Revised Specification

The game will have two modes:

## Build Mode:

* All the basic Gizmos: Square, Circle, Triangle, Left Flipper, Right Flipper.
* Being able to rotate gizmos. Clicking rotates the shape once by 90 degrees.
* When a user tries to place a gizmo on top of another one, an error is displayed.
* Connection functionality between the gizmos:

When a gizmo is connected to another it means that when the ball hits the first gizmo, it activates the second gizmo as well, as if it it had hit them both at the same time.

There is a noticeable edge case: If a flipper and a gizmo are connected and the connection is activated while the flipper is in motion e.g. because the user pressed a button, the interaction will be ignored i.e. it will not override the flipper’s motion.

* When a gizmo shape (Square, Circle, Triangle) is connected to another shape, the default action is a change of colour (for the Absorber this is the only possible action).
  + An add-on functionality would be to ask the user what action they want between:
    - Changing the colour
    - Changing the shape of the gizmo randomly or in a sequence or between chosen shapes.

We have to make sure that there aren't any problems when for example a shape changes from a circle to a square and suddenly there are corners where there previously weren't. We should check that the ball would not collide with the newly created matter. Should it collide, then the system waits until the ball leaves the area of collision before changing the shape.

* We should show a grid.
* The ability to remove gizmos and connections between them. We could add a button that clears the whole board as well.
* As an add-on to the core functionality a user will be able to edit the gravity and friction coefficients.

## Run mode:

### Gravity:

The assignment uses L/s (lengths per second) to define speed, a length being the size of 1 grid square. Default gravity should be 25 Grid squares per second (squared).

### Friction:

In the Physics code given to us, the way collisions are calculated include a reflection coefficient . A reflection coefficient of 1 means that the ball doesn't lose speed upon bouncing off of the object. A coefficient of 0.5 means the ball loses half of its speed upon collision. This is how friction will be handled upon collision. If the reflection coefficient is bigger than 1 then the ball acquires speed upon rebounding with a gizmo.

* Airborne friction will be handled by scaling the velocity of the ball using the frictional constants *mu* and *mu*2. “For sufficiently small *delta\_t*'s you can model friction as *Vnew* = *Vold* \* (1 - *mu* \* *delta\_t* - *mu*2 \* |*Vold*| \* *delta\_t*).
  + The default value of *mu* should be 0.025 per second.
  + The default value of *mu2* should be 0.025 per L.”

### Other:

* We need to have a stepper function (tick) during run mode. There are 20 ticks a second so each tick lasts for 50 ms.
* The play area has a minimum requirement of 20x20 lengths (L). By these constraints, the minimum canvas size is 20x20 pixels where the size of a square bumper is 1x1 pixels.
* Import and save from files (just the grid and not the state of a running game.)
* We should display the ball speed during run mode. When the ball is launched its speed is 50 L/s by default. The maximum speed is 200 and the minimum is -200.
* In the Model, we have to make sure to model the gizmos as lines and circles. **Make sure that there are zero radius circles at the end of lines.**
* Make sure to make JUnit tests for the Model (hopefully the Controller and View as well).