Validation Testing

Add Gizmo

Test 1

Purpose of test: ensuring that a gizmo is added correctly onto an empty slot of the playing board.

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Add” tab, select gizmo of choice
3. Select an empty cell in the board for the gizmo to be added

Expected Output(s):

- Contextual message should display “gizmo has been successfully added”.
- The empty cell that the user clicked on should now be occupied by the selected gizmo.
- The newly added gizmo should also appear in the position in run mode (File -> Run Mode)

Test 2

Purpose of test: ensuring that an error message is given when an attempt to add a gizmo to an occupied slot of the board is made.

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Add” tab, select gizmo of choice
3. Select a cell that is already occupied by an existing gizmo

Expected Output(s):

- Contextual message should display “Sorry! Gizmo cannot be placed in an occupied cell)
- No change is made to the selected cell

Move Gizmo

Test 3

Purpose of test: ensuring that a gizmo is moved correctly onto an empty slot of the playing board.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select “Move”
3 Select an existing gizmo on the board to be moved
4. Select an empty cell in the board for the selected gizmo to be moved to

Expected Output(s):

- Contextual message should display “gizmo has been successfully moved”.
- The empty cell that the user clicked on should now be occupied by the selected gizmo with the old cell the gizmo was held in now being empty.
- The moved gizmo should also appear in the new position in run mode (File -> Run Mode)

Test 4

Purpose of test: ensuring that an error message is given when an attempt to move a gizmo to an occupied slot of the board is made.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select “Move”
3 Select an existing gizmo on the board to be moved
4 Select an occupied cell in the board for the selected gizmo to be moved to

Expected Output(s):

- Contextual message should display “Sorry! Gizmo cannot be moved to an occupied cell)
- No change is made to the selected gizmo cell and the cell the user wants to move the gizmo to

Delete Gizmo

Test 5

Purpose of test: ensuring that a gizmo is deleted correctly.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select “Delete”
3 Select an existing gizmo on the board to be deleted

Expected Output(s):

- Contextual message should display “gizmo has been successfully deleted”.
- The cell selected which contained a gizmo should now be empty
- The deleted gizmo should not appear in run mode (File -> Run Mode)

Test 6

Purpose of test: ensuring that only gizmos can be deleted.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select “Delete”
3 Select any cell which does not contain a gizmo

Expected Output(s):

- Contextual message should display “nothing to delete”.
- No changes should be made to the selected cell

Rotate Gizmo Clockwise

Test 7

Purpose of test: ensuring that the rotate clockwise button successfully turns the selected gizmo 90 degrees clockwise.

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Edit” tab, select the clockwise rotation button
3. Select an existing gizmo on the board to be rotated

Expected Output(s):

- Contextual message should display “gizmo successfully rotated clockwise”.
- Gizmo is rotated 90 degrees clockwise
- Rotated gizmo should still be contained within its own cell
- The rotated gizmo is also displayed correctly in run mode (File -> Run Mode)

Rotate Gizmo Anti-Clockwise

Test 8

Purpose of test: ensuring that the rotate anti-clockwise button successfully turns the selected gizmo 90 degrees anti-clockwise.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select the anti-clockwise rotation button
3 Select an existing gizmo on the board to be rotated

Expected Output(s):

- Contextual message should display “gizmo successfully rotated anti-clockwise”.
- Gizmo is rotated 90 degrees anti-clockwise
- Rotated gizmo should still be contained within its own cell
- The rotated gizmo is also displayed correctly in run mode (File -> Run Mode)

Add Connection

Test 9

Purpose of test: ensuring that a gizmo is successfully connected to another gizmo (primarily the bumpers)

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select the “Add Connection” Button
3 Select an existing gizmo on the board
4 Select a second existing gizmo on the board to link its action to the previously selected gizmos trigger.
5 Go to Run mode (File -> Run mode)
6 Click the play button

Expected Output(s):

- Contextual message should display “Connection successfully added”.
- When the ball collides with the first selected gizmo, the action for the second gizmo should be triggered e.g. if a square bumper is connected to a flipper, every time the ball collides with the square bumper, the flipper will activate i.e. rotate then return to its original position.

