CMSC 447

Software Test Description (STD)

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Signature** |
| Caroline Cocca | POC/Team Lead | Caroline Cocca |
| James Gough | Simulation Dev | James Gough |
| Stuart Reilly | Junior Developer | Stuart Reilly |
| Kelley Schmidt | Documentation and  Diagrams Technician | Kelley Schmidt |
| Ryan Messett | Data Guru | Ryan Messett |
| Sang Nguyen | Visual Specialist | Sang Nguyen |

[1 Scope 3](#_3znysh7)

[1.1 Identification 3](#_1t3h5sf)

[1.2 System overview 3](#_4d34og8)

[1.3 Document overview 3](#_2s8eyo1)

[2 Referenced documents 3](#_3rdcrjn)

[3 Test preparations](#_lnxbz9) 4

[4 Test descriptions 4](#_3whwml4)

# Scope

## Identification

This document applied to the specific testing we shall conduct for our Python implementation of Conway’s Game of Life. The application is made to run on Windows 10 via command prompt. The Game of Life is a cellular automaton that is a single-player game with the only user input needed being the initial setup of the starting cell(s) and menu item selections. The cells are contained in squares of a grid, with each square of the grid being a specific x-y coordinate. The cells multiply, shrink, and move to form patterns according to specified algorithms based on their placement and the placement of surrounding cells. The goal of the game is to create interesting patterns through various starting cell coordinates. Title is “Game of Life”, version number 1, release number pending.

## System overview

The purpose of this application is to demonstrate a classic cellular automaton with client-specific requirements. The input of the initial state of the game is the only input required in order to play the game. The software is a visual program and various aspects can be configured by the user. The system on which the software will run is the Windows 10 operating system. A mouse is required for the system to be able to navigate the software. The users of this software are the developers, graders, and the client. There are no project sponsors as this project will have no monetary cost to produce. The acquirer for our software application is the client, Dr. Jon Squire. The developers consist of our software engineering team - Kelley Schmidt, Caroline Cocca, James Gough, Sang Nguyen, Ryan Messett, and Stuart Reilly. The support agencies for our project would be the teaching assistants, Professor Cain, and the client if support is needed. Planned operating sites not applicable. Other relevant documents are the Software Design Description, Software Requirements Specification, Software User Manual, and Software Test Report.

## Document overview

The purpose of this document is to outline specific test protocols for ensuring the “Game of Life” application is functional to the client’s requirements. Hardware and software preliminary requirements are included in this document in order for the test system to be able to run the application. Test cases for each requirement are included as well as how to measure the test was a success/failure, how to carry out the test, and what the inputs/outputs of the tests shall be.

# Referenced documents

Conway's Game of Life. (2019, March 24). Retrieved from <https://en.wikipedia.org/wiki/Conway's_Game_of_Life>

Tkinter -- Python interface to Tcl/Tk. (2019). Retrieved from <https://docs.python.org/2/library/tkinter.html>

Pandas: Python Data Analysis Library. (2019). Retrieved from <https://pandas.pydata.org>

# Test preparations

**Tests 1-10:**

* **Software preparation:** The game\_of\_life.py file should be downloaded and run in its active state on a Windows 10 operating system.

# Test descriptions

**Test 1: Requirement 1**

* **Test case(s):** 
  + Case 1: Identify that when the color GREEN is selected from the GRID/SPEED dropdown menu, GREEN appears in the squares selected on the grid.
  + Case 2: Identify that when the color RED is selected from the GRID/SPEED dropdown menu, RED appears in the squares selected on the grid.
  + Case 3: Identify that when the color BLUE is selected from the GRID/SPEED dropdown menu, BLUE appears in the squares selected on the grid.
* **Traceability:** This test verifies the requirement that the color of a square clicked on the grid shall change to the color most recently selected on the color dropdown menu.
* **Prerequisite conditions:** In order to test this requirement, the game must be in the active state on the system.
* **Test inputs:** The button for the color must be selected. A square or multiple squares must be clicked on the game board.
* **Expected test results:** The squares selected on the game board change green when green is selected, red when red is selected, and blue when blue is selected.
* **Criteria for evaluation:** Observation of the color selected and inspecting the color on the grid.
* **Test procedure:** Run the executable game file. Click the RED option from the GRID/SPEED menu. Select multiple squares on the grid and observe their color. Select the GREEN option from the GRID/SPEED menu. Select multiple squares on the grid and observe their color. Select the BLUE option from the GRID/SPEED menu. Select multiple squares on the grid and observe their color.
* **Assumptions and Constraints:** N/A

