Testing Plan

Puzzle Dots Game

Team Llama Food

Methodology

To test our game, we have decided to use black-box methodology, such as unit testing for each component and system tests to cover each user story. For *black-box testing*, we want to test if each level can be completed. We want to be able to move the dots correctly in the right direction, select the right color groups, and blend the dot colors accurately. If there are collisions, we need to test that dots will not move or accidentally blend if they are illegal moves. Once the dots are all in the right place, we have to test to see if the win state is correctly detected and the success sound is played to inform the user. After the level is complete, the next level must be loaded after and progress saved. Because black-box testing is more about functionality we want to have other people who did not code the project to playtest it. Testers should just be able to understand and specify what the desired output should be for a given input into the program.

System Testing

To test the game, we begin at the start screen where we press the start button and complete each level. We want to test if the key presses and clicks do the correct action, the level can be completed, and a winner is recognized. By testing the keys [WASDPR] we should be able to see the right dots moving on the screen, being highlighted, and dot directions changing on the screen.

W should move dots up, S should move dots down, S changes dot direction to the left, D changes dot direction to the right, P takes the player back to the start screen, and R resets the current level.

To test the rules, such as collision and blending. We need to test each colored dot: red, orange, yellow, green, purple, and blue. In our rules, we should see secondary and primary colors absorbing, colliding, and blending.

We should also be able to hear a click sounds when we select a dot, a chime sound when we complete a level, and a chop sound if there is a collision. With each move, the moves counter should be increasing and each current level should be displayed. After

each level is completed, the moves counter and level counter should reset or change. After a level is completed, a message should pop stating that the level is completed and to continue to the next one by pressing ‘w’. The new level should be displayed after the keypress. This should continue for as long as there are more levels.

For the level screen, any level can be played. Each level button should be associated with the correct level. Thus you can start at any if you get stuck on one.
For the instructions screen, understanding how to play should be clear to the player.
For the start screen, all buttons should be registered to the correct screen.
For the credits screen, a picture of the team and the members should be displayed.

Another important test is cross browser functionality. We need to check if the game works on different browsers such as Firefox, Chrome, and our favorite, Internet Explorer.

User Stories

Sprint1

User story 1: “As a product owner, I want a prototype that can demonstrate the core features of the game, so I know it is heading in the right direction.”
Task 1: Design a control scheme.
    Test: Ask for comments on the game screens.
Task 2: Set up a development workspace.
    Test: Explain our workflow.
Task 3: Create basic art resources to be used in alpha.
    Test: Expose asset folders.
Task 3: Develop a game engine.
    Test: Present game code.
Task 4: Design and implement three basic levels.
    Test: Load first three tutorial levels.

User story 2: “As a product owner, I need an integration system, so the team can collaborate.”
Task 1: Set up github repository and add team members, TAs, and professor.
    Test: Share the GitHub repository link.
Task 2: Create google docs and powerpoints for collaboration and deliverables.

Test: Make sure github account, google docs, and powerpoints exists.

User story 3: “As a product owner, I want potential features demonstrated, so I can decide how to prioritize development.”
Task 1: Implement “collisions” into the game engine.
Test: Show collision behavior for each type of dot collision.
Task 2: Explanation of controls and game rules.
Test: Make sure Instructions Screen has instructions that are readable, make sure dots can collide.

User story 4: “As a developer or programmer, I want to have a flexible code structure so I understand what changes done that arise during collaborative development.”
Task 1: Review each others code and make sure they have comments.
Test: Show game code comments.
Task 2: Make sure code is in separate files, functions, and easy to understand.
Test: Present game source code and assets.
Task 3: Test code to make sure revision didn’t break the working copy after every commit.
Test: Verify last commit results in a working master branch.

