Gizmoball Preliminary Design

Table of Contents

Revised Specifications

Gizmoball is a version of pinball; an arcade game in which the purpose is to keep a ball moving around, without falling off to the bottom of the playing area. The player controls a set of flippers that can bat at the ball as it falls. The advantage of Gizmoball is that it allows users to construct their own machine layout by placing gizmos on the playing field.
Gizmoball has a graphical user interface with two modes, **building mode** and **running mode**.

In building mode, a user can:

1. create and edit square, circular, and triangular **bumpers** on the playing surface,
2. create and edit **flippers**,
3. **connect** the action of flippers and bumpers to triggers, such as a keyboard key being hit or one of the bumpers being collided with, and
4. **save and load** the user's game configuration to and from a file.

In running mode, the user can **play** the game. User can **switch** between modes from within each mode.

Build Mode

User can:

1. Add any of the available types of gizmos to the playing area (squares, circles, triangles, flippers, absorbers).
2. Move a gizmo from one place to another on the playing area.
3. Apply a 90-degree clockwise rotation to any gizmo.
4. Connect a particular gizmo’s trigger to a particular gizmo’s action. The standard gizmos produce a trigger when hit by the ball, and exhibit at most one action. Triggers can be connected to the actions of many gizmos and a gizmo’s action can be activated by many triggers.
5. Connect a key-press trigger to the action of a gizmo. Each keyboard key generates a unique trigger when pressed. Key-press triggers can also be connected to the actions of many gizmos.
6. Delete a gizmo from the playing area.
7. Add a ball to the playing area and specify the position and velocity.
8. Save or load game configurations. The user must name the file and they must have the option of saving the file in a standard format or in a special format, which handles special features of the implementation. The saved file must include information about all the gizmos in the playing area, all the connections and the current position and velocity of the ball.
9. Switch to running mode.
10. Quit the Application.

In building mode, an attempt to place a gizmo in such a way that it overlaps a previously placed gizmo or the boundary of the playing area should be rejected.

Running Mode

User can:

1. Run or stop the game.
2. Rotate flipper 90 degrees by pressing a keyboard key.
3. Press keys in which the user can create triggers that may be connected to the action of the gizmos.
4. Switch to building mode at any time, even if the flipper is still in motion. If the user makes this request, the switching will be delayed until the flipper reaches the end of its trajectory. A delay would also apply to a transitioning gizmo.
5. Quit the application.

Gizmoball should:

1. Provide visually smooth animation of the motion of the ball.
2. The ball by default must have a diameter of approximately 0.5L.
3. Ball velocities must range at least from 0.01 L/sec to 200 L/sec. A stationary ball of velocity 0L/sec must also be defined.
4. Provide a frame rate of 20 frames per second to ensure a smooth animation.
5. Ensure that the interaction between the ball and the gizmos is as realistic as possible.
6. Continually modify the velocity of the ball to account for the effects of gravity.
7. Continually modify the velocity of the ball to account for the effects of friction.

Playing Area

To describe dimensions in the playing area, we define L to be the basic distance unit, equal to the edge length of a square bumper.

1. The playing area must be at least 20 L wide by 20 L high. That is, 400 square bumpers could be placed on the playing area without overlapping. The upper left corner is (0, 0) and the lower right corner is (20, 20).
2. The origin of each of the standard gizmos is the upper left-hand corner of its bounding box. The furthest location a gizmo can be placed is (19, 19) on a 20L x 20L board.
3. The origin of a ball is at its centre.
4. During building mode, Gizmos should "snap" to a 1 L by 1 L grid. That is, a user may only place gizmos at locations (0, 0), (0, 1), (0, 2), and so on.
5. During running mode, the animation grid may be no coarser than 0.05 L by 0.05 L.
6. Rotating flippers can be animated somewhat more coarsely.

Standard Gizmos

There are seven standard gizmos that must be supported: bumpers (square, circular, and triangular), flippers (left and right), absorbers and outer walls.