**Test 2: Requirement 2:**

* **Test case(s):** 
  + Case 1: Once the user selects the WRITE FILE button, windows explorer is launched.
  + Case 2: The game throws a proper exception if the file specified by the user does not have the correct extension.
  + Case 3: While the game is still opened, the contents of the saved file are immediately checked against the current game state.
  + Case 4: Test for the case that the user is not allowed to save the state of an empty grid to a file.
  + Case 5: The colors of the current grid points match the colors specified in the saved file.
  + Case 6: The time from the user clicking “save” in windows explorer to the time that windows explorer closes is less than or equal to 2 seconds.
* **Traceability:** This test verifies the requirement that the program shall allow for the saving of grid state to a file.
* **Prerequisite conditions:** The game must be running and there must be squares selected on the grid in order to save to a file of a specific color. There must be sufficient storage on the system to save the file.
* **Test inputs:** Click a color specified in the GRID/SPEED menu. Click to select squares on the grid. Click the WRITE FILE button. Text is input to specify the file name in windows explorer.
* **Expected test results:** 
  + It is expected that windows explorer launches when WRITE FILE is selected.
  + It is expected that the appropriate exceptions occur when there is invalid file input.
  + It is expected that the file is checked against the current game board and matches.
  + It is expected that the program will not allow the user to save an empty board.
  + It is expected that the color of each point is correctly saved to the file.
  + The time to save is expected to be no greater than 2 seconds.
* **Criteria for evaluation:** Windows explorer is launched when WRITE FILE is selected, should happen instantly. Errors are observed for invalid file input. The coordinates are compared with the game coordinates, must match. Error message is displayed when attempting to save empty board. Game board coordinate color must be one of the three colors described, otherwise test is invalid. Save time less than or equal to 2 seconds.
* **Test procedure:**
  + WRITE FILE button selected, observe that windows explorer launches.
  + Tester inputs invalid file extension when writing a file to observe error message.
  + Back-end function compares saved file and game board automatically.
  + Tester attempted to write a game state for an empty board, observes error message.
  + Tester observes the color of the saved game point and the current point on the grid.
  + Timing function activated when “save” is selected in windows explorer, tester observes error if greater than 2 seconds to save and close windows explorer and appropriate error message occurs.
* **Assumptions and Constraints:** N/A

**Test 3: Requirement 3**

* **Test case(s):**
  + Case 1: A text box for the file name input appears when READ FILE is selected.
  + Case 2: An appropriate error message appears when an invalid file name is specified.
  + Case 3: Contents of an existing save file are formatted appropriately.
  + Case 4: User not permitted to read an empty file into the game.
  + Case 5: The colors and grid points saved in the specified file appear in the grid displaying the correct color and at the correct coordinates once the file is loaded.
  + Case 6: Time between loading a valid file to its contents being displayed in the game is less than or equal to 2 seconds.
  + Case 7: Current game state is overwritten when data is loaded from a file.
* **Traceability:** This test verifies the requirement that the program shall allow for the loading of a saved grid state from a file.
* **Prerequisite conditions:** The game must be running, may be paused or active. There may or may not be any modifications already made to the game board.
* **Test inputs:** READ FILE button must be clicked. Text is typed into a text box to specify the file to be loaded.
* **Expected test results:** 
  + A text box appears successfully upon clicking READ FILE.
  + An error message appears successfully and re-prompts when invalid file name is specified.
  + An error message appears successfully and re-prompts when contents of file are not formatted correctly.
  + An error message appears successfully and re-prompts when user attempts to read in empty file.
  + Color and grid coordinates displayed after loading a file match the color and grid coordinates specified in the file.
  + An error message appears when loading time is greater than 2 seconds.
  + Game board does not retain any prior information to loading the file.
* **Criteria for evaluation:** Text box is observable and functional. Error messages are observable and make sense for the error that has occurred. Colors and point positions on grid match the data in the file. Game board visibly re-configures to new inputs from file. Load time less than or equal to 2 seconds.
* **Test procedure:**
  + Select the READ FILE button and observe that there is a text box for file name.
  + Try inputting an invalid file name and observe that there is a proper error message.
  + Input an empty file and observe that there is a proper error message.
  + Input a file with incorrectly formatted data and observe that there is a proper error message.
  + Observe the colors and grid points loaded to the game and compare them to the loaded file contents.
  + Timing function tests loading time, tester observes error message if time exceeds 2 seconds.
  + Observe that the previous game state is overwritten after loading a file.
* **Assumptions and Constraints:** N/A