Sprint 2
User story 1: “As a product owner, I want a fully functional level, so I can begin testing the game on potential players.”
Task 1: Create a GUI which handles player input, event handlers, keyboard inputs, mouse events.
Test: Load the game menu.
Task 2: Create separate graphics object to draw components, group them together, and handles dots.
Test: Load some game screens.
Task 3: Create a game loop that can update game states and render the puzzle game.
Test: Play a game level.
Task 4: Implement three basic levels.
Test: Play three levels.

User story 3: “As a product owner, I want a solid idea of what can be included in the final release, so I can begin pitching the game. ”

Task 1: Work on game architecture and structure code. Create sensible module structure and avoid having too many dependencies.

Test: Review game source code for consistency.

Task 2: Break up individual source files for development and then unify at end.

Test: Code inspection.

Task 3: Load some external resources such as gfx objects and other assets.

Test: Code inspection.

User story 2: "As a player I want the game to recognize I am a winner, so I can feel accomplished."

Task 1: Work on state management. Figure out game values (clearing and setting them) and game modes (load, play, pause).

Test: Check that each dot is correctly placed into the correct spaces. Check to see if message pops up if game is won.

Task 2: Interactive sound when a puzzle has been solved.

Test: Listen for sound when puzzle is solved.

Sprint 3

User story 1: "As a player I want a progression of levels, so I can learn how to play and be challenged as I progress."

Task 1: Make function to add levels to the game.

Test: Load some levels from the level menu.

User story 2: "As a product owner, I want the release to be thoroughly tested, so I know it won't crash on users."

Task 1: Have each team member play the game and complete the levels. Try to find regressions in the code.

Test: As each team member is playing the game or coding it, have them test the game's functionality.

Task 2: Find outsiders/friends to play test the games. Analyze how they learn how to play the game and what they have difficulties with.

Test: Have friends/TAs play game and get feedback. Make changes as needed.

User story 4: "As a player, I want to see how well I did on a level, so I can try to improve and compete with friends."

Task 2: Draw the score to the game canvas.

Test: Play the game and count how many moves. Compare with moves counter.

User story 5: As a product owner, I want a credits and title screen, so the development team can let people know they made the game.
Task 1: Create title and credit screen object. Make it appealing to players.
Test: Load the game menu and request comments on the layout.

User Story 6: As a player, I want the game to look and sound good, so I can enjoy it more.
Task 1: Animate fluid game menu and screen transitions.
Test: Load the main menu and try some options.
Task 2: Add sounds.
Test: Try triggering each sound in the sounds folder in-game.
Task 3: Add good lighting and effects with bright color palette.
Test: Present the game and request comments on the color theme.

Unit Tests

TEST CASES

Test ID	Description	Expected Result	Actual Result
Level Playtesting	Have people outside the team play each level	Each level can be completed and the levels progress smoothly in difficulty	
Menu	Click “Start”	You are brought to the first level	
Menu	Click “Instructions”	You are shown instructions for how to play	
Menu	Click “Credits”	The Credits screen is shown	
Menu	Click “Levels”	You are brought to the level selection screen.	
Level Selection	Click on a level	A game is initialized at the level selected	

Level Selection	Press “p”	You are brought to the main menu	
Credits	Press “p”	You are brought to the main menu	
Instructions	Press “p”	You are brought to the main menu	
Gameplay	Press “p”	You are brought to the main menu	
Gameplay	Click a primary color	All dots with the same color as the dot clicked on are highlighted	
Gameplay	Click a secondary color repeatedly	Dot selection is toggled between the primary colors which form the secondary color.	
Gameplay	Press “w” with dots selected	All highlighted dots move forward that won’t collide with each other and the move count is incremented.	
Gameplay	Press “a” with dots selected	All highlighted dots are rotated counter-clockwise and the move count is incremented	
Gameplay	Press “d” with dots selected	All highlighted dots are rotated clockwise and the move count is incremented	
Gameplay	Complete a level	Completion sound effect is played, and all input besides ‘w’ and ‘p’ are ignored	
Gameplay	Press ‘w’ after completing a level	You are brought to the next level	