Validation Testing Documentation

Use case: Switch to Build Mode

Test 1:

Purpose of Test:

Switch to Build Mode

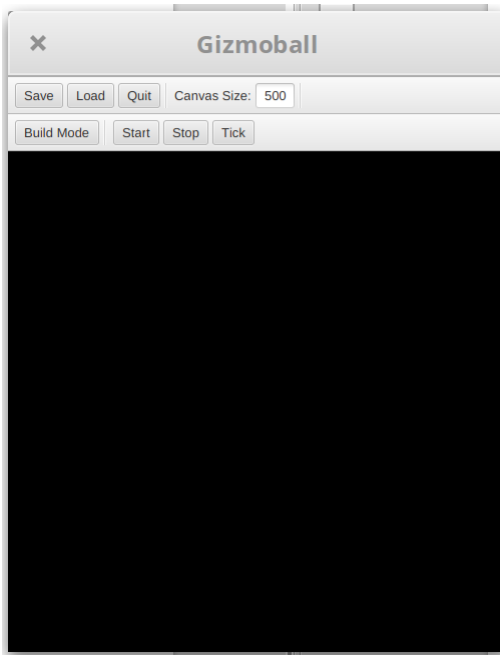
Test inputs:

1. Select Build Mode Button

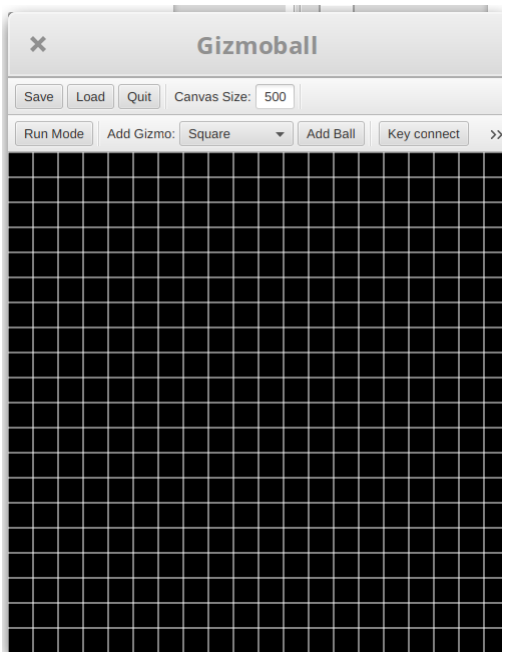
Expected Outputs:

Build Mode GUI appears, with buttons allowing the user to create a custom scenario.

i.e Add Square, Circle, buttons etc.



Game in Run Mode



Game in Build Mode

Results:

The game was run and it presented the game in run mode. The “build mode” button was clicked and the interface layout changed, new buttons appeared and a grid layout was presented. This is as intended.

Use case: Adding Gizmo

Test 1:

Purpose of Test:

Add Gizmo (Normal Path)

Test inputs:

1. Select Add Gizmo Button

2. Select desired gizmo from dropdown list/combo box

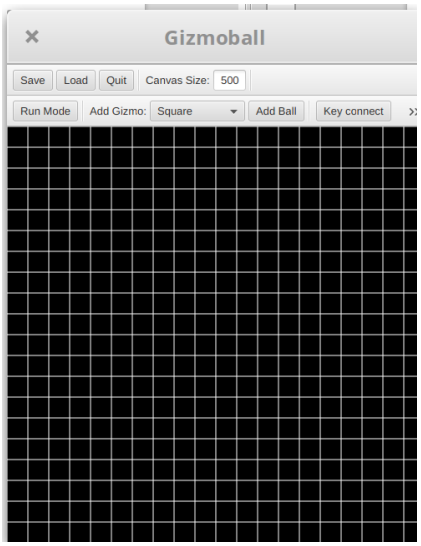
3. Select Empty Square

Expected Outputs:

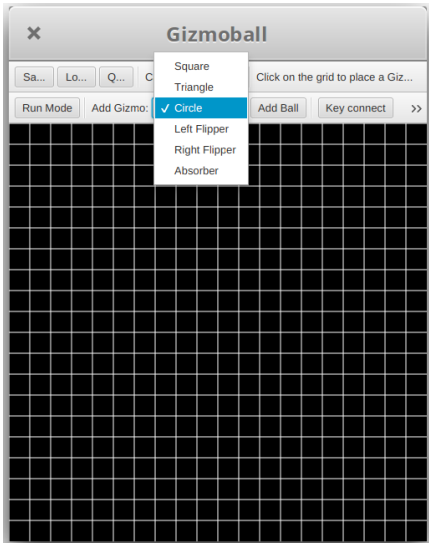
Square Gizmo Appears on Empty Square

Results:

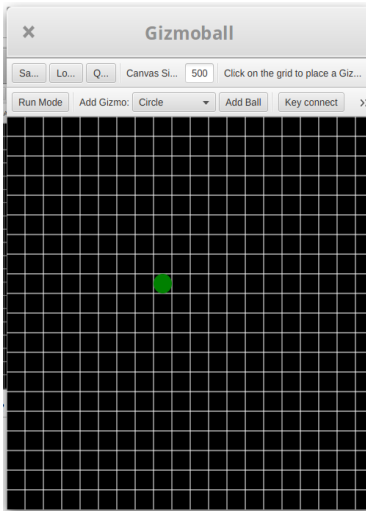
The game was in build mode and from the drop down menu, “circle” was selected. An empty grid square was then selected and a circle was successfully placed in this square. This behaviour was as intended.



Game in Build Mode



Circle Selected from Dropdown



Empty box selected to place a Gizmo

Test 2:

Purpose of Test:

Add Gizmo on Occupied Square (Alternative Path)

Test Inputs:

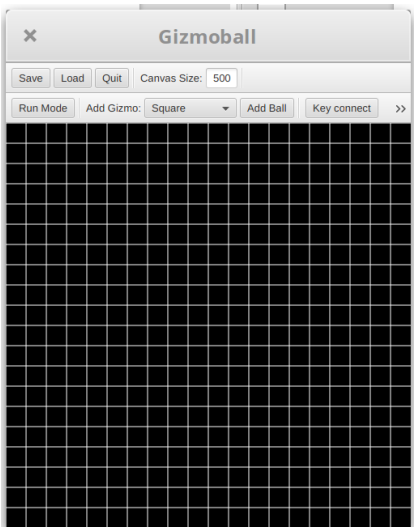
1. Select Add Gizmo Button

2. Select desired gizmo from dropdown list/combo box

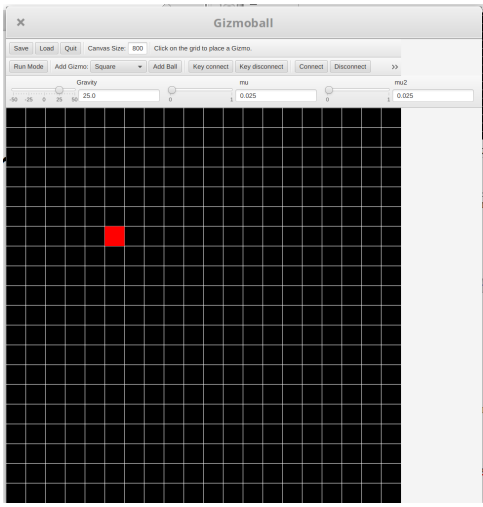
3. Select Occupied Square

Test Outputs:

Error Message Rejecting Gizmo Due to Occupied Square.



Game in Build Mode



Rejected Add Gizmo

Results:

The game was in build mode and from the drop down menu, “circle” was selected. An occupied grid square was then selected and an error message appeared warning that it could not be placed in the occupied square. This behaviour was as intended.

Use case: Move existing Gizmo

Test 1:

Purpose of Test:

Move existing Gizmo (Normal Path)

Test Inputs:

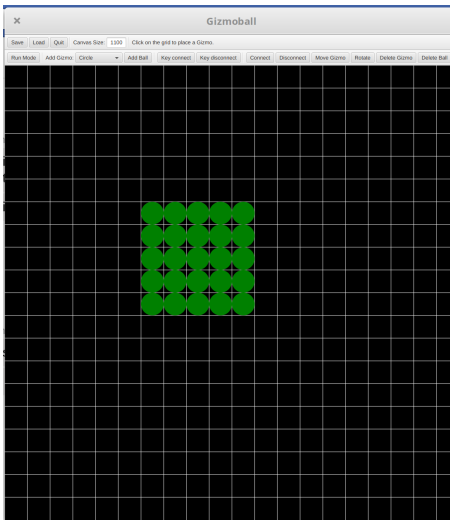
1. Select Move Button

2. Select Gizmo to be moved

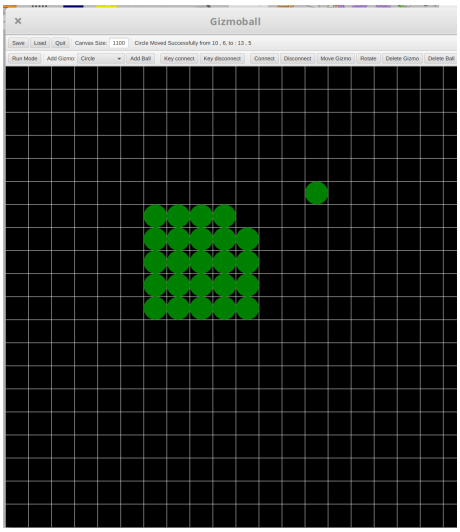
3. Select Empty Square

Test Outputs:

Gizmo is moved to new location.



Game in Build Mode with Gizmos



Game in Build Mode w/ Moved Gizmo

Results:

The game was in build mode and some gizmos were added to the grid. The “Move gizmo” button was selected and a gizmo was clicked then an empty grid square was clicked. The clicked gizmo moved to the unoccupied, selected square. This behaviour was as intended.

Test 2:

Purpose of Test:

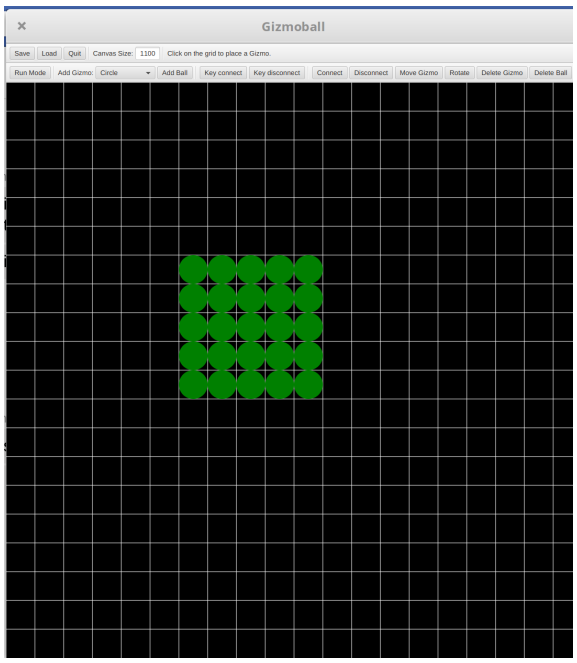
Move existing Gizmo to occupied space (Alternative Path)

Test Inputs:

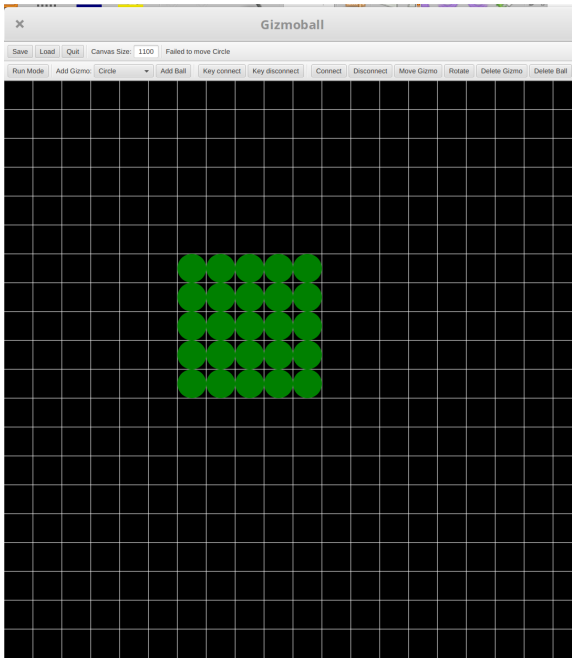
1. Select Move Button
2. Select Gizmo to be moved
3. Select Occupied Square

Test Outputs:

Gizmo isn't moved and error message rejecting gizmo move due to occupied square.



