Moving Circle vs Static Line – Pseudo Code

Introduction:

LNS is a line segment with end points **P0** and **P1** and outward normal $\hat{\boldsymbol{N}}$.

Circle is centered by **B** and has radius **R**. It is moving with velocity $\vec{V}$ per one frame.

$\mathbf{B_s}$ is the starting position of **B**, $\mathbf{B_e}$ is the end position of **B**, $\mathbf{B_i}$ is the intersection position of **B** (if any collision).

Problem:

Detect the collision time, the collision position of Circle, and the reflected position after bouