**Validation Testing Strategy**

Test Number: 1

Purpose of Test: Start game by clicking the “Start” button from the main GUI.

Test Inputs: run gizmoball application by running the game from a desktop. When the GUI has loaded, click the “Start” button to begin gameplay. Board should then change from build game mode into play mode. Check if the game board is now displaying in the interface. Scan through GUI and ensure that gizmos and ball are displayed in the correct position (specified through pre-set board OR build mode map).

Expected Inputs: User clicks the start button from main interface to begin game.

Expected Outputs: Check Start button – Start button + Gizmoball game view

* 1. Normal Flow – Game begins in run mode letting the user start the Gizmoball game
  2. Alternative Path – Button does nothing / Game does not start due to a fault within the system
  3. Alternative Path – Interface displays gizmos, however not as the user has buit the game, i.e. some of the gizmos are in a different place from when it was originally placed, proving a glitch in the system

Test Number: 2

Purpose of Test: Stop gameplay by clicking the “Stop” button displayed in the game interface. Board should be in run mode AND running, for the STOP command to continue so check that the board has met these conditions. After these conditions have been met, the gameplay should halt (check that the board is in run mode BUT NOT running). Gameplay should have stopped and key presses should not affect the state of the game.

Expected Inputs: User clicks the stop button from main interface to stop game.

Expected Outputs: Check Stop button – Stop button + Board

* 1. Normal Flow – Games stops running in gameplay mode
  2. Alternative Path – Button does nothing / Game fails to stop and continues in gameplay, signalling a possible error in the system. User may have to force game shutdown if game fails to stop.

**RUN MODE**

Test Number: 3

Purpose of Test: Load Model (from save // game in progress)  
Check if player’s click on button “Load Model” triggers code tied to it. Board should be in run mode in order to continue with loading a previous save. Check if file selected is in the correct format (correct file extension). Saved game state should appear in the GUI. User should be able to continue from saved state (so maybe change Start > Continue after a map has been loaded).

Expected Input: User clicks the “Load Game” button from the GUI, looking to load a previously built game

Expected Output: Previously saved file (game data) is loaded into game, and the GUI displays the gizmoball build as expected.

* 1. Normal Flow – Model is loaded from a saved file and everything is in correct position, i.e. gizmos display in game as previously saved without corruption
  2. Alternative Path – Button does not work, preventing the saved game from loading caused by an error in the system
  3. Alternative Path – user clicks load game, however the file has been corrupted or deleted, preventing the game from being loaded
  4. Alternative Path – game is loaded, however the gizmos are not displayed in the correct place as previously saved

Use Case: Reload Model

Test Number: 4  
  
Purpose of Test: Check if user’s click on button “Reload Model” works as expected. Current map should then be “refreshed” (map should reload from pre-built map or user made map). Check if the correct map has been loaded (check gizmo locations and compare to saved gizmo locations).

Expected Input: click load game from start state

Expected Output: Gizmoball is loaded in run mode gameplay state

1. Check Reload – Reload Button + Board
   1. Normal Flow – Board is reloaded into start state
   2. Alternative Path - Button doesn’t work caused by an error in the system

/\* NEED TO FINISH CORRECTING DOCUMENT FORMAT FROM HERE DOWN, SO ITS THE SAME FORMT AS ABOVE \*/

Use Case: Save Model  
  
Check if player’s click on button “Save Model” triggers code tied to it. User should be in build mode and at least 4 gizmo’s (some progress) should have been placed in build mode (just some basic counter to keep track?) in order for this code to continue. After file is saved check if file exits and if it does continue, if not ERROR.

1. Check Save – Save button + Board
   1. Normal Flow – Board is saved correctly in user directory
   2. Alternative Path(s) – Button does nothing / File does not save correctly / Save window doesn’t pop up

Use Case: Build Game-play  
  
Check if player’s click on button “Build Mode” triggers code tied to it. Board should then change from Run mode into Build mode.

1. Check Build Mode – Build Mode Button
   1. Normal Flow – Game enters Build Mode
   2. Alternative Path(s) - Button does nothing / Game remains in run mode

Use Case: Quit  
  
Check if player’s click on button “Quit” triggers code tied to it. Check game has stopped and program closes (print something in console).

1. Check Quit – Quit Button
   1. Normal Flow – Game quits / Halts execution
   2. Alternative Path(s) – Button does nothing / Game does not quit

Use Case: Add Gizmo  
  
Check if player’s click on button “Add Gizmo” triggers code tied to it. (Button only available in build mode so no need to check if in build mode?). Check if a gizmo is added to the board in the correct position (where the user clicked).

1. Check Add Gizmo – Add Gizmo Button + Board
   1. Normal Flow – Gizmo is added to board where specified
   2. Alternative Path(s) - Button does nothing / Gizmo is not added to where the user specified

**Use Case: Delete Object**Check if player’s click on button “Delete Object” triggers code tied to it.  
Used to enter a sort of “delete mode”. User clicks button then clicks on an object they want to delete. Exits the “delete mode” after one deletion to prevent accidental deletes. Check if there is something on the board to delete (check if not empty). Targeted object should be deleted, check that there is no object in the position of where the user clicked.

1. Check Delete Object – Delete Object Button + Board
   1. Normal Flow – Specified Gizmo is deleted from board
   2. Alternative Path(s) - Button does nothing / Gizmo is not deleted

Use Case: Add Absorber  
  
Check if player’s click on button “Add Absorber” triggers code tied to it. Absorber should be added where the user specified (check where user clicked and that the object added is an absorber). Also check if there is already an object in the space clicked, if there is, do not add an absorber.

