***Dankest Dungeon Final Report***



**CS 440**

**at the**

**University of Illinois Chicago**

**December 2020**

**Table of Contents**

[List of Figures 6](#_Toc57072990)

[List of Tables 7](#_Toc57072991)

[I. Project Description 8](#_Toc57072992)

[1. Project Overview 8](#_Toc57072993)

[2. Project Domain 8](#_Toc57072994)

[3. Relationship to Other Documents 8](#_Toc57072995)

[4. Naming Conventions and Definitions 8](#_Toc57072996)

[a. Definitions of Key Terms 8](#_Toc57072997)

[b. UML and Other Notation Used in This Document 9](#_Toc57072998)

[c. Data Dictionary for Any Included Models 11](#_Toc57072999)

[II. Project Deliverables 12](#_Toc57073000)

[5. First Release 12](#_Toc57073001)

[6. Second Release 13](#_Toc57073002)

[7. Comparison with Original Project Design Document 13](#_Toc57073003)

[III. Testing 14](#_Toc57073004)

[8. Items to be Tested 14](#_Toc57073005)

[9. Test Specifications 14](#_Toc57073006)

[10. Test Results 17](#_Toc57073007)

[IV. Inspection 20](#_Toc57073008)

[11. Items to be Inspected 20](#_Toc57073009)

[12. Inspection Procedures 20](#_Toc57073010)

[13. Inspection Results 22](#_Toc57073011)

[V. Recommendations and Conclusions 23](#_Toc57073012)

[VI. Project Issues 24](#_Toc57073013)

[14. Open Issues 24](#_Toc57073014)

[15. Waiting Room 25](#_Toc57073015)

[16. Ideas for Solutions 26](#_Toc57073016)

[17. Project Retrospective 27](#_Toc57073017)

[VII. Glossary 28](#_Toc57073018)

[VIII. References / Bibliography 28](#_Toc57073019)

[IX. Index 28](#_Toc57073020)

### ****List of Figures****

[Figure 1: Game UML 10](#_Toc57072384)

[Figure 2: Initial Level Design 1 12](#_Toc57072385)

[Figure 3: Initial Level Design 2 13](#_Toc57072386)

[Figure 4: Final Level Design 13](#_Toc57072387)

### ****List of Tables****

[Table 1: Game Database 11](#_Toc57072391)

# Project Description

Dankest Dungeon is a mobile application created by Bohn Jell Entertainment. It is a 2D adventure game where players can explore dungeons to escape the dungeon. Be quick, or monsters will eat you alive!

## Project Overview

Dankest dungeon is an adventure game where the player will need to explore different rooms within a dungeon to escape. The game consists of different chests, doors, walls, monsters, power-ups, and more that will help the play navigate their way through the system. This is a simple game that any player of any age can understand by tapping through their mobile device.

## Project Domain

The game is a mobile game, designed for Android and iOS for now. All the player needs to do is tap on the screen to press on menu option, move through tiles in the game, or write commands in the command bar. In future releases, we may consider adding the games to desktops so that players of all background can fully benefit from the game.

## Relationship to Other Documents

Within this document we will occasionally reference the technical report written by Arthur Mezheritskiy, Bennett Maciorowski, Chris Lee, and Ovidiu Bahnean by the name of “Dungeons and Dank” from February 2018. Of specific importance for this report, the sections labelled:  
1. Project Overview  
2. The Purpose of the Project  
3. The Scope of the Work.  
4. The Scope of the Product.

## Naming Conventions and Definitions

The below sections serve as a sort of dictionary that defines the “language” of Dankest Dungeon. Key terms are created deliver quick and concise communication.

### Definitions of Key Terms

*Motivation*

Naming convention is essential to every game. Some words are shortened so that players can quickly communicate to each other during times of urgency. Providing the player with some insight of the terminology should help the player understand the different “language” that exists within the game.

*Examples*

DGN: This is the shortened version of the word “dungeon”. “Dungeon” is a commonly used word in the game, and is shortened for quick communication. One may say: “Help, how did you pass that DGN?”

GC: This is the shortened version of “gold coins”. In every level, you can pick up coins from tiles, or find them in the chests. When you enter the shop, you buy items or power ups with GC.

HP: This is the shortened version of “hit points”. Enemies chase you around the map and if they reach you, you will lose hit points. You are only allowed to be attacked by an enemy some finite number of times before having to restart the level.

INV: This is the shortened version of the word “inventory”. All items such as keys, power ups, and other items are stored into the inventory and can be used for future use.

LVL: This is the shortened version of the word “level”. Providing a short version of this word can save space on the screen when we have to display the word. It coincides with the difficulty of the level. For example: “LVL 6” is more difficult than “LVL 4”.

PTS: This is the shortened version of the word “points”. This is synonymous with the score that player has in the top right corner of the screen. The score can be increased as the player explores the map.

REST: This the shortened version of the word “restart”. When the player has lost all their hit points, we can say that they’ve been hit with a restart. They have to start at the beginning of the level again. If someone were to lose all their hit points, they might respond with: “I’ve been rested!”

TP: This is the shortened version of the word “teleport”. There is an item in the game where you can teleport to different tiles within the dungeon. The shop keeper will sell this item as “TP spell”.

+: This is the shortened version of “power up”. Power ups allow you to move to different tiles in a special way, let you attack enemies, and more. If a player were to refer to the teleport power up, they may respond with: “TP+”.

### UML and Other Notation Used in This Document

To keep things organized, we created a UML diagram to showcase the structure of our project. UML diagrams make is so that anybody can get a general idea of how the blueprint of the project looks. If something needs to be pointed out during a meeting with a client or a scrum team, then it is easy to see what and which parts of the project would be affected if some change is made.