**Test 4: Requirement 4**

* **Test case(s):** 
  + Case 1: The state of the game is paused.
  + Case 2: The state of the game is unpaused.
* **Traceability:** This test verifies the requirement that the program has the ability to be paused or unpaused.
* **Prerequisite conditions:** The game is in the active state.
* **Test inputs:** 
  + Button click RUN from the ACTIONS dropdown menu.
  + Button click STOP from the ACTIONS dropdown menu.
* **Expected test results:** 
  + When RUN is selected, the game is expected to proceed at the specified speed and life count/step count continue appropriately.
  + When STOP is selected, the game is expected to pause at the given game state.
* **Criteria for evaluation:** The tester evaluates the state of the selected pixels, life count, and step count with the press of STOP and RUN. Board must respond to the button click.
* **Test procedure:**
  + Click RUN from the actions menu and observe that the state of the game continues, life count continues, and step count continues at the specified speed.
  + Click STOP from the actions menu and observe that all board activity ceases.
* **Assumptions and Constraints:** N/A

**Test 5: Requirement 5**

* **Test case(s):**
  + Case 1: Game rate speeds up to a specified speed, reflected by step count rate.
  + Case 2: Game rate slows down to a specified speed, reflected by step count rate.
* **Traceability:** This test verifies the requirement that the speed of the rate of gameplay can be changed while the game is running.
* **Prerequisite conditions:** The game must be running in an active state. There may or may not be grid points selected on the grid.
* **Test inputs:** 
  + The button FASTER is selected from the GRID/SPEED dropdown menu.
  + The button SLOWER is selected from the GRID/SPEED dropdown menu.
* **Expected test results:** 
  + The rate of gameplay is expected to speed up by twice the current speed when FASTER is selected.
  + The rate of gameplay is expected to slow down by half the amount of the current speed when SLOWER is selected.
* **Criteria for evaluation:** The rate of the step count is monitored.
* **Test procedure:** FASTER is clicked from the GRID/SPEED menu. The speed of step execution must be observed to increase to 2x the current speed. SLOWER is clicked from the GRID/SPEED menu. The speed of step execution must be observed to decrease to ½ the current speed.
* **Assumptions and Constraints:** N/A

**Test 6: Requirement 6**

* **Test case(s):**
  + Case 1: The Beehive test case from the Game of Life Wikipedia is placed in the grid and the program is run
  + Case 2: The Beacon test case from the Game of Life Wikipedia is placed in the grid and the program is run
  + Case 3: The Glider test case from the Game of Life Wikipedia is placed in the grid and the program is run
* **Traceability:** This test verifies the requirement that the program displays a life count to the user.
* **Prerequisite conditions:** The game is in the active state on the system
* **Test inputs:** COLOR selection with a click, grid square selections with clicks, the RUN menu option selection with a click
* **Expected test results:** The life count updates accurately for each test case so that it accurately reflects the number of colored squares active in the grid. Specifically,
  + In case 1, the life count should always be 6 while the game is running with the Beehive pattern
  + In case 2, the life count should alternate between 6 and 8 while the game is running with the Beacon pattern. The Beacon pattern should alternate between the shape of 6 and the shape of 8 in accordance with the Game of Life rules
  + In case 3, the life count should always be 5 while the game is running with the Glider pattern. The Glider should move across the grid in accordance with the Game of Life rules
  + As squares are being initially selected in any test case, life count should increase by one with each selection to reflect the addition of a colored square
* **Criteria for evaluation:** There is no acceptable range for error on the life count value; the test is not passed if the life count is updated inaccurately at any point. The life count should update within 0.5 seconds of the addition or removal of a colored square.
* **Test procedure:** Each test case pattern is added to the grid separately. The life count is observed to increase by 1 with the addition of each square, and the life count is observed for each test case for 10 steps while the game is running.
* **Assumptions and Constraints:** N/A

