Plan for Testing

Puzzle Dots Game

Team Llama Food

To test our game, we have decided to use a mix of black-box methodology and white-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 is completable and winnable. 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 cannot 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 all the colored spaces contain the correct dots and the success sound is played to inform the user. After the level is complete, the next level must be loaded after and progress is 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.

For *white-box testing*, we want to be able to test that each dot state is recognized and that the code is working as intended. We want to be able to catch any defects early on and and prevent any later types of errors. We hope to use control flow testing, data flow testing, and branch testing on our code. By doing unit testing and system testing we expect to cover most paths. For system testings, we plan on testing specific inputs, interaction, output specified via text/and or UML. Likewise, we will test equivalence classes of inputs and use test cases for select classes.

TEST CASES

Our objective is to find as many defects as possible in as few test cases as possible. We plan to organize our tests in these tables as below.

Test ID	Description	Expected Result	Actual Result