1. **Square Bumper**
    a. A square shape with edge length 1L
    b. Trigger: generated whenever the ball hits it
    c. Action: none required
    d. Coefficient of reflection: 1.0
2. **Circular Bumper**
    a. A circular shape with diameter 1L
    b. Trigger: generated whenever the ball hits it
    c. Action: none required
    d. Coefficient of reflection: 1.0
3. **Triangular Bumper**
    a. A right-triangular shape with sides of length 1L and hypotenuse of length Sqrt(2)L
    b. Trigger: generated whenever the ball hits it
    c. Action: none required
    d. Coefficient of reflection: 1.0
4. **Flipper**
    a. Flippers come in two different varieties, left flippers and right flippers.
    b. A flipper should never extend outside its bounding box or be placed in such a way, which would cause this scenario.
    c. In run-mode, when a flipper is triggered, it sweeps 90 degrees in the predefined direction. If triggered again, it sweeps back to the initial position.
    d. When a flipper's action is triggered, the flipper rotates at a constant angular velocity of 1080 degrees per second to a position 90 degrees away from its starting position. When its action is triggered a second time, the flipper rotates back to its original position at an angular velocity of 1080 degrees per second.
    e. The standard coefficient of reflection for a flipper is 0.95. However, when computing the behaviour of a ball bouncing off the flipper, the linear velocity of the part of the flipper that contacts the ball must be accounted; therefore the ball may leave the flipper with a higher energy than it had when it reached it.
5. **Absorber**
    a. A rectangle with integral-length sides
    b. Trigger: generated whenever the ball hits it
    c. Action: shoots out a stored ball (see below)
    d. Coefficient of reflection: not applicable; the ball is captured
    e. When a ball hits an absorber, the absorber stops the ball and holds it (unmoving) in the bottom right-hand corner of the absorber. The ball's centre is .25L from the bottom of the absorber and .25L from the right side of the absorber.

f. If the absorber is holding a ball, then the action of an absorber, when it is triggered, is to shoot the ball straight upwards in the direction of the top of the playing area. By default, the initial velocity of the ball should be 50L/sec.
g. If the absorber is not holding the ball, or if the previously ejected ball has not yet left the absorber, then the absorber takes no action when it receives a trigger signal.
h. Absorbers cannot be rotated.

6. **Outer Walls**
    a. Impermeable barriers surrounding the playfield.
    b. Trigger: generated whenever the ball hits it
    c. Action: none required
    d. Coefficient of reflection: 1.0
    e. A Gizmoball game supports exactly one set of outer walls. The user cannot move, delete, or rotate the outer walls. The outer walls lie just outside the playing area:
    f. There is one horizontal wall just above the y=0L coordinate.
    g. There is one horizontal wall just below the y=20L coordinate.
    h. There is one vertical wall just to the left of the x=0L coordinate.
    i. There is one vertical wall just to the right of the x=20L coordinate.

File Format:

1. Gizmoball files are command scripts. The files contain not just a list of gizmos and their positions, but rather commands for exercising all the Gizmoball functionality, including placing, deleting, and modifying gizmos.
2. Each line of a Gizmoball file contains a command. Each command consists of an opcode and zero or more arguments; spaces and/or tabs separate these components. For Example:
    a. Triangle T1 0 7 (the program creates a new triangular bumper at the location (0, 7), and that the triangle can later be referred to in the file by the name "T1")
    b. Square S1 10 2 (the program creates a new square bumper at the location (10, 2), and that the triangle can later be referred to in the file by the name "S1")
    c. Circle C1 4 3 (the program creates a new triangular bumper at the location (4, 3), and that the triangle can later be referred to in the file by the name "C1")
3. The “Rotate” command rotates a gizmo by 90-degrees in the clockwise direction. Rotating a flipper does not change its bounding box but does change its pivot.
    a. Rotate T1
4. The “Move” command moves a gizmo to the specified position.
    a. Move S1 19 17
5. The “Delete” command deletes the specified gizmo.
    a. Delete C1
6. There are also commands for connecting triggers to actions. For example, the "Connect" command specifies that the flipper's action should be triggered whenever the ball hits the square bumper:

    a. Square S3 13 13
    b. LeftFlipper LF1 5 18
    c. Connect S3 LF1
7. The KeyConnect command specifies that the action of a gizmo is associated with a particular key being pressed or released:
    a. RightFlipper RF1 9 18
    b. KeyConnect key 32 down RF1
    c. KeyConnect key 32 up RF1
8. Outer walls can also trigger various actions; there is a special identifier reserved for it (This command would cause the ball hitting any of the outer walls trigger the action of the gizmo named by "GIZ")
    a. Connect OuterWalls GIZ
9. The Ball command specifies the position and velocity of the ball. Because the ball can be at intermediate points within a particular square, the coordinates are specified as floating point numbers:
    a. Ball B1 14.2 4.5 -3.4 -2.3 (This places a ball with its center at (14.2, 4.5) and an initial velocity of 3.4L per second to the left and 2.3L per second upward.)
10. The Gravity and Friction commands can be used to set global properties of the game. Each takes floating point values as arguments such as:
    a. Gravity 16.0 (reduces the gravity in the game to only 16L/sec^2)
    b. Friction 0.0 0.0 (remove any effects of friction)

Use Cases

Name: Run gizmoball

Preconditions:

Play view is open.

Triggers:

User clicks the Run button.

Basic Course of Events:

1. Ball starts moving with the previously specified speed and direction.
2. Flippers can be flipped if the user pressed the predefined key.
3. Loop until user clicks Stop.

Postconditions:

Ball is reset into original position.

Name: Place a gizmo

Preconditions:

Build view is open

Triggers:

User clicked on a gizmo on the build menu

Basic Course of Events:

1. User moves mouse over to where they want to place gizmo. Gizmo snaps to tile under mouse as it moves.

2. User clicks and gizmo snaps to the tile under mouse.

3. Menu pops up that allows user to edit (using drop down menus) the action, trigger and colour of the gizmo depending on what type of gizmo it is.

4. User clicks okay.

5. Menu closes and gizmo added to tile.

Alternative Course of Events:

2. User clicks on tile that already contains a gizmo.

3. Alert is shown that this is an invalid gizmo placement.

4. Go to step 1 in main path.

2. User clicks a button on the build menu.

3. Building the gizmo is cancelled.

4. User clicks cancel

5. Building the gizmo is cancelled.

4. User clicks close.

5. Gizmo is added to tile with the default colour and without any action/trigger information.

Postconditions:

A new gizmo is now in the position the user placed it in.

Gizmo is correct colour.

Triggers and Actions are set up if user set them up.

Name: Edit a gizmo

Preconditions:

Build view is open

There exists one gizmo on the board.

Triggers:

User clicks on edit button and the gizmo in either order OR User right clicks on gizmo.

Basic Course of Events:

1. Menu pops up that allows user to edit the trigger, action, position (in coords) or colour of the gizmo.

2. User makes edits and clicks OK.

3. Menu closes and edits are made.

Alternative Course of Events:

2. User clicks Cancel.

3. Menu closes.

2. User makes edits that are not possible (ie negative coords).

3. Alert is shown describing the issue.

4. Got to step 1 in main path.

Postconditions:

Selected Gizmo is edited according to the user's edits.

Name: Move a gizmo.

Preconditions:

Build view is open

There exists one gizmo on the board.

Triggers:

User starts to click and drag gizmo

Basic Course of Events:

1. User clicks and drags gizmo in to position.
2. Gizmo snaps to the tile under mouse.
3. User releases mouse click.
4. Gizmo is saved in new position.

Alternative Course of Events:

3. User releases mouse click over invalid position.
4. User is shown alert explaining the issue.
5. Gizmo snaps back to original position.

1. User edits position via coordinates. Follow the 'Edit a gizmo' use case.

Postconditions:

Gizmo is in new position.

Name: Rotate a gizmo

Preconditions:

Build view is open

There exists one gizmo on the board.

Triggers:

User clicks on rotate button and the gizmo in either order

Basic Course of Events:

1. Selected gizmo rotates 90 degrees clockwise.

Postconditions:

Selected gizmo is rotated

Name: Set key for flipper trigger

Preconditions:

Build view is open

There exists one flipper on the board.

Triggers:

User clicks on edit button and the flipper in either order OR User right clicks on the flipper.

