

Software Unit and System Test Documentation

Graze Save Survive

Version 1.0

12 April 2024

<https://github.com/holiday-pettijohn/GrazeSaveSurvive> (currently private)

Department of Math and Computer Science, Biola University

Revisions

Overview. This document exists to specify the tests for the entire system, define the testing system, and record the test results.

Target Audience. Stakeholders

Project Team Members. Holiday Pettijohn, Jacob Shirota, Caleb Zuniga, Nicklas Kuo

Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Jacob Shirota	Initial documentation	04-08-2024
0.2	Jacob, Holiday, Nicklas	Test cases	04-12-2024

Signatures of Approval

Version 1.0		
Name	Signature	Date
Holiday Pettijohn	Holiday Pettijohn	04-12-2024
Jacob Shirota	Jacob Shirota	04-12-2024
Caleb Zuniga	Caleb Zuniga	04-12-2024
Nicklas Kuo	Nicklas Kuo	04-12-2024

Table of Contents

1. Introduction.....	4
1.1. System Overview.....	4
2. Test Plan.....	4
2.1. Features to be Tested.....	4
2.2. Features not to be Tested.....	4
2.3. Testing Tools and Environment.....	4
3. Test Cases.....	5
3.1. Case 1.....	5
3.2. Case 2.....	5
3.3. Case 3.....	5
3.4. Case 4.....	5
3.5. Case 5.....	6
3.6. Case 6.....	6
3.7. Case 7.....	6
3.8. Case 8.....	6
3.9. Case 9.....	7
3.10. Case 10.....	7
3.11. Case 11.....	7
3.12. Case 12.....	8
4. Additional Material.....	8
4.1. Appendix A. Test Logs.....	8
5. Overall Takeaways.....	13

1. Introduction

1.1. System Overview

Graze Save Survive is a 2D rogue-lite game inspired by *Vampire Survivors* which consists of various menu interfaces and a gameplay map in which a Player sprite, controlled by user keyboard input, is able to move around the screen and interact with other generated Enemy sprites, which appear over incremental time periods until a certain threshold is reached. During gameplay, the user can collect Item sprites which are stored to a persistent SQLite database and can be interacted with from a specific menu to modify their gameplay.

2. Test Plan

2.1. Features to be Tested

- 2.1.1. Runs: mechanic testing, expected and unexpected input combos, stress testing with enemies and projectiles
- 2.1.2. Items: change to player stats, upgrade menu config, persistence between runs
- 2.1.3. Menus: navigation, buttons
- 2.1.4. Audio Outputs: audio slider

2.2. Features not to be Tested

- 2.2.1. Formal Semantics: Conditions which should always be true (“there is one player”, “one menu appears at once”, etc) are never violated during testing but do not warrant their own specific tests to verify.
- 2.2.2. Performance Optimality: We are not testing how well the game is optimized. We are not seeking to test for and improve FPS (frames per second) or memory usage, as we have reasonable expectations that user systems can run the program with acceptable resource consumption.

2.3. Testing Tools and Environment

- 2.3.1. *Test Staffing*. Each member of the development team will be responsible for testing certain requirements and procedures. In addition to this, each member will introduce one third-party independent tester to aid in testing gameplay requirements.
- 2.3.2. *Test Environment*. Most tests will be performed within the Godot editor's Debug environment, with all visible debugging features (eg. Visible Collision Boxes) disabled. Each member of the development team and their independent tester will do so on the development team member's personal computer.
Tests on the SQLite database will also use an external tool to directly read the contents of the SQLite database.