1. Check Add Absorber – Add Absorber Button + Board
   1. Normal Flow – Absorber is added where user specified
   2. Alternative Path(s) - Button does nothing / Absorber not added / Not added to correct position

Use Case: Add Flipper  
  
Check if player’s click on button “Add Flipper” triggers code tied to it. Flipper should be added where the user specified (check where user clicked and that the object added is a flipper). Also check if there is already an object in the space clicked, if there is, do not add a flipper.

1. Check Add Flipper – Add Flipper Button + Button
   1. Normal Flow – Flipper is added where user specified
   2. Alternative Path(s) - Button does nothing / Flipper not added / Not added to correct position

Use Case: Key Connect  
  
Check if player’s click on button “Bind Key” triggers code tied to it. User then clicks on a flipper or absorber and a window will pop up. Check if the object clicked is valid. User then enters a VALID key to associate with the object. Then make sure that the key is bound to the correct flipper or absorber. Key bind that has just been made by the user should then be added to the current game.

1. Check Key Connect – Key Connect Button + Board
   1. Normal Flow – Gizmo is connected to a key press
   2. Alternative Path(s) – Button does nothing / Gizmo is not connect to a key press

Use Case: Key Disconnect  
  
Check if player’s click on button “Unbind Key” triggers code tied to it. User clicks on an object and the program should unbind any key associated with it. Check that the object now has 0 keys bound to it.

1. Check Key Disconnect – Key Disconnect Button + Board
   1. Normal Flow – Gizmo no longer has a key connected to it
   2. Alternative Path(s) - Button does nothing / Gizmo still has a key connection

Use Case: Add Ball  
  
Check if player’s click on button “Add Ball” triggers code tied to it. Ball should be added where the user specified (check where user clicked and that the object added is a ball). Also check if there is already an object in the space clicked, if there is, do not add an absorber. **Ensure that there are no other balls on the board, if there are, delete the balls and the check again to make sure board has no balls. (Do this on button click?)**

1. Check Add Ball – Add Ball Button + Board
   1. Normal Flow – Ball is added to the board
   2. Alternative Path(s) - Button does nothing / Ball is not added

Use Case: Rotate  
  
Check if player’s click on button “Rotate” triggers code tied to it. User will then click on an object. Check if object is there, if yes, continue. Code should then rotate the object 90 degrees. Check that the object has been rotated (basic variable that is updated after rotate?).

1. Check Rotate – Rotate Button + Board
   1. Normal Flow – Selected gizmo is rotated
   2. Alternative Path(s) - Button does nothing / Gizmo is not rotated (check co-ords)

Use Case: Clear Board  
  
Check if player’s click on button “Clear Board” triggers code tied to it. Runs the code. Check that there is nothing on the board after clear has been run.

1. Check Clear Board – Clear Board Button + Board
   1. Normal Flow – Clears the board of gizmo and ball
   2. Alternative Path(s) - Button does nothing / Board still has gizmos on it

Use Case: Move  
  
Check if player’s click on button “Move” triggers code tied to it. Check that board is not empty. User clicks on an object, then on the preferred location of where they would like to move it. Check that the user has clicked on a valid object (not empty). After object has been moved check that it is in the correct position by comparing mouse click location to current object position.

1. Check Move – Move Button + Board
   1. Normal Flow – Gizmo is moved to position specified
   2. Alternative Path(s) - Button does nothing / Gizmo doesn’t move / Doesn’t remove old gizmo position

Use Case: Connect  
  
Check if player’s click on button “Connect gizmo” triggers code tied to it. Check that board is not empty. User clicks on a gizmo and then another, these are now connected. Check that the 2 gizmos are connected / linked together. Exit Connect mode after 1 link is created to minimize accidental links.

1. Check Connect – Connect Button + Board
   1. Normal Flow – 2 Gizmos are now connected/linked
   2. Alternative Path(s) - Button does nothing / Gizmos are not connected/linked

Use Case: Disconnect  
  
Check if player’s click on button “Disconnect gizmo” triggers code tied to it. Check that board is not empty and that a current link / connection between 2 gizmos exists. User clicks on a linked gizmo and then the associated gizmo, these are no longer connected. Check that the 2 gizmos are no longer connected / linked together. Exit disconnect mode after 1 link is been deleted to minimize accidental disconnects.

1. Check Disconnect – Disconnect Button + Board
   1. Normal Flow – Gizmo link is destroyed
   2. Alternative Path(s) - Button does nothing / Gizmos are still connected/linked

Use Case: Change Friction   
  
Check if slider “Change Friction” triggers code tied to it. User chooses value on a slider. Check that the number displayed on the GUI is the same as the value for friction.

1. Check Change Friction – Change Friction Slider + Board
   1. Normal Flow – Friction variable is changed
   2. Alternative Path(s) - Slider does nothing / Friction is changed but not to correct value

Use Case: Change Gravity   
  
Check if slider “Change Gravity” triggers code tied to it. User chooses value on a slider. Check that the number displayed on the GUI is the same as the value for gravity.

1. Check Change Gravity – Change Gravity Slider + Board
   1. Normal Flow – Gravity variable is changed
   2. Alternative Path(s) - Slider does nothing / Gravity is changed but not to correct value

Use Case: Ball Velocity