# Validation Test Cases

Test 1

Purpose: To test if the run button switches the mode from build mode to run mode.

Test Inputs: Using the mouse to press the “Run Mode” button on the GUI.

Expected Outputs: The game view switches from the build view to the running view such that building options and the build grid are removed and the “Play” button is added to the GUI.

Actual Outputs: As expected, build mode features are removed, and start, tick and stop buttons are displayed.

**Test 2**

**Purpose:** To test if the build button switches the mode from running mode to build mode.

**Test Inputs:** Using the mouse to press the “Build Mode” button on the GUI.

**Expected Outputs:** The game view switches from the running view to the build view such that the “Play” button is removed from the GUI and build buttons are added to the GUI. A grid should also appear of the game board.

Actual Outputs: As expected, run mode features are removed, and build mode features such as buttons to add a ball, triangle and square are displayed along with a grid on the board.

Test 3

Purpose: To test if the “Play” button starts the game timer and begins running the game.

Test Inputs: Using the mouse to press the “Play” button on the GUI.

Expected Outputs: The ball begins moving, collisions are visible between the ball and other gizmos. Flippers and other gizmos are able to be triggered.

Actual Outputs: As expected, the ball starts moving and if it touches any gizmos, proper collision detection is visible and if it hits the absorber, it is absorbed and then released from the top right corner of the absorber. The flippers and other gizmos are also able to be triggered.

Test 4

**Purpose:** To test if the “Stop” button stops the game timer.

**Test Inputs:** Using the mouse to press the “Stop” button on the GUI.

**Expected Outputs:** The GUI remains in run mode however the board freezes, the ball stops moving, key connected actions, and triggers do not function when activated.

Actual Outputs: As expected, everything comes to a standstill, GUI doesn’t switch modes but everything such as the ball freezes and until play or tick is clicked.

**Test 5**

**Purpose:** To test if the “Quit” button gives a user a prompt to quit and quits on confirmation.

**Test Inputs:** Using the mouse to press the “Quit” button and to using the mouse to press the “Yes” button in the confirmation box.

**Expected Outputs:** Upon confirmation the game window closes and ceases to run.

Actual Outputs: As expected, a message asking to confirm the action appears and if yes is clicked, we exit the application.

**Test 5.1**

**Purpose:** To test if the “Quit” button gives a user a prompt to quit and does not quit on deny.

**Test Inputs:** Using the mouse to press the “Quit” button and to using the mouse to press the “No” button in the confirmation box.

**Expected Outputs:** Upon denial the game window does not close and continues to run in the mode it was in prior to pressing the “Quit” button.

Actual Outputs: As expected, everything runs as normal if no is clicked in the message that appears when quit is clicked.

**Test 6**

**Purpose:** To test if the “Load” function gives the user a prompt to load a game file and loads the specified file.

**Test Inputs:** Using the mouse to select the “Load” button and to use the mouse to navigate to the specified file in the file browser, select the file, and press “load”.

**Expected Outputs:** The game file is successfully parsed and loaded onto the board. The gizmos and connections stored in the file should be visible on the game board in the files specified positions.

Actual Outputs: As expected, the files directories appear, and if you navigate and find a valid file and press load, everything is loaded onto the board in their correct positions.

**Test 6.1**

**Purpose:** To test if the “Load” function gives the user an appropriate message if the file fails to load but remains robust and returns to its previous state.

**Test Inputs:** Using the mouse to select the “Load” button and to use the mouse to navigate to the specified file in the file browser, select the file, and press “load”.

**Expected Outputs:** The game file is unable to load. The user receives an appropriate message informing them that the file could not load. Gizmoball does not crash and stays in the mode it was in prior to pressing the “Load” button.

Actual Outputs: As expected, an error message appears to say that file wasn’t loaded and everything goes back to the previous state before load was clicked.

**Test 7**

**Purpose:** To test if the “Save” function gives the user a prompt to save a game board under their specified name and saves the game board in the standard file format.

**Test Inputs:** Using the mouse to select “Save” button and to use the mouse to navigate to the specified directory they would like to save a file in. Then using the keyboard to specify the name that the file will be saved under.

**Expected Outputs:** The game file is saved to the desired directory with the desired name. Upon opening the file, the elements are stored in the standard file format (GizmoType GizmoName Xcoord YCoorD). Loading this file displays the same board prior to saving.