Remove Connection

Test 10

Purpose of test: ensuring that a gizmo is successfully disconnected to another gizmo (primarily the bumpers)

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select the “Remove Connection” Button
3 Select an existing gizmo on the board which is connected to another existing gizmo
4 Go to Run mode (File -> Run mode)
5 Click the play button

Expected Output(s):

- Contextual message should display “Connection removed successfully”.
- When the ball collides with the selected gizmo, the action for the second gizmo should be not be triggered

Test 11

Purpose of test: ensuring that a when a connection that does not exist is attempted to be disconnected nothing happens

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Edit” tab, select the “Remove Connection” Button
3. Select an existing gizmo that has no connections

Expected Output(s):

- Contextual message should display “No connections to disconnect”.

Add Key Connection

Test 12

Purpose of test: ensuring that a key connected to a gizmo when pressed, will trigger the gizmo’s action

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Edit” tab, select the “Add Key Connection” Button
3 Select an existing gizmo on the board
4 Select a key for the gizmo’s trigger to be linked to
5 Go to Run mode (File -> Run mode)
6 Click the play button
7 Press the assigned key

Expected Output(s):

- Contextual message should display “Key connection successfully added”.
- Every time the selected key is pressed, the action associated with the connected gizmo will trigger.

Remove Key Connection

Test 13

Purpose of test: ensuring that a key is successfully disconnected from a gizmo and when pressed, will no longer trigger the gizmo’s action

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Edit” tab, select the “Remove Key Connection” Button
3. Select an existing gizmo on the board that is already linked to a key press
4. Go to Run mode (File -> Run mode)
5. Click the play button
6. Press the key that was previously assigned

Expected Output(s):

- Contextual message should display “Key successfully disconnected”.
- The key should no longer trigger the gizmo’s action

Add Ball

Test 14

Purpose of test: ensuring that a ball is successfully added to the board

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Add” tab, select the ball gizmo to be added
3. Enter the initial velocity for the ball
4. Enter the initial direction for the ball
5. Select an empty space on the board

Expected Output(s):

- Contextual message should display “Ball successfully added”.
- In run mode (File -> Run Mode -> Play), The ball should move in the specified direction.
- The ball should move with the specified initial velocity which can be tracked in the side panel

Test 15

Purpose of test: ensuring that a ball does not get added to the board if its initial position overlaps an existing gizmo or is outside of the playing area.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 In the “Add” tab, select the ball gizmo to be added
3 Enter the initial velocity for the ball
4 Enter the initial direction for the ball
5 Select an area outside of the playing field

Expected Output(s):

- Contextual message should display “Ball cannot be placed here”.
- No ball should be added to the board

Test 16

Purpose of test: ensuring that a ball is correctly handled when placed inside an absorber

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. In the “Add” tab, select the ball gizmo to be added
3. Enter the initial velocity for the ball
4. Enter the initial direction for the ball
5. Select a cell occupied by an absorber

Expected Output(s):

- Contextual message should display “Ball successfully added”.
- The ball should be trapped in the top right hand corner of the absorber until the key press connected to the absorber is triggered where the absorber will fire the ball straight upwards

Save Configuration

Test 17

Purpose of test: ensuring that the save function correctly saves the current game settings to a file

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Select File -> Save Configuration
3 Enter a filename
4 Click save

Expected Output(s):

- Contextual message should display “Save successful”.
- File should be saved in the selected folder with the specified filename

Test 18

Purpose of test: ensuring that the save function correctly saves on top of a new file if the user chooses to do so

Test Input Steps:

1. Build mode is selected (File -> Build mode)
2. Select File -> Save Configuration
3. Navigate to and select an existing Gizmoball save file
4. Click save and confirm

Expected Output(s):

- Contextual message should display “Save successful”.
- The existing save file should be overwritten

Load Configuration

Test 19

Purpose of test: ensuring that the game settings within a saved file is correctly loaded in

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Select File -> Load Configuration
3 Navigate to the folder with the saved file
4 Click on the saved file
5 Click “ok”

Expected Output(s):

- Contextual message should display “Load successful”.