*Content*

The UML diagram is made up of many different classes. We have three classes for the levels, five for the different types of tiles, and a main driver class. The three levels are grid panes that are made up of tiles. We have four different types of tiles. We have a wall tile, a player tile, a door tile, and a chest tile. Each level can be made up into a combination of different tiles. The main driver will run the application.

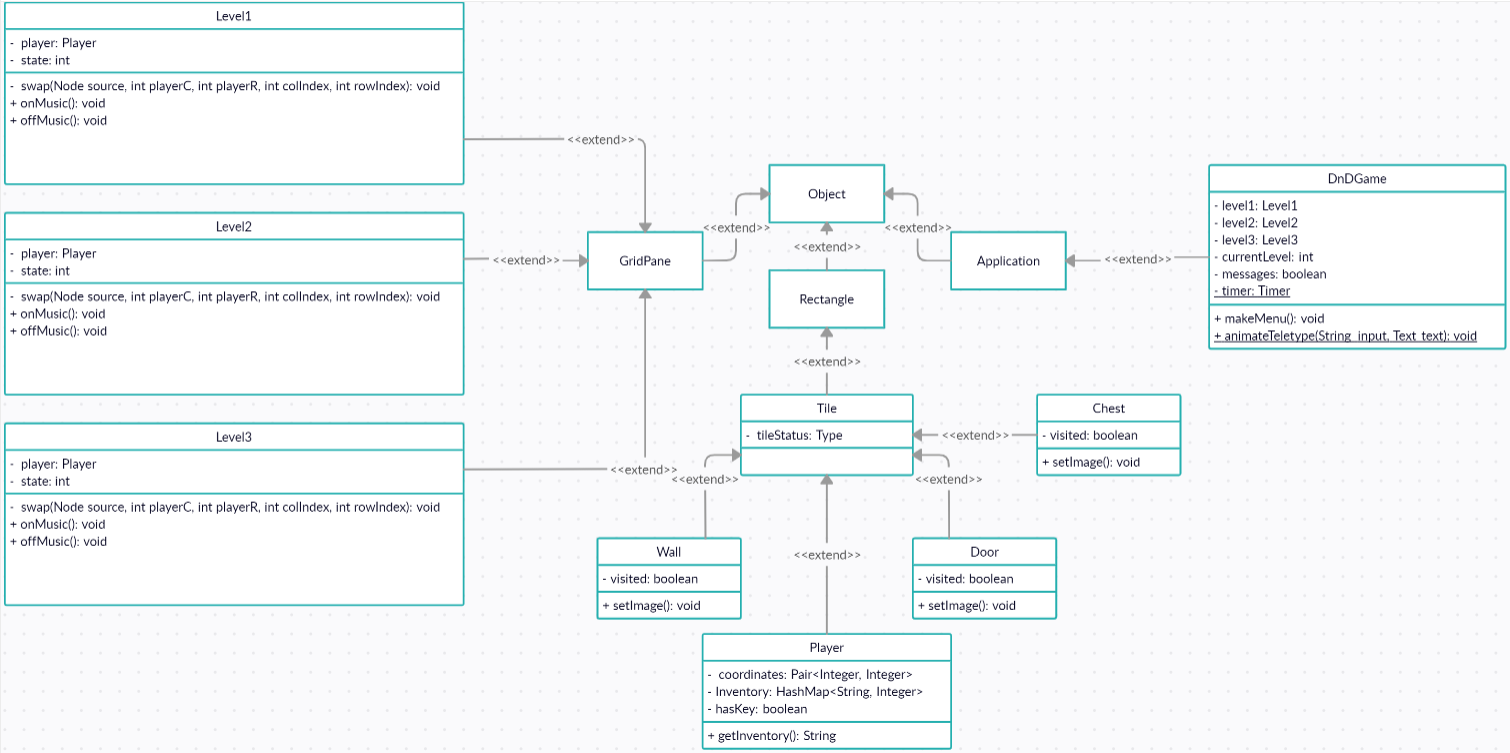
*Motivation*

Every tile is a rectangle. All regular tiles, walls, players, doors, and chests inherit the properties of that rectangle. Similarly, all levels inherit the properties of a GridPane. Lastly, the DnDGame class extends application. All of the classes are extended from some other class, and some classes may be compositions of other classes.

*Considerations*

Some considerations we would like to take is to condense the levels and have them extend to a Level class. If this is done, the level classes can be put into an ArrayList of levels, and it will be easier to access them in DnDGame. Also, instead of the player having a hash map for an inventory, we can make an inventory class which can extend from a hash map and add more functionality to it. Although the UML may look simple, it can still be broken down further to support more encapsulation. It may be the case that if we have more classes, accessing different objects may be easier.

*Example*



**Figure 1: Game UML**

### Data Dictionary for Any Included Models

*Content*

All of the information will be from the player’s Google or Apple account. The game itself does not store any information from the user, but instead we are given their account information to flag any bots and to redeem any in-app purchases.

*Motivation*

The diagram below was created to extract data from the user. We are using this data to keep track of which players are legitimate and which players are not. If Dankest Dungeon were to get saturated with bots, then the game would lose its authenticity. Since Dankest Dungeon is Bohn Jell Entertainment’s one and only game, then a lot of weight hinges on removing these bots.