Actual Outputs: As expected, the board is saved with the standard format automatically added after a file name is provided.

**Test 7.1**

**Purpose:** To test if the “Save” function gives the user an appropriate message if the file fails to save but remains robust and returns to its previous state.

**Test Inputs:** Using the mouse to select “Save” button and to use the mouse to navigate to the specified directory they would like to save a file in. Then using the keyboard to specify the name that the file will be saved under.

**Expected Outputs:** The game file is unable to be saved. The user receives an appropriate message informing them that the file could not be saved. Gizmoball does not crash and stays in the mode it was in prior to pressing the “Save” button.

Actual Outputs: As expected, an error message appears if the file isn’t saved and the current state of the application is the one before save was clicked initially.

**Test 8 – changed ( makes use of a Jslider now)**

**Purpose:** To test if the “Gravity” function allows the user to enter a specified gravity setting and that change is reflected in the movement of the ball.

**Test Inputs:** Use the JSlider to set the gravity to the desired value.

**Expected Outputs:** The gravity is changed, this can be observed by the ball moving either more slowly or more quickly on the game board.

Actual Outputs: As expected, the gravity has been changed and looking at the speed of the ball we can see this as it will now be quicker or slower than before depending on the change.

**Test 9**

**Purpose:** To test the “Square” function, when the gizmo is selected and a clear point on the board is selected the gizmo is successfully added to the board.

**Test Inputs:** Click on the grid to add the gizmo to any clear location on the board.

**Expected Outputs:** Upon clicking an empty location on the board the square gizmo should be added to the board. The user should be able to repeatedly click, which will cause the same gizmo to be repeatedly added to the board.

Actual Outputs: As expected, when we click a location on the grid that has not been used yet, the gizmo is able to be added.

**Test 9.1**

**Purpose:** To test that the “Square” function prevents the user from adding a square on top of another gizmo and displays an appropriate error message.

**Test Inputs:** Click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the square gizmo is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Square” mode.

Actual Outputs: As expected, an error message appears if we for example try to add a square on a location where a circle already is placed.

**Test 10**

**Purpose:** To test if a the “Left Flipper” functions correctly.

**Test Inputs:**  Using the mouse select the “Left Flipper” from the gizmo drop-down menu and click add. Then click a clear location on the grid to add the flipper.

**Expected Outputs:** Upon clicking an empty location on the board the “Left Flipper” gizmo should be added to the board. The user should be able to repeatedly click, which will cause the same gizmo to be repeatedly added to the board.

Actual Outputs: As expected, a left flipper is added if where you click on the grid, its bounding box doesn’t collide with another gizmo or walls.

**Test 10.1**

**Purpose:** To test that the “Left Flipper” function prevents the user from adding a left flipper gizmo on top of another gizmo (or within its 4x4 grid space) and displays an error appropriate message.

**Test Inputs:** Using the mouse select the “Left Flipper” gizmo from the drop-down menu. Then, click the “Add” button. Now click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the “Left Flipper” gizmo is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Left Flipper” mode.

Actual Outputs: As expected, we get an error message if the flipper is attempted to be added onto a grid place where another gizmo is placed or if it is placed next to a gizmo or a wall such that it’s bounding box (it’s grid space) collides to another gizmo or wall.

**Test 10.2**

**Purpose:** To test that the “Left Flipper” function prevents the user from adding a left flipper where its flipper would extend outside the bounds of the play area and displays an appropriate error message.

**Test Inputs:** Using the mouse select the “Left Flipper” gizmo from the drop-down menu. Then, click the “Add” button. Now place the gizmo in a location 1 grid space from the right wall.

**Expected Outputs:** Upon clicking a grid space 1 space from the right wall on the board the “Left Flipper” gizmo is not added to the board and a message is displayed to the user saying that the flipper cannot be placed there. The user should however remain in the “Left Flipper” mode.

Actual Outputs: As expected, if the left flipper is added next to the wall such that extending it would cause it to go out of the boundary walls then we receive an error message explaining placing the flipper on such a location is not possible.

**Test 11**

**Purpose:** To test if a the “Right Flipper” functions correctly.

**Test Inputs:**  Using the mouse select the “Right Flipper” from the gizmo drop-down menu and click add. Then click a clear location on the grid to add the flipper.