3. Test Cases

3.1. Case 1

- 3.1.1. *Purpose.* Test the function of the buttons in the menus (Requirements 2.1.n)
- 3.1.2. *Inputs.* Left-mouse click the “Upgrades” button, “Options” button, and “Exit” button from the Menu. Left-mouse click the “Back” button from the Upgrades Menu. Adjust the audio slider in the Options Menu all the way down (left), then all the way up (right), then click the “Back” button.
- 3.1.3. *Expected Outputs.* The program starts on the main menu. The upgrade menu should appear after the “Upgrades” button is pressed. The main menu should reappear after the “Back” button is pressed. The options menu should appear after the “Options” button is pressed. The main menu should reappear after the “Back” button is pressed. The program should terminate when the “Exit” button is pressed.
- 3.1.4. *Pass/Fail Criteria.* Pass if the menus appear in the expected sequence, and the audio fades out and in when the slider is adjusted. The program exits at the end. Fail otherwise.
- 3.1.5. *Test Procedure.* The previously listed inputs were executed.

3.2. Case 2

- 3.2.1. *Purpose.* Verify that player movement is functional.
- 3.2.2. *Inputs.* Use the WASD keys to move the player on the screen.
- 3.2.3. *Expected outputs.* The player should move at a consistent speed in four directions and diagonally. When multiple opposite keys are pressed, they should cancel out each other.
- 3.2.4. *Pass/Fail Criteria.* Pass: The player responds to keyboard movement.
Fail: The player does not move, or does not move in the correct direction.

3.3. Case 3

- 3.3.1. *Purpose.* Verify that the player cannot leave the map.
- 3.3.2. *Inputs:* Use the controls to move the player to all four edges of the map.
- 3.3.3. *Expected outputs.* The player is not able to move fully out of the map.
- 3.3.4. *Pass/Fail Criteria.* Pass: The player cannot leave the map from any direction. Fail: The player can leave.

3.4. Case 4

- 3.4.1. *Purpose:* Verify the player can perform a ranged and melee attack.
- 3.4.2. *Inputs:* Press “space” during a run for a melee attack. Press “p” to perform a ranged attack.
- 3.4.3. *Expected outputs:* The melee attack should display a sprite in the direction the player is facing for less than a second, and then disappear. The ranged attack should create a projectile in the direction of the player.

- 3.4.4. *Pass/Fail Criteria:* Pass: The controls should perform the appropriate attack with the expected attack behavior. Fail: Pressing the control does not attack, or the attack is not performed in the right direction.

3.5. Case 5

- 3.5.1. *Purpose:* Verify that enemies spawn each wave.
- 3.5.2. *Inputs:* Play the game as preferred to survive 5 waves.
- 3.5.3. *Expected Outputs:* The moment a wave increments, enemies spawn. Each wave has an increased amount of enemies that spawn.
- 3.5.4. *Pass/Fail Criteria:* Pass: Enemies spawn in increasing numbers each round. Fail: Enemies do not spawn each wave, or there the number of enemies is nonincreasing.

3.6. Case 6

- 3.6.1. *Purpose:* Verify that the enemies can be damaged and die by player attacks.
- 3.6.2. *Inputs:* Move to close proximity to the enemy and perform a melee attack. Then, move away from an enemy and land a projectile (ranged) attack.
- 3.6.3. *Expected Outputs:* The enemy should take damage when attacked. When their health reaches zero, they disappear.
- 3.6.4. *Pass/Fail Criteria:* Pass: Enemies take damage and die. Fail: Enemies do not take damage when attacked, or do not die when health is reduced to zero.

3.7. Case 7

- 3.7.1. *Purpose:* Verify enemies move in the proper directions.
- 3.7.2. *Inputs:* Move the player around the enemies in a circle.
- 3.7.3. *Expected Outputs:* The melee enemies should follow the player, and the ranged enemies move towards the enemy if they are far away. Otherwise, they move in the opposite direction.
- 3.7.4. *Pass/Fail Criteria:* Pass: The enemies behave in the expected manner. Fail: They do not move in the direction expected.

3.8. Case 8

- 3.8.1. *Purpose:* Verify that enemies drop XP, and these can be picked up by the player.
- 3.8.2. *Inputs:* Kill a ranged and melee enemy with the player's attacks. Move to the enemy upon death.
- 3.8.3. *Expected Outputs:* The enemies should drop an XP orb upon death. When it comes in contact with the player, the orb should disappear, and the XP bar increments.