*Examples*



Table 1: Game Database

*Considerations*

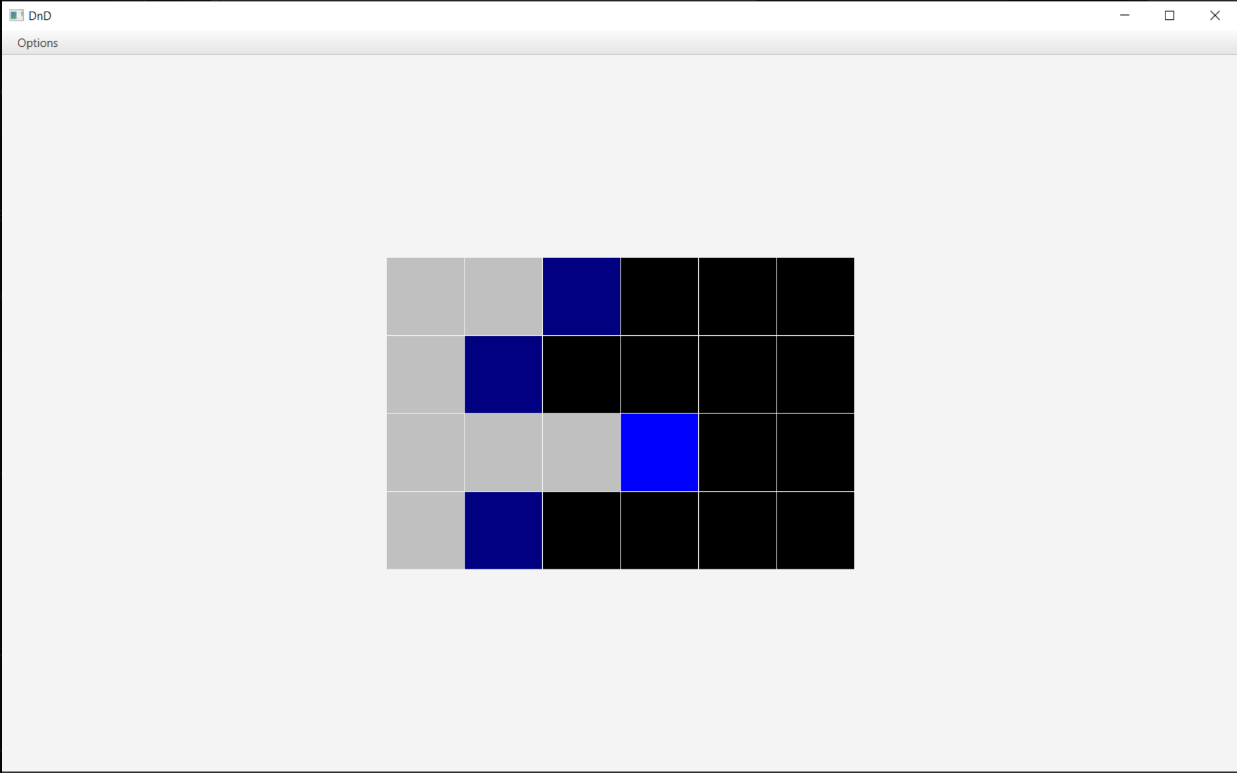
Although we do not want to over-saturate the game, we may consider that a user can have multiple accounts. Some players may want to start from scratch again to re-live their past gameplay. However, if too any accounts are being created for malicious reasons, then we here at Bohn Jell Entertainment reserve the right to delete unused accounts and accounts with malicious intent.

# Project Deliverables

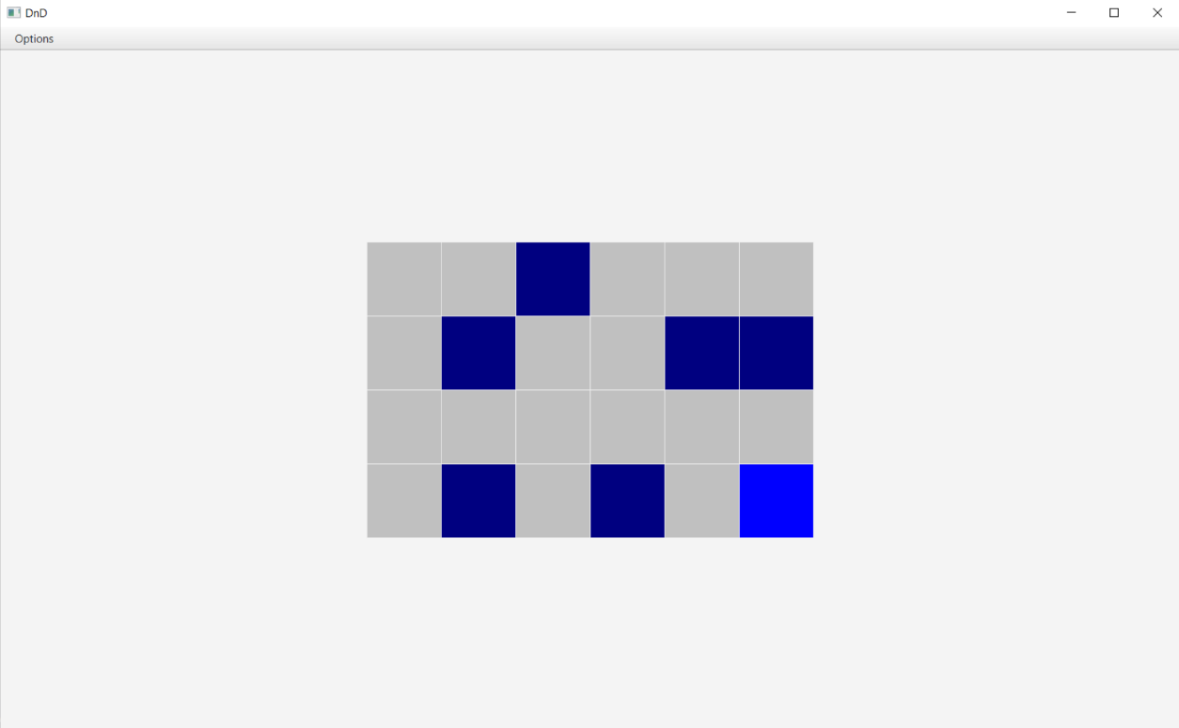
This is what we made in terms of progress per release:

## First Release

Date released: 09/Oct/20



**Figure 2: Initial Level Design 1**



**Figure 3: Initial Level Design 2**

The first release as a result of our first sprint outlined the initial workings and flow of the game. The main menu was completed, along with a rough draft of Level 1. Not all graphical and audio features were implemented (music/SFX), but basic playability was completed to give players an idea of how the game worked.

## Second Release

Date released: 06/Nov/20.



Figure 4: Final Level Design

The second release as a result of our second sprint had us creating and implementing background/lore for the game, as well as polishing up the main menu screen, adding a title screen, and adding graphics and additional music and SFX for a better user experience.

## Comparison with Original Project Design Document

This Project is based on the Technical Report written by Arthur Mezheritskiy, Bennett Maciorowski, Chris Lee, and Ovidiu Bahnean by the name of “Dungeons and Dank” from February 2018.

The final product ended up being quite close in implementation to the report described, a 2-Dimensional roguelike puzzle game. Implementation of the level design focusing on the dragon character stems straight from the report. The game relies heavily on puzzle-based level design and exploration by the player and is designed for mobile and touchpad usage.

As documented above, we were able to complete the main part of the project. However, the technical report went much further than our game. In the technical report they described 6 hours of total gameplay, along with additional items and characters that we were unable to implement in time. Though we were unable to implement the volume of work described in the technical report, we completed the initial level and graphical design in the spirit of the game, and if more time was allowed, could build on what was made and implement more and more puzzle-based levels.

The bulk of the work in creating the game is coding to allow the ability for designers to create levels of their own design quickly and easily. We reached that point, and further level creation is now much faster than it was initially.

# Testing

## Items to be Tested

There are 4 main categories of testing as shown below with (coder => tester) labelled in parentheses:

* 1. GUI (Sam => Luke)
  2. Logic/Code (Andrew => Alex)
  3. Audio (Alex => Sam)
  4. Playability (fun) (Luke => Andrew)

## Test Specifications

**ID# - gui1- GUIMainMenu**

**Description:** There are three buttons on the main menu screen for Dankest Dungeon.

**Items covered by this test:** “Play now!”, “About Us”, “Exit” buttons.

**Test Procedures:**  
- Buttons do not respond to any right-click actions.  
- (1) Ensure mouse rollover elicits a response from the buttons (both visually and audially).  
- (2) Begins game from first level when “Play now!” is left-clicked.  
- (3) Enters about us screen when “About us” button is left-clicked.  
- (4) Exits game entirely when “Exit” button is left-clicked.

**Input Specification:** Manual testing of left and right-click operations on each of the three buttons. 6 initial tests, with repeated entrance/exit of "About us” button.

**Output Specifications:**  
(1) Text turns blue on mouse rollover, reverts to original white text when mouse hover removed from button.  
(2) Begins game from level 1  
(3) Enters about us scene, and text begins displaying (rolling text display)  
(4) Exits

**Pass/Fail Criteria:** Successfully completes specifications 1-4.

**ID# - log1- LogicGameplayLevelOne**

**Description:** Ensure items and characters respond appropriately to input for level 1 of the game.

**Items covered by this test:** Level 1.

**Test Procedures:**  
- (1) Click on tile out of range of current tile.  
- (2) Click on tile with fog.  
- (3) Click on grass tile.  
- (4) Click on wall tile.  
- (5) Click on chest tile.  
- (6) Click on door tile without key.  
- (7) Click on door tile with key.  
- (8) Click on every tile in level 1 to ensure fog of war disappears.

**Input Specification:** Manual testing of operations on each of the three buttons. Tests will be completed multiple times in different orders to ensure accuracy.

**Output Specifications:**   
- (1) No movement/response.  
- (2) un-fogs clicked-on tile.  
- (3) Move to grass tile, audio of walking on grass plays.  
- (4) Player stays in current tile, audio of walking into wall plays.  
- (5) Player received key from chest.  
- (6) Player stays in current tile.  
- (7) Player moves on to level 2.  
- (8) Ensure audio replays when tile stepped on a second time, ensure player can only receive 1 key from chest.

**Pass/Fail Criteria:** Successfully completes specifications 1-8.

**ID# - audio1- AudioMusicAndSFX**

**Description:** Ensure all audio tracks play successfully and without glitching, and all SFX is consistent throughout the game.

**Items covered by this test:** All audio tracks and SFX in the game.

**Test Procedures:**  
- (1) Title screen music plays.  
- (2) Main Menu music plays.  
- (3) Main menu buttons hover SFX.  
- (4) Level 1 music plays.  
- (5) Level 2 music plays, level 1 music stops playing.  
- (6) Level 3 music plays, level 2 music stops playing.  
- (7) Grass tile SFX plays when moved on to grass tile.  
- (8) Wall tile SFX plays when moved on to wall tile.  
- (9) Chest tile SFX plays when moved on to chest tile.  
- (10) Door tile SFX plays when moved on to door tile.

