CMSC 447

Software Test Description (STD)

|  |  |  |
| --- | --- | --- |
| Name | Role | Signature |
| Holly Bennett | Customer/Sponsor |  |
| Khaled Elgendy | POC, Student Developer |  |
| Rachael McKenzie | Student Developer |  |
| Ryan Miller | Student Developer |  |
| Aarti Patel | Student Developer |  |
| Connor Thomas | Student Developer |  |
| Jie Zhou | Student Developer |  |

Last Updated: 5/5/2019

[**1. Scope**](#_snm4q8lrzjsf) **8**

[1.1 Identification](#_dcre7bgzcryn) 8

[1.2 System Overview](#_sspae7fwxie) 8

[1.3 Document Overview](#_eqeh7q63rf9c) 8

[**2. Referenced Documents**](#_gqpviets1gca) **8**

[**3. Test Preparations**](#_x144w2prlghf) **9**

[3.1 Test 1: Cell Survival](#_gm8vc69v57kq) 9

[3.1.1 Hardware Preparations](#_7p59mtecbanr) 9

[3.1.2 Software Preparations](#_wra6jque2sqv) 9

[3.1.3 Other Pre-Test Preparations](#_6lvu5dva4ig8) 9

[3.2 Grid Parameters](#_tk07nznkax0h) 9

[3.2.1 Hardware Preparations](#_e2hu2b46iz0m) 9

[3.2.2 Software Preparations](#_hyc4qtcvgwr7) 9

[3.2.3 Other Pre-Test Preparations](#_kpwwdp5tna5w) 9

[3.3 Website Options](#_11fu2sj4zeng) 10

[3.3.1 Hardware Preparations](#_1xaq9pwo721f) 10

[3.3.2 Software Preparations](#_uh31p1k100w3) 10

[3.3.3 Other Pre-Test Preparations](#_t33y4ibh8pyd) 10

[3.4 Stable State Detection](#_6r6alvio1byr) 10

[3.4.1 Hardware Preparations](#_xntw2p2dop6n) 10

[3.4.2 Software Preparations](#_7korwbqc296p) 10

[3.4.3 Other Pre-Test Preparations](#_et4e585fahex) 10

[3.5 Error Detection](#_u26ao06ylkqm) 10

[3.5.1 Hardware Preparations](#_i2ikqafrqrb1) 10

[3.5.2 Software Preparations](#_dvkpnhjeinyz) 10

[3.5.3 Other Pre-Test Preparations](#_mdb0ihdec31d) 11

[3.6 Hardware and Software Accessibility](#_830ao3gh7oiy) 11

[3.6.1 Hardware Preparations](#_dl0ty4v2bt7q) 11

[3.6.2 Software Preparations](#_f4v5sn6vknxd) 11

[3.6.3 Other Pre-Test Preparations](#_357qfn7iumgq) 11

[3.7 Website Display](#_ei2x8vhfga46) 11

[3.7.1 Hardware Preparations](#_cv54zp7pxnmk) 11

[3.7.2 Software Preparations](#_vcw5zyvlxqa) 11

[3.7.3 Other Pre-Test Preparations](#_u4l9e2vlec29) 11

[**4. Test Descriptions**](#_6dlct9jibie) **12**

[4.1 Test 1: Cell Survival](#_m920tyj2h1zc) 12

[4.1.1 Test 1-1: Cell Survival](#_e50g5lgtg5oo) 12

[4.1.1.1 Requirements Addressed](#_6ks08mmalz3m) 12

[4.1.1.2 Prerequisite Conditions](#_vfwzs3qg57k7) 12

[4.1.1.3 Test Inputs](#_k893f1ixqy7l) 12

[4.1.1.4 Expected Test Results](#_v9czfu5oycsc) 13

[4.1.1.5 Criteria for Evaluating Results](#_ti6h4g2zkjs5) 13

[4.1.1.6 Test Procedure](#_u65rth3m6ka9) 13

[4.1.1.7 Assumptions and Constraints](#_ma6gqwec2vte) 13

[4.1.2 Test 1-2: Cell Death](#_92cxufvzb1ap) 13

[4.1.2.1 Requirements Addressed](#_ro658xvnukbr) 13

[4.1.2.2 Prerequisite Conditions](#_son44ce2scrp) 13

[4.1.2.3 Test Inputs](#_ou8vfe93x93d) 13

[4.1.2.4 Expected Test Results](#_g9py8phq1eqv) 14

[4.1.2.5 Criteria for Evaluating Results](#_xkyc5onxdz1p) 14

[4.1.2.6 Test Procedure](#_5da5id5a2tah) 14

[4.1.2.7 Assumptions and Constraints](#_f1wzs41lujp0) 15

[4.1.3 Cell Revival](#_xdfp0n9t4hd8) 15

[4.1.3.1 Requirements Addressed](#_v0koeg198rj0) 15

[4.1.3.2 Prerequisite Conditions](#_6qcnvaebzzjh) 15

[4.1.3.3 Test Inputs](#_jw2fo97z8ti2) 15

[4.1.3.4 Expected Test Results](#_aie91wb3ukit) 15

[4.1.3.5 Criteria for Evaluating Results](#_sq4rc4gjwu5e) 16

[4.1.3.6 Test Procedure](#_sj3f9v3swvv0) 16

[4.1.3.7 Assumptions and Constraints](#_v0b939759nu5) 16

[4.2 Test 2: Grid Parameters](#_2can4rz1uj0r) 16

[4.2.1 Test 2-1: Square Grid](#_i1nkbghfq2h) 16

[4.2.1.1 Requirements Addressed](#_e0ve95uh8du5) 16

[4.2.1.2 Prerequisite Conditions](#_za4qdurrm78v) 17

[4.2.1.3 Test Inputs](#_m4hztkcwfajv) 17

[4.2.1.4 Expected Test Results](#_e401pzgagmvq) 17

[4.2.1.5 Criteria for Evaluating Results](#_ufm5sluojy1v) 17

[4.2.1.6 Test Procedure](#_5sz38pg3nb7) 17

[4.2.1.7 Assumptions and Constraints](#_6piafg791u3y) 17

[4.2.2 Test 2-2: Grid Resolution](#_dlcm6xa5eb5b) 17

[4.2.2.1 Requirements Addressed](#_o0ba0hc2nf1) 17

[4.2.2.2 Prerequisite Conditions](#_vl4oknrcvej1) 18

[4.2.2.3 Test Inputs](#_ewy2gu7h4ahq) 18

[4.2.2.4 Expected Test Results](#_dad1jfvg43kr) 18

[4.2.2.5 Criteria for Evaluating Results](#_6xy9p0kim41a) 18

[4.2.2.6 Test Procedure](#_fshq3dnt8uw0) 18

[4.2.2.7 Assumptions and Constraints](#_1wv3qi5qq476) 18

[4.2.3 Test 2-3: Grid Node Size](#_qczkw3qftqwm) 18

[4.2.3.1 Requirements Addressed](#_z752fg3g5pyq) 18

[4.2.3.2 Prerequisite Conditions](#_17pkiivkhzcc) 18

[4.2.3.3 Test Inputs](#_ysgj7ckbqcfj) 19

[4.2.3.4 Expected Test Results](#_uswik5m3bt14) 19

[4.2.3.5 Criteria for Evaluating Results](#_9c2pe0sbj1ws) 19

[4.2.3.6 Test Procedure](#_e6jioy396is) 19

[4.2.3.7 Assumptions and Constraints](#_4ylz26rj9xvw) 19

[4.2.4 Test 2-4: Node Distinction](#_pdaa1kgyv2qw) 19

[4.2.4.1 Requirements Addressed](#_82p26p8nz270) 20

[4.2.4.2 Prerequisite Conditions](#_nd36nmsq26pu) 20

[4.2.4.3 Test Inputs](#_1h1tjtjk2oh9) 20

[4.2.4.4 Expected Test Results](#_96xhpc8r02yb) 20

[4.2.4.5 Criteria for Evaluating Results](#_od08dzcs7vo) 20

[4.2.4.6 Test Procedure](#_rnnvg2qo1c9q) 20

[4.2.4.7 Assumptions and Constraints](#_kvtkoxdtreas) 20

[4.3 Test 3: Website Options](#_bxmft126546g) 20

[4.3.1 Test 3-1: Cell Solitude](#_m5brf37ih3nh) 20

[4.3.1.1 Requirements Addressed](#_mynaj09lnbwk) 20

[4.3.1.2 Prerequisite Conditions](#_2e9gg8sc0dts) 21

[4.3.1.3 Test Inputs](#_qc10qvwzk971) 21

[4.3.1.4 Expected Test Results](#_tesy88fgz9xn) 21

[4.3.1.5 Criteria for Evaluating Results](#_czyzxvw83vs9) 21

[4.3.1.6 Test Procedure](#_p2n4k5o7szq4) 21

[4.3.1.7 Assumptions and Constraints](#_8df3czrcolwr) 22

[4.3.2 Test 3-2: Cell Overpopulation](#_njpj8wyroeot) 22

[4.3.2.1 Requirements Addressed](#_51dpel49a59h) 22

[4.3.2.2 Prerequisite Conditions](#_c5k6frxnall7) 22

[4.3.2.3 Test Inputs](#_pq1j20xjmj9z) 22

[4.3.2.4 Expected Test Results](#_koui34rjuhm) 22

[4.3.2.5 Criteria for Evaluating Results](#_yzvzikxyj04v) 23

[4.3.2.6 Test Procedure](#_xmcgnifvxu9j) 23

[4.3.2.7 Assumptions and Constraints](#_p0hrmzp5n13b) 23

[4.3.3 Cell Revival](#_n6jpg0gdnn93) 23

[4.3.3.1 Requirements Addressed](#_px2vuy855cob) 24

[4.3.3.2 Prerequisite Conditions](#_pgylmufd6rdl) 24

[4.3.3.3 Test Inputs](#_6dq3nd61i4ax) 24

[4.3.3.4 Expected Test Results](#_h54zm2437hbs) 24

[4.3.3.5 Criteria for Evaluating Results](#_vsa0blqs35rb) 24

[4.3.3.6 Test Procedure](#_r4uhbpv590kk) 25

[4.3.3.7 Assumptions and Constraints](#_uxnla1eeh6mu) 25

[4.3.4 Test 3-4: Background Color](#_h29xg9a1wmkd) 25

[4.3.4.1 Requirements Addressed](#_pm164t9rjpv) 25

[4.3.4.2 Prerequisite Conditions](#_1yah69yieyf0) 25

[4.3.4.3 Test Inputs](#_r0kohgzfbyo9) 25

[4.3.4.4 Expected Test Results](#_9mn2c2ccd22) 25

[4.3.4.5 Criteria for Evaluating Results](#_y7mkwet26rg6) 26

[4.3.4.6 Test Procedure](#_27k87nsso9ae) 26

[4.3.4.7 Assumptions and Constraints](#_nctaub1wg8fs) 26

[4.3.5 Test 3-5: Cell Color](#_jzy1siuzm3xm) 26

[4.3.5.1 Requirements Addressed](#_67ba08s0fpfz) 26

[4.3.5.2 Prerequisite Conditions](#_vcgpnnorm08r) 26

[4.3.5.3 Test Inputs](#_44rtzw7vfcqh) 26

[4.3.5.4 Expected Test Results](#_rtbkp49yc92l) 26

[4.3.5.5 Criteria for Evaluating Results](#_pnjkfcmctfg5) 26

[4.3.5.6 Test Procedure](#_8bg13i7t2l3t) 27

[4.3.5.7 Assumptions and Constraints](#_y96icz38okwl) 27

[4.3.6 Test 3-6: Cell Shape](#_j1b0azz2d53g) 27

[4.3.6.1 Requirements Addressed](#_xurugywzkbei) 27

[4.3.6.2 Prerequisite Conditions](#_aq7kb87sfw9w) 27

[4.3.6.3 Test Inputs](#_uucdom1pb8rw) 27

[4.3.6.4 Expected Test Results](#_ys4evmgi2mym) 27

[4.3.6.5 Criteria for Evaluating Results](#_9e7vk8ubhrsq) 27

[4.3.6.6 Test Procedure](#_q6esoi972sy5) 27

[4.3.6.7 Assumptions and Constraints](#_divrx5vn3mxb) 28

[4.3.7 Test 3-7: Grid Size](#_xmm3rnv3wvtu) 28

[4.3.7.1 Requirements Addressed](#_rr80tn4ukv99) 28

[4.3.7.2 Prerequisite Conditions](#_hezmfzbhiywy) 28

[4.3.7.3 Test Inputs](#_4wyhh98p2d9s) 28

[4.3.7.4 Expected Test Results](#_fi3zhey6pha2) 28

[4.3.7.5 Criteria for Evaluating Results](#_t0rzbzg1yp8e) 28

[4.3.7.6 Test Procedure](#_nqweduhmn99z) 28

[4.3.7.7 Assumptions and Constraints](#_hc3ty3jejs6c) 29

[4.3.8 Test 3-8: Maximum Iterations](#_tae4x3me46yu) 29

[4.3.8.1 Requirements Addressed](#_xq79cad1f3l0) 29

[4.3.8.2 Prerequisite Conditions](#_yhd28yru6n38) 29

[4.3.8.3 Test Inputs](#_1phb3stvf7fj) 29

[4.3.8.4 Expected Test Results](#_5gnmih1p9w5q) 29

[4.3.8.5 Criteria for Evaluating Results](#_qyrzm2fbyetf) 29

[4.3.8.6 Test Procedure](#_x5blbstl27ks) 29

[4.3.8.7 Assumptions and Constraints](#_v007dtgl65z8) 29

[4.3.9 Test 3-9: Change Iterations](#_3rohxhvug9kb) 29

[4.3.9.1 Requirements Addressed](#_7vuaq7dg222r) 30

[4.3.9.2 Prerequisite Conditions](#_eb5uvyka0xcc) 30

[4.3.9.3 Test Inputs](#_psgoxvt80p6o) 30

[4.3.9.4 Expected Test Results](#_hvwm2m3fc34o) 30

[4.3.9.5 Criteria for Evaluating Results](#_fcngkye73eg2) 30

[4.3.9.6 Test Procedure](#_aik2fcah7zkg) 30

[4.3.9.7 Assumptions and Constraints](#_vrskcdq0bd96) 30

[4.3.10 Test 3-10: File Upload](#_yvrsdpdfwnl) 30

[4.3.10.1 Requirements Addressed](#_14zhbpugnwu4) 31

[4.3.10.2 Prerequisite Conditions](#_wonm6i5qavbp) 31

[4.3.10.3 Test Inputs](#_km8mh4j9xr8w) 31

[4.3.10.4 Expected Test Results](#_cu9hmju1ovuj) 31

[4.3.10.5 Criteria for Evaluating Results](#_ki66799yileg) 31

[4.3.10.6 Test Procedure](#_aymprnq02mjt) 31

[4.3.10.7 Assumptions and Constraints](#_ok3cim3j4rls) 31

[4.3.11 Test 3-11: Multiple Speeds](#_fahvptcorqw5) 31

[4.3.11.1 Requirements Addressed](#_r4i7mk3ktfzo) 31

[4.3.11.2 Prerequisite Conditions](#_1v55ts3xyf5x) 31

[4.3.11.3 Test Inputs](#_vofti5ntnrpe) 32

[4.3.11.4 Expected Test Results](#_z10kzknjpvo4) 32

[4.3.11.5 Criteria for Evaluating Results](#_s2otkvj8un03) 32

[4.3.11.6 Test Procedure](#_48oj0pbvbdy2) 32

[4.3.11.7 Assumptions and Constraints](#_4w634gbd7gai) 32

[4.4 Test 4: Stable State Detection](#_ym9aoic1f157) 32

[4.4.1 Test 4-1: Same State](#_ovj6thtfxyre) 32

[4.4.1.1 Requirements Addressed](#_ht23qk842o16) 32

[4.4.1.2 Prerequisite Conditions](#_ey5f7ltlco8h) 32

[4.4.1.3 Test Inputs](#_pqekbsw7wgcj) 33

[4.4.1.4 Expected Test Results](#_54amjlyx6bpw) 33

[4.4.1.5 Criteria for Evaluating Results](#_t0r03spmmpvo) 33

[4.4.1.6 Test Procedure](#_5ruecosu3nra) 33

[4.4.1.7 Assumptions and Constraints](#_b93feqx9uzwu) 34

[4.4.2 Test 4-2: Oscillating States](#_84sz27bnwtob) 34

[4.4.2.1 Requirements Addressed](#_8r7o00lldkk2) 34

[4.4.2.2 Prerequisite Conditions](#_7i945ypb45z0) 34

[4.4.2.3 Test Inputs](#_37t5a1nxn4fu) 34

[4.4.2.4 Expected Test Results](#_vyjpwqh95owt) 34

[4.4.2.5 Criteria for Evaluating Results](#_tqck1ks5uyda) 34

[4.4.2.6 Test Procedure](#_vs9v8qe1rf51) 35

[4.4.2.7 Assumptions and Constraints](#_j7ky54qx9zhz) 35

[4.5 Test 5: Error Detection](#_uygack596027) 35

[4.5.1 Test 5-1: Grid Size Error](#_emsjyt8gv2c5) 35

[4.5.1.1 Requirements Addressed](#_d9wjhakbmhfw) 35

[4.5.1.2 Prerequisite Conditions](#_5vr400mqkyqa) 35

[4.5.1.3 Test Inputs](#_ea73m0t36kf1) 35

[4.5.1.4 Expected Test Results](#_gr5oh471ac7m) 36

[4.5.1.5 Criteria for Evaluating Results](#_coljp2a5n3s9) 36

[4.5.1.6 Test Procedure](#_c8hsbx4yg8or) 36

[4.5.1.7 Assumptions and Constraints](#_uwukw6qz71wx) 36

[4.5.2 Test 5-2: Iteration Selection Error](#_gaz47vdn7dml) 36

[4.5.2.1 Requirements Addressed](#_35byeera73fj) 36

[4.5.2.2 Prerequisite Conditions](#_22nj97po3lcs) 36

[4.5.2.3 Test Inputs](#_d81de73m4sb7) 36

[4.5.2.4 Expected Test Results](#_w80q4qdi3dbw) 36

[4.5.2.5 Criteria for Evaluating Results](#_n59vigu7bagh) 37

[4.5.2.6 Test Procedure](#_zfi44n94cdsn) 37

[4.5.2.7 Assumptions and Constraints](#_5cmbunfaf2s) 37

[4.5.3 Test 5-3: File Errors](#_r1h78hvxk3y9) 37

[4.5.3.1 Requirements Addressed](#_o3ovhdhylpmx) 37

[4.5.3.2 Prerequisite Conditions](#_e9ogwyeivnt8) 37

[4.5.3.3 Test Inputs](#_uir7zxag4kl7) 37

[4.5.3.4 Expected Test Results](#_2philnrxhys2) 37

[4.5.3.5 Criteria for Evaluating Results](#_8lic5r9drwxg) 38

[4.5.3.6 Test Procedure](#_yshkrw5wmzww) 38

[4.5.3.7 Assumptions and Constraints](#_mogpsrvcua4) 38

[4.6 Test 6: Hardware and Software Accessibility](#_gjk6o2it1i9t) 38

[4.6.1 Test 6-1: Mac Accessibility](#_dykhwu8lef) 38

[4.6.1.1 Requirements Addressed](#_iut6lsgrmur) 38

[4.6.1.2 Prerequisite Conditions](#_ud7vta2chudz) 38

[4.6.1.3 Test Inputs](#_bcsb0ma98xax) 39

[4.6.1.4 Expected Test Results](#_lyeazbmoa0g5) 39

[4.6.1.5 Criteria for Evaluating Results](#_jzqbfbdmxa3) 39

[4.6.1.6 Test Procedure](#_qoulxqcaivi2) 39

[4.6.1.7 Assumptions and Constraints](#_gzvda12gbxms) 39

[4.6.2 Test 6-2: Chrome Accessibility](#_gj6w447zyw41) 39

[4.6.2.1 Requirements Addressed](#_4s2x2gprzo3k) 39

[4.6.2.2 Prerequisite Conditions](#_3xvuhd1s65js) 39

[4.6.2.3 Test Inputs](#_gao0z2x3she1) 39

[4.6.2.4 Expected Test Results](#_nr70knnsmvsv) 39

[4.6.2.5 Criteria for Evaluating Results](#_r95kgpgoqs9h) 39

[4.6.2.6 Test Procedure](#_ntnhzjgnebk6) 39

[4.6.2.7 Assumptions and Constraints](#_760aapdymw5s) 40

[4.7 Test 7: Website Display](#_fw33k3iwrrjm) 40

[4.7.1 Test 7-1: Display Number of Cells](#_otoul9vtp0mu) 40

[4.7.1.1 Requirements Addressed](#_j5crsey3za2g) 40

[4.7.1.2 Prerequisite Conditions](#_qls080lwop9a) 40

[4.7.1.3 Test Inputs](#_jqjfu91jmf3) 40

[4.7.1.4 Expected Test Results](#_em7ww89c4fjp) 40

[4.7.1.5 Criteria for Evaluating Results](#_3oo2s1cxp1bl) 40

[4.7.1.6 Test Procedure](#_n4gtbs3lgr74) 41

[4.7.1.7 Assumptions and Constraints](#_t5kg3mfnetyq) 41

[4.7.2 Test 7-2: Display Number of Iterations](#_9ndfwn7s4k3) 41

[4.7.2.1 Requirements Addressed](#_o14qhqsdyl0t) 41

[4.7.2.2 Prerequisite Conditions](#_35n5tmylslp3) 41

[4.7.2.3 Test Inputs](#_ye0yf5c073xz) 41

[4.7.2.4 Expected Test Results](#_i2klwbz4c151) 41

[4.7.2.5 Criteria for Evaluating Results](#_fpidndg3csyi) 42

[4.7.2.6 Test Procedure](#_mkrmlvllq9z3) 42

[4.7.2.7 Assumptions and Constraints](#_ra94ziot823x) 42

[**5. Requirements Traceability**](#_o7z362htgit7) **42**

# 

# 1. Scope

## 1.1 Identification

This document applies to the latest version of the Conway’s Game of Life simulation developed by Team Segmentation Fault and sponsored by Holly Bennett. Currently the software is in the implementation stage of development.

## 1.2 System Overview

The purpose of the software is to model Conway’s Game of Life, a cellular automaton that evolves based upon its initial state. The software will take the form of a website that is accessible and executable by a Chrome browser. It will possess the ability to adjust various factors of the simulation, such as initial state, factors for the cell survival, reproduction, death, and be able to identify and halt the simulation when a stable state is reached. The sponsor and user of the software is Holly Bennett, while the developers consist of Khaled Elgendy, Rachael McKenzie, Ryan Miller, Aarti Patel, Connor Thomas, and Jie Zhou, henceforth collectively referenced as Team Segmentation Fault or simply Segmentation Fault.

## 1.3 Document Overview

The purpose of this document is to describe the tests used to ensure that the software meets requirements, including the necessary test preparations, the test procedure, the expected test results and the criteria used to evaluate them, and the assumptions, constraints, and limitations imposed on the test cases.

# 2. Referenced Documents

Software Requirements Specification for Team Segmentation Fault’s Game of Life, Ver. 1.0, Last Updated: 4/20/2019

Software Design Description for Team Segmentation Fault’s Game of Life, Ver 1.0, Last Updated: 4/20/2019

# 3. Test Preparations

## 3.1 Test 1: Cell Survival

### 3.1.1 Hardware Preparations

A Mac laptop shall be used to test the software. It must contain a functional network adapter and have access to a network. The laptop must be capable of displaying a 1080p resolution. Beyond this the software does not require any specific hardware, as the software application is not resource-intensive.

### 3.1.2 Software Preparations

The software that is to be tested is Segmentation Fault’s Game of Life, which shall be stored and hosted on Github. It should be confirmed that the website is online and accessible before testing begins. A Chrome browser ver. 72, should be installed on the Mac and used to conduct the tests. Before each test is conducted, the ‘New’ button shall be used in the software in order to reset it to a default state (see the software manual for reference).

### 3.1.3 Other Pre-Test Preparations

Test files shall be made in advance before the tests begin. Members of the test team should come prepared to record test results.

## 3.2 Grid Parameters

### 3.2.1 Hardware Preparations

See 3.1.1.

### 3.2.2 Software Preparations

See 3.1.2.

### 3.2.3 Other Pre-Test Preparations

See 3.1.3.

## 3.3 Website Options

### 3.3.1 Hardware Preparations

See 3.1.1.

### 3.3.2 Software Preparations

See 3.1.2.

### 3.3.3 Other Pre-Test Preparations

See 3.1.3.

## 3.4 Stable State Detection

### 3.4.1 Hardware Preparations

See 3.1.1.

### 3.4.2 Software Preparations

See 3.1.2.

### 3.4.3 Other Pre-Test Preparations

See 3.1.3.

## 3.5 Error Detection

### 3.5.1 Hardware Preparations

See 3.1.1.

### 3.5.2 Software Preparations

See 3.1.2.

### 3.5.3 Other Pre-Test Preparations

See 3.1.3.

## 3.6 Hardware and Software Accessibility

### 3.6.1 Hardware Preparations

See 3.1.1.

### 3.6.2 Software Preparations

See 3.1.2.

### 3.6.3 Other Pre-Test Preparations

See 3.1.3.

## 3.7 Website Display

### 3.7.1 Hardware Preparations

See 3.1.1.

### 3.7.2 Software Preparations

See 3.1.2.

### 3.7.3 Other Pre-Test Preparations

See 3.1.3.

# 4. Test Descriptions

## 4.1 Test 1: Cell Survival

### 4.1.1 Test 1-1: Cell Survival

The purpose of this test case, known as Test 1-1, is to ensure that cells in the game will survive when they posses a given number of neighbors. This shall be done through the uploading of a set test file, then observation over a given number of iterations.

#### 4.1.1.1 Requirements Addressed

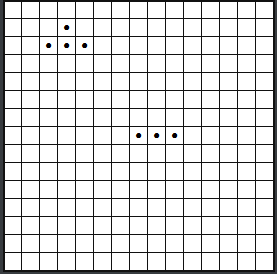
This test case addresses requirement 3.2.1.4, which states that living cells shall survive if they have 2 or 3 neighbors.

#### 4.1.1.2 Prerequisite Conditions

The software shall be prepared according to the test preparations in section 3.1. The game shall be returned to a default state by use of the ‘New’ button. The game speed shall be reduced to ‘Slow’ to allow for the easier observation of results. A test file, Text1-1.txt shall be prepared for use as input.

#### 4.1.1.3 Test Inputs

A file known as Test1-1.txt shall be prepared and used for this test. It will contain two unique clusters of cells, one with three cells and another with four. One cell shall serve as the primary cell, the others will serve as neighbors for the cell to be observed. When uploaded, the file shall produce the following game board:



This same file will be used for multiple test iterations.

#### 4.1.1.4 Expected Test Results

When any given cell in the game has either two or three neighbors, that cell will continue to live.

#### 4.1.1.5 Criteria for Evaluating Results

To pass the test, the game must meet the expected result after each number of iterations. These expectations are as follows:

1. After 1 iteration, both cell (2,3) and (7,8) will be alive.
2. After 2 iterations, cell (2,3) will have died. (7,8) will continue to live.
3. After 10 iterations, cell (2,3) will still be dead, (7,8) will still continue to live.

#### 4.1.1.6 Test Procedure

1. Reset the board using the ‘New’ button.
2. Upload the test file Test1-1.txt
3. Set the amount of iterations, starting at 1 and increasing which each subsequent test.
4. Set the game speed to slow to allow for easier observation of results.
5. Enable play.
6. Record results
7. Repeat the test for 1, 2, and 10 iterations.

#### 4.1.1.7 Assumptions and Constraints

For this test, it is assumed that the file upload will work correctly. It is also assumed that the game board will update and display the cell data correctly.

### 4.1.2 Test 1-2: Cell Death

The purpose of this test case, known as Test 1-2, is to ensure that cells in the game will die over when they do not possess a given number of neighbors. This shall be done through the uploading of a set test file, then observation over a given number of iterations.

#### 4.1.2.1 Requirements Addressed

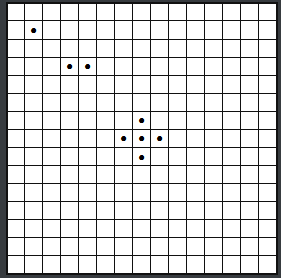
This test addresses requirements 3.2.1.1 and 3.2.1.2, which state that living cells shall die if they have less than 2 or greater than 3 neighbors.

#### 4.1.2.2 Prerequisite Conditions

The software shall be prepared according to the test preparations in section 3.1. The game shall be returned to a default state by use of the ‘New’ button. The game speed shall be reduced to ‘Slow’ to allow for the easier observation of results. A test file, Text1-2.txt shall be prepared for use as input.

#### 4.1.2.3 Test Inputs

A test file known as Test1-2.txt shall be prepared and used for this test. It shall contain several clusters of cells: a single, solitary cell (1,1), two cells at (3,3) and (3,4), and a large cluster of 4 at (7,7), (7,8), (7,6), (8,7), and (6,7) respectively. When the test file is uploaded, it should produce the following game board:



This file shall be used for multiple test iterations.

#### 4.1.2.4 Expected Test Results

When any cell in the game possesses either less than 2 or greater than 3 neighbors, that cell will die.

#### 4.1.2.5 Criteria for Evaluating Results

To pass this test, the game must meet the expected test results after each number of iterations. For testing purposes, these expectations are as follows:

1. After 1 iteration, the cells at (1,1), (3,3), and (3,4), not having the required number of neighbors, should be dead. The cell at (7,7), having too many neighbors, will be dead. The remainder of the cells should continue to survive.
2. After 2 iterations, the cells at (1,1), (3,3), and (3,4) will continue to remain dead. (7, 7) should also be dead; furthermore, more cells at (7,6), (7,8), (6,7), and (8,7) will have all died. New cells should have been added to the board.
3. After 10 iterations, more cells should have died. All of the original set will be dead, as well as the majority of the revived cells. The game will have settled into a stable oscillating state.

#### 4.1.2.6 Test Procedure

1. Reset the board using the ‘New’ button.
2. Upload the test file Test1-2.txt
3. Set the amount of iterations, starting at 1 and increasing which each subsequent test.
4. Set the game speed to slow to allow for easier observation of results.
5. Enable play.
6. Record results
7. Repeat the test for 1, 2, and 10 iterations.

#### 4.1.2.7 Assumptions and Constraints

For this test, it is assumed that the file upload will work correctly. It is also assumed that the game board will update and display the cell data correctly.

### 4.1.3 Cell Revival

The purpose of this test case, known as Test 1-3, is to ensure that cells in the game will revive when they posses a given number of neighbors. This shall be done through the uploading of a set test file, then observation over a given number of iterations.

#### 4.1.3.1 Requirements Addressed

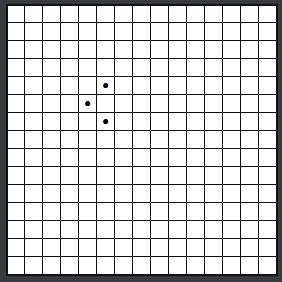
This test addresses requirement 3.2.1.3, which states that dead cells shall revive if they have 3 living neighbors.

#### 4.1.3.2 Prerequisite Conditions

The software shall be prepared according to the test preparations in section 3.1. The game shall be returned to a default state by use of the ‘New’ button. The game speed shall be reduced to ‘Slow’ to allow for the easier observation of results. A test file, Text1-3.txt shall be prepared for use as input.

#### 4.1.3.3 Test Inputs

A test file known as Test1-3.txt shall be prepared and used for this test. It shall contain a single cluster of cells located at (4,5), (5,4), and (6,5), with location (5,5) empty and the target for testing purposes. When uploaded, the test file will produce the following game board:



#### 4.1.3.4 Expected Test Results

When a given space on the grid is surrounded by three living cells, a cell will be revived in that space.

#### 4.1.3.5 Criteria for Evaluating Results

To pass the test, the software must meet the expected test results. To do this, after using file Test1-3.txt as input, a cell should revive in space (5,5) after one iteration.

#### 4.1.3.6 Test Procedure

1. Reset the board using the ‘New’ button.
2. Upload the test file Test1-3.txt
3. Set the amount of iterations, starting at 1 and increasing which each subsequent test.
4. Set the game speed to slow to allow for easier observation of results.
5. Enable play.
6. Record results

#### 4.1.3.7 Assumptions and Constraints

For this test, it is assumed that the file upload will work correctly. It is also assumed that the game board will update and display the cell data correctly.

## 4.2 Test 2: Grid Parameters

### 4.2.1 Test 2-1: Square Grid

The purpose of this test case is to ensure that the game board, also known as the grid, will continue to hold it’s square shape when it’s size is adjusted.

#### 4.2.1.1 Requirements Addressed

This test addresses requirement 3.2.1.5, which states that the game shall be implemented on a square grid.

#### 4.2.1.2 Prerequisite Conditions

The software shall be prepared according to the instructions given in 3.2. The ‘New’ button shall be used to return the game to a default state. No test file is necessary for this test.

#### 4.2.1.3 Test Inputs

The only input necessary for this test is to enter various sizes for the game board.

#### 4.2.1.4 Expected Test Results

The default board should be rectangular. The board should continue to remain rectangular, despite any changes to the board size.

#### 4.2.1.5 Criteria for Evaluating Results

To pass this test, the board must meet all of the expected test results, both before the testing begins and while the testing takes place. Any failure to abide by them shall be counted as a fail sure.

#### 4.2.1.6 Test Procedure

Once any necessary preparations have been made: the test shall proceed as follows:

1. Click the ‘New’ button
2. Count the number of squares making up the x and y axises of the grid.
3. Record result.
4. Click ‘New’ again.
5. This time, enter size values for the grid.
6. Count the number of squares making up the x and y axises of the grid.
7. Record result.
8. Repeat steps 5 through 7 two more times to ensure consistency.

#### 4.2.1.7 Assumptions and Constraints

This test is constrained by another component of the software, namely whether the ability to change the grid size has been implemented and is functioning correctly. Thus, it is forced to assume that this is the case. If it is not, then this test should be delayed until such time as valid results can be obtained.

### 4.2.2 Test 2-2: Grid Resolution

The purpose of this test case is to ensure that the game shall display in the required resolution.

#### 4.2.2.1 Requirements Addressed

This test addresses requirement 3.2.1.6, which states that the game shall display in 1080p resolution.

#### 4.2.2.2 Prerequisite Conditions

The page will be designed to take up the entire page of a 1080p screen.

#### 4.2.2.3 Test Inputs

N/A. To test load the page on a chrome browser window.

#### 4.2.2.4 Expected Test Results

Page should take up the entire screen of browser window. If the computer has a zoom applied to the display setting than the page will be larger than the browser window and scrolling will be needed.

#### 4.2.2.5 Criteria for Evaluating Results

To pass this test the entire page should be displayed correctly on a chrome browser on a 1080p display

#### 4.2.2.6 Test Procedure

1. Open google chrome
2. Paste: “<https://ryamil1.github.io/CMSC447Project/>” (excluding quotes) into the url bar and press enter
3. Record results.

#### 4.2.2.7 Assumptions and Constraints

1. It is assumed that the display will be set to a 1920p-1080p aspect ratio
2. It is expected that the screen will be set to 100% zoom.

### 4.2.3 Test 2-3: Grid Node Size

The purpose of this test case is to ensure that the grid nodes will not change size, no matter what the grid size is set to.

#### 4.2.3.1 Requirements Addressed

This test addresses requirement 3.2.1.7, which states that the game shall display nodes at a fixed size, regardless of the world size.