Basic Course of Events:

1. Menu pops up that allows user to edit the trigger, action, position (in coords) or colour of the gizmo.
2. User selects keyboard under the trigger field.
3. User is advised to press the key they wish to use as the trigger
4. User presses key.
5. User clicks OK.
6. Menu closes and edits are made.

Alternative Course of Events:

2. User clicks Cancel.
3. Menu closes.

4. User presses ESC to cancel changing the trigger
5. Trigger is set back to original trigger.
6. Go to step 2 in main path.

Name: Save Game

Preconditions:

Game is not running.

Triggers:

User clicks File > Save

Basic Course of Events:

1. Save dialogue is displayed.
2. User chooses where to save file and clicks Save
3. File is written to disk and save dialogue closes.

Alternative Course of Events:

2. User clicks cancel
3. Save dialogue closes.

Postconditions:

File is written to disk.

Name: Load Game

Preconditions:

Game is not running

Triggers:

User clicks File > Load

Basic Course of Events:

1. File chooser dialogue is displayed.
2. User chooses which file to load and clicks Load.
3. File chooser is closed and file is loaded into application and tiles are generated.

Alternative Course of Events:

2. User chooses a non-gizmoball file and clicks Load.
3. File chooser is closed and alert is shown saying that this is not a gizmoball file.
4. If the user clicks Retry, alert is closed and go to step 1 in main path. If the user clicks cancel, alert is closed.

2. User clicks cancel.
3. File chooser is closed.

Postconditions:

Tiles are generated on the board according to the save file.

Name: Reset board

Preconditions:

Build view is open.

Triggers:

User clicks File > 'Reset Board'

Basic Course of Events:

1. If board is not saved
2. 'Are you sure?' alert is shown.
3. User clicks Yes.
4. Alert closes.
5. All gizmos are removed from the board.

Alternative Course of Events:

3. User clicks no
4. Alert closes.
5. All remains how it was originally.

Postconditions:

Board is cleared of all gizmos.

Name: Quit game

Triggers:

User clicks File > Quit

Basic Course of Events:

1. If board is not saved
2. 'Are you sure?' alert is shown.
3. User clicks Yes.
4. Alert closes.
5. Application closes.

Alternative Course of Events:

3. User clicks no.
4. Alert closes.
5. All remains how it was originally.

Postconditions:

Application closes.

Name: Connect two gizmos

Preconditions:

Build view is open.

There exists two gizmos.

Triggers:

User clicks the connect button and then two gizmos.

User clicks on edit button and a gizmo in either order OR User right clicks on a gizmo.

Basic Course of Events:

1. User clicks the connect button.
2. User clicks the first gizmo
3. User moves mouse to next gizmo while connection graphic is displayed between the first gizmo and the mouse.
4. User clicks the second gizmo.
5. Gizmos highlight for a moment then connection graphic disappears.

Alternative Course of Events:

1. User clicks on edit button and then the first gizmo OR Use right clicks on the first gizmo.
2. Menu pops up that allows user to edit the trigger, action, position (in coords) or colour of the gizmo.
3. User selects 'Trigger Another Gizmo' in the action drop down list.
4. Menu closes, graphic displays 'Choose gizmo'.
5. Go to step 3 in main path.