- 3.8.4. *Pass/Fail Criteria:* Pass: XP drops, it can be picked up, and it increments the bar. Fail: Any of these three do not happen.

3.9. Case 9

- 3.9.1. *Purpose:* Verify that the player can be damaged and die.
- 3.9.2. *Inputs:* When enemies spawn, move to the melee enemy and make contact with it. Do the same with a ranged enemy and their projectile. Take damage until the player has 0 health.
- 3.9.3. *Expected Outputs:* The player takes damage when each of these occur. The run should end, and the results screen appears, when the health reaches 0.
- 3.9.4. *Pass/Fail Criteria:* Pass: The player takes damage when either of these contacts occur. When the player's health is zero, the results screen appears. Fail: At least one of these contacts does not cause the player to take damage, or the results screen does not appear when health reaches 0.

3.10. Case 10

- 3.10.1. *Purpose:* Verify that the Player begins each run with a "blank slate".
- 3.10.2. *Inputs:* Start a run, play as necessary, start a second run.
- 3.10.3. *Expected Outputs:* The player should start the second run with full health (5) and no XP gained (0).
- 3.10.4. *Pass/Fail Criteria:* Pass: Player HP is 5 and XP is 0. Fail: Any stats that are different.
- 3.10.5. *Test Procedure.* Enter the Run scene with a breakpoint set to after Player instantiation but before the first wave of Enemies are instantiated. Pass this breakpoint during the first run. then play until the player gains XP and loses any amount of HP. On the second run, observe the "hp" and "xp" variables of the Player scene from the Godot debugger.

3.11. Case 11

- 3.11.1. *Purpose.* Test the player can win the game and that the results screen reflects the player's performance
- 3.11.2. *Inputs.* Modify the initial Wave Count to satisfy the conditions to "win" sooner than normal. Return to the Main Menu via clicking the Menu button on the Results screen.
- 3.11.3. *Expected Outputs.* After the 9th wave, all of the enemies should disappear and the results screen should pop up with an indication that the player "won". The player level and enemy kill count should also appear.
- 3.11.4. *Pass/Fail Criteria.* Pass if the results screen pops up with the aforementioned information. Fail if it does not appear regardless of how long the player waits, or if it does not contain the appropriate information.

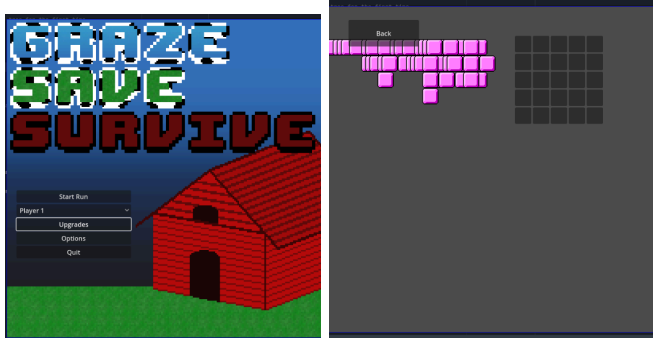
- 3.11.5. *Test Procedure.* After beginning the Run scene, wait (without causing Player death) until the Wave Timer runs out, then observe the Results screen that appears.

