will happen, I will try to test if I get the same time twice which one goes up higher on the leader board. I will also try to test what happens if I never do anything and just let the clock run. Dec. 5

Version 6 I will rerun all the previously mentioned tests, and test any refinements that are done in this version. Dec. 8 and 9th

SECTION 7. BIBLIOGRAPHY

List any books, manuals or other resources that you used to construct the testing document. (I don't really know what to say here? The testing document?)

Sara Turner 11/27/2014 2:50 AM

Comment [1]: I'll stick the Pygame, PyMap and Sqlite3 links here.

SECTION 8. WORK ASSISTANCE STATEMENT

As usual, write a work assistance statement for the document.

The testing document was assembled with assistance from members of Team 11: Scott Richards, David Warthen, Nic Ward, Asher Yusim, Samantha Steinberg and Sara Turner.

Some final thoughts about the Testing Document.

The document must be professional in appearance. Use a word processor and make sure to use the spelling checker and grammar checker. Be sure to give credit to anyone and any source that you use. The notes for Lecture 12 illustrate some of the tests that you might have for a simple calculator. You should look at that example and adapt it to your game.