#### 4.2.3.2 Prerequisite Conditions

The grid nodes will be designed according to requirement 3.2.1.7. Requirement 3.2.2.8 will need to completed in-order to adjust the grid size

#### 4.2.3.3 Test Inputs

Various grid sizes will be used. Grid size parameters will be in the following format (x,y) with x being the value to be entered in the Grid Size X column on the page and y being the value to be entered in the Grid Size Y column of the page.

First Test: (1,1)

Second Test: (5,5)

Third Test: (15,15)

Fourth Test: (30,30)

Fifth Test: (50,40)

Sixth Test: (100,40)

#### 4.2.3.4 Expected Test Results

The grid nodes should maintain the same size regardless of the change in grid size

#### 4.2.3.5 Criteria for Evaluating Results

The test will be evaluated using visual inspection. No noticeable change should be detected in node size as the total grid size increases

#### 4.2.3.6 Test Procedure

1. Load the page
2. Enter the First Test in section 4.2.3.3
3. Press New
4. Inspect size of single node visually
5. Enter Second Test in section 4.2.3.3
6. Press New
7. Inspect to see if there has been a noticeable change in size
8. Record results.
9. Repeat steps 5-8 for each of the remaining Test Inputs in section 4.2.3.3

#### 4.2.3.7 Assumptions and Constraints

1. It is assumed that the screen will have a 1920-1080p screen resolution
2. It is assumed zoom will be set to 100% on the computer display

### 4.2.4 Test 2-4: Node Distinction

The purpose of this test case is to ensure that the nodes in the game grid are visually distinct.

#### 4.2.4.1 Requirements Addressed

This test addresses requirement 3.2.1.8, which states the game shall feature nodes which are visually distinguishable from one another

#### 4.2.4.2 Prerequisite Conditions

The game grid must be implemented before this test.

#### 4.2.4.3 Test Inputs

A grid size, in this case 10, must be entered into the ‘Grid Size’ boxes. There is no other input required.

#### 4.2.4.4 Expected Test Results

The amount of nodes displayed should match that entered into the ‘Grid Size’ boxes.

#### 4.2.4.5 Criteria for Evaluating Results

To pass this test it should be possible to count that the number of nodes displayed in the game grid matches that entered, as this proves they are easily discerned from each other.

#### 4.2.4.6 Test Procedure

1. Enter 10 into both the X and Y ‘Grid Size’ boxes.
2. Count the number of nodes.
3. Record result

#### 4.2.4.7 Assumptions and Constraints

It is assumed that the game board has been implemented.

## 4.3 Test 3: Website Options

### 4.3.1 Test 3-1: Cell Solitude

The purpose of this test is to ensure that the website both possesses the ability to adjust the minimum number of neighbors required before a cell dies and that said implementation functions correctly.

#### 4.3.1.1 Requirements Addressed

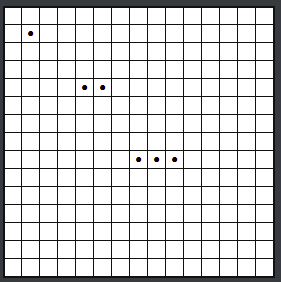
This test addresses requirement 3.2.2.1 and 3.2.2.4, which states the website shall implement the ability to adjust the number of neighbors for a cell to die from solitude.

#### 4.3.1.2 Prerequisite Conditions

All preparations in section 3.3 should be completed. All aspects of cell survival, but most importantly, cell survival and death based upon the default number of neighbors, should be completed before beginning this test. A test file, Test3-1.txt, shall be prepared, which shall be used to ensure consistent test results.

#### 4.3.1.3 Test Inputs

This test requires the use of a test file, known as Test3-1.txt. In addition, it requires input into the website, specifically the ‘Neighbors to Survive’ input box. 3 sets of values shall be used in testing, specifically 1, 2, and 3. The test file should produce the following game board:



#### 4.3.1.4 Expected Test Results

Cells who do not meet the given input for minimum number of neighbors will die. Cells which do meet this minimum (but do not meet the maximum) will continue to survive.

#### 4.3.1.5 Criteria for Evaluating Results

Each test case used must meet the expected test results. These results will be inspected visually and recorded for each test. For each test, the following results are expected:

1. With a 1 neighbor minimum, the solitary cell at (1,1) will die.
2. With a 2 neighbor minimum, both the solitary cell and the cells at (4,4) and (4,5) will die.
3. With a 3 neighbor minimum, all of the original cells will die; however, new cells may be revived depending upon those settings.

#### 4.3.1.6 Test Procedure

1. Click the ‘New’ button to reset the game board.
2. Set the number of iterations to 1.
3. Set game speed to slow.
4. Upload Test3-1.txt.
5. Set ‘Neighbors to Survive: X’ to 1.
6. Run game.
7. Record results.
8. Click ‘New’ to reset the board.
9. Repeat steps 4 through 8 using 2 and 3 neighbors as input.

#### 4.3.1.7 Assumptions and Constraints

This test assumes that the game board is functioning correctly and that the file upload functions correctly. It assumes that cell death, survival, and revival are functional before this test is performed. As such, it is constrained by these components.

### 4.3.2 Test 3-2: Cell Overpopulation

The purpose of this test is to ensure that the website both possesses the ability to adjust the maximum number of neighbors required before a cell dies and that said implementation functions correctly.

#### 4.3.2.1 Requirements Addressed

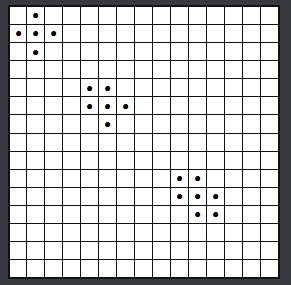
This test addresses requirement 3.2.2.2 and 3.2.2.4, which states the website shall implement the ability to adjust the number of neighbors for a cell to die from overpopulation.

#### 4.3.2.2 Prerequisite Conditions

All preparations in section 3.3 should be completed. All aspects of cell survival, but most importantly, cell survival and death based upon the default number of neighbors, should be completed before beginning this test. A test file, Test3-2.txt, shall be prepared, which shall be used to ensure consistent test results.

#### 4.3.2.3 Test Inputs

This test requires the use of a test file, known as Test3-2.txt. In addition, it requires input into the website, specifically the ‘Neighbors to Survive’ input box. 3 sets of values shall be used in testing, specifically 4, 5, and 6. The test file should produce the following game board:



#### 4.3.2.4 Expected Test Results

Cells who do not meet the given input for maximum number of neighbors will die. Cells which do meet this maximum (but do not meet the minimum) will continue to survive.

#### 4.3.2.5 Criteria for Evaluating Results

Each test case used must meet the expected test results. These results will be inspected visually and recorded for each test. For each test, the following results are expected:

1. With a 4 neighbor maximum, the cell at (1,1) will survive, while (5,5) and (10,10) will die.
2. With a 5 neighbor maximum, (1,1) and (5,5) will survive, while (10,10) will die.
3. With a 6 neighbor maximum, (1,1), (5,5), and (10,10) all survive.

#### 4.3.2.6 Test Procedure

1. Click the ‘New’ button to reset the game board.
2. Set the number of iterations to 1.
3. Set game speed to slow.
4. Set ‘Neighbors to Survive: Y’ to 4.
5. Upload Test3-2.txt.
6. Run game.
7. Record results.
8. Click ‘New’ to reset the board.
9. Repeat steps 4 through 8 using 5 and 6 neighbors as input.

#### 4.3.2.7 Assumptions and Constraints

This test assumes that the game board is functioning correctly and that the file upload functions correctly. It assumes that cell death, survival, and revival are functional before this test is performed. As such, it is constrained by these components.

### 4.3.3 Cell Revival

The purpose of this test is to ensure that the website both possesses the ability to adjust the number of neighbors required before a cell revives and that said implementation functions correctly.

#### 4.3.3.1 Requirements Addressed

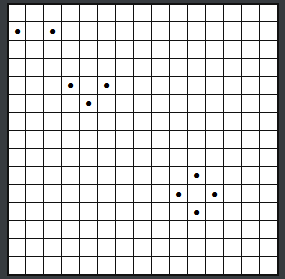
This test addresses requirement 3.2.2.3, which states the website shall implement the ability to adjust the number of neighbors for a cell to revive from death.

#### 4.3.3.2 Prerequisite Conditions

All preparations in section 3.3 should be completed. All aspects of cell survival, but most importantly, cell survival and death based upon the default number of neighbors, should be completed before beginning this test. A test file, Test3-3.txt, shall be prepared, which shall be used to ensure consistent test results.

#### 4.3.3.3 Test Inputs

This test requires the use of a test file, known as Test3-3.txt. In addition, it requires input into the website, specifically the ‘Neighbors to Revive’ input box. 3 sets of values shall be used in testing, specifically 0, 2, and 4. The test file should produce the following game board:



#### 4.3.3.4 Expected Test Results

Cells who meet the given input for revival will revive. Those who do not, will not.