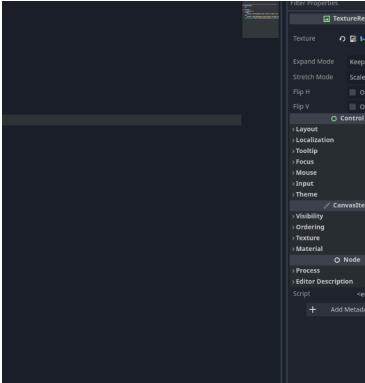
3.12. Case 12


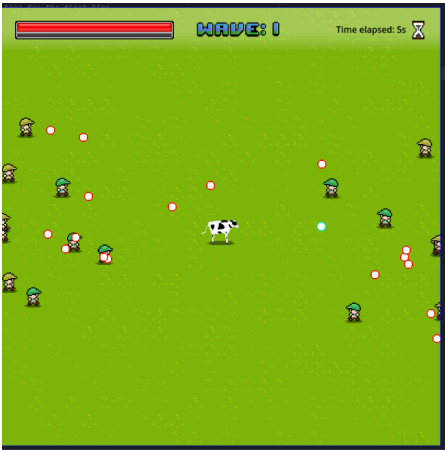
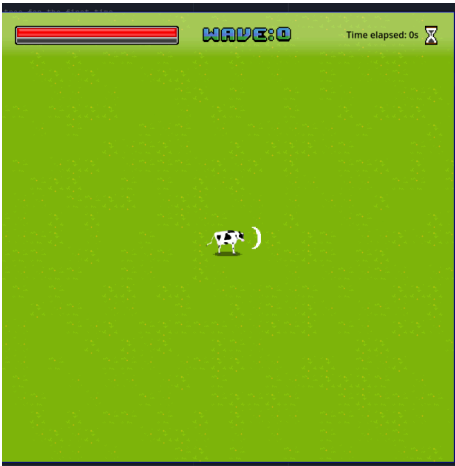
- 3.12.1. *Purpose.* Test the integrated SQLite database, including all SELECT queries, INSERT queries, and data parsing.
- 3.12.2. *Inputs.* As a prerequisite, use a copy of the "database.sqlite3" file that has an empty "player_item_inventory" table. Enter the Run scene. Move the Player over a Tile. View all entries of the "player_item_inventory" table after.
- 3.12.3. *Expected Outputs.* The "player_item_inventory" table of the "database.sqlite3" should contain an entry for the tile(s) picked up with an item_id corresponding to the correct entry in the "item" table.
- 3.12.4. *Pass/Fail Criteria.* Pass if the "player_item_inventory" table contains entries for every tile picked up, which can be confirmed by comparing the generated Tile in the Run scene with the bitmap datafield of the item in the "item" table. Fail if one or more picked up Tiles is not present in the "player_item_inventory" table, or if an entry in the table corresponds to an incorrect item.
- 3.12.5. *Test Procedure.* After beginning with a cleaned copy of "database.sqlite3", enter the Run scene and play until an Enemy dying causes a Tile to appear. Move the Player sprite to pick up the Tile, then end the Run by dying. Examine the contents of "database.sqlite3" using an external tool.


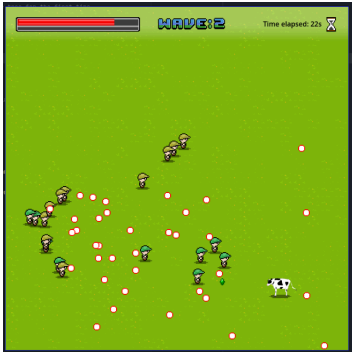
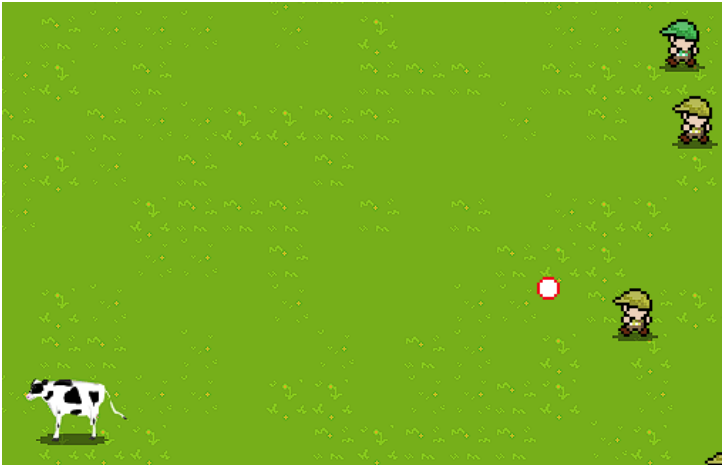
4. Additional Material