**Input Specification:** Manual testing of operations for every scene in the game. Testing will be conducted through headphones and speakers separately, and ideally on multiple computers/runtime environments.

**Output Specifications:**  
- (1) Title screen music is heard through speakers/headphones.  
- (2) Main Menu music is heard through speakers/headphones.  
- (3) Main menu buttons hover SFX is heard through speakers/headphones.  
- (4) Level 1 music is heard through speakers/headphones.  
- (5) Level 2 music is heard through speakers/headphones, level 1 music stops playing.  
- (6) Level 3 music is heard through speakers/headphones, level 2 music stops playing.  
- (7) Grass tile SFX is heard through speakers/headphones when moved on to grass tile.  
- (8) Wall tile SFX is heard through speakers/headphones when moved on to wall tile.  
- (9) Chest tile SFX is heard through speakers/headphones when moved on to chest tile.  
- (10) Door tile SFX is heard through speakers/headphones when moved on to door tile.

**Pass/Fail Criteria:** Successfully completes specifications 1-10.

**ID# - play1- Playability**

**Description:** Ensure gameplay is smooth, meaning transitions between levels, feedback is timely on all button clicks and player movement.

**Items covered by this test:** Play through the game multiple times and attempt to break the game to find errors.

**Test Procedures:**  
- (1) Title screen button works as expected.  
- (2) Main Menu buttons work as expected. Feedback is timely.  
- (3) Level 1 player movement is as expected. Only legal moves are accepted. Feedback and movement is timely (<100ms).  
- (4) Level 2 player movement is as expected. Only legal moves are accepted. Feedback and movement is timely (<100ms).  
- (5) Level 3 player movement is as expected. Only legal moves are accepted. Feedback and movement is timely (<100ms).  
- (6) Exit button works at each level to exit back to main menu.

**Input Specification:** Manual testing of operations for every scene in the game. General gameplay testing to test for playability and enhance player fun.

**Output Specifications:**   
- (1) Title screen button moves player to main menu when clicked.  
- (2) Main Menu buttons move player to level 1 for Play now, About us screen for About us button, and exit the game for the Exit button.  
- (3) Level 1 movement is smooth and seamless and as expected.  
- (4) Level 2 movement is smooth and seamless and as expected.  
- (5) Level 3 movement is smooth and seamless and as expected.  
- (6) Exit button works at each level and returns player back to main menu.

**Pass/Fail Criteria:**   
Successfully completes specifications 1-6.

## Test Results

**ID# - gui1- GUImainMenu**

**Date(s) of Execution:** 11/22/2020, 11/23/2020.

**Staff conducting tests:** Luke Austin.

**Expected Results:**  
(1) Text turns blue on mouse rollover, reverts to original white text when mouse hover removed from button for all buttons.  
(2) Begins game from level 1 when “Play now!” button is pressed.  
(3) Enters about us scene, and text begins displaying (rolling text display) when “About us“ button is pressed.  
(4) Exits game when Exit button is pressed.

**Actual Results:** 11/22/2020, 11/23/2020  
(1) Text turns blue on mouse rollover, reverts to original white text when mouse hover removed from button for all buttons.  
(2) Begins game from level 1 when “Play now!” button is pressed.  
(3) Enters about us scene, and text begins displaying (rolling text display) when “About us“ button is pressed.  
(4) Exits game when Exit button is pressed.

**Test Status:** All tests passed.

**ID# - log1- LogicGameplayLevelOne**

**Date(s) of Execution:** 11/19/2020 (multiple run-throughs), 11/20/2020

**Staff conducting tests:** Alex Choi.

**Expected Results:**  
- (1) Click on tile out of range of current tile, no movement/response.  
- (2) Click on tile with fog, un-fogs clicked tile.  
- (3) Click on grass tile, moves to grass tile and plays audio.  
- (4) Click on wall tile, stay on current tile and play wall audio.  
- (5) Click on chest tile, player received key.  
- (6) Click on door tile without key, player stays on current tile.  
- (7) Click on door tile with key, player moves to next level (level 2).  
- (8) Click on every tile in level 1 to ensure fog of war disappears, audio replays, key only able to be received from chest once.

**Actual Results:** 11/19/2020  
- (1) Click on tile out of range of current tile, no movement/response.  
- (2) Click on tile with fog, un-fogs clicked tile.  
- (3) Click on grass tile, moves to grass tile and **audio does not play consistently**.  
- (4) Click on wall tile, stay on current tile and play wall audio.  
- (5) Click on chest tile, player received key.  
- (6) Click on door tile without key, player stays on current tile.  
- (7) Click on door tile with key, player moves to next level (level 2).  
- (8) Click on every tile in level 1 to ensure fog of war disappears, audio replays, key only able to be received from chest once.

**Actual Results:** 11/20/2020  
- (1) Click on tile out of range of current tile, no movement/response.  
- (2) Click on tile with fog, un-fogs clicked tile.  
- (3) Click on grass tile, moves to grass tile and audio plays.  
- (4) Click on wall tile, stay on current tile and play wall audio.  
- (5) Click on chest tile, player received key.  
- (6) Click on door tile without key, player stays on current tile.  
- (7) Click on door tile with key, player moves to next level (level 2).  
- (8) Click on every tile in level 1 to ensure fog of war disappears, audio replays, key only able to be received from chest once.

**Test Status:** Pass. After initial tests on 11/19/2020, some audio bugs were found but subsequently fixed for 11/20/2020 testing.