**Expected Outputs:** Upon clicking an empty location on the board the “Right Flipper” gizmo should be added to the board. The user should be able to repeatedly click, which will cause the same gizmo to be repeatedly added to the board.

Actual Outputs: As expected, a right flipper is added if where you click on the grid, its bounding box doesn’t collide with another gizmo or walls.

**Test 11.1**

**Purpose:** To test that the “Right Flipper” function prevents the user from adding a “Right Flipper” gizmo on top of another gizmo (or within its 4x4 grid space) and displays an error appropriate message.

**Test Inputs:** Using the mouse select the “Right Flipper” gizmo from the drop-down menu. Then, click the “Add” button. Now click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the “Right Flipper” gizmo is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Right Flipper” mode.

Actual Outputs: As expected, we get an error message if the right flipper is attempted to be added onto a grid place where another gizmo is placed or if it is placed next to a gizmo or a wall such that it’s bounding box (it’s grid space) collides to another gizmo or wall.

**Test 11.2**

**Purpose:** To test that the “Right Flipper” function prevents the user from adding a “Right Flipper” where its flipper would extend outside the bounds of the play area and displays an appropriate error message.

**Test Inputs:** Using the mouse select the “Right Flipper” gizmo from the drop-down menu. Then, click the “Add” button. Now place the gizmo in a location 1 grid space from the left wall.

**Expected Outputs:** Upon clicking a grid space 1 space from the right wall on the board the right flipper gizmo is not added to the board and a message is displayed to the user saying that the flipper cannot be placed there. The user should however remain in the “Right Flipper” mode.

Actual Outputs: As expected, if the right flipper is added right next to the wall such that extending it would cause it to go out of the boundary walls then we receive an error message explaining that placing the flipper on such a location is not possible.

**Test 12**

**Purpose:** To test that the “Friction” function allows the user to enter a specified friction setting and that change is reflected in the movement of the ball.

**Test Inputs:** Use the JSlider to set the friction to the desired value

**Expected Outputs:** The friction is changed, this can be observed by the ball moving either more slowly or more quickly on the game board.

Actual Outputs: As expected, the friction is applied as per the selection by the user and this can be seen as the ball moves quicker or slower depending on the value of the friction selected.

**Test 13 – changed from connect gimzo**

**Purpose:** To test that the “Add Trigger” function allows the user to connect 2 gizmos together.

**Test Inputs:** The user selects the “Add trigger” button using the mouse. Then using the mouse, the user selects the first gizmo and then the second gizmo. A message appears informing the user that the two have been connected.

**Expected Outputs:** During run mode the first gizmo should trigger the second gizmos action upon a collision with the ball.

Actual Outputs: As expected, once you click add trigger, you click on two gizmos, and upon clicking the second one, you will get a message that tells you that the trigger has been applied.

**Test 13.1**

**Purpose:** To test that the “Add Trigger” function allows the user to connect another gizmo to the a previously connected pair of gizmos without pressing the connect gizmo button again.

**Test Inputs:** The user chooses a third gizmo to connect to the first gizmo. Then the user is notified that the gizmo is connected.

**Expected Outputs:** During run mode the third gizmo should trigger the first gizmos action upon a collision with the ball.

Actual Outputs: The expected output isn’t what happens as a trigger is added but you must click the add trigger button once again.

**Test 14 – changed from disconnect gizmos**

**Purpose:** To test that the “Remove Trigger” function allows the user to disconnect two connected gizmos.

**Test Inputs:** Using the mouse the user selects the “Remove Trigger” button. Then using the mouse, the user selects the first gizmo and then the second gizmo. A message appears informing the user that the two have been disconnected.

**Expected Outputs:** During run mode the first gizmo will not trigger the second previously connected gizmos action upon a collision with the ball.

Actual Outputs: As expected, once remove trigger is clicked and two gizmos that were connected are clicked, then upon the second click, we get a message telling us that the trigger has been removed

**Test 14.1**

**Purpose:** To test that the “Remove Trigger” function does not allow the user to disconnect two gizmos that are not connected.

**Test Inputs:** Using the mouse the user selects the “Remove Trigger” button. Then using the mouse, the user selects the first gizmo and then the second gizmo. Then the user is notified that the second gizmo is not connected to the first one.

**Expected Outputs:** When the user tries to disconnect the first gizmo from the second one an error message appears notifying the user that the second gizmo is not connected to the first one.