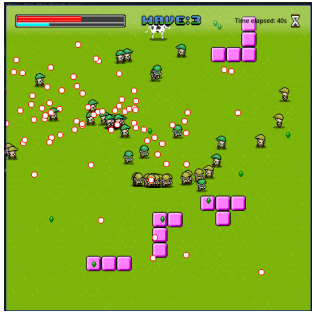

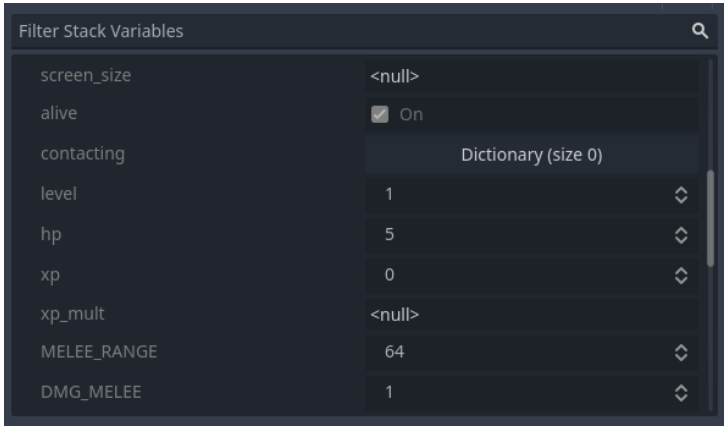
4.1. Appendix A. Test Logs

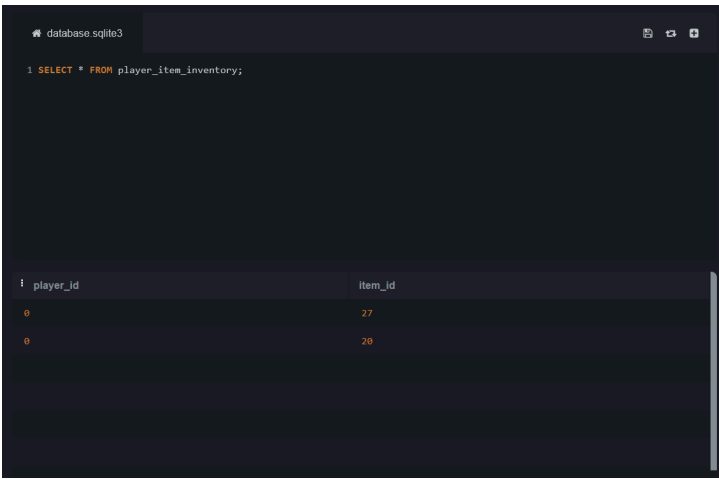
Test Case	Log	Screenshot
1	<p>[04-12-24 22:57] PASSED. The menus work as expected, and the game closed successfully. Audio does fade out based on the slider, but music does not play on the Options menu. Going back out to the main menu reflects changes to the slider however.</p> <p>This build was on a version without the upgrades menu tweaks fully merged, and the appearance is not final. It is merely to test the menu</p>	

	transition functionality.	
2	<p>[04-12-24 23:06] PASSED. Player was able to move up, down, side to side, and diagonally. Player stayed still if opposing buttons were pressed. Screenshot is of player position logs.</p>	<pre>(1054.298, 151.0387) (1054.41, 150.9275) (1054.536, 150.8014) (1054.701, 150.6363) (1054.822, 150.5156) (1054.936, 150.4007) (1055.06, 150.2773) (1055.178, 150.1587) (1055.292, 150.0452) (1055.406, 149.9313) (1055.518, 149.8187) (1055.63, 149.7075) (1055.77, 149.567) (1055.889, 149.4482) (1056.007, 149.3305) (1056.125, 149.2122) (1056.256, 149.0809) (1056.373, 148.9645) (1056.508, 148.8295) (1056.631, 148.7065) (1056.774, 148.5633) (1056.895, 148.4422) (1057.015, 148.3221) (1057.015, 148.3221) --- Debugging process stopped</pre>
3	<p>[4-12-24] PASSED. Printing the player position during gameplay, we can see the player is unable to leave the map in either of the four corners. The map is 1600x800, so the player is constrained between [0,0] and [1600,800].</p> <p><i>Incident:</i> The player can be partially obscured when standing at the edge of the map. This does not violate any</p>	<pre>Player positon:(0, 800) Player positon:(0, 800) Player positon:(0, 800) Player positon:(0, 800) Player positon:(0, 800) Player positon:(0, 800) Player positon:(0, 0) Player positon:(0, 0) Player positon:(0, 0) Player positon:(0, 0)</pre>