**ID# - audio1- AudioMusicAndSFX**

**Date(s) of Execution:** 11/20/2020, 11/22/2020

**Staff conducting tests:** Sam Alammar.

**Expected Results:**  
- (1) Title screen music plays.  
- (2) Main Menu music plays.  
- (3) Main menu buttons hover SFX plays.  
- (4) Level 1 music plays, loops when finished playthrough continuously (tested up to 20 minutes).  
- (5) Level 2 music plays, loops when finished playthrough continuously (tested up to 20 minutes). level 1 music stops playing.  
- (6) Level 3 music plays, loops when finished playthrough continuously (tested up to 20 minutes). level 2 music stops playing.  
- (7) Grass tile SFX plays when moved on to grass tile.  
- (8) Wall tile SFX plays when moved on to wall tile.  
- (9) Chest tile SFX plays when moved on to chest tile.  
- (10) Door tile SFX plays when moved on to door tile.

**Actual Results:** 11/20/2020, 11/22/2020  
- (1) Title screen music plays.  
- (2) Main Menu music plays.  
- (3) Main menu buttons hover SFX plays.  
- (4) Level 1 music plays, loops when finished playthrough continuously (tested up to 20 minutes).  
- (5) Level 2 music plays, loops when finished playthrough continuously (tested up to 20 minutes). level 1 music stops playing.  
- (6) Level 3 music plays, loops when finished playthrough continuously (tested up to 20 minutes). level 2 music stops playing.  
- (7) Grass tile SFX plays when moved on to grass tile.  
- (8) Wall tile SFX plays when moved on to wall tile.  
- (9) Chest tile SFX plays when moved on to chest tile.  
- (10) Door tile SFX plays when moved on to door tile.

**Test Status:**  Passed on multiple setups and through headphones and speakers.

**ID# - play1- Playability**

**Date(s) of Execution:** 11/23/2020

**Staff conducting tests:** Andrew Macatangay.

**Expected Results:**  
- (1) Title screen button moves player to main menu when clicked.  
- (2) Main Menu buttons move player to level 1 for Play now, About us screen for About us button, and exit the game for the Exit button.  
- (3) Level 1 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (4) Level 2 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (5) Level 3 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (6) Exit button works at each level and returns player back to main menu.  
  
**Actual Results:** 11/23/2020  
- (1) Title screen button moves player to main menu when clicked.  
- (2) Main Menu buttons move player to level 1 for Play now, About us screen for About us button, and exit the game for the Exit button.  
- (3) Level 1 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (4) Level 2 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (5) Level 3 movement follows only legal in-game movement. SFX for tiles plays accurately and appropriately.  
- (6) Exit button works at each level and returns player back to main menu.

**Test Status:** Pass.

# Inspection

## Items to be Inspected

For the purpose of this inspection, our team has selected the most important areas of concern. This includes all event handlers that deal with sound effects or music, all secondary classes dealing with special tiles and the player, the secondary classes dealing with the levels themselves, and the function of how each class works together in the main driver class.

## Inspection Procedures

As per the inspection, a total of 4 meetings were held in order to get the code in order. The meetings were intended to keep all group members up-to-date on the plans and findings of the inspections. Out of the 4 meetings, 3 were held in order to check up progress on the program and how it pertains to the checklist. The first meeting dealt with the planning stages of the coding inspections and set a precedent for what we as a company are looking for in the code. The second inspection was utilized as a progress report for how the code was satisfying conditions of the checklist below. The third meeting was solely used to address a major issue in the code that was found in a code inspection. The last meeting was used to recheck the code and how it satisfied conditions of the checklist and to give final words and advice pertaining to the coding. As time went on, more items were added to the checklist and certain items that were deemed unnecessary were taken off the checklist. There were certain issues that were addressed that were not on the checklist since they pertained more toward the style of the coding and not necessarily the actual function. All the results were reviewed electronically, during live meeting sessions.

Bohn Jell Entertainment Code Inspection Checklist – Java

1. Intention of Design and Specification

[ ] Does the code perform its intended function?

[ ] Does the code perform multiple functions?

[ ] Is the code designed in such a way that it may be reused for future programs?

[ ] Is the code logically sound and does its organization make sense?

[ ] Is the code clear and concise?

1. Initialization of Variables

[ ] Are all import statements necessary and actually being used properly?

[ ] Are all variables initialized before use?

[ ] Do any variables project a type-cast error?

[ ] Are variables declared in the proper scope?

[ ] Are constructor utilized to instantiate a class properly?

[ ] Are all variable names pertinent to their function?

[ ] Are all the primitive data type variables used and instantiated properly?

[ ] Are there any static variables or functions and if so are they being used properly?

1. Method Calls

[ ] Are all inherited classes properly extending the scope of the intended parent class?

[ ] Are parameters of the proper type for the method being called?

[ ] Are the methods being called properly?

[ ] Are method names pertinent to their functionality?

[ ] Are return values in the proper type and being used properly?

[ ] Does the method organization make sense?

[ ] Can certain method classes be combined and still make sense logically?

[ ] If a method calls another method class, is the call necessary and is it properly utilized?

1. Object Comparison

[ ] Are the object comparisons used necessary?

[ ] Do all object comparisons have proper boundary conditions?

1. Graphical User Interface

[ ] Are all the event handlers written properly without errors?

[ ] Are all event handlers used?

[ ] Are all event handlers necessary?

[ ] Is the GUI void of any grammatical errors?

[ ] No nested event handlers

[ ] Are the usage of GUI elements clear and concise?

1. Computations and Comparisons

[ ] Are there any common syntax errors still present in the code?

[ ] Is operator precedence being maintained?

[ ] Is integer arithmetic used in a necessary way?

