SNO

SUPER NINTENDO ONLINE

TEST PLAN

Michael Kelly
mkelly01@my.fit.edu

Keith Johnson
kjohns07@my.fit.edu

Eric Wells
wellse@my.fit.edu

Faculty Sponsor

Dr. William H. Allen
wallen@cs.fit.edu

Test Plan

For our project we will be employing two testing methods. At the beginning of our project when we are unable to test full features of our program, we will be relying on unit tests to ensure the correctness of our code. Once we are to the point that we are able to run actual ROMs in some capability, we will begin performing play-tests, where we run a game and attempt to play it, noting any bugs we encounter along the way. For our program to be considered successful, it will need to pass all of the unit tests we have written for it, as well as complete a play test without finding any significant bugs.

Unit Testing

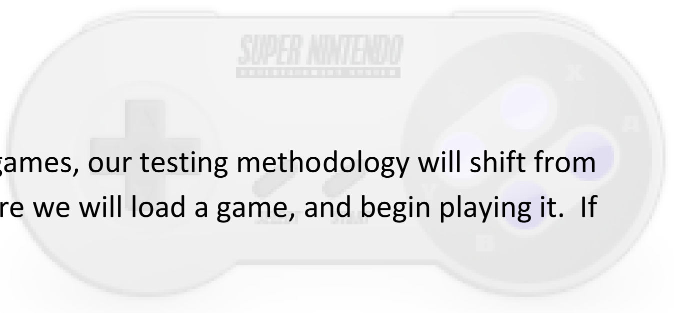
We will be conducting our unit tests using JUnit. This is a powerful unit testing framework for Java, which allows us to rapidly develop test code as we create the code. We plan for every CPU instruction to contain a set of unit tests that will attempt to fully test each instruction for correctness. This will allow us to know that we are writing correct code as we go along, before we are actually able to run a full program. We will be building our project with Apache Ant, which has the ability to generate JUnit reports. We will be using these to track our progress and make sure we do not introduce regressions to previously functioning code.

Once we have our code working and capable of running programs, as we find bugs and fix them, we will add more unit tests to make sure that we do not reintroduce the issue in the future. The benefit of having all the unit tests will allow us to restructure the program or perform refactoring without fear of breaking the code during the process. If the tests pass before the refactoring, and then again after the refactoring, we can be more confident in our changes. This will allow us to restructure our code if desired more easily and without fear.

Test Case Example for the Increment CPU Instruction

There is a CPU instruction for implementing the CPU Accumulator register. We will have 3 different tests for this instruction. We will set the initial value to zero, and then run the instruction to verify that it incremented to 1. Then we will set it to 0xFFFF, increment it, and make sure it wraps around to zero. The last test will set it to 0x7FFF, and increment it, checking that the negative flag gets set. This is just an example for one instruction. There are a possible 256 instructions for the CPU, so we will have a large test suite by the time the CPU is implemented.

Play Testing



Once we get to the point that we have playable games, our testing methodology will shift from unit testing to play testing. This is a process where we will load a game, and begin playing it. If we notice a bug or graphical fault, we will write a unit test to reproduce it if possible, and then fix the code. Once it has been fixed and passes the new unit test, we will restart the play test and attempt to recreate the bug.

Test Plans Specific to Each Requirement

The following is a list of each requirement specified, along with the steps required to test the features.

Requirement #001

1. Run our JUnit test suite.
2. Verify that all tests pass

Requirement #002

1. Choose to load a ROM from the local computer
2. Verify that the ROM loads and play begins.
3. Repeat with a ROM loaded from the remote server.

Requirement #003

1. Choose to save the game to a local file
2. Save the game state
3. Verify that a gamesave file was created on your system

Requirement #004

1. Choose to load a ROM file
2. Verify that the information shown about the game is what is expected.(I.E that the game name is correct, and various other metadata.)
3. Load an invalid ROM file
4. Verify that the load fails gracefully, and no information is displayed

Requirement #005

1. Choose to load a valid ROM file
2. Verify that the game loads
3. Load an invalid ROM file
4. Verify that the load fails gracefully

Requirement #006

1. Configure the program to load a ROM file from a remote server.
2. Test loading for a valid server location
3. Test loading for an invalid server location
4. Test loading for valid location, but invalid file type
5. Only the valid file, valid location should function properly

Requirement #007

1. Load a ROM
2. Ensure that keyboard input generates the correct response from the system

Requirement #008

1. Load a ROM
2. Verify that sounds can be heard during gameplay

Requirement #009

1. Load a ROM
2. Verify that it plays correctly
3. Load an invalid ROM
4. Verify that it does not work

Requirement #010

1. Load a ROM that supports multiple players
2. Configure the input for multiple players
3. Begin play with more than one person
4. Verify that the controls work for both players, and the game plays properly

Requirement #011

1. Load a ROM and being playing
2. Press the designated pause button
3. Verify that gameplay is suspended
4. Press the unpause button
5. Verify that gameplay resumes correctly

Requirement #012

1. Load a ROM and being playing
2. Open up the memory viewer
3. Verify that the contents of memory are displayed, and appear correct

Requirement #013

1. Load a ROM and begin playing
2. Press the designated screenshot button
3. Choose a location to save the screenshot file to
4. Verify that the file saved matches the output of the screen when the screenshot was taken

Requirement #014

1. Embed the program on a web page, making sure that it is configured to automatically load a ROM file from a remote server, along with remote save data.
2. Verify that the game loads, and playback begins from the save state.
3. Test with the configuration pointing to both valid and invalid ROMs, as well as valid and invalid network locations.

Requirement #015

1. Watch a new user navigate to a page with the software on it
2. Time how long the user takes to begin working with the software
3. Compare to the 1minute allowable time as specified in the requirement

Requirement #016

1. Load the program
2. Access the settings/configuration menu
3. Verify all required options are present, and that their values can be changed
4. Save the settings and close the program
5. Open the program again, and verify that the settings have been loaded correctly

Requirement #017

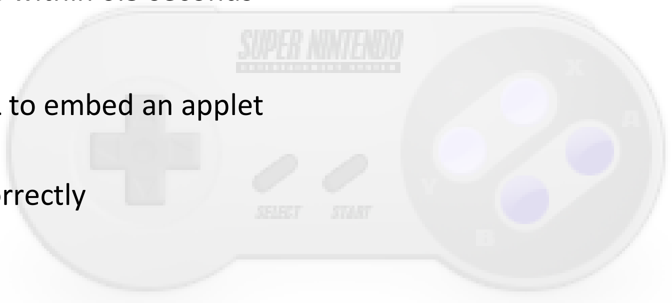
1. Load a ROM and being playback
2. Display the FPS counter
3. Verify that the FPS maintains at least 30fps for 95% of the time

Requirement #018

1. Load a ROM and being playback
2. Press a key
3. Verify that action is taken by the software within 0.5 seconds

Requirement #019

1. Create a web page with the correct HTML to embed an applet
2. Load the web page
3. Verify that the program loads and runs correctly



Requirement #020

1. Load the program
2. Access the settings/configuration menu
3. Verify that a new user understands what each of the options does

Requirement #021

1. Load a ROM and being playback
2. Toggle the display of performance statistics
3. Verify that the statistics are shown
4. Toggle again
5. Verify that they are removed

Requirement #022

1. Create a new web page with the code for the applet
2. Configure the width/height of the applet to something custom
3. Load the page, and verify that the display has adjusted to work within the confines of the new space
4. Change the width/height again
5. Reverify

Requirement #023

1. Download the distributable package
2. Verify that there are no copyrighted ROM files in the distribution