#### 4.3.3.5 Criteria for Evaluating Results

Each test case used must meet the expected test results. These results will be inspected visually and recorded for each test. For each test, the following results are expected:

1. With revival set at 0, no cells will revive.
2. With revival set at 2, the cells at (1,1), (0,1), (2,1), (3,4), (5,3), (5,5), (9,9), (9,11), (11,9), and (11,11) will revive.
3. With revival set at 4, only the cell at (10,10) will revive.

#### 4.3.3.6 Test Procedure

1. Click the ‘New’ button to reset the game board.
2. Set the number of iterations to 1.
3. Set game speed to slow.
4. Upload Test3-3.txt.
5. Set ‘Neighbors to Revive’ to 0.
6. Run game.
7. Record results.
8. Click ‘New’ to reset the board.
9. Repeat steps 4 through 8 using 2 and 4 neighbors as input.

#### 4.3.3.7 Assumptions and Constraints

This test assumes that the game board is functioning correctly and that the file upload functions correctly. It assumes that cell death, survival, and revival are functional before this test is performed. As such, it is constrained by these components.

### 4.3.4 Test 3-4: Background Color

The purpose of this test is to ensure that the website functionality which allows for the background color of the game board is implemented correctly.

#### 4.3.4.1 Requirements Addressed

This test addresses requirement 3.2.2.5, which states the website shall implement the ability to adjust the background color of the game using 16 bit colors.

#### 4.3.4.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.4.3 Test Inputs

For this test, three colors will be chosen from the menu provided by the color button. All colors should be different, and none should be white.

#### 4.3.4.4 Expected Test Results

The background of the game will change to the color chosen in the grid color popup.

#### 4.3.4.5 Criteria for Evaluating Results

A visual examination of the game board and comparison to the color chosen in the menu may be made. If they are identical, then the expected test results have been achieved.

#### 4.3.4.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Click on the ‘Grid Color’ button.
3. Select any color other than white from the popup that appears.
4. Hit ‘OK’.
5. Record results.
6. Repeat steps 1 through 5 twice more with different colors to ensure consistency.

#### 4.3.4.7 Assumptions and Constraints

This test assumes that the game board has been implemented.

### 4.3.5 Test 3-5: Cell Color

The purpose of this test is to ensure that the website functionality which allows for the user to change the color of the cells is implemented correctly.

#### 4.3.5.1 Requirements Addressed

This test addresses requirement 3.2.2.6, which states the website shall implement the ability to adjust the color of cells using 16 bit colors.

#### 4.3.5.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.5.3 Test Inputs

For this test, three colors will be chosen from the menu provided by the color button. All colors should be different, and none should be black. Any test file may be used, provided it contains at least one cell.

#### 4.3.5.4 Expected Test Results

The color of the cells in the game will change to the color chosen in the cell color popup.

#### 4.3.5.5 Criteria for Evaluating Results

A visual examination of the cells and comparison to the color chosen in the menu may be made. If they are identical, then the expected test results have been achieved.

#### 4.3.5.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Upload any test file with valid coordinates.
3. Click on the ‘Cell Color’ button.
4. Select any color other than white from the popup that appears.
5. Hit ‘OK’.
6. Record results.
7. Repeat steps 1 through 5 twice more with different colors to ensure consistency.

#### 4.3.5.7 Assumptions and Constraints

This test assumes that the game board has been implemented.

### 4.3.6 Test 3-6: Cell Shape

The purpose of this test is to ensure that the website functionality which allows for the user to change the shape of the cells is implemented correctly.

#### 4.3.6.1 Requirements Addressed

This test addresses requirement 3.2.2.7, which states the website shall implement the ability to select the shape of the cells. The default shape shall be circles. Additional selectable shapes shall include triangles and squares.

#### 4.3.6.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.6.3 Test Inputs

For this test, a shape will be selected as input from the ‘Cell Shape’ dropdown menu. Any test file may be used to display cells, provided it contains at least one cell.

#### 4.3.6.4 Expected Test Results

When a file is first uploaded, the cells will display as circles. After a different shape is selected, the game will update to show the cells in that shape.

#### 4.3.6.5 Criteria for Evaluating Results

A visual examination of the cells and comparison to the shape chosen may be used to verify results. If they are identical, the test is passed.

#### 4.3.6.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Upload any test file with valid coordinates.
3. Click on the ‘Cell Shape’ button.
4. Select Square
5. Record results.
6. Repeat steps 1 through 4, only using Triangle in place of Square.

#### 4.3.6.7 Assumptions and Constraints

This test assumes that the game board has been implemented.

### 4.3.7 Test 3-7: Grid Size

The purpose of this test is to ensure that the website functionality which allows for the user to change the size of the grid is implemented correctly.

#### 4.3.7.1 Requirements Addressed

This test addresses requirement 3.2.2.8, which states the website shall implement the ability to change the size of the grid in the game.

#### 4.3.7.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.7.3 Test Inputs

For this test, two values, one X, one Y, will be entered into the appropriate boxes labelled ‘Grid Size’. Any test file may be used to trigger the size change.

#### 4.3.7.4 Expected Test Results

When a given X and Y are entered into the website, the game board will increase or decrease in size to match the values given.

#### 4.3.7.5 Criteria for Evaluating Results

A visual inspection of the grid board, namely counting one row and one column to determine the number of nodes present, may be performed. If these numbers match the values entered into Grid Size, then the test is passed.

#### 4.3.7.6 Test Procedure

1. Reset the game using ‘New’.
2. Input a value into ‘Grid Size: X’.
3. Input a value into ‘Grid Size: Y’.
4. Upload a test file.
5. Observe and record the results.
6. Repeat steps 1 through 5 with different values to confirm.

#### 4.3.7.7 Assumptions and Constraints

This test assumes that the game board has been implemented.

### 4.3.8 Test 3-8: Maximum Iterations

The purpose of this test is to ensure that the game will run to the maximum number of iterations specified.

#### 4.3.8.1 Requirements Addressed

This test addresses requirement 3.2.2.9, which states the website shall run the game to a maximum of 216,000 iterations.

#### 4.3.8.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.8.3 Test Inputs

For this test, 216,000 iterations will be entered into the input box labeled ‘Number of Iterations’. A test file will be prepared and used in this test.

#### 4.3.8.4 Expected Test Results

It is expected that the game will continue until 216,000 iterations have been met.

#### 4.3.8.5 Criteria for Evaluating Results

To pass this test, the game must continue until 216,000 iterations have been reached, as indicated by the Number of Iterations counter located underneath the game board. If this happens, then the test is passed.

#### 4.3.8.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Set the ‘Number of Iterations’ to 216,000
3. Upload the test file.
4. Click ‘Play’.
5. Wait for game to complete
6. Record results.

#### 4.3.8.7 Assumptions and Constraints

This test assumes that the board, cell death and survival, and the iteration counter are working correctly. It is constrained by these other components.

### 4.3.9 Test 3-9: Change Iterations

The purpose of this test is to ensure that the website functionality to change the number of iterations has been correctly implemented.

#### 4.3.9.1 Requirements Addressed

This test addresses requirement 3.2.2.10, which states the website shall implement the ability to change the default number of maximum iterations to a value between 1 and 216,000.

#### 4.3.9.2 Prerequisite Conditions

Beyond the game board being implemented, there are no prerequisite conditions.

#### 4.3.9.3 Test Inputs

For this test, various test iterations will be entered into the input box labeled ‘Number of Iterations’. A test file will be prepared and used in this test.

#### 4.3.9.4 Expected Test Results

It is expected that the game will continue until the given number of iterations have been met.

#### 4.3.9.5 Criteria for Evaluating Results

To pass this test, the game must continue until the given number of iterations have been reached, as indicated by the Number of Iterations counter located underneath the game board. If this happens, then the test is passed.

#### 4.3.9.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Set the ‘Number of Iterations’ to 1,000.
3. Upload the test file.
4. Click ‘Play’.
5. Wait for game to complete
6. Record results.
7. Repeat steps 1 through 6 with 10,000 and 100,000 as inputs.

#### 4.3.9.7 Assumptions and Constraints

This test assumes that the board, cell death and survival, file upload, and the iteration counter are working correctly. It is constrained by these other components.

### 4.3.10 Test 3-10: File Upload

The purpose of this test is to ensure that the file upload function of the website is implemented correctly.

#### 4.3.10.1 Requirements Addressed

This test addresses requirement 3.2.2.11 and 3.2.1.9, which states the website shall accept a .txt text file containing grid coordinates as input for the game.

#### 4.3.10.2 Prerequisite Conditions

This test requires that the game board has already been implemented. There are no other prerequisites.

#### 4.3.10.3 Test Inputs

Test file Test1-3.txt may be reused for this test.

#### 4.3.10.4 Expected Test Results

When the test file is uploaded, the game board will update will cells at the given coordinates.

#### 4.3.10.5 Criteria for Evaluating Results

When test file Test1-3.txt is used, cells will appear at (5,4), (4,5), and (6,5). If this occurs, then the test is passed.

#### 4.3.10.6 Test Procedure

1. Click ‘New’ to reset the game board.
2. Upload file Test1-3.txt.
3. Observe and record results.

#### 4.3.10.7 Assumptions and Constraints

This test assumes that the game board has been implemented correctly.

### 4.3.11 Test 3-11: Multiple Speeds

The purpose of this test is to ensure that the ability to select multiple speeds for the game has been properly implemented into the website.

#### 4.3.11.1 Requirements Addressed

This test addresses requirement 3.2.2.12 and 3.2.2.13, which state the website should be able to run the game at multiple speeds and that the user should be able to select between them.

#### 4.3.11.2 Prerequisite Conditions

This test requires that the game board and file upload have already been implemented. There are no other prerequisites.

#### 4.3.11.3 Test Inputs

Test file Test1-3.txt may be reused for this test. Speeds will be selected from the ‘Speed’ dropdown menu.

#### 4.3.11.4 Expected Test Results

It is expected that the game speed will change according to what speed is selected.

#### 4.3.11.5 Criteria for Evaluating Results

To pass this test, there must be a noticable difference between the slow, normal, and fast speeds, with slow being the slowest and fast being the fastest.

#### 4.3.11.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Set ‘Speed’ to Slow.
3. Set iterations to 10.
4. Upload Test1-3.txt.
5. Time the amount of time it takes for the game to run.
6. Record results.
7. Repeat steps 1 through 6 for normal and fast.

#### 4.3.11.7 Assumptions and Constraints

This test assumes that the game board and file upload have been implemented correctly.

## 4.4 Test 4: Stable State Detection

### 4.4.1 Test 4-1: Same State

The purpose of this test is to ensure that the game will detect and halt if a stable state consisting of an unchanging pattern has been reached.

#### 4.4.1.1 Requirements Addressed

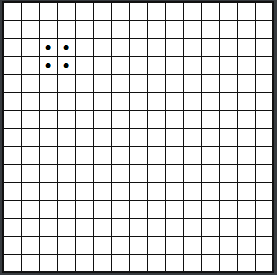
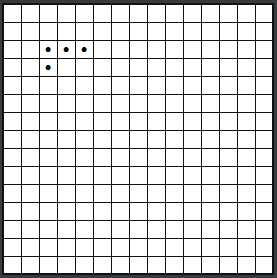
This test addresses requirement 3.2.4.1, which states the game shall stop if it is in the same state for two iterations.

#### 4.4.1.2 Prerequisite Conditions

This test requires that the game board and file upload have been implemented. In addition, that the game follows the rules set out by the parameters the user can change. No prerequisites beyond this.

#### 4.4.1.3 Test Inputs

This test will use two test files to verify it can detect an unchanging state. The files are Test4-1.1.txt and Test4-1.2.txt. All test files will will use the default values loaded in with the website. Below are the tables Test4-1.1.txt and Test4-1.2.txt will produce respectively.

#### 4.4.1.4 Expected Test Results

The test files will stop iterating once they are the same state for two iterations.

#### 4.4.1.5 Criteria for Evaluating Results

For Test4-1.1.txt, the game will iterate once and stop. The ‘Iteration #:’ will read 1. For Test4-1.2.txt, the game will iterate until ‘Iteration #:’ is 4. The board will change for each iteration except for number 3 and 4. Both of those iterations will have the same board.

#### 4.4.1.6 Test Procedure

1. Reload the website to get default parameters.
2. Click ‘Choose File’ and upload Test4-1.1.txt.
3. Run game.
4. Record states and iteration numbers until the game stops.
5. Click the ‘New’ button to reset the board.
6. Repeat steps 2-4 with Test4-1.2.txt.

#### 4.4.1.7 Assumptions and Constraints

This test assumes that the game functions correctly, the file upload button works and all parameters deciding cell death, survival and revival are working correctly.

### 4.4.2 Test 4-2: Oscillating States

The purpose of this test is to ensure the game will detect and halt if a stable state consisting of an oscillating pattern has been reached.

#### 4.4.2.1 Requirements Addressed

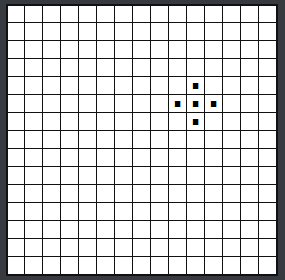
This test addresses requirement 3.2.4.2, which states the game shall stop if it is oscillating between two states.

#### 4.4.2.2 Prerequisite Conditions

This test requires that the game board and file upload have been implemented. In addition, that the game follows the rules set out by the parameters the user can change. No prerequisites beyond this.

#### 4.4.2.3 Test Inputs

This test will use a test file, Test4-2.txt, to ensure that it can detect an oscillating state. All other values will be the default when the website is loaded. When uploaded, the test file will produce the following result:



#### 4.4.2.4 Expected Test Results

The test files will stop iterating once an oscillating game pattern has been reached.

#### 4.4.2.5 Criteria for Evaluating Results

This test is considered passed if, when the test file is loaded, the game proceeds to 8 iterations then halts due to a stable state being reached.

#### 4.4.2.6 Test Procedure

1. Reload the website to get default parameters.
2. Click ‘Choose File’ and upload Test4-2.txt.
3. Run game.
4. Record states and iteration numbers until the game stops.

#### 4.4.2.7 Assumptions and Constraints

This test assumes that the game functions correctly, the file upload button works and all parameters deciding cell death, survival and revival are working correctly.

## 4.5 Test 5: Error Detection

### 4.5.1 Test 5-1: Grid Size Error

The purpose of this test is to ensure that the website will recognize and display an error when an invalid grid size has been entered.

#### 4.5.1.1 Requirements Addressed

This test addresses requirement 3.2.2.8, which states the website shall implement the ability to change the size of the grid in the game.

#### 4.5.1.2 Prerequisite Conditions

This test requires that the grid size adjusting feature has been implemented. There are no other prerequisites.

#### 4.5.1.3 Test Inputs

The following are grid size parameters in the form (X,Y). Enter each of the following separately

1. (101,15)
2. (15,41)
3. (101,41)
4. (a,15)
5. (15,a)
6. (a,a)
7. (15.1,15)
8. (15,15.1)
9. (15.1,15.1)
10. (-1,15)
11. (15,-1)
12. (-1,-1)

#### 4.5.1.4 Expected Test Results

Each of the entries should return the error, “GRID HEIGHT(Y) MUST BE INT IN RANGE [1,40]. GRID WIDTH(X) MUST BE INT IN RANGE [1,100] Default Set.” The grid should also default to 15 X 15.

#### 4.5.1.5 Criteria for Evaluating Results

To pass the tests the correct error must be displayed along with the correct default grid size.

#### 4.5.1.6 Test Procedure

1. Load Page
2. Run first test input
3. Refresh page
4. Run next test input
5. Refresh page
6. Repeat steps 4-5 for the remaining test inputs.

#### 4.5.1.7 Assumptions and Constraints

None

### 4.5.2 Test 5-2: Iteration Selection Error

The purpose of this test is to ensure that the website will detect and display an error when an invalid number of iterations has been entered.

#### 4.5.2.1 Requirements Addressed

This test addresses requirement 3.2.2.9, which states the game shall have a maximum iteration count of 216,000.

#### 4.5.2.2 Prerequisite Conditions

This test has no prerequisites.

#### 4.5.2.3 Test Inputs

In the “Enter Number of Iterations” input field enter 216001

#### 4.5.2.4 Expected Test Results

The game should not start when play is pressed. The number of Iterations in the input field should reset to 10. An error should display in the error box saying: “ERROR: NUMBER OF ITERATIONS MUST BE IN RANGE 1-216000”.

#### 4.5.2.5 Criteria for Evaluating Results

To pass the tests the correct error must be displayed. The game should not start when press is played and the number in the input field should be reset to 10.

#### 4.5.2.6 Test Procedure

1. Load Page
2. Enter 216001 in the “Enter Number of Iterations” field
3. Press Play

#### 4.5.2.7 Assumptions and Constraints

None

### 4.5.3 Test 5-3: File Errors

#### 4.5.3.1 Requirements Addressed

This test addresses requirement 3.2.2.11, which states the website shall accept a .txt text file containing grid coordinates as input for the game. This test will determine if the program correctly rejects files with improper formatting.

#### 4.5.3.2 Prerequisite Conditions

This test requires that the file upload feature, file parser, and the grid display be complete.

#### 4.5.3.3 Test Inputs

4.5.3.3.1 Upload file parse\_notCoords

4.5.3.3.2 Upload file parse\_tooManyCoords

4.5.3.3.3 Upload file parse\_tooFewCoords

4.5.3.3.4 Upload file parse\_outOfBounds

4.5.3.3.5 Upload file parse\_decimals

#### 4.5.3.4 Expected Test Results

4.5.3.4.1

Test 4.5.3.3.1 should return an error message in the error field that says “FILE ERROR: FILE CONTAINS NON-INTEGER VALUES”

4.5.3.4.2

Test 4.5.3.3.2 should return an error message in the error field that says “FILE ERROR: COORDINATE FORMAT IS INCORRECT”

4.5.3.4.3

Test 4.5.3.3.3 should return an error message in the error field that says “FILE ERROR: COORDINATE FORMAT IS INCORRECT”

4.5.2.4.4

Test 4.5.3.3.4 should return an error message in the error field that says “FILE ERROR: COORDINATES OUTSIDE OF SPECIFIED GRID RANGE”

4.5.2.4.5

Test 4.5.3.3.5 should return an error message in the error field that says “FILE ERROR: FILE CONTAINS NON-INTEGER VALUES”

#### 4.5.3.5 Criteria for Evaluating Results

To pass the tests the correct error must be displayed for each test input.