Game in Build Mode with Gizmos



Failed Move Gizmo Invocation

Results:

The game was in build mode and some gizmos were added to the grid. The "Move gizmo" button was selected and an occupied square was selected. The clicked gizmo did not move an informative message displayed in the upper menu bar. This behaviour was as intended.

Use case: Add Absorber

Test 1:

Purpose of Test:

Add Absorber Gizmo (Normal Path)

Test Inputs:

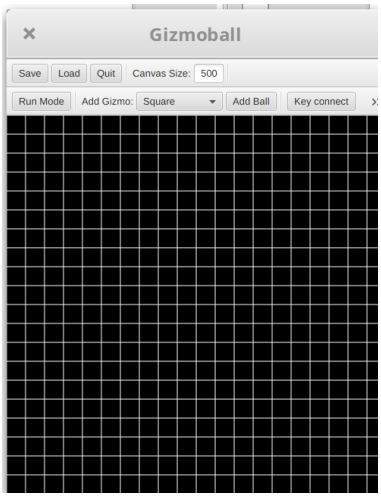
1. Select Add Gizmo Button

2. Select Absorber from the dropdown list/combo box

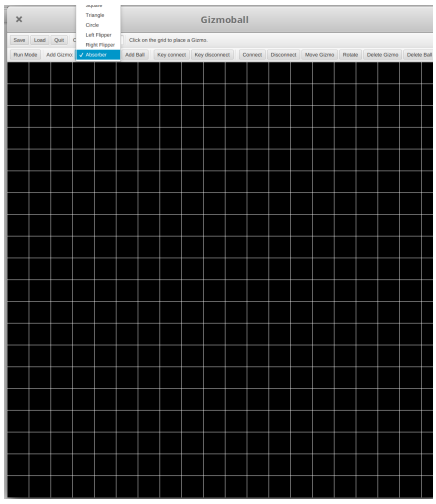
3. Drag from start square to end square (With all empty space between start and end)

Test Outputs:

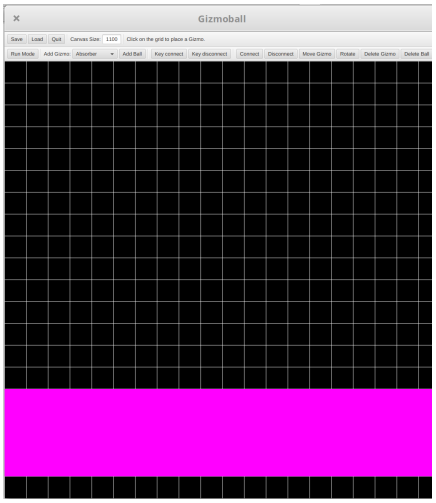
Absorber is placed between the start square and end square that the user dragged.



Game in Build Mode



Absorber Selected from Dropdown



Absorber Drawn in Build Mode

Results:

The game was in build mode and absorber was selected from the add gizmo drop down menu. An empty grid square was selected and held selected as the mouse was dragged to another unoccupied square with no occupied square in between. A new absorber was drawn. This behaviour was as intended.

CS308 Group W12

Test 2:

Purpose of Test:

Add Absorber Gizmo to area where an object already exists (Alternative Path)

Test Inputs:

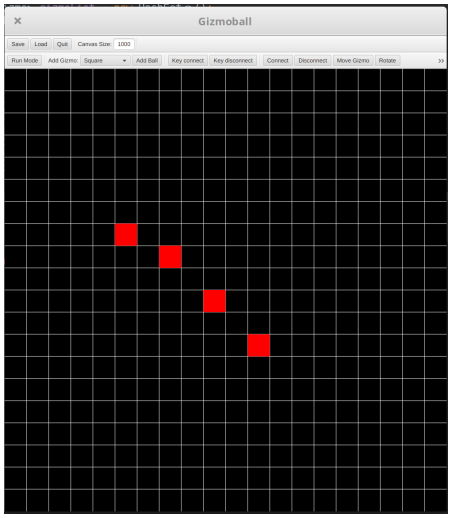
1. Select Add Gizmo Button

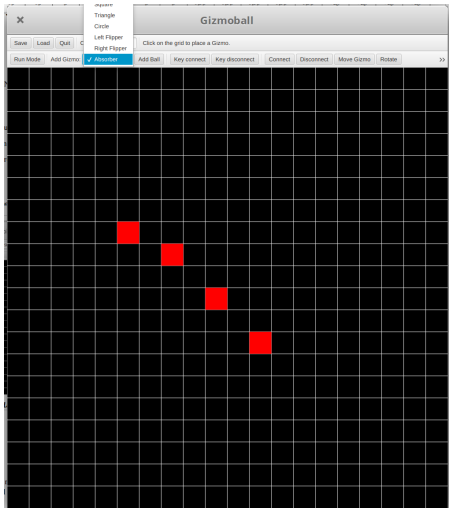
2. Select Absorber from the dropdown list/combo box

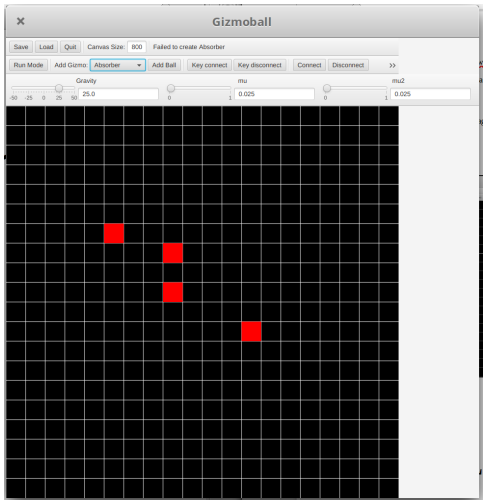
3. Drag from start square to end square (With an object placed in-between start and end points)

Test Outputs:

Absorber is rejected and error message is display about area already occupied by an existing gizmo.







Failed Add Absorber Message

Results:

The game was in build mode and several gizmos were placed on the screen, add absorber was selected from the add gizmo drop down menu. An empty grid square was selected and held selected as the mouse was dragged to another unoccupied square with the placed gizmos in between. The new absorber is rejected and an error message presented itself. This behaviour was as intended.

Use case: Rotate Gizmo

Test 1:

Purpose of Test:

Rotate Gizmo (Normal Path)

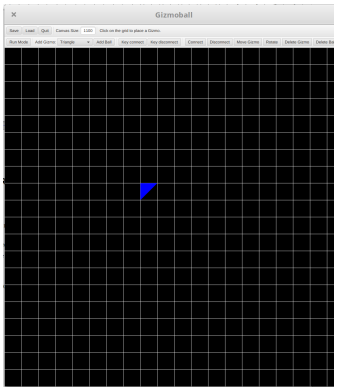
Test Inputs:

1. Selects Rotate Button

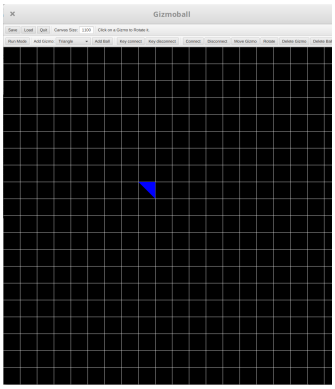
2. Selects Gizmo on the board

Test Outputs:

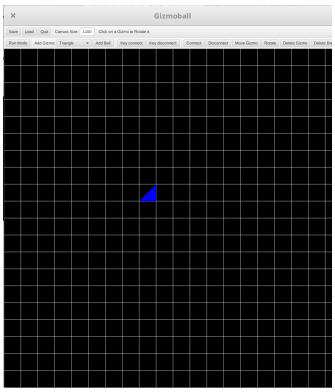
Gizmo is now rotated 90 degrees



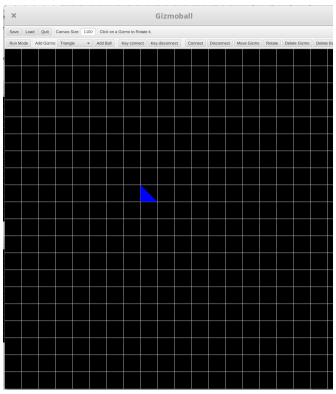
An Non-Rotated Triangle



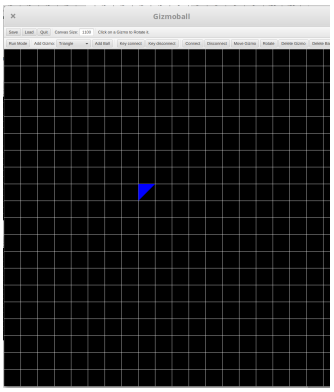
Triangle Gizmo Rotated Once



Triangle Gizmo Rotated Twice



Triangle Gizmo Rotated Thrice



Triangle Gizmo Rotated 4 Times

Results:

The game was in build mode and a triangle was placed in the board. The “rotate’ button was selected and the triangle gizmo was clicked several times until it fully rotated. The gizmo rotated 90 degrees each time. This behaviour was as intended.