**Test 7: Requirement 7**

* **Test case(s):**
  + Case 1: A single step is executed
  + Case 2: The game is run (multiple steps being executed in succession)
* **Traceability:** This test verifies the requirement that the program displays a step count to the user.
* **Prerequisite conditions:** The game is must be in the active state on the system.
* **Test inputs:** A click of the menu item STEP and the menu item RUN
* **Expected test results:** The step count increases by one with each completed step.
* **Criteria for evaluation:** This number should update within 0.5 seconds of the step executing.
* **Test procedure:** Run a single step and observe that the step count has increased within the specified time by one. Allow the game to run continuously for ten steps and observe that the step count has increased within the specified time by one for each step completed.
* **Assumptions and Constraints:** N/A

**Test 8: Requirement 8**

* **Test case(s):**
  + Case 1: The grid lines are turned on
  + Case 2: The grid lines are turned off
* **Traceability:** This test verifies the requirement that the program shall allow the user to toggle grid lines on or off.
* **Prerequisite conditions:** The game is in the active state on the system.
* **Test inputs:** A click on the menu item SHOW GRID and the menu item NO GRID
* **Expected test results:** When SHOW GRID is clicked, the grid lines appear on the GUI. When NO GRID is clicked, the grid lines are removed from the GUI.
* **Criteria for evaluation:** The grid lines should appear within 0.5 seconds of selecting SHOW GRID and disappear within 0.5 seconds of selecting NO GRID.
* **Test procedure:** While the grid lines are off, the option SHOW GRID will be clicked and appearance of grid lines will be verified. While the grid lines are on, the option NO GRID will be clicked and the disappearance of grid lines will be verified.
* **Assumptions and Constraints:** N/A

**Test 9: Requirement 9**

* **Test case(s):**
  + Case 1: The menu is checked to see if COLOR options are available to the user
* **Traceability:** This test verifies the requirement that the user may select from at least 3 color options.
* **Prerequisite conditions:** The game is in the active state on the system.
* **Test inputs:** N/A
* **Expected test results:** When viewing the GRID/SPEED dropdown menu, a COLOR option exists which allows the user 3 separate options of color to choose from.
* **Criteria for evaluation:** N/A
* **Test procedure:** Open the GRID/SPEED dropdown menu and observe if the COLOR option is visible to the user with 3 options of color.
* **Assumptions and Constraints:** N/A

**Test 10: Requirement 10**

* **Test case(s):**
  + Case 1: The Beehive test case from the Game of Life Wikipedia is placed in the grid and the program is run
  + Case 2: The Beacon test case from the Game of Life Wikipedia is placed in the grid and the program is run
  + Case 3: The Beehive test case from the Game of Life Wikipedia is placed in the grid and the program is run. After 10 steps, the PAUSE option is selected and the Glider test case from the Game of Life Wikipedia is placed in the grid. The program is run.
* **Traceability:** This test verifies the requirement that the word “Stable” is displayed to the user when the entire grid contains only stable oscillating patterns
* **Prerequisite conditions:** The game is in the active state on the system.
* **Test inputs:** Color selection with a click, grid square selections with clicks, the RUN menu option selection with a click, PAUSE menu option selection with a click
* **Expected test results:** The word “Stable” shall appear after one complete oscillation where the entire grid contains only stable patterns and shall disappear after a step where the entire grid no longer contains only stable patterns. Specifically,
  + In Case 1, the word “Stable” should appear after two steps (one complete oscillation/return to start of pattern) and remain for the duration of the game being run.
  + In Case 2, the word “Stable” should appear after three steps (one complete oscillation/return to start of pattern) and remain for the duration of the game being run
  + In Case 3, the word “Stable” should appear after two steps and remain until the Glider test case is added to the grid. Once the Glider is added, the word “Stable” should disappear.
* **Criteria for evaluation:** The word “Stable” should appear within 0.5 seconds of one complete stable oscillation of the entire grid. The word “Stable” should disappear within 0.5 seconds of one unstable step of the entire grid.
* **Test procedure:** The specified test cases are added to the grid and the game is observed continuously running for at least 10 steps. The upper right corner of the GUI is observed to verify that the word Stable appears within the specified time according to the Criteria for Evaluation, and disappears within the specified time according to the Criteria for Evaluation.
* **Assumptions and Constraints:** N/A