Actual Outputs: As expected, if two unconnected gizmos are attempted to be disconnected then an error message appears telling the user that the gizmos are not connected.

**Test 15**

**Purpose:** To test that the “Move” function allows the user to move a gizmo to another clear location on the board.

**Test Inputs:** Using the mouse the user selects the “Move” button. Then using the mouse again, the user selects a gizmo from the board they would like to move. Then selects a clear position to move the gizmo.

**Expected Outputs:** When the user selects the clear position to move the specified gizmo, the gizmo is moved upon click and the user remains in “Move” mode.

Actual Outputs: As expected, once move is clicked and the user clicks on a gizmo to move then upon clicking a desired location, that gizmo is moved to that location.

**Test 15.1**

**Purpose:** To test that the “Move” function does not allow the user to move a gizmo to an occupied location on the board.

**Test Inputs:** Using the mouse the user selects the “Move” button. Then using the mouse again, the user selects a gizmo from the board they would like to move. Then selects an occupied position to move the gizmo.

**Expected Outputs:** When the user tries to move the gizmo an error message will appear notifying the user that the position is occupied. The user remains in “Move” mode.

Actual Outputs: As expected, if the user tries to move a gizmo to a location already occupied then an error message telling the user this appears, and the gizmo remains at the place it was before moving was initiated on it. The user can then click on the gizmo again to move it elsewhere without clicking the move button again.

**Test 16**

**Purpose:** To test that the “Remove” function allows the user to delete a gizmo.

**Test Inputs:** Using the mouse the user selects the “Remove” button. Then using the mouse again, the user selects a gizmo from the board they would like to remove.

**Expected Outputs:** Upon clicking on the gizmo the user would like to remove the gizmo is removed from the gameboard.

Actual Outputs: As expected, if a user clicks on a gizmo after clicking the remove button then that gizmo is deleted from the board.

**Test 17**

**Purpose:** To test that the “Rotate” function allows the user to rotate a gizmo.

**Test Inputs:** Using the mouse the user selects the “Rotate” button. Then using the mouse again, the user selects a gizmo from the board they would like to rotate.

**Expected Outputs:** Upon clicking on the gizmo the user would like to rotate the gizmo is rotated 90 degrees clockwise on the gameboard at its current position. The user remains in “Rotate” mode.

Actual Outputs: As expected, if the user clicks on a gizmo such as a triangle after clicking the rotate button then that gizmo is rotated 90 degrees clockwise in its current position i.e within the square it is in.

**Test 18 – changed from connect key**

**Purpose:** To test that the “Add Keybind” function allows the user to connect a key to a gizmo.

**Test Inputs:** Using the mouse the user selects the “Add Keybind” button. Then a message asking the user to insert key to bind appears, so after a key is inserted, you next click the gizmo to bind it to.

**Expected Outputs:** The user is notified that the key is now connected to the gizmo. When pressing the key in running mode the gizmo’s action will be triggered.

Actual Outputs: As expected, once a key is connected to a gizmo, a message appears to tell the user of this. Then in run mode, if a space bar was bind with a left flipper then upon clicking the space bar, that flipper is triggered.

**Test 18.1**

**Purpose:** To test that the “Add Keybind” function allows the user to overwrite a key connection to a gizmo.

**Test Inputs:** Using the mouse the user selects the “Add Keybind” button. Then the user selects the key they want to bind to a gizmo. The user then selects the gizmo they want to bind the key to by clicking the specified gizmo.

**Expected Outputs:**  The key selected is now also connected to the gizmo.

Actual Outputs: As expected, since the gizmo was already connected to a certain gizmo, adding a new key bind simply adds it as a second one. Meaning both keys can be used to trigger the gizmo and the first one isn’t overridden.

**Test 19 – No warning message**

**Purpose:** To test that the “Remove KeyBind” function allows the user to disconnect a key from a gizmo.

**Test Inputs:** The user selects the ““Remove KeyBind” button. Then selects the key they would like to remove the connection from, then click the gizmo.

**Expected Outputs:** A message appears asking the user if they want to remove the connection. If the user selects yes, the connection is removed if the user selects no the connection remains.

Actual Outputs: As expected, the connection between the key and gizmo is removed after remove keybind is selected and the user selected the key and its connected gizmo that it now wants to disconnect.