Test 2:

Purpose of Test:

Rotate Gizmo that cannot be rotated (Alternative Path)

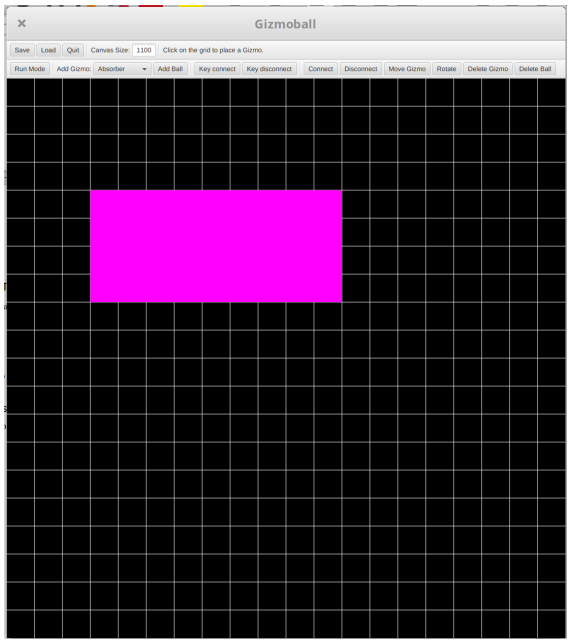
Test Inputs:

1. Selects Rotate Button

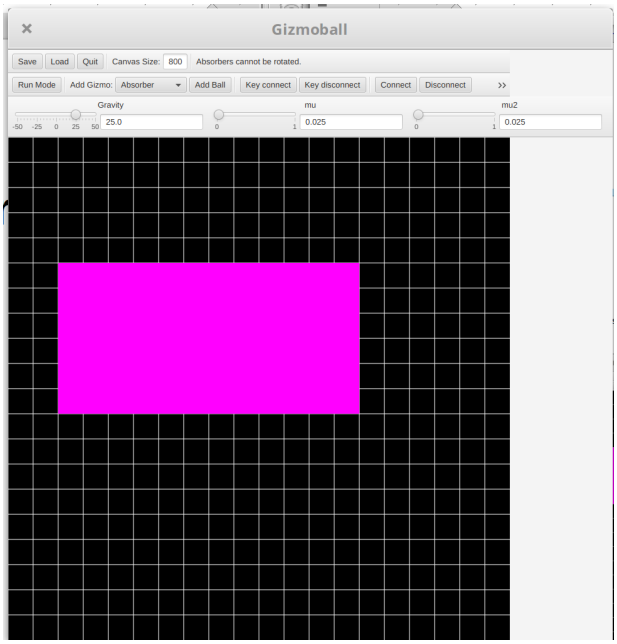
2. Selects Gizmo that can't rotate

Test Outputs:

Gizmo doesn't rotate and Error message saying gizmo cannot be rotated.



Game in Build Mode w/ Absorber



Rotate Invoked on Non-Triangle Reject Message

Results:

The game was in build mode and an absorber was placed. The rotate button was clicked and the absorber was then selected. The game rejected the rotate and printed a message to the GUI. This behaviour was as intended.

Use case: Connect Gizmo

Test 1:

Purpose of Test:

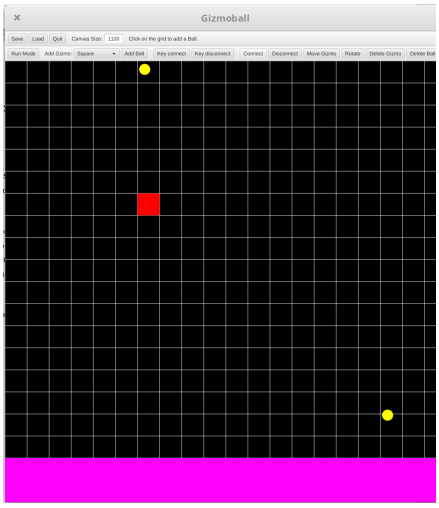
Connect Action to Gizmo (Normal Path)

Test Inputs:

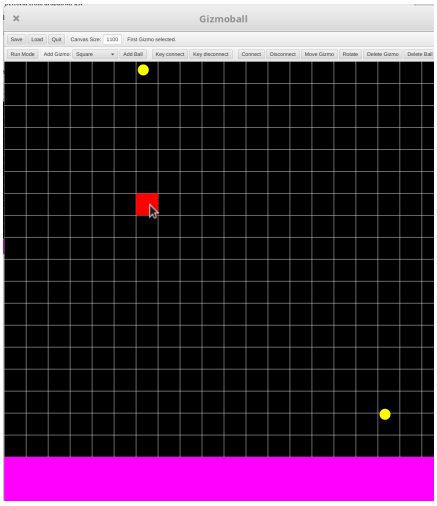
1. Selects Gizmo Connect Button
2. Selects Action to perform from dropdown list
3. Selects Gizmo which when hit triggers the target gizmo
4. Selects Target Gizmo to perform selected action

Test Outputs:

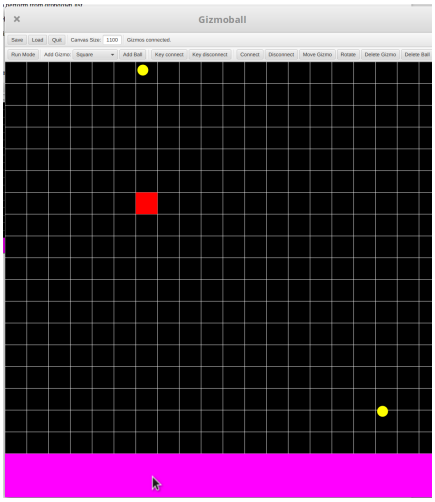
When Selected gizmo is hit the target gizmo performed the desired action.



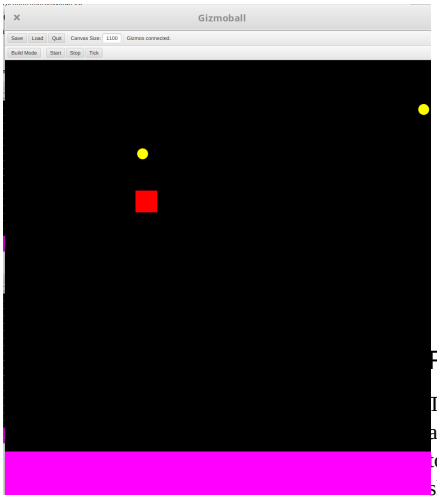
Game in Build Mode with



Connect- 1st Gizmo Selected



Connect- Second Gizmo Selected



Absorber Activated by Square Collision

Results:

The game was in build mode and a square and absorber were drawn. A ball was placed to bounce up and down on the square and a ball was placed to fall into the absorber. The gizmos were connected together to trigger the absorber when a ball collides with the square. The ball caught in the absorber was shot out when the other ball collided with the square. This behaviour was as intended.

Test 2:

Purpose of Test:

Connect Action to Gizmo that doesn't support the action (Alternative Path)

Test Inputs:

1. Selects Gizmo Connect button

2. Selects action to perform from dropdown list

3. Selects gizmo which when hit triggers the target gizmo

4. Selects target gizmo which cannot perform selected action

Test Outputs:

Error Message rejecting connection due to invalid action

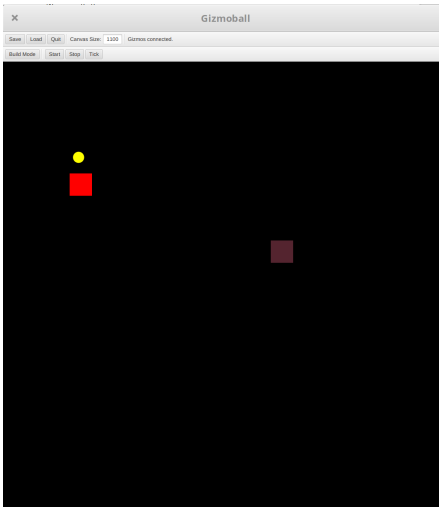
Change to Feature:

This test was now obsolete as all gizmos can now connect to each other an activate their relevant features. Upon a collision. These are as follows:

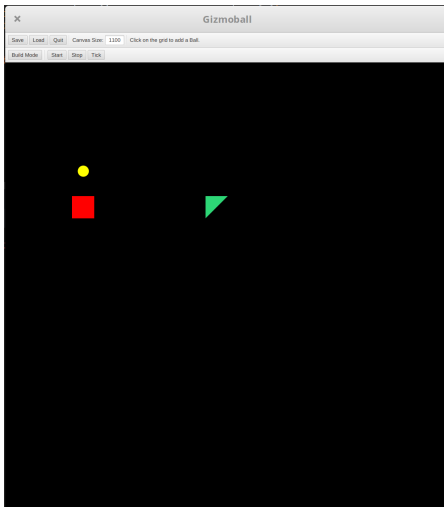
Square/Circle/Triangle- Change colour

Flipper- Rotates 90 degrees

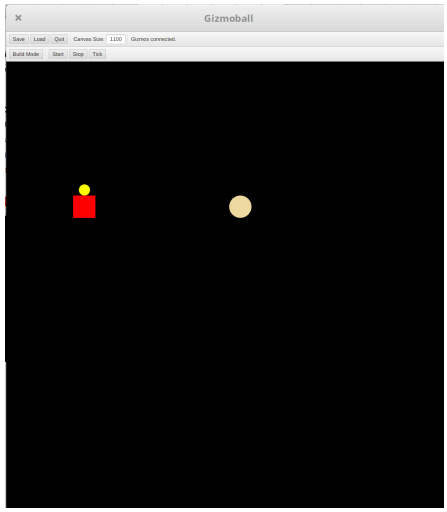
Absorber- Shoots out a caught ball



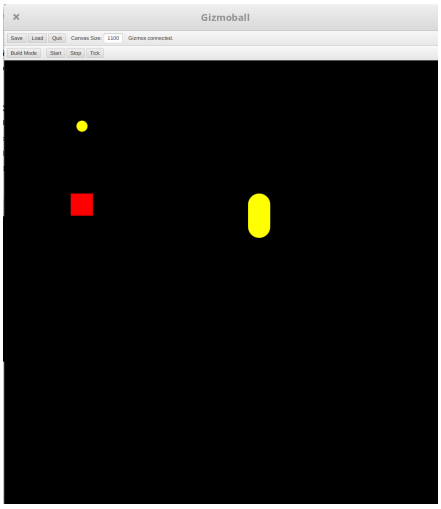
Square Changing Colour Triggered



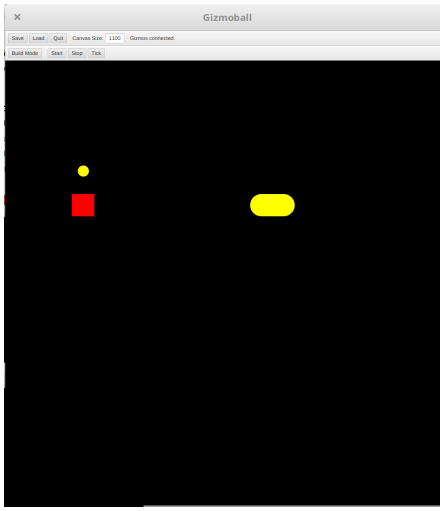
Triangle Changing Colour Triggering



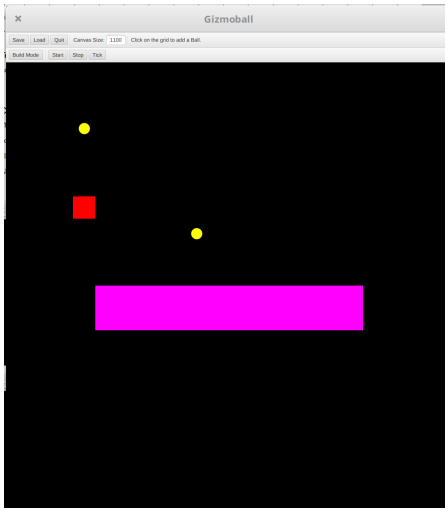
Circle Changing Colour Triggering



Flipper Before Triggered by



Flipper After Triggered by



Absorber Before Triggered by

Results:

The varying gizmos were connected to the square and executed their action when triggered by a ball colliding with the square. This behaviour was as intended.