- The loaded file should match the saved file i.e. The board layout, key connections, initial ball speed, initial ball direction, friction and gravity should match.

Test 20

Purpose of test: ensuring that files unrelated to Gizmoball cannot and should not be loaded in.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Select File -> Load Configuration
3 Navigate to any file unrelated to Gizmoball
4 Click “ok”

Expected Output(s):

- Contextual message should display “File cannot be loaded”.
- No files should be loaded and the game should be in the same state

Quit

Test 21

Purpose of test: ensuring that the application will prompt the user to save the current game state before quitting Gizmoball.

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Select File -> Quit
3 Click yes when the application prompts to save a file
4 Save the current game settings

Expected Output(s):

- Gizmoball should close after the file has been successfully saved

Test 22

Purpose of test: ensuring that the application will still close even if the user declines to save the current game settings

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Select File -> Quit
3 Click no when the application prompts to save a file

Expected Output(s):

- Gizmoball should close after the file has been successfully saved

Change Gravity

Test 23

Purpose of test: ensuring that the change in gravity will affect the ball in run mode

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Go to settings and adjust gravity to your liking
3 Click apply settings
4 Switch to run mode (File -> Run mode)
5 Click Play

Expected Output(s):

- Reduction in gravity will result in the ball bouncing much higher whilst increasing gravity will have the opposite effect

Change Friction

Test 24

Purpose of test: ensuring that the change in friction will affect the ball in run mode

Test Input Steps:

1 Build mode is selected (File -> Build mode)
2 Go to settings and adjust friction 1 and/or fraction 2 to your liking
3 Click apply settings
4 Switch to run mode (File -> Run mode)
5 Click Play

Expected Output(s):

- Higher friction settings will result in the ball having greater velocity after colliding with a gizmo whilst a reduction in friction will have the opposite effect

Tick

Test 25

Purpose of test: ensuring that the tick button moves the gameplay by one frame

Test Input Steps:

1 Run mode is selected (File -> Run Mode)
2 Click Play to view the normal play speed of Gizmoball
3 Click the stop button
4 Click the tick button

Expected Output(s):

- With each click of the Tick button, the state of the Gizmoball should progress by one frame

Reset

Test 26

Purpose of test: ensuring that the reset button causes the Gizmoball to jump back to its initial state

Test Input Steps:

1 Run mode is selected (File -> Run Mode)
2 The current(initial) state should be the state that is returned to when the reset button is hit
3 Click the play button
4 Click the stop button
5 Click the reset button

Expected Output(s):

- After click the reset button, the state should be returned to as it was in Step 2.

Add Absorber

Test 27

Purpose of test: ensuring that an absorber is added correctly and functions properly

Test Input Steps:

1 Run mode is selected (File -> Run Mode)
2 In the “Add” tab select Add Absorber button
3 Click on empty cell
4 Drag across multiple cells to adjust preferred size
5 Add a key connection to the absorber

Expected Output(s):

- Contextual message should display “Absorber successfully added”
- When the ball comes in contact with an absorber, the ball should be “trapped” in the top right corner of the absorber
- Once the key press associated with the absorber is triggered, the absorber with fire the ball directly upwards

Test 28

Purpose of test: ensuring that an absorber does not overlap any existing gizmos

Test Input Steps:

1. Run mode is selected (File -> Run Mode)
2. In the “Add” tab select Add Absorber button
3. Click on a cell that already contains an existing gizmo
4. OR Click on empty cell and drag across multiple cells where at least one of these cells is already pre occupied

Expected Output(s):

- Contextual message should display “Cell(s) selected is pre occupied, absorber could not be added”