[ ] Are all proper logical and relational operators correctly being used?

[ ] Are the error conditions pertinent to what is being computed?

[ ] Does the code require any implicit type conversions?

[ ] If there are nested if statements, are all the else statements necessary?

[ ] Do all loop conditions make sense logically?

[ ] Are all parentheses and braces aligned properly?

[ ] Are all logical true or false statements necessary and being used properly?

## Inspection Results

Inspection: Game mechanic and SFX Event Handlers

Inspector: Andrew P. Macatangay

Date: 11/20/2020 4:20 PM

Errors Revealed:

* 1. Sound effects and music were not working properly for stage 1 and 3 even though they previously worked for stage 1.
  2. Volume for certain event handlers was set incorrectly
  3. Certain event handlers were superfluous
  4. Image resources and sound effects were not being referenced properly

Solutions for Errors:

* 1. This problem was remedied by removing certain nested event handlers.
  2. Changed the volume value to one that was acceptable.
  3. Removed event handlers that were never used and made sure that they were unimportant.
  4. Changed some of the import statements and the file pathing to the correct resource folder.

Reinspection for point 3 was done in order to make sure that all extra event handlers were indeed unimportant.

Inspection: Special tiles and Player class

Inspector: Luke Austin

Date: 11/21/2020 4:00 PM

Errors Revealed:

* 1. Certain tiles classes were not being properly used in the program.
  2. The rock class was not loading the proper image.

Solutions for Errors:

* 1. Used inheritance in order to extend the Tile class.
  2. Utilized an image of appropriate file type.

Inspection: Level Classes

Inspector: Sam Alammar

Date: 11/20/2020 6:00 PM

Errors Revealed:

* 1. Levels were not resizing properly from level 2 to level 3.
  2. For-loop for creating the level was going out of scope.
  3. Added a wall out-of-bounds of the playable area

Solutions for Errors:

* 1. Changed the parameters for the for-loop and resized the grid pane that the level was situated on.
  2. Changed the for-loop boundary conditions.
  3. Changed the wall class area to be in-bounds.

Reinspection occurred for point 1 to make sure the resizing was working properly.

Inspection: Main Driver Class

Inspector: Alex Choi

Date: 11/19/2020 4:00 PM

Errors Revealed:

* 1. Certain design elements were not being properly aligned in the grid pane.
  2. Organization of certain design elements were not in proper order.
  3. Introductory video was not being called properly.
  4. Grammatical errors in certain messages.

Solutions for Errors:

* 1. Changed the alignment of the design elements.
  2. Reorganized the Vbox and certain buttons.
  3. Created a media object instead.
  4. Fixed all the grammatical errors for the messages with the issue.

# Recommendations and Conclusions

Based upon all testing and inspection parameters, the program passes all the procedures that it was subject to. All the tests were run again after the coding reached maturity to make sure that there were no regressive errors, in that certain working parts of coding are no longer working since the last round of testing. This could be since an addition to the code that uses portions of the older code may break the functionality of that code. Certain parameters were reinspected just to make sure that all changes reflected the intended purpose of the code. At the basic level, all coding inspections problems were addressed properly and all suggestions by the coding inspector were considered and implemented into their respective coding sections. As of right now, no further testing or inspection procedures are necessary as no new snippets of code are being added, however, at any point in the future if there are new additions, the code will need to be retested and reinspected.

# Project Issues

## Open Issues

At Bohn Jell Entertainment, we still need a form of income for both our workers, and for the welfare of the game. Advertisements, in-app purchases, and subscriptions are all techniques to help combat this dilemma.

*Content*

Dankest Dungeon is not without its faults. There are still issues to monetize the game. Since Dankest Dungeon is Bohn Jell Entertainment’s only game, we need to be able to provide for our workers, and benefit as a company.

*Motivation*

Our motivation is to bring gaming to a wider audience. Free games do not come without a cost. In order to cater to a larger, audience, we need to be able to maintain the game, and expand to add more features. We feel that if we can add more features, that the game will be more playable, players would be satisfied, and the company will be able to continuously operate.

*Examples*

Some examples of ways to monetize the game is to add in-app purchases. During the intermission of each level, we can have a shop keeper where players can buy power ups and other items.

Another example is to create a monthly subscription. This subscription would only cost about 5 US dollars per month which is on par with a lot of other subscription-based applications.

We may include a perk where a paid player can participate in unlimited dungeons. We may have an energy system where a player can only play so many levels before entering a cooldown where the player must wait before being able to play again. The last form of monetization can come in the form of advertisements.

*Considerations*

Some considerations we would have to take is input from the players. A lot of the time, mobile games die off because of a focus on monetization rather than the state of the game. Some games in the market are only meant to be money-grabbers, and we do not want Dankest Dungeon to be one of those games. Although Bohn Jell Entertainment relies solely on one game, we believe that the satisfaction of the players is more important. Dankest Dungeon would not be able to survive without this.

If changes needed to be made to cater to free players, then we can add lower-level updates to cater to those players. The updates will not only cater to high level players who spend money on the game. To further progress in the game, each player should be given the voice to influence the game. At the end of the day, Dankest Dungeon belongs to the players, and not to the company.

With all these considerations, we hope to make Dankest Dungeon the most successful mobile application in the market and hope that our product can supply our players with the best gaming experience!

## Waiting Room

Sometime after launch, the game can get stale and repetitive, that’s where add-ons and patch updates will be introduced to include more variety. Below, we are going to discuss some potential updates and possible new features to include into this game to make it live longer.