#### 4.5.3.6 Test Procedure

1. Load Page
2. Use input for test 4.5.3.4.1
3. Refresh page
4. Use input for test 4.5.3.4.2
5. Refresh page
6. Use input for test 4.5.3.4.3
7. Refresh page
8. Use input for test 4.5.3.4.4

#### 4.5.3.7 Assumptions and Constraints

None

## 4.6 Test 6: Hardware and Software Accessibility

### 4.6.1 Test 6-1: Mac Accessibility

The purpose of this test is to ensure that the website is accessible on the proper hardware.

#### 4.6.1.1 Requirements Addressed

This test addresses requirement 3.1.1, which states the website shall be accessible from a macbook running OS X High Sierra.

#### 4.6.1.2 Prerequisite Conditions

There are no prerequisite conditions for this test.

#### 4.6.1.3 Test Inputs

There are no inputs for this test.

#### 4.6.1.4 Expected Test Results

When the website address is entered, the website will display on the Mac’s screen.

#### 4.6.1.5 Criteria for Evaluating Results

This test is passed if the Mac can successfully reach the website.

#### 4.6.1.6 Test Procedure

1. Boot up the Mac.
2. Open Chrome.
3. Navigate to <https://ryamil1.github.io/CMSC447Project/>.

#### 4.6.1.7 Assumptions and Constraints

There are no assumptions for this test.

### 4.6.2 Test 6-2: Chrome Accessibility

The purpose of this test is to ensure that the website is accessible on the proper software

#### 4.6.2.1 Requirements Addressed

This test addresses requirement 3.1.2, which states the website shall be accessible and able to run from Chrome version 72.

#### 4.6.2.2 Prerequisite Conditions

There are no prerequisite conditions for this test.

#### 4.6.2.3 Test Inputs

There are no inputs for this test.

#### 4.6.2.4 Expected Test Results

When the website address is entered into Chrome, the website will display on the Mac’s screen.

#### 4.6.2.5 Criteria for Evaluating Results

This test is passed if the website displays correctly when the proper address is entered.

#### 4.6.2.6 Test Procedure

1. Open Chrome.
2. Navigate to <https://ryamil1.github.io/CMSC447Project/>.

#### 4.6.2.7 Assumptions and Constraints

There are no assumptions for this test.

## 4.7 Test 7: Website Display

### 4.7.1 Test 7-1: Display Number of Cells

The purpose of this test is to ensure that the website displays the correct number of living cells and that the counter updates accordingly..

#### 4.7.1.1 Requirements Addressed

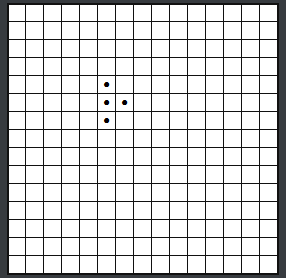
This test addresses requirement 3.2.3.2, which states that the website shall display a count of how many cells are alive during each iteration.

#### 4.7.1.2 Prerequisite Conditions

The game grid and file upload must be working for this test.

#### 4.7.1.3 Test Inputs

A test file, Test7.txt, shall be used for this test. When uploaded, it should produce the following game board:



#### 4.7.1.4 Expected Test Results

When an iteration passes in the game, the counter for the number of cells still living will update.

#### 4.7.1.5 Criteria for Evaluating Results

This test is passed if the number of cells on the board matches that of the counter on the website. As such, for this test file, after 10 iterations only 12 cells should be alive.

#### 4.7.1.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Set the game speed to Slow.
3. Upload the test file Test7.txt.
4. Click ‘Run’.
5. Observe and record results.

#### 4.7.1.7 Assumptions and Constraints

It is assumed that the other components of the website are functioning correctly before starting this test.

### 4.7.2 Test 7-2: Display Number of Iterations

The purpose of this test is to ensure that the website displays the correct number of iterations and that the counter updates accordingly..

#### 4.7.2.1 Requirements Addressed

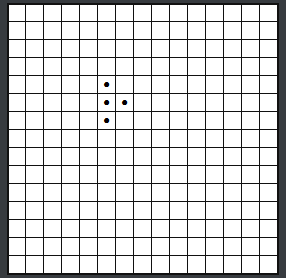
This test addresses requirement 3.2.3.1, which states that the website shall display how many iterations the game has been running.

#### 4.7.2.2 Prerequisite Conditions

The game grid and file upload must be working for this test.

#### 4.7.2.3 Test Inputs

A test file, Test7.txt, shall be used for this test. When uploaded, it should produce the following game board:



#### 4.7.2.4 Expected Test Results

When an iteration passes in the game, the iteration counter on the website will update accordingly.

#### 4.7.2.5 Criteria for Evaluating Results

This test is passed if, when an iteration passes, the iteration counter updates. Using the given test file, 10 iterations should pass before the game is halted due to reaching a stable state.

#### 4.7.2.6 Test Procedure

1. Click ‘New’ to reset the game.
2. Set the game speed to Slow.
3. Upload the test file Test7.txt.
4. Click ‘Run’.
5. Observe and record results.

#### 4.7.2.7 Assumptions and Constraints

It is assumed that the other components of the website are functioning correctly before starting this test.

# 5. Requirements Traceability

|  |  |
| --- | --- |
| CSCI Requirement | Test Case |
| 3.1.1 Mac Accessibility | 6-1 |
| 3.1.2 Chrome Accessibility | 6-2 |
| 3.2.1.1 The game shall implement the rule, by default, “If a living cell has 1 or 0 living neighbors, it will die from solitude.” | 1-2  Steps 1 through 7 address this requirement. |
| 3.2.1.2 The game shall implement the rule, by default, “If a living cell has 4 or more living neighbors, it will die from overpopulation.” | 1-2  Steps 1 through 7 address this requirement |
| 3.2.1.3 The game shall implement the rule, by default, “If a dead cell has 3 living neighbors, it will be revived.” | 1-3 |
| 3.2.1.4 The game shall implement the rule, by default, “If a living cell has 2 or 3 neighbors, it survives.” | 1-1 |
| 3.2.1.5 The game shall be implemented on a square grid. | 2-1 |
| 3.2.1.6 The game shall display in a 1080p resolution. | 2-2 |
| 3.2.1.7 The game shall display nodes at a fixed size, regardless of the size of the world. | 2-3 |
| 3.2.1.8 The game shall feature nodes which are visually distinguishable from one another. | 2-4 |
| 3.2.1.9 The game shall parse grid coordinates provided by a user as cells that are living at the start of the game. | 3-10  Steps 1 through 3 address this requirement |
| 3.2.2.1 The website shall implement the ability to adjust the number of neighbors for a cell to die from solitude. | 3-1  Steps 1 through 9 address this requirement. |
| 3.2.2.2 The website shall implement the ability to adjust the number of neighbors for a cell to die from overpopulation. | 3-2  Steps 1 through 9 address this requirement |
| 3.2.2.3 The website shall implement the ability to adjust the number of neighbors for a cell to be revived. | 3-3 |
| 3.2.2.4 The website shall implement the ability to adjust the number of neighbors for a cell to survive. | 3-1  Steps 1 through 9 address this requirement.  3-2  Steps 1 through 9 address this requirement |
| 3.2.2.5 The website shall implement the ability to adjust the background color of the game using 16 bit colors. | 3-4 |
| 3.2.2.6 The website shall implement the ability to adjust the color of cells using 16 bit colors. | 3-5 |
| 3.2.2.7 The website shall implement the ability to select the shape of the cells. The default shape shall be circles. Additional selectable shapes shall include triangles and squares. | 3-6 |
| 3.2.2.8 The website shall implement the ability to change the size of the grid in the game. | 3-7, 5-1 |
| 3.2.2.9 The website shall run the game to a maximum of 216,000 iterations. | 3-8, 5-2 |
| 3.2.2.10 The website shall implement the ability to change the default number of maximum iterations to a value between 1 and 216,000. | 3-9 |
| 3.2.2.11 The website shall accept a .txt text file containing grid coordinates as input for the game. | 3-10  Steps 1 through 3 address this requirement.  5-3 |
| 3.2.2.12 The website should be able to run the game at multiple speeds. | 3-11  Steps 1 through 7 address this requirement. |
| 3.2.2.13 The website should allow the user to adjust the speeds between selectable options. | 3-11  Steps 1 through 7 address this requirement. |
| 3.2.3.1 The website shall display how many iterations the game has been running. | 7-2 |
| 3.2.3.2 The website shall display a count of how many cells are alive during each iteration. | 7-1 |
| 3.2.4.1 The game shall stop if it is in the same state for two iterations. | 4-1 |
| 3.2.4.2 The game shall stop if it is oscillating between two states. | 4-2 |

# 