**Test 19.1**

**Purpose:** To test that the “Remove KeyBind” function does not allow the user to disconnect a key from a gizmo if the gizmo does not have a key connection.

**Test Inputs:** The user selects the “Remove KeyBind” button. Then types in the key before selecting the gizmo they would like to remove the connection from

**Expected Outputs:** An error message appears notifying the user that the chosen gizmo does not have a key connection.

Actual Outputs: As expected, an error message appears to tell the user that no keybind was detected.

**Test 20**

**Purpose:** To test that the “Absorber” function allows the user to add an absorber to the field.

**Test Inputs:** The user selects the “Add Absorber” button. Then the user selects the start position of the absorber clicking a position on the board, and then selects the end of the absorber by dragging the mouse over a clear area on the board and releasing the mouse at the end position, creating the absorber.

**Expected Outputs:** The absorber is added to the playing board for the user to see and objects on the board to interact with.

Actual Outputs: As expected, the absorber is added onto the board between the start click and the end click after the mouse drag.

**Test 20.1**

**Purpose:** To test that the “Absorber” function does not allow the user to add an absorber to the board in a position that is already occupied.

**Test Inputs:** The user selects the “Absorber” button. Then the user selects the start position of the absorber clicking a position on the board, and then selects the end of the absorber by dragging the mouse over an occupied area of the board and releasing the mouse at the end position, creating the absorber.

**Expected Outputs:** An error message notifies the user that the area within the absorber was already occupied. The absorber is not added to the board, but the user remains in “Absorber” mode.

Actual Outputs: As expected, the user is not able to add the absorber in an occupied spot on the grid and an error message tells the user of this.

**Test 21**

**Purpose:** To test the “Triangle” function, when the gizmo is selected and a clear point on the board is selected the gizmo is successfully added to the board.

**Test Inputs:** Click on the grid to add the gizmo to any clear location on the board.

**Expected Outputs:** Upon clicking an empty location on the board the triangle gizmo should be added to the board. The user should be able to repeatedly click, which will cause the same gizmo to be repeatedly added to the board.

Actual Outputs: As expected, when we click a location on the grid that has not been used yet, the gizmo is able to be added.

**Test 21.1**

**Purpose:** To test that the “Triangle” function prevents the user from adding a triangle on top of another gizmo and displays an appropriate error message.

**Test Inputs:** Click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the triangle gizmo is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Triangle” mode.

Actual Outputs: As expected, an error message appears if we for example try to add a triangle on a location where a circle already is placed.

**Test 22**

**Purpose:** To test the “Circle” function, when the gizmo is selected and a clear point on the board is selected the gizmo is successfully added to the board.

**Test Inputs:** Click on the grid to add the gizmo to any clear location on the board.

**Expected Outputs:** Upon clicking an empty location on the board the circle gizmo should be added to the board. The user should be able to repeatedly click, which will cause the same gizmo to be repeatedly added to the board.

Actual Outputs: As expected, when we click a location on the grid that has not been used yet, the gizmo is able to be added.

**Test 22.1**

**Purpose:** To test that the “Circle” function prevents the user from adding a circle on top of another gizmo and displays an appropriate error message.

**Test Inputs:** Click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the circle gizmo is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Circle” mode.

Actual Outputs: As expected, an error message appears if we for example try to add a circle on a location where a square already is placed.

**Test 23**

**Purpose:** To test the “Ball” function, when the ball is selected and a clear point on the board is selected the ball is successfully added to the board.

**Test Inputs:** Click on the grid to add a Ball to any clear location on the board.

**Expected Outputs:** Upon clicking an empty location on the board the ball should be added to the board. The user should be able to repeatedly click, which will cause a new ball to be repeatedly added to the board each time

Actual Outputs: As expected, when we click a location on the grid that has not been used yet, the ball is able to be added.

**Test 23.1 – able to add ball on top of each other**

**Purpose:** To test that the “Ball” function prevents the user from adding a ball on top of a gizmo and displays an appropriate error message.

**Test Inputs:** Click on a location already filled by another gizmo.

**Expected Outputs:** Upon clicking a non-empty location on the board the ball is not added to the board and an error message is displayed to the user saying that the space is occupied. The user should however remain in the “Ball” mode.

Actual Outputs: As expected, an error message appears if we for example try to add a ball on a location where a circle already is placed.