Use case: Key Connect

Test 1:

Purpose of Test:

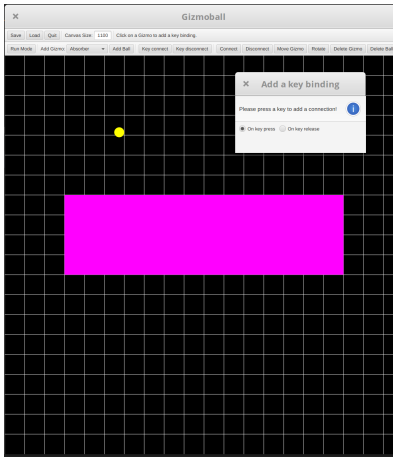
Connection Action to Key Press (Normal Path)

Test Inputs:

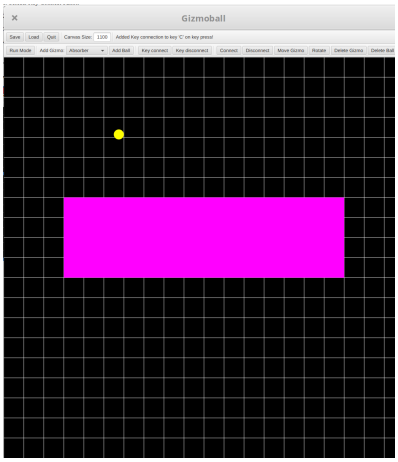
1. Selects Key Connect button
2. Selects action to perform from dropdown list
3. Selects key which triggers the action
4. Selects target gizmo to perform action

Test Outputs:

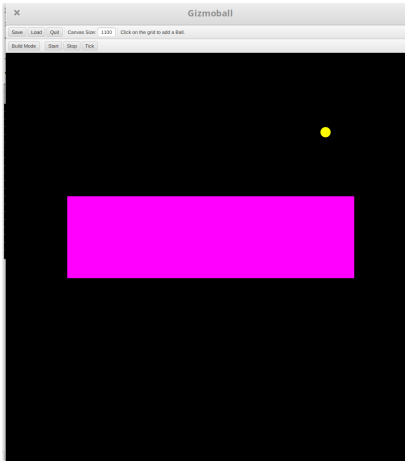
When Key is pressed the target gizmo performs the desired action.



Instructions to Add a Key Binding



Key Bind Message



Absorber Triggered by Bound Key

Results:

An absorber was drawn in build mode and the key connect button was selected. The absorber was selected and an option message appeared, notifying to specify whether or not the key trigger is on press down or release. The ‘C’ key was chosen and the message box disappeared. A confirmation message appeared in the toolbar saying that ‘C’ was bound. The game was played and the absorber shot out the caught ball when ’C’ was pressed. This behaviour was as intended.

Test 2:

Purpose of Test:

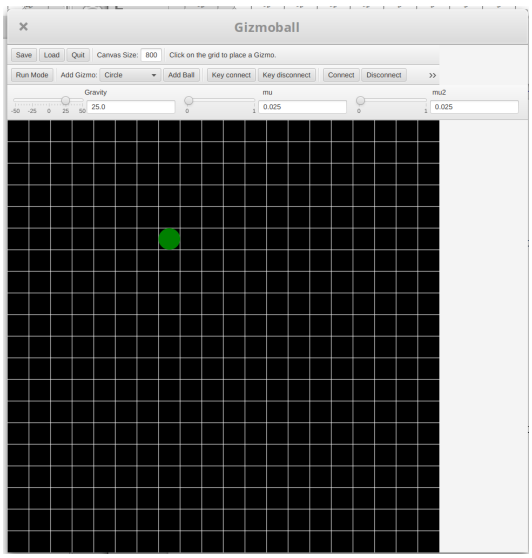
Connection Action to Key Press that doesn’t support the action (Alternative Path)

Test Inputs:

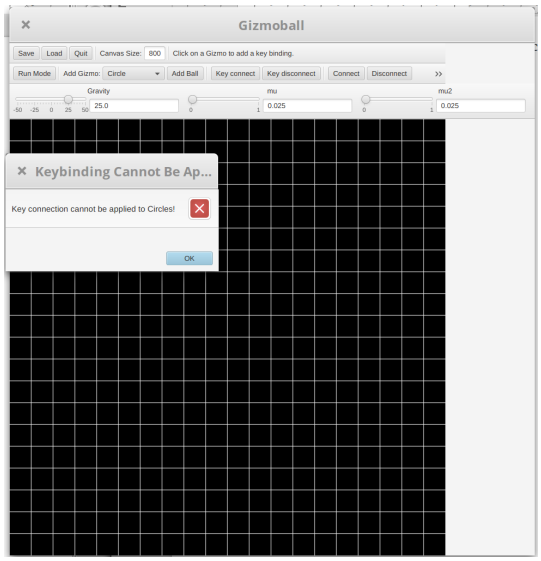
1. Selects Key Connect button
2. Selects action to perform from dropdown list
3. Selects key which triggers the action
4. Selects target gizmo which cannot perform action

Test Outputs:

Error Message rejecting connection due to invalid action



Game in Build Mode w/ Circle



Key Connect Error Message for Circle Bind

Results:

A circle was drawn in build mode then the key connect button was pressed. The circle was selected and the game presented an error message informing that circles cannot be bound to keys as they have no action. This behaviour was as intended.

Use case: Delete Gizmo

Test 1:

Purpose of Test:

Delete Gizmo

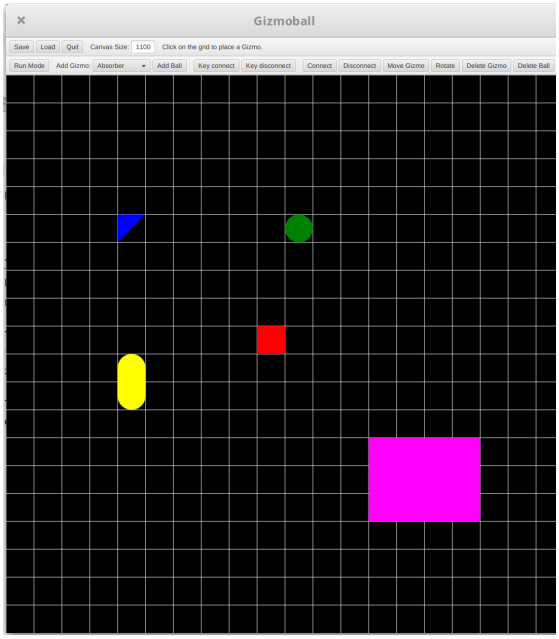
Test Inputs:

1. Selects Delete button

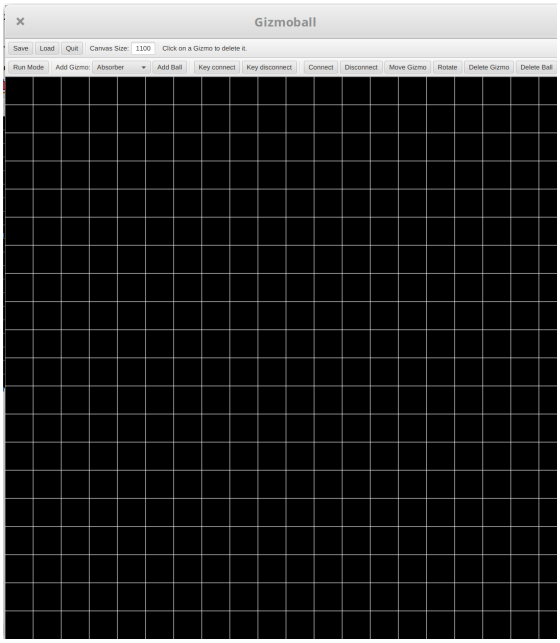
2. Selects gizmo to be deleted

Test Outputs:

Gizmo should be removed from the board.



Build Mode with Various Gizmos



Build Mode with Gizmos Deleted

Results:

All types of Gizmo were placed in the board in build mode then the delete gizmo button was selected. One by one the paced gizmos were clicked and they disappeared form the board. This behaviour was as intended.

Use case: Place Ball

Test 1

Purpose of test:

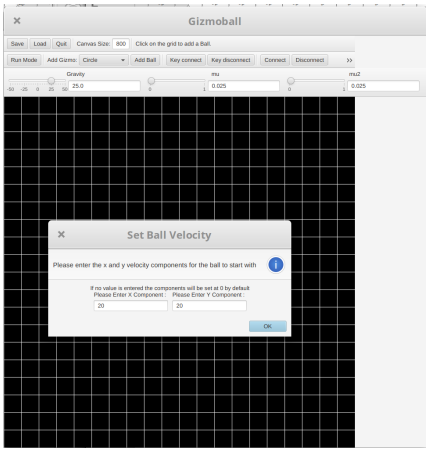
User places a ball on an empty grid square and specifies a velocity with a valid value. (Normal and intended path)

Test Inputs:

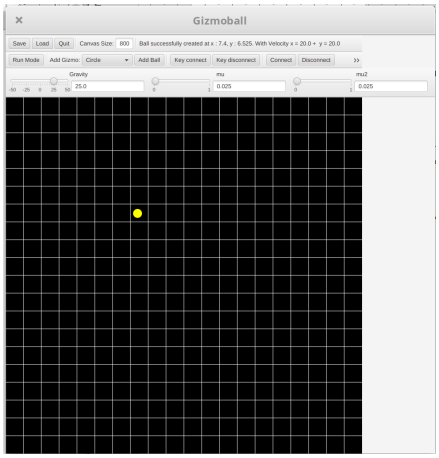
1. Select the ball element in build mode.
2. Select an empty grid square.
3. Enter the speed as a number within the data type size limits of a double.
4. Enter the angle at which the ball is going, as a number within the data type size limits of a double.
5. Select confirm/ok.

Expected Outputs:

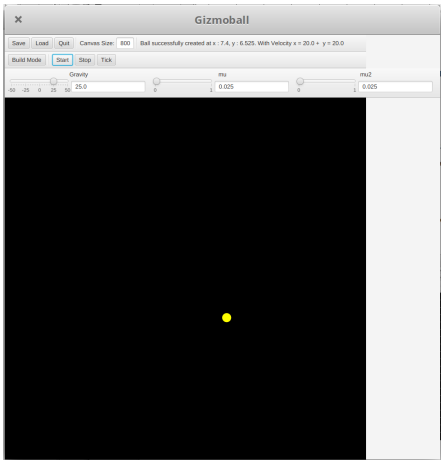
A ball appears at the selected grid square and will have the specified initial velocity.



Add Ball Velocity Form



Ball Placed in Build Mode



Ball Moves at Given Velocity

Results:

In build mode, the add ball button was selected and an empty grid square was clicked on. A fillable form was presented to give velocities for the x and y components of the ball. Upon completion the ball was added to the board. The game was run to check if the velocity was as specified and it was. This behaviour was as intended.

