CARPENTIER Simon, CARPENTIER François TD2 TPD

STEP 3

The algorithm has an update function that updates the position and bounces of each ball. However, it updates balls that are not in motion, which is unnecessary.

```python
def update(ball, deltaTime):
    ball["x"] += deltaTime * ball["dx"]
    ball["y"] += deltaTime * ball["dy"]

    dx = ball["dx"]
    dy = ball["dy"]
    l = math.sqrt(dx ** 2 + dy ** 2)
    if l > constants.EPSILON_SPEED:
        dx = dx / l * constants.FRICTION
        dy = dy / l * constants.FRICTION

        ball["dx"] -= dx * deltaTime
        ball["dy"] -= dy * deltaTime
    else:
        ball["dx"] = 0
        ball["dy"] = 0


    # bouncing
    if ball["x"] < constants.BALL_RADIUS:
        ball["x"] =  constants.BALL_RADIUS
        ball["dx"] = -ball["dx"] * constants.DAMP_COEF
        ball["dy"] *= constants.DAMP_COEF
    elif ball["x"] > constants.LAYOUT_PLAY_AREA_WIDTH - constants.BALL_RADIUS:
        ball["x"] = constants.LAYOUT_PLAY_AREA_WIDTH - constants.BALL_RADIUS
        ball["dx"] = -ball["dx"] * constants.DAMP_COEF
        ball["dy"] *= constants.DAMP_COEF

    if ball["y"] < constants.BALL_RADIUS:
```

The goal is to try to do the update only on moving balls, so we will use a solution that has been used in step one.

```python
def update(ball, deltaTime):
    p = ball
    if ((p["dx"] != 0) or (p["dy"] != 0)):
        ball["x"] += deltaTime * ball["dx"]
        ball["y"] += deltaTime * ball["dy"]

        dx = ball["dx"]
        dy = ball["dy"]
        l = math.sqrt(dx ** 2 + dy ** 2)
        if l > constants.EPSILON_SPEED:
            dx = dx / l * constants.FRICTION
            dy = dy / l * constants.FRICTION

            ball["dx"] -= dx * deltaTime
            ball["dy"] -= dy * deltaTime
        else:
            ball["dx"] = 0
            ball["dy"] = 0


        # bouncing
        if ball["x"] < constants.BALL_RADIUS:
            ball["x"] =  constants.BALL_RADIUS
            ball["dx"] = -ball["dx"] * constants.DAMP_COEF
            ball["dy"] *= constants.DAMP_COEF
        elif ball["x"] > constants.LAYOUT_PLAY_AREA_WIDTH - constants.BALL_RADIUS:
            ball["x"] = constants.LAYOUT_PLAY_AREA_WIDTH - constants.BALL_RADIUS
            ball["dx"] = -ball["dx"] * constants.DAMP_COEF
            ball["dy"] *= constants.DAMP_COEF
```

Using the direction of the balls, we look at which balls are in motion, and which balls we will update.