*Content*

Dankest Dungeon by itself is a steady paced game, but there is always room for expanding upon what already exists in the sense of scalability. We are considering adding items, power ups, enemies, special dungeons, and more!

*Motivation*

As you go through the dungeons that the game has to offer, you might want items and perks that can make the game run faster and smoother at the same time to match the players’ needs. As such, we thought about adding additional features that can be used for in order to make the game go along in a pace that a veteran player would enjoy.

*Considerations*

We can consider seasonal events, such as Christmas, and can add character and item cosmetics to change the aesthetics of the game in general. We can also consider the fact that the players will get to the point of clearing the game and in need of more content, to which we can add harder dungeons and introduce extreme difficulty for some dungeons that add high rewards.

Speaking about high stakes. We can add gatcha (loot boxes) into the game, for the players who want more excitement and more rewards. It can be a character or an item gatcha. Depending on the rarity of said character/item, it can be harder to get/lower chances of the item dropping.

We can also consider that new players who join the game will be confused with all the new updates. To aid any new player, we can make a more tutorial system that keeps said player in the loop.

At some point down the road people will ask for interactivity with other players in one way or another. There is merit in adding an online system, starting off with just trading items, then slowly expanding on it until you are able with 2-3 other players. This enforces cooperative gameplay, making it much more enjoyable.

## Ideas for Solutions

Right now, Dankest Dungeon is struggling to make ends meet. We need capital to support our team members and to expand our game.

Your text goes here . . .

*Content*

We at Bohn Jell Entertainment plan to monetize the game by using in-app purchases, monthly subscriptions, and advertisements. We believe that the usage of these in our game will be sufficient to support the game and all its infrastructure.

*Motivation*

Our motivation is to bring gaming to a wider audience. Free games do not come without a cost. In order to cater to a larger, audience, we need to be able to maintain the game, and expand to add more features. We feel that if we can add more features, that the game will be more playable, players would be satisfied, and the company will be able to continuously operate.

*Considerations*

Some examples of ways to monetize the game is to add in-app purchases. During the intermission of each level, we can have a shop keeper where players can buy power ups and other items. We plan to make this free, with the addition of buying gold so that paid players can boost their way to higher levels. This would allow free players to still enjoy the game without having to pay, and it gives the option to paid players to play at a faster rate.

Another example is to create a monthly subscription. This subscription would only cost about 5 US dollars per month which is on par with a lot of other subscription-based applications. We can create perks for these members. At the start of every month, we can add exclusive dungeons that only paid players can access. The dungeons can include more gold and power ups.

We may include a perk where a paid player can participate in unlimited dungeons. We may have an energy system where a player can only play so many levels before entering a cooldown where the player must wait before being able to play again. This is to deter players from overplaying the game and to spike the demand of being able to play. Players may buy energy so that they can continue.

The last form of monetization can come in the form of advertisements. To reduce our advertising costs, we can add advertisements to our game to reattain part of the spending. Players may watch advertisements to be able to receive energy, gold, or even visit other dungeons. This is a great way for free players to be able to progress faster, at the cost of their time. The goal is to add these advertisements only when the player decides to watch an advertisement, rather than flooding their screen with advertisements mid-game.

## Project Retrospective

Dankest Dungeon reached some highs and lows through the development of the project. The general vision of the game was successful, but we would have liked to add more features to create a more interactive and comprehensive game.

*Content*

We were able to make great strides into developing a game, but it was not done in the most efficient manner. Our group of four members all worked on the same parts at every single turn. Along with this, expediting the time for each task was not ideal.

*Motivation*

At Bohn Jell Entertainment, we are in no way perfect. Looking back at what we have created, we feel that Dankest Dungeon is still a work in progress. To continue expanding, we will encounter even more obstacles, and will need to consider them for future success. We as a company, must keep in mind the value of our players to create a game that is suitable to all.

*Considerations*

When it came down to development, we found it inefficient for all members of the group to have to code and do documentation. It would be more efficient if we split up into pairs and two of us worked on the code, while the other two would work on documentation. If we did this, it would be easier to maintain code since we don’t need to relay each update to four people. We would only need to talk in pairs. By splitting up into pairs, the two members can get into a groove of either coding or just documentation and not have to worry about switching gears.

Another thing we fell back on was not having enough time to finish what we set out to do. We wanted to add enemies, power ups, and intermission screen, and more rooms to each dungeon, but we did not have enough time to do it. There were issues on how the code was structured which made adding new components harder. Every single component of the game should have been split up into smaller classes, and should have used various design patterns and techniques for inheritance. Without a sufficient knowledge in this, development took longer than it should be.

# Glossary

**AI:** Artificial Intelligence.

**Bohn Jell Entertainment:** The name of the company that made the game!

**Dankest:** Something that is epic or cool.

**Gatcha:** Is the implementation of a gatcha mechanic, which is mainly paying for a random virtual item. Example such as loot boxes.

**GUI:** Graphical User Interface.

**Hit Points (HP):** The health of a creature, quantified in a number of points.

**SFX:** Sound effects.

**Spell:** An incantation that a creature creates to invoke a supernatural action such as flying or creating a fire ball.

**Trogdor:** The name of the dragon.

# References / Bibliography

|  |  |
| --- | --- |
| [1] | Robertson and Robertson, Mastering the Requirements Process. |
| [2] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |
| [3] | B. M. C. L. O. B. Arthur Mezheritskiy, "Dungeons and Dank," 2018. |

# Index

problem 20

project 7, 11, 19, 25

release 10, 11

Test 2, 12, 13, 14, 15, 16, 17, 18

UML 2, 4, 7, 8, 26