Test 2+

Purpose of test:

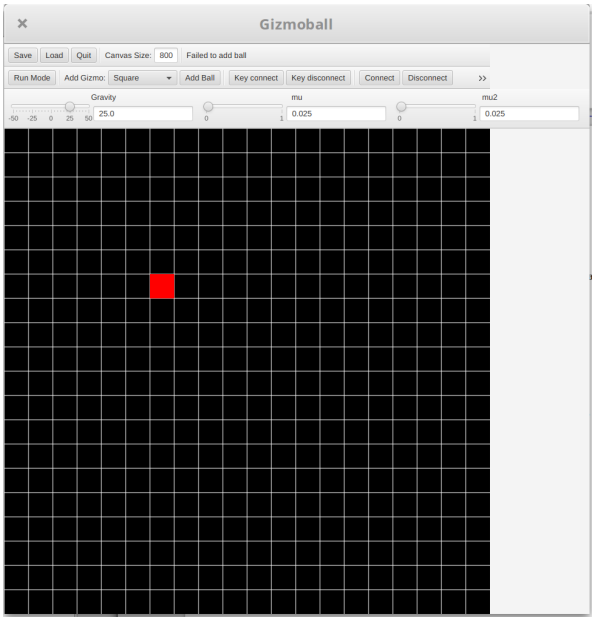
User places a ball on a grid square already occupied by a gizmo (one test for each gizmo excluding absorber). (Invalid use as balls and gizmos cannot occupy the same space, alternate path.)

Test Inputs:

1. Select the ball element in build mode.
2. Select a grid square occupied by another gizmo

Expected Outputs:

A warning is issued by the program, explaining that the ball cannot occupy the same space as a gizmo unless it is an absorber.



Rejected Add Ball Message

Results:

In build mode, the add ball button was selected and an occupied grid square was clicked on.=The game this placement and presented an error message in the GUI. This behaviour was as intended.

Use case: Save File

Test 1

Purpose of test:

The user builds a board layout and saves it to a new file to be loaded correctly at another instance. (Normal and intended path)

Test Inputs:

1. Select the save file button in build mode (whether or not there are elements on the board).
2. Enter a filename for the save file that does not already exist in the current directory.
3. Select the confirm/ok button to save the file.

Expected Outputs:

A new file is created in the current working directory of the program with the specified name and containing the correct information to build the board layout that was present when the save file functionality was invoked.

Results:

Gizmos were placed in build mode and the save button was selected. The dialogue box appeared to specify file location and a name was given. The window closed upon confirmation and the file appeared in the directory. This behaviour was as intended. See figures after all file load/save tests.

Test 2

Purpose of test:

The user builds a board layout and saves it to a file to be loaded correctly at another instance, but specifies a name already used in the current working directory of the program. The user then enters a different name. (Normal, but alternate path)

Test Inputs:

1. Select the save file button in build mode (whether or not there are elements on the board).
2. Enter a filename for the save file that already exists in the current working directory of the program and confirm.
3. The program warns the user that the filename is already in use in prompts them to enter a different name or overwrite the already present file.
4. Enter a different name.
5. Select the confirm/ok button to save the file.

Expected Outputs:

A new file is created in the current working directory of the program with the new specified name that is not the same as any other filename, and containing the correct information to build the board layout that was present when the save file functionality was invoked.

Test 3

Purpose of test:

The user builds a board layout and saves it to a file to be loaded correctly at another instance, but specifies a name already used in the current working directory of the program. The user overwrites the file with the same name already present in the current working directory. (Normal, but alternate path)

Test Inputs:

1. Select the save file button in build mode (whether or not there are elements on the board).
2. Enter a filename for the save file that already exists in the current working directory of the program and confirm.
3. The program warns the user that the filename is already in use in prompts them to enter a different name or overwrite the already present file.
4. Select the overwrite button.

Expected Outputs:

The file in the current working directory of the program with the specified name is overwritten with the correct information to build the board layout that was present when the save file functionality was invoked.

Changes:

Test 2 and 3 for file saving and loading do not needed tested as they test a Java library's functionality, not Gizmoball's.

Use case: Load File

Test 1

Purpose of test:

The user loads a file in the current working directory of the program with a specified filename, that contains no errors and builds the board and can then be edited as normal in build mode. The board initially is empty, containing no elements on it. (Normal and intended path)

Test Inputs:

1. Select the load file button in build mode.
2. Enter a filename for a file that exists with that name in the current working directory (click it in the file chooser window).
3. Select the confirm/ok button to load the file.

Expected Outputs:

The board is filled with the elements specified by the file chosen with the specified filename.

Results:

On an empty board, a file we had prepared was loaded upon clicking the load file button and selecting it in the directory. The board loaded the elements specified in the file and the game ran perfectly. See figures after all file load/save tests.

Test 2

Purpose of test:

The user loads a file in the current working directory of the program with a specified filename, that contains no errors and builds the board and can then be edited as normal in build mode. The board contains some already placed elements. (Normal, but alternate path)

Test Inputs:

1. Select the load file button in build mode.
2. Enter a filename for a file that exists with that name in the current working directory (click it in the file chooser window).
3. Select the confirm/ok button to load the file.
4. The user is issued a warning that their current board will be wiped and can choose to continue or to not.
5. Select to discard current board content.

Expected Outputs:

The board is wiped of all the content that was previously on it and filled with the elements specified by the file chosen with the specified filename.

Results:

On an occupied board, a file we had prepared was loaded upon clicking the load file button and selecting it in the directory. The board cleared, then loaded the elements specified in the file and the game ran perfectly. See figures after all file load/save tests.

CS308 Group W12

Test 3

Purpose of test:

The user loads a file in the current working directory of the program with a specified filename, that contains an error. The program should reject this file and stop attempting to read and load from it. The original board layout should remain. (Handling errors, alternative path)

Test Inputs:

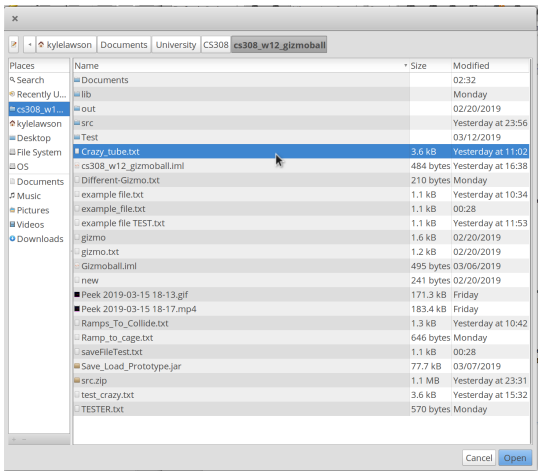
1. Select the load file button in build mode.
2. Enter a filename for a file that exists with that name in the current working directory (click it in the file chooser window).
3. Select the confirm/ok button to load the file.
4. Select the ok/confirm button to dismiss the error warning window.

Expected Outputs:

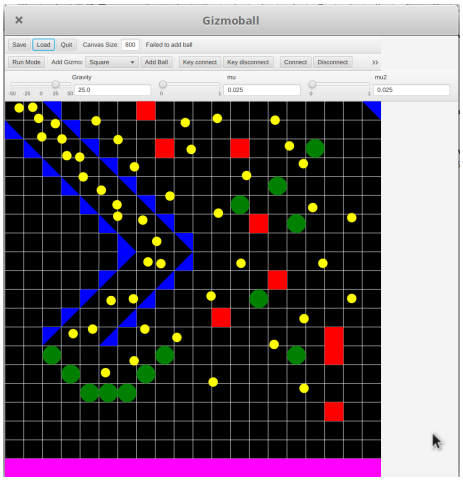
The board should not have changed from its original layout from before the load file button was selected.

Results:

The load file button was selected and in the file chooser window, a faulty file was chosen. The game rejected the file and displayed an error message explaining that it does not conform to the standard text format. See figures after all file load/save tests.

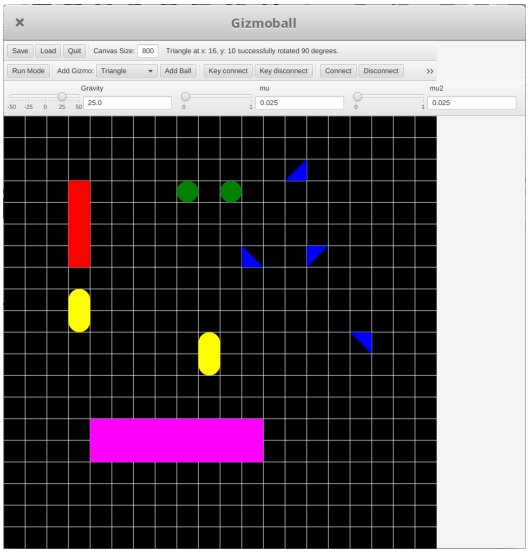


File Chooser to Load a file

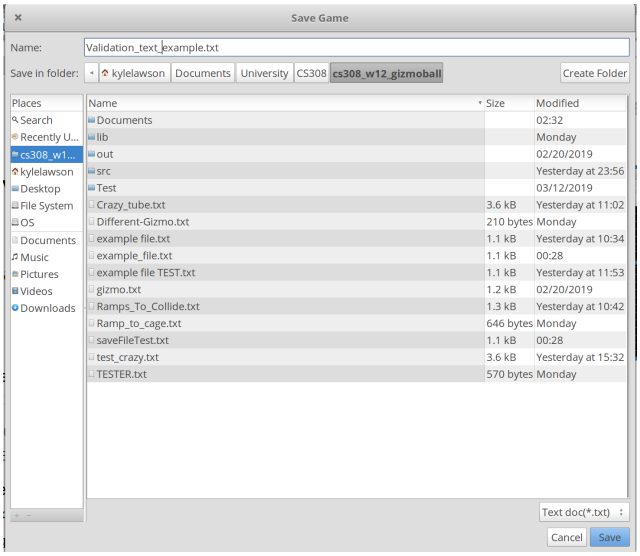


File Loaded

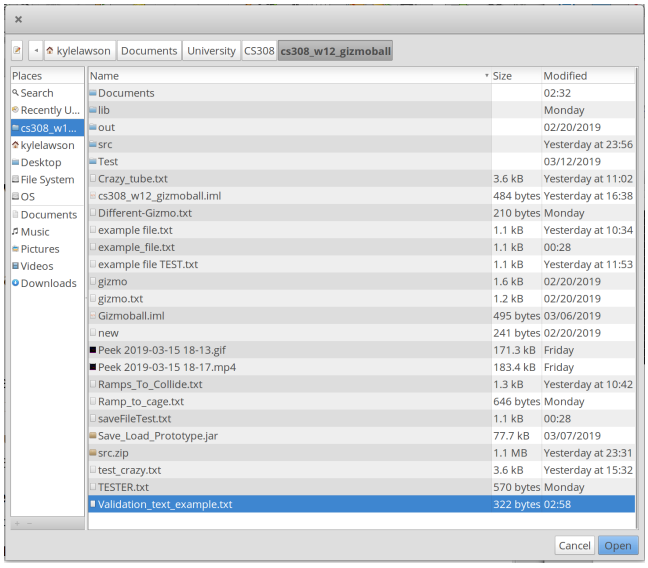
CS308 Group W12



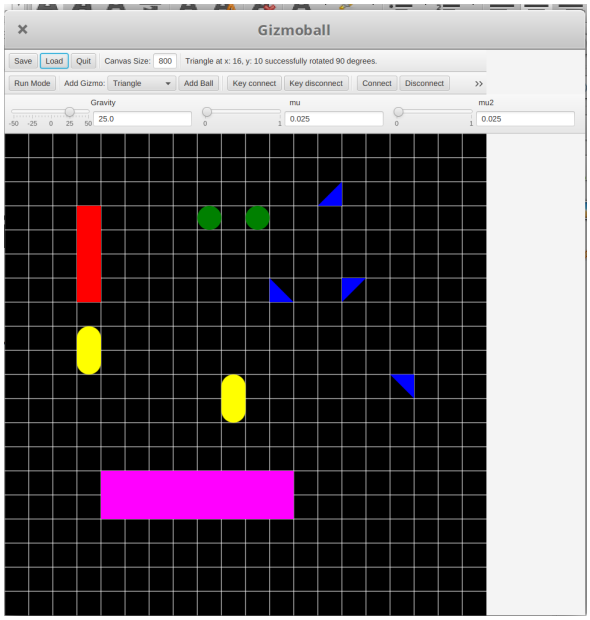
Game Drawn Out with all Type of Gizmos



Saving Layout to Validation_Test_example.txt



Loading Validation_Test_example.txt



Loaded Layout Same as Before

Use case: Switch from Build Mode to Run Mode

Test 1

Purpose of test:

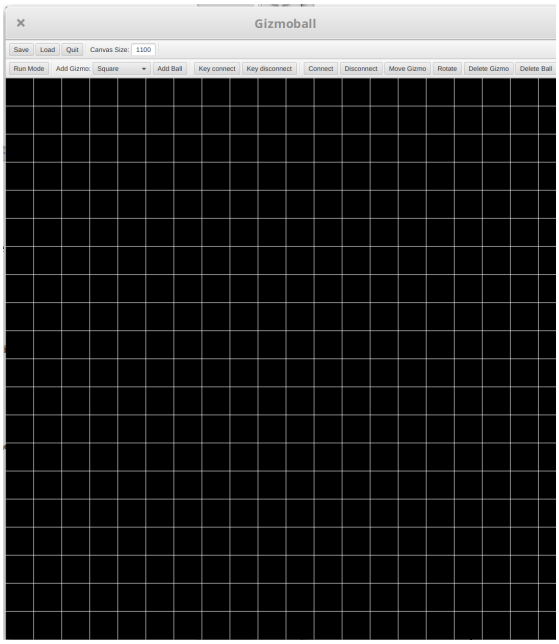
The user switches to run mode to play the game with an empty board layout. (Normal and intended path)

Test Inputs:

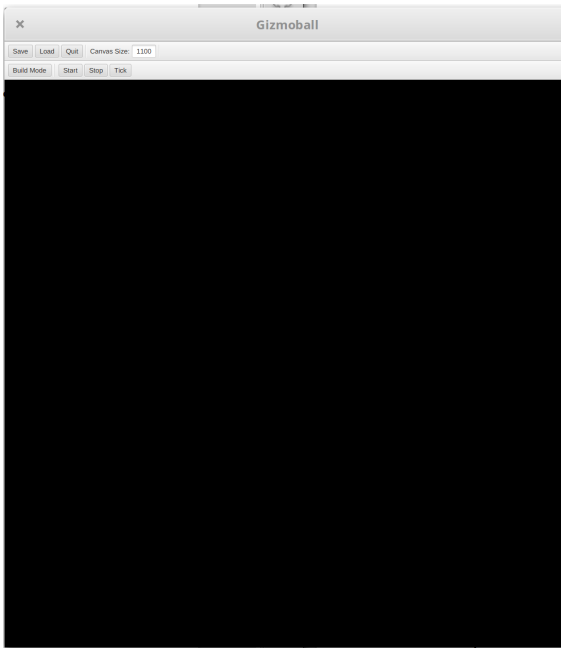
1. Select the Run Mode button in build mode with an empty board.

Expected Outputs:

The GUI changes layout, displaying only the empty board and no info as to the speed of the ball as it does not exist.



Game in Build Mode



Game in Run Mode

Results:

The game was in build mode with an empty grid and the run mode button was selected. The board changed layout, removing the build mode buttons, replacing them with run mode buttons. The grid also disappeared. This was as intended.

Test 2

Purpose of test:

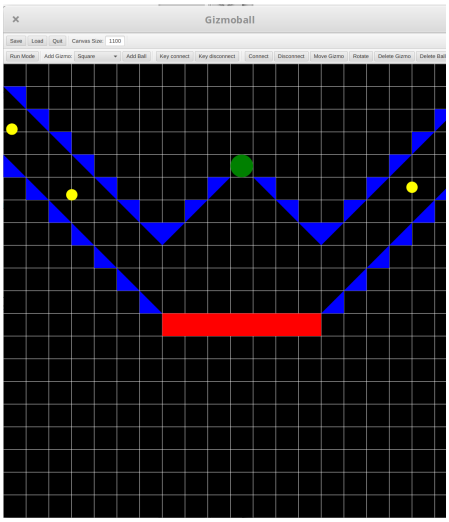
The user switches to run mode to play the game with a filled board in build mode. (Normal and intended, but alternate path)

Test Inputs:

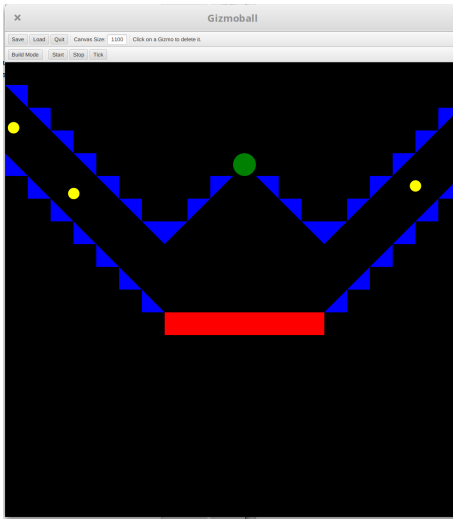
1. Select the Run Mode button in build mode with a board containing elements.

Expected Outputs:

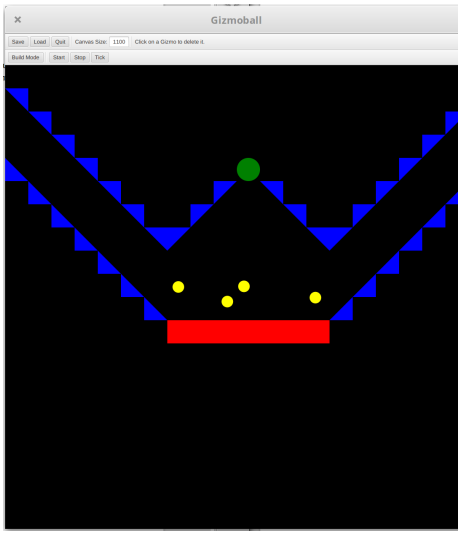
The GUI changes layout, displaying only the board containing identical elements and layouts from its state in the build mode. The game begins with the ball(s) moving at its initial velocity as specified in build mode.



Game in Build Mode



Game in Run Mode (Not Started)



Game in Run Mode (Started)

Results:

The game was in build mode with an occupied grid and the run mode button was selected. The board changed layout, removing the build mode buttons, replacing them with run mode buttons. The grid also disappeared but the gizmos and balls remained. The game was started and it ran fine. This was as intended.

Use case: Ball hits a Gizmo

Test 1+

(Test for each Gizmo other than absorber)

Purpose of test:

The ball moves through empty space then collides with a gizmo (including walls of the grid but excluding absorbers) and rebounds, one of the most common and normal events in run mode. (Normal path)

Test Inputs:

1. Build a board with a gizmo and a ball, initialising the ball to move with a velocity to hit the gizmo.
2. Select the switch the Run Mode button.

Expected Outputs:

The game should start and the ball will begin to move with the specified velocity toward the gizmo. Collision should have no clipping and velocity should change according to pre-collision velocity, gizmo shape and friction coefficient.

Test 2

Purpose of test:

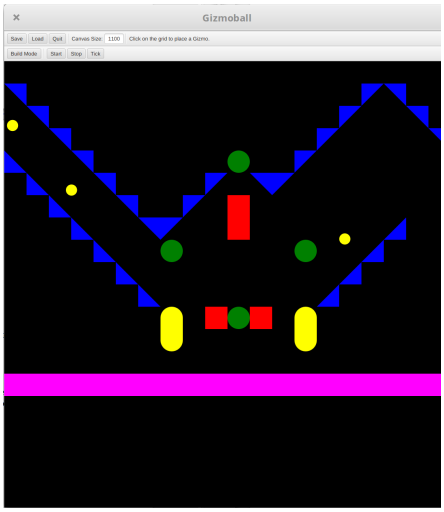
The ball moves through empty space then collides with an absorber (including walls of the grid but excluding absorbers) and the game reacts appropriately. (Normal but alternate path)

Test Inputs:

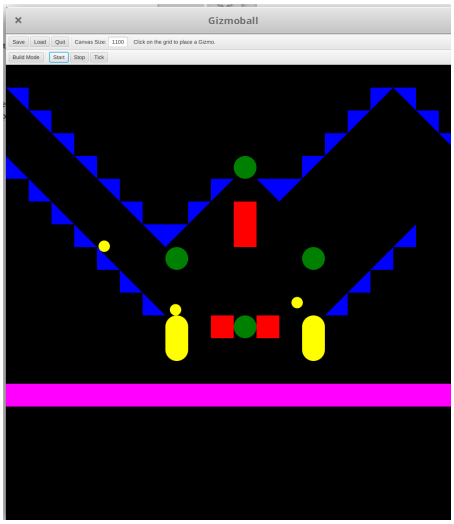
1. Build a board with an absorber and a ball, initialising the ball to move with a velocity to hit the absorber.
2. Select the switch the Run Mode button.

Expected Outputs:

The game should start and the ball will begin to move with the specified velocity toward the absorber. On collision the ball should instantly move to the top right hand corner of the absorber and be fired directly out at 50L/s. The speed can be verified by observing the speed measurement.



Game in Run Mode (Not Started)



Game in Run Mode (Started with Collisions)

CS308 Group W12

Results:

A board was built to include all types of gizmo. The game was ran and observed. The ball hit all the gizmos and collided perfectly, as for the absorber, the ball moved to the far right hand side of it until shot back out by key press of bound key. This was as intended.

Use case: Ticker

Test 1

Purpose of test:

Make the Run mode play on manually triggered ticks instead of an automatically recurring tick. Increment by one tick where the ball moves into empty space. (Normal path)

Test Inputs:

1. Select run mode button to start a game with a built board containing elements including a moving ball.
2. Select the switch the Run Mode button.
3. After the ball begins to move and when it is well away from elements it could collide with, select the tick button.
4. Press the tick button a few times while not near any gizmos.

Expected Outputs:

The game should start and the ball will begin to move. When tick is pressed, the ball should move a distance equal to the distance it would travel in 50ms, then pause.

Test 2

Purpose of test:

Make the Run mode play on manually triggered ticks instead of an automatically recurring tick. Increment by one tick where the ball moves to collide with a gizmo. (Normal path)

Test Inputs:

1. Select run mode button to start a game with a built board containing elements including a moving ball.
2. Select the switch the Run Mode button.
3. After the ball begins to move and when it is near a gizmo and about to collide with it, select the tick button.
4. Press the tick button until the ball collides and rebounds.

