**Physics loop**

**Description**

The movement of the ball is driven by the timer. The motion is simulated by updating the ball’s coordinates at a timer rate that is referred to as *tick time* with consequent redrawing of the board to reflect a change in the ball’s position. The tick time rate is chosen to be 20 times per second (50 milliseconds). The physics loop is invoked at every tick time. It consists of the following actions:

**if** event = timer **then**

1. Collision detection

For each physical segment calculate the time until collision:

**if** time until collision < shortest time **then**

set shortest time to time until collision;

update velocity;

1. Update position

**if** time until collision > tick time **then**

update of the ball’s position using tick time;

**else**

update of the ball’s position using time until collision;

update the velocity vector to change the direction/speed of the ball;

1. Trigger action

**if** trigger is present **then**

iterate through gizmo collection;

invoke trigger action on each;

1. Handle flipper

**if** flipper is active **then**

move flipper;

1. Apply forces:

apply gravity;

apply friction;

1. Notify observers and redraw;

First, any change in the ball’s position is determined by its velocity. Velocity is a vector with two components: a direction of the ball’s movement in the two dimensions and its speed (or magnitude). The initial velocity gives the ball a momentum for its motion. The velocity vector is then affected by the forces of gravity and friction as well as collisions, which change the direction and potentially the speed of the ball at every tick time.

Second, the changes in position also depend on whether or not there is an obstacle within the *tick* *time* which signifies an impending collision. Tick time serves as a reference time interval to determine the proximity of other objects with which the ball may collide.

1. **Collision Detection**

Collision detection mechanism relies on calculating the time until collision by iteratively comparing the times it takes the ball to collide with all of the existing segments on the plane and selecting the minimum value (*mintime*). The *mintime* is then compared to the tick time to determine whether or not a collision will occur within the tick time. At this stage, the change in time or ∆t is determined. If multiple collisions are detected within the tick time, then the board is redrawn and the physics loop is restarted.

1. **Updating the position**

The distance travelled by the ball (S) per tick time is determined by the velocity vector (V) and time (T):

*S = V \* T (1)*

The time (T) refers to ∆t that take the value of either the *tick time* (case 1) or *time until collision* (case 2). The position of the ball is updated at each tick time by adding the distance travelled (s) to the current X and Y coordinates.

**Case 1: Unhindered motion**

If the mintime is greater than the tick time, then no collision occurs within the tick interval and the tick time is taken to calculate the distance travelled (S).

**Case 2: Collision resolution**

If the mintime is less than the tick time, then there is an impending collision. Collisions induce the change in the ball’s velocity and position that are accounted for by implementing the following steps:

1. the ball’s position is adjusted to be adjacent to the surface of the object it is about to collide with by applying the formula (1), where time (T) is *time until collision*;
2. the velocity vector of the ball is updated to emulate the change in direction (by inverting either x or y components) and change in speed (depending on an appropriate coefficient of reflection);
3. a trigger action is invoked if a trigger is attached to the gizmo;
4. if activated, synchronise flipper(s) with the ball’s movement.
5. if multiple collisions are detected within tick time, redraw the board and restart the physics loop.
6. **Triggering**

When a collision is registered and the ball is adjacent to a line segment or circle that is a part of the gizmo, a trigger action is invoked on that gizmo objects if one is defined.

1. **Flippers**

Once *∆t* is determined, check is flippers are triggered. Then use the value of *∆t* to update the position of the flipper(s).

1. **Gravity and Friction**

Collision detection and handling mechanisms are independent from the effects of gravity and friction on the ball. Gravity and friction are factored in after the collision sequence as separate methods to make velocity consistent during the collision detection and handling. Both methods take ∆t as a parameter.

Gravity increments the magnitude of Y component of the velocity vector at each tick time to imitate the constant acceleration of the ball downwards. Given that the gravity is 25L/sec2 as per specifications, the Y component of velocity is incremented by 1.25L per ∆t.

Friction decrements the magnitudes of X and Y components of the velocity vector per ∆t to imitate the resistive force of the surface. The velocity of the moving ball is scaled using the two frictional constants µ and µ1 with the values of 0.025 per second and 0.025 per L respectively in the equation (2).

*Vnew = Vold \* (1 – µ\*∆t - µ1\* |Vold| \* ∆t)* (2)

1. **Repainting**

The view is redrawn at the tick time by notifying the model through notifyObserver() method to reflect the change in position of the ball. Repainting normally occurs at the end of physics, unless there are multiple collisions within a single tick time.