	<p>requirement, nor does it cause the test to fail.</p>	<pre>Player positon:(1600, 0) Player positon:(1600, 0) Player positon:(1600, 0) Player positon:(1600, 0) Player positon:(1600, 800) Player positon:(1600, 800) Player positon:(1600, 800) Player positon:(1600, 800)</pre> 
4	<p>[04-12-24 23:06] PASSED. Player melee and ranged attacks appear as intended and deal damage to enemies when colliding with them. The ranged attack disappears upon hitting an enemy. The melee attack, on the other hand, will hit all enemies in its trajectory.</p> <p><i>Incident:</i> Collision allows hitboxes to overlap, meaning that even though the projectile isn't meant to pierce enemies, it can have a small area of effect which was not intended.</p>	 
5	<p>[4-12-24] PASSED. Observing the output log, we can see that each</p>	<pre>Wave 0: Spawning 20 enemies Created 20 enemies at (1600, 800)</pre>

	<p>wave, the game spawns an increasing number of enemies. This is tested for the first five waves.</p> <p><i>Incident:</i> Since the number of enemies spawned has float values, there is imprecision with the exact number spawned. Still, this does not violate the progressive spawning requirement.</p>	<div>Wave 1: Spawning 22 enemies</div> <div>Wave 2: Spawning 24.5947934199881 enemies</div> <div>Wave 3: Spawning 27.4743856376931 enemies</div> <div>Wave 4: Spawning 30.5560632861832 enemies</div>
6	<p>[04-12-24 22:55] PASSED. Enemies die once hit by enough melee/ranged attacks, or a combination thereof. Enemies have a tendency to bunch up and die together. This may have game design implications but does not violate any requirements</p>	<div></div> <div></div>
7	<p>[4-12-24] PASSED. For melee enemies move towards the player as seen in the screenshots. Ranged enemies stop at a distance. When the player approaches a ranged enemy, they walk away.</p>	

		
8	[4-12-24 23:13] PASSED. When enemies are hit by enough attacks (melee or projectile), they die and drop XP, as pictured (the little green gems).	
9	[4-12-24 23:13] PASSED. When making contact with an enemy projectile or a melee enemy, the player takes damage with a sound effect and the health bar goes down. Once the health is depleted, the game over variant of the results screen appears.	
10	[4-12-24 23:10] PASSED. Player scene was instantiated with expected "hp" and "xp" values on second Run.	

11	<p>[04-12-24 23:38] PASSED. After the Wave Timer ran up, the Results screen popped up containing the text “You Won!”. The Player sprite did not play the death animation, as expected. The Menu button led to the Main menu. No enemy waves spawned after the Results screen occurred.</p> <p><i>Incident:</i> The Player sprite was still able to be moved when the Results screen was active. This was not expected behavior but did not cause the test to fail.</p>	
12	<p>[04-12-24 22:49] PASSED. Player sprite “picking up” a Tile during a Run causes two messages to be printed to console in succession, indicating the database successfully opened and closed. Afterwards, expected data was in the “player_item_inventory” table.</p>	

5. Overall Takeaways

The test cases themselves passed, meaning all mandatory and otherwise high priority requirements were satisfied. Some faults and/or design issues became apparent, but they did not interfere with the expected outputs. Some polishing could be done to eliminate edge cases or balance the game per game design principles, but the product is in a deliverable state.