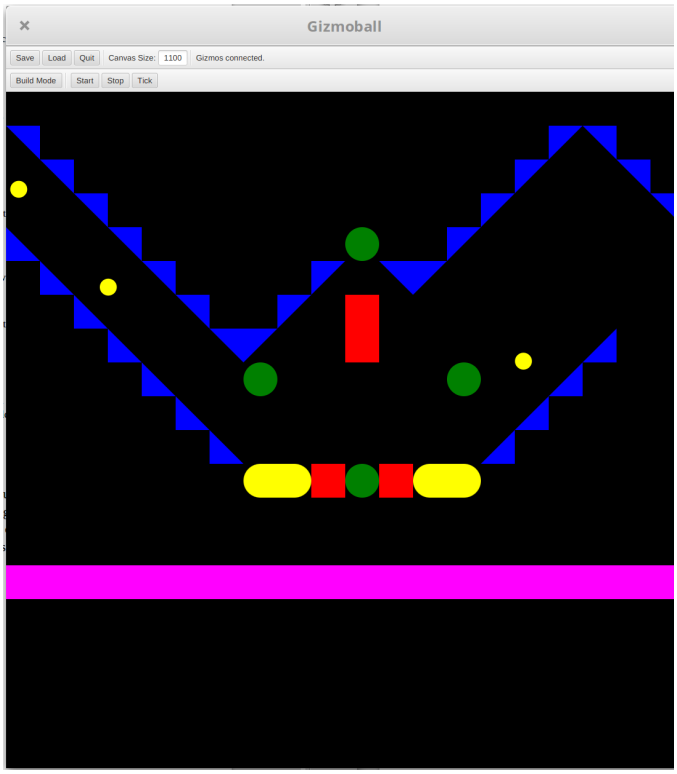
Expected Outputs:

The game should start and the ball will begin to move. When tick is pressed, the ball should move a distance equal to the distance it would travel in 50ms, unless it were to collide with a gizmo in that distance, then it only moves for that distance and the ball should appear to be nicely touching the gizmo, then rebound for a distance equal to the distance it would travel in 50ms, or until it hits another gizmo. The game should then pause after a tick completes.

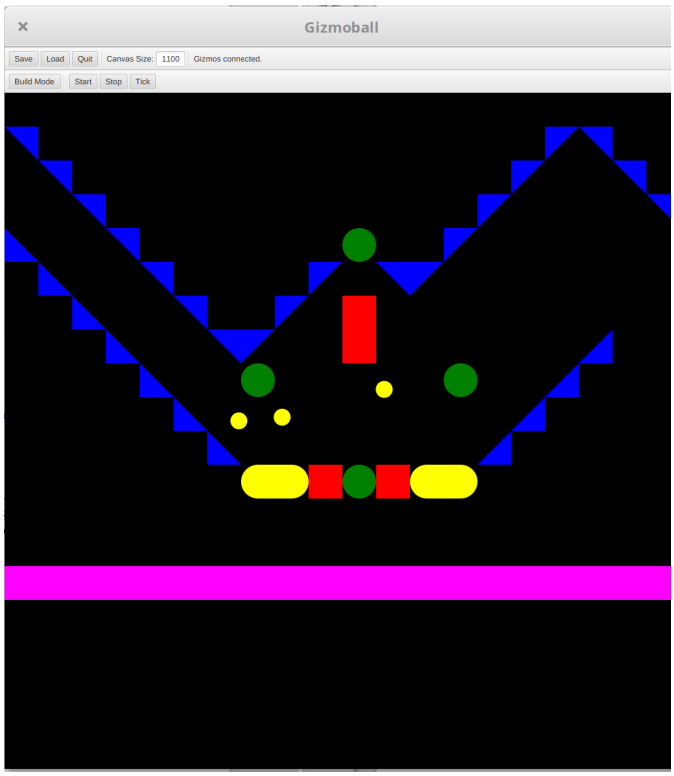
Results:

A board was built to include all types of gizmo. The game was ran and observed. The tick button was pressed and the game paused, it was clicked several times and the game was observed. When no balls were going to hit anything fro greater than 50ms, the game moved the balls as if 50ms had passed, and if they would have collided with something in under 50ms, the game moved until then and paused, showing the perfect collisions. This was as intended.

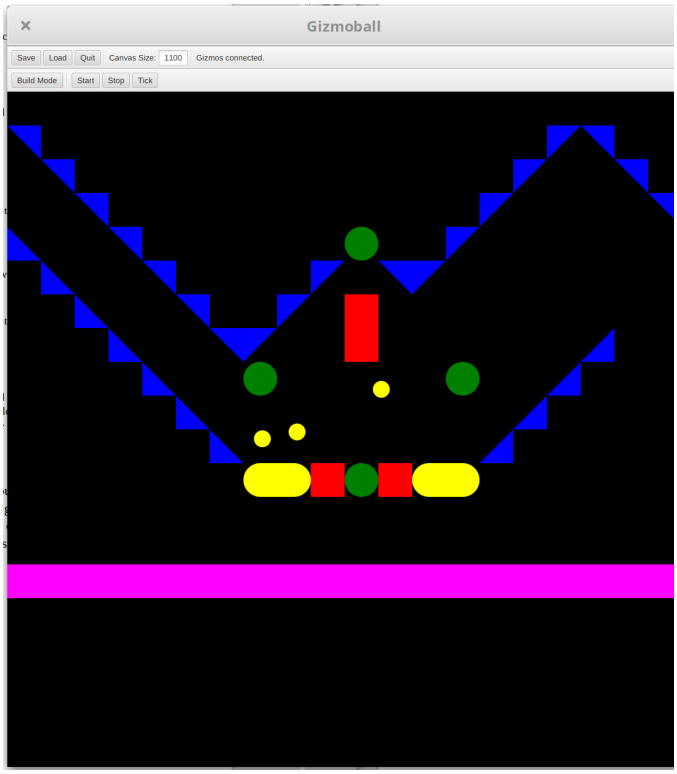
CS308 Group W12



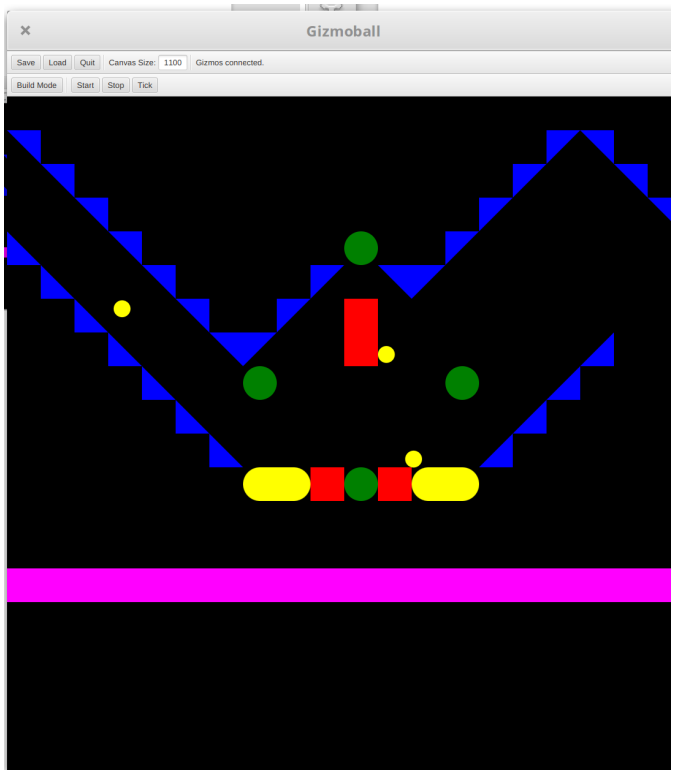
Game in Run Mode (Not Started)



Game in Run Mode (Started)



Game Ticked by 50ms (No Collision)



Game Ticked and Stopped at Collision

Use case: Quit

Test 1

Purpose of test:

The game is running and the game has been quit. (Normal path)

Test Inputs:

1. Select run mode button to start a game with a built board containing elements including a moving ball.
2. Once the game begins, select the quit button.

Expected Outputs:

The window for Gizmoball should close without warning.

Test 2

Purpose of test:

The game is in build mode and the game has been quit, the user wants to save the data before quitting. (Alternate path)

Test Inputs:

1. The game is in build mode, select the quit button.
2. Select the save option from the quitting warning.
3. Enter a filename and save as normal.

Expected Outputs:

A warning window will be issued to ask the user if they wish to save the board being built. The user selects the option to save to a file and the saved file should contain the information to build the board that was being built when the quit button was invoked.

Test 3

Purpose of test:

The game is in build mode and the game has been quit, the user does not want to save the data before quitting. (Alternate path)

Test Inputs:

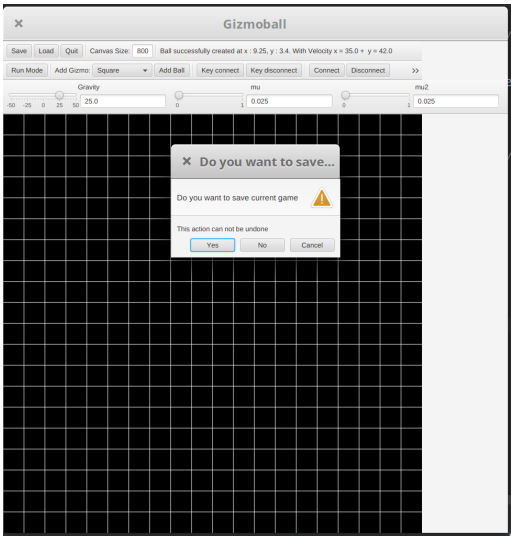
1. The game is in build mode, select the quit button.
2. Select the quit option form the warning issued by quitting.

Expected Outputs:

A warning window will be issued to ask the user if they wish to save the board being built. The user selects the quit without saving their board. The game window closes and no files are saved to the current working directory.

Results:

The board was occupied with gizmos and balls and ran. The game was then closed using the quit button. The window closed without warning. As for Tests 2 and 3, we present a warning to the user to save first if they try and close in build mode. This was as intended.



Save Warning on Close

Use case: Setting Friction

Test 1

Purpose of test:

Entering a non-default and valid value for friction in build mode and running the game with it.

Test Inputs:

1. In build mode select add ball button.
2. Enter valid speed and direction values.
3. Select the confirm/ok button to add the ball.
4. In the friction field enter a positive number up to the maximum value of a double.
5. Select the run mode button.

Expected Outputs:

The game should start and the ball should begin to move, changing in speed according to the value of friction entered. If the value is greater than the default value, it will reduce in speed at a greater rate, and the opposite for smaller value of friction. The speed value can also be monitored.

Test 2

Purpose of test:

Entering a non-valid value for friction in build mode such as a String and running the game with it.

Test Inputs:

1. In build mode select add ball button.
2. Enter valid speed and direction values.
3. Select the confirm/ok button to add the ball.
4. In the friction field enter a String of letters.
5. Select the run mode button.

Expected Outputs:

The game should issue a warning to the user saying the friction value entered is not valid and return them to the build mode screen.

Results:

A layout was loaded and the run twice, each with different levels of friction, 0.025 and 0.5. The behaviour of the balls were observed. On 0.025, the balls maintained far more momentum than 0.5 friction. This was as intended.

Use case: Setting Gravity

Test 1

Purpose of test:

Entering a non-default and valid positive value for gravity in build mode and running the game with it.

Test Inputs:

1. In build mode select add ball button.
2. Enter valid speed and direction values.
3. Select the confirm/ok button to add the ball.
4. In the gravity field enter a positive number up to the maximum value of a double.
5. Select the run mode button.

Expected Outputs:

The game should start and the ball should begin to move, changing in speed and direction according to the value of gravity entered. If the magnitude of the value entered is greater than the default value of gravity, it will reduce in speed at a greater rate when travelling toward the top of the screen (then increase the speed of the ball when travelling toward the bottom of the screen), and the opposite for a smaller magnitude value of gravity. The speed value can also be monitored.