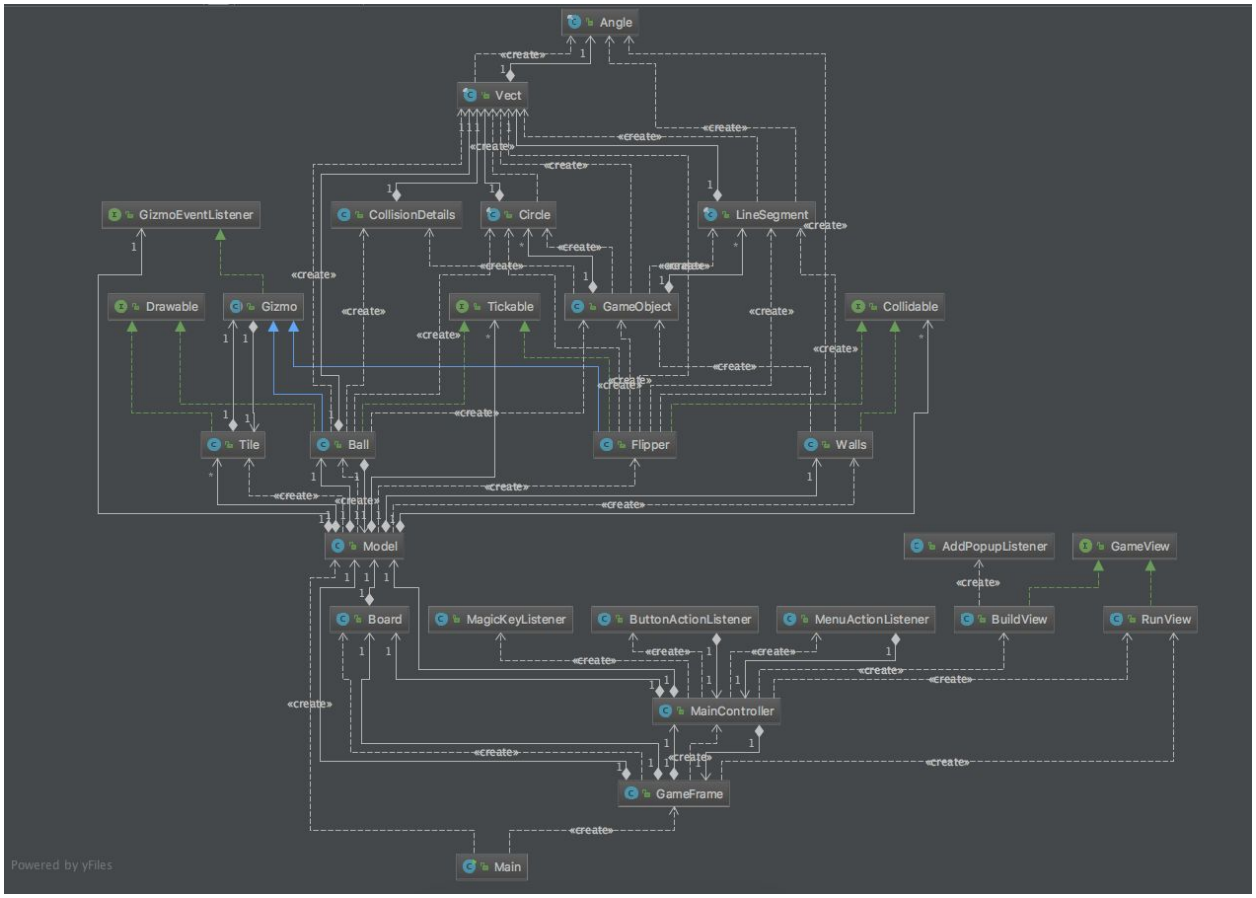
4. User right clicks anywhere on the board.
5. Connection graphic disappears..

Postconditions:

When the first gizmo is triggered, it triggers the second gizmo.

Class Diagram

Below is a graphical representation of our class hierarchy. For our system we are using the Model View Controller object oriented design pattern to ensure the encapsulation of the various subsystems. All Swing elements are contained within the View package and employed to build our applications GUI. The main highlight of which is the GameView class, an interface from which RunView and BuildView. As for the Controller package, this consists of a MainController class that can call on MenuActionListener, MagicKeyListener and ButtonActionListener allowing us to decouple ActionListeners from their corresponding View classes and streamline the method call process. Our Model contains logical representation of the elements of GizmoBall such as Wall, Flipper, Tile. It communicates with the Controller and the View via the use of interfaces such as GizmoEventListener and Drawable. The Model provides the functional core of the application and notifies dependent components about data changes using the Observer design pattern.



Physics Loop

1. Initialise the tick time for the ball
2. Initialise the Collision Details cd to timeUntilCollision()
3. Initialise timeUntilCollision(tc) to cd.getTuc()
4. Loop for all Gizmos and check every line and corner of every Gizmo
5. Trigger activates
6. Check for external forces(gravity and friction)
7. If the minimum timeUntilCollision > than the “tick time” then
8. No collision occurs, so move the ball for tick time then
9. redraw
10. Else
11. If the minimum timeUntilCollision <= than the “tick time” then
12. Collision occurs, so move the ball for timeUntilCollision then
13. redraw
14. Set the ball to its velocity after the collision
15. Notify observers and redraw updated view

Triggering System Description

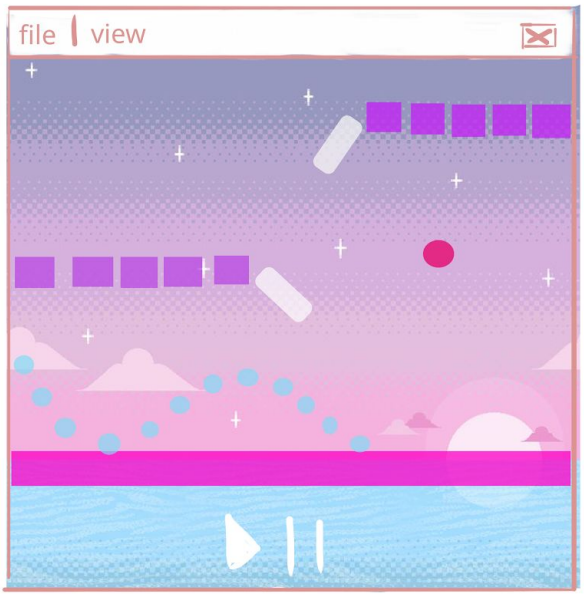
1. Gizmos can be triggered by gizmo actions or key presses (including key up and key down)
2. Action-trigger connections can be unbound
3. Action-trigger connections can be rebound, overwriting the old one with a new one
4. A gizmos action can be associated with many triggers and vice versa
5. Mouse events are not allowed to be assigned as triggers to gizmos
6. Users connect the actions and triggers of gizmos only in build mode
7. Connections can also be established when a gizmoball file is loaded, provided these connections are specified in the file and are in the correct format
8. Connections specified in a gizmoball file must adhere to the same rules enforced as when a user specifies them
9. Gizmoball files that assign an action to a mouse event will fail to be read
10. A gizmo’s action can be activated by its own trigger
11. If a trigger occurs midway through a tick, it’s corresponding action occurs during the following tick.
12. During a tick, if one action is triggered multiple times, it is activated only once.

GUI Sketches and Screenshots

Sketches

Sketches representing ideal view are seen below, these include the Gizmoball build mode, run mode and the shape editor dialogue.