Test 2

Purpose of test:

Entering a non-default and valid negative value for gravity in build mode and running the game with it.

Test Inputs:

1. In build mode select add ball button.
2. Enter valid speed and direction values.
3. Select the confirm/ok button to add the ball.
4. In the gravity field enter a negative number up to the minimum negative value of a double.
5. Select the run mode button.

Expected Outputs:

The game should start and the ball should begin to move, changing in speed and direction according to the value of gravity entered. If the magnitude of the value entered is greater than the default value of gravity, it will increase in speed at a greater rate when travelling toward the top of the screen (then decrease the speed of the ball when travelling toward the bottom of the screen), and the opposite for a smaller magnitude value of gravity. The speed value can also be monitored.

Test 3

Purpose of test:

Entering a non-valid value for gravity in build mode such as a String and running the game with it.

Test Inputs:

1. In build mode select add ball button.
2. Enter valid speed and direction values.

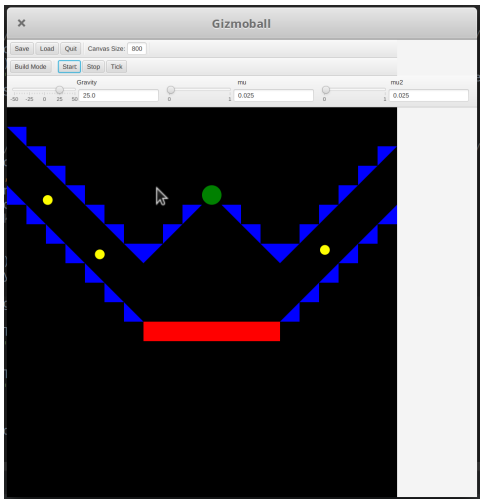
3. Select the confirm/ok button to add the ball.
4. In the gravity field enter a String of letters.
5. Select the run mode button.

Expected Outputs:

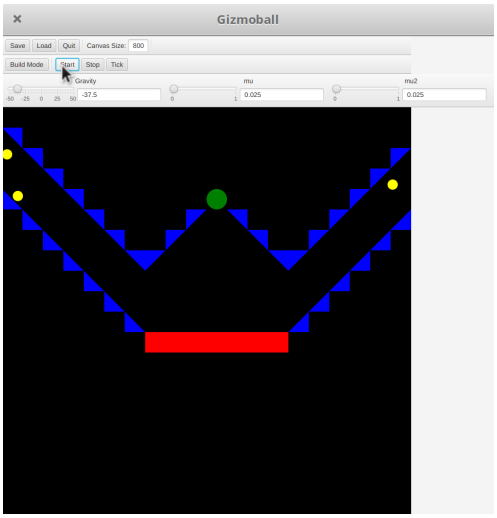
The game should issue a warning to the user saying the gravity value entered is not valid and return them to the build mode screen.

Results:

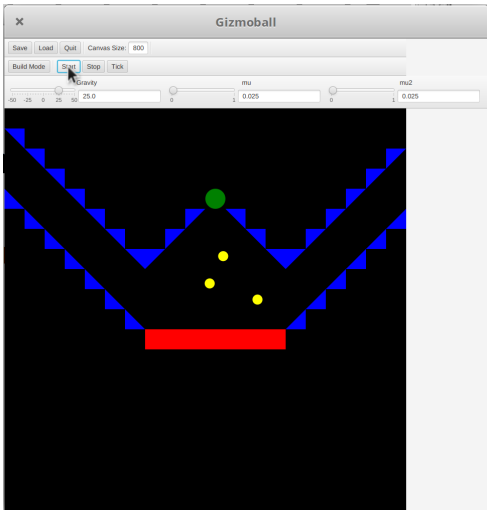
A layout was loaded and thrice. Once with positive gravity, once with negative gravity and once with 0 gravity. In positive gravity the ball went down, negative, the ball was pulled up and 0 the ball came to a halt as there was no acceleration. This was as intended.



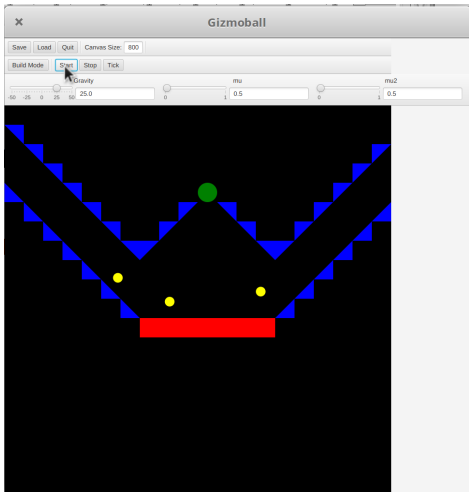
Positive Gravity



Negative Gravity



0.025 Friction



0.5 Friction

Use case: Removing Connections

Test 1

Purpose of test:

The user removes an existing connection between two elements in build mode.

Test Inputs:

1. In build mode select add gizmo button
2. Select a free grid square to place the gizmo.
3. Select add gizmo button.
4. Select a free grid square to place the gizmo.
5. Select add connection button.
6. Select one of the gizmos, then select the other gizmo.
7. Select the remove connection button.
8. Select the connection line between the two gizmos on which a connection was just made.
9. Select the run mode button.
10. Test connection that was made then removed.

Expected Outputs:

When the connection is clicked after the remove connection button is selected, the line should disappear.

The game should start and the connection between the two triggerable gizmos should not be present and the action that would trigger them does nothing.

Purpose of test:

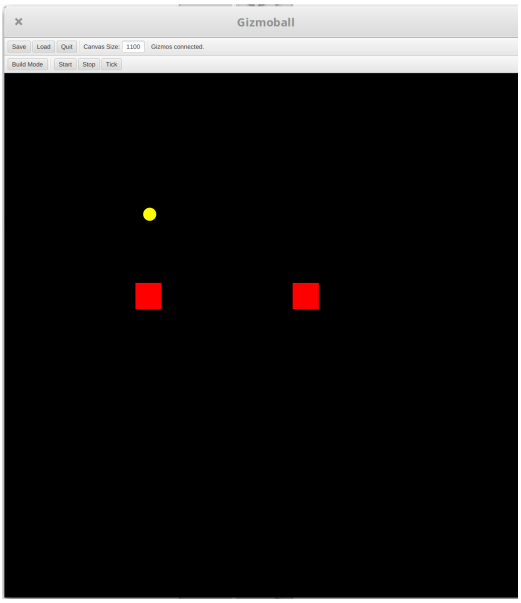
The user clicks anything but a connection line in build mode.

Test Inputs:

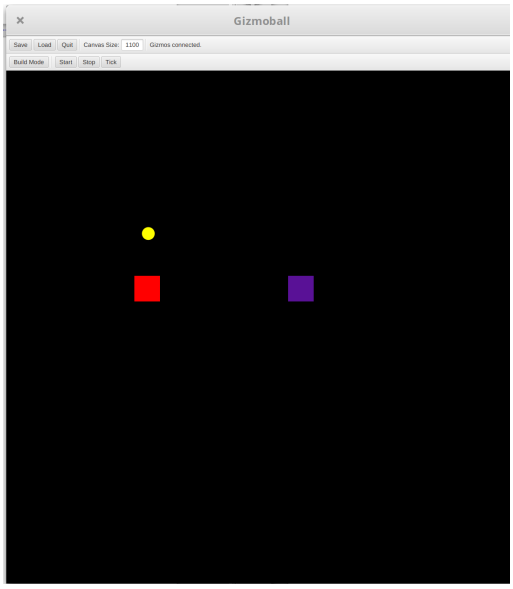
1. In build mode select the remove connection button.
2. Select anything on the board other than a connection line.

Expected Outputs:

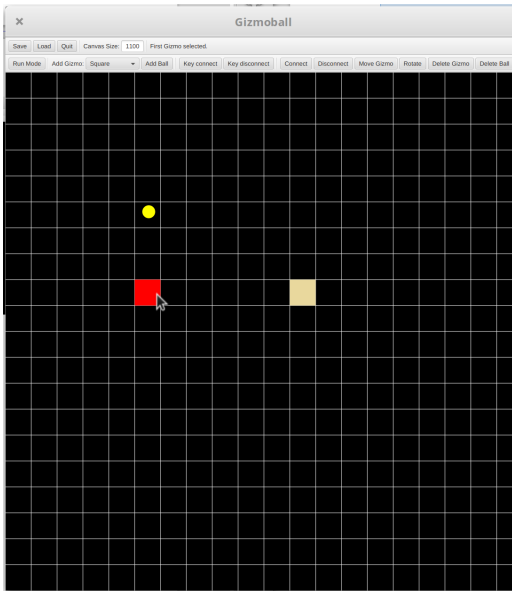
There is no functional or visible difference to the building process.



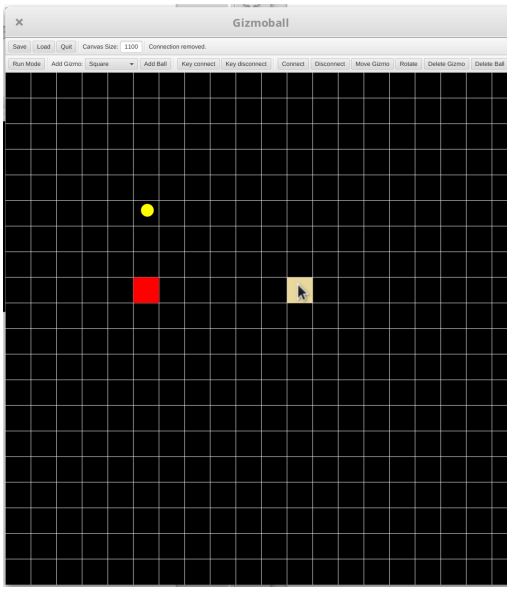
Game in Run Mode (Connection not Triggered)



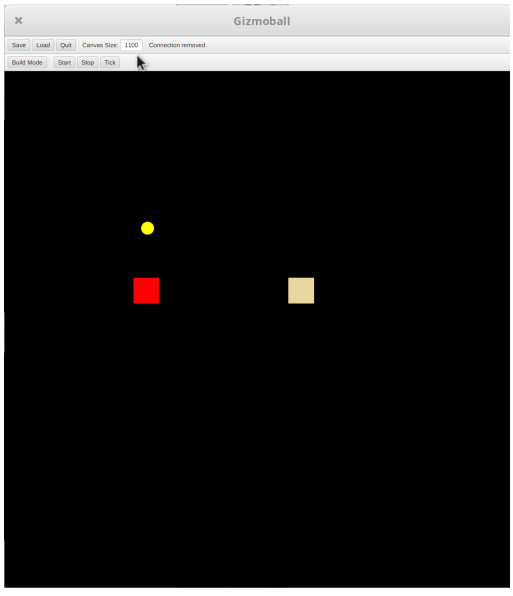
Game in Run Mode (Connection Triggered)



Game in Build Mode (First Connection Disconnect Gizmo



Game in Build Mode (Second Connection Disconnect Gizmo Selected)



Game in Run Mode (No Connection

DISCLAIMER: GUI styling does not resemble final submission as this was done before styling in final stages. Styling is non-functional and thus does not affect the validity of tests.