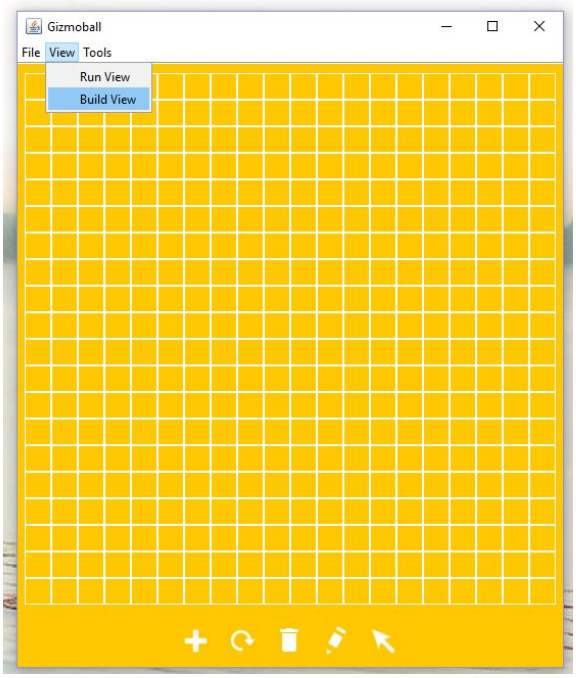
Building View

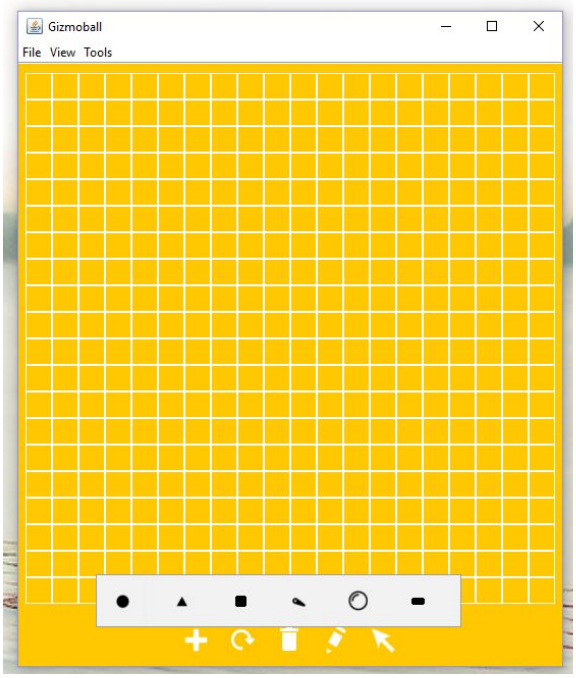


Running View

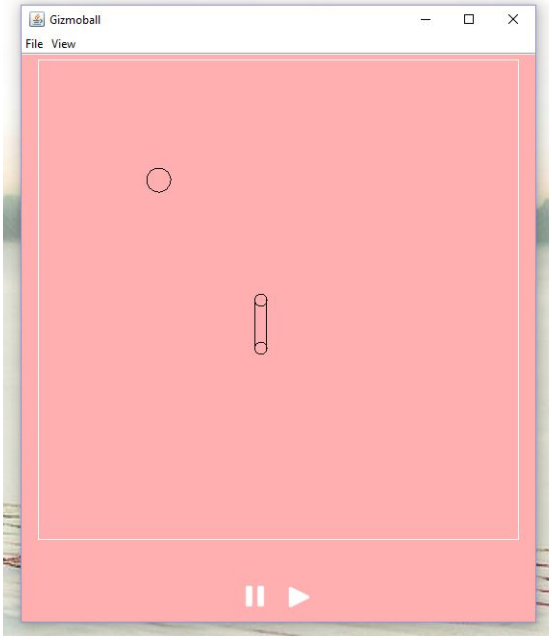
Screenshots

Screenshots of the view as it currently is are show below and include the Gizmoball build mode, run mode and the shape editor dialogue.

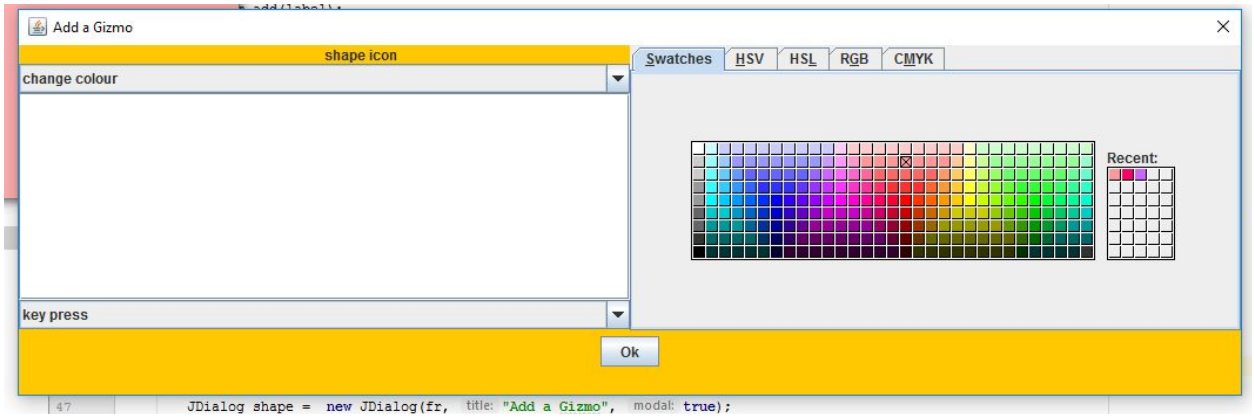




Build mode screenshots

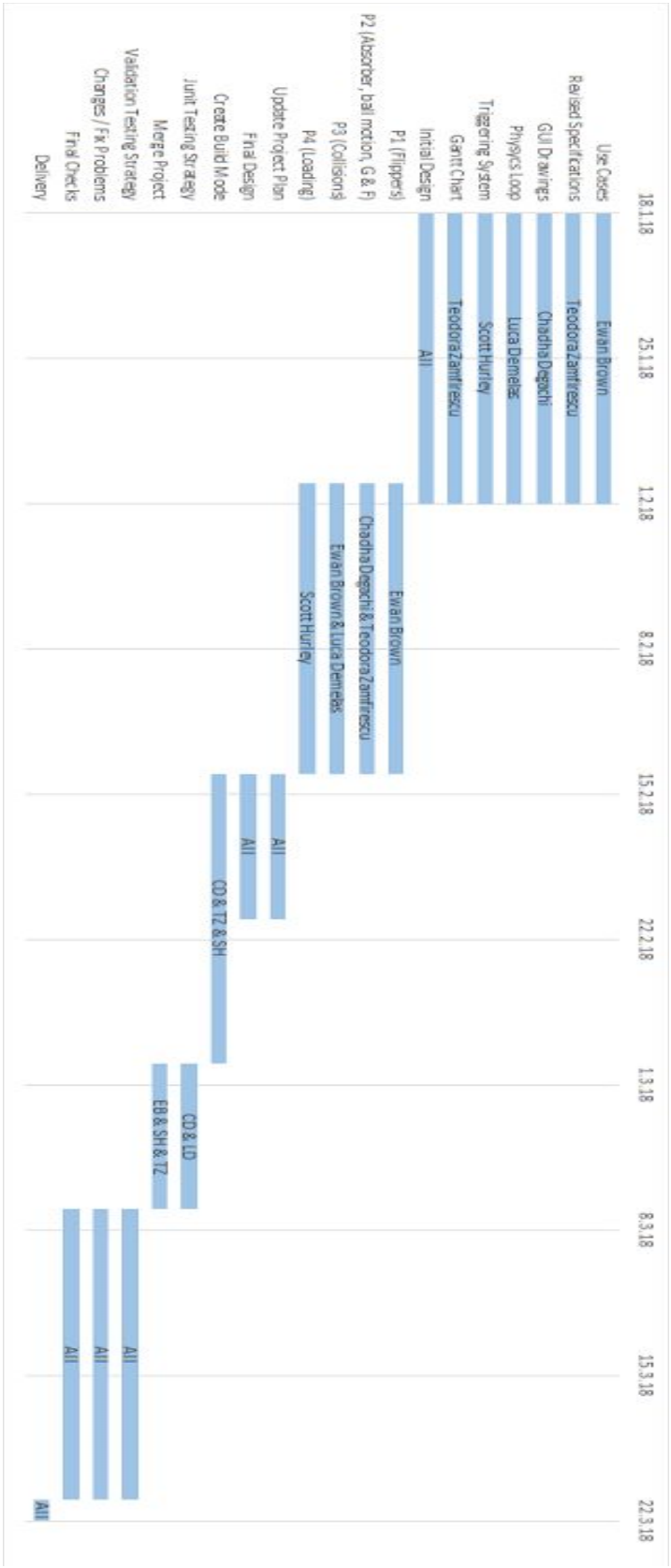


Running mode screenshot



Edit a Gizmo Dialogue

Gantt Chart


Use Cases
Revised Specifications
GUI Drawings
Physics Loop
Triggering System
Gantt Chart
Initial Design
P1 (Flippers)
P2 (Absorber, ball motion, G & F)
P3 (Collisions)
P4 (Loading)
Update Project Plan
Final Design
Create Build Mode
Junit Testing Strategy
Merge Project
Validation Testing Strategy
Changes / Fix Problems
Final Checks
Delivery
Ewan Brown
Teodora Zamfirescu
Chadha Degachi
Luca Demetas
Scott Hurley
Teodora Zamfirescu
All
Ewan Brown
Chadha Degachi & Teodora Zamfirescu
Ewan Brown & Luca Demetas
Scott Hurley
All
All
CD & TZ & SH
CD & LD
EB & SH & TZ
All
All
All
All
18.1.18
25.1.18
1.2.18
8.2.18
15.2.18
22.2.18
1.3.18
8.3.18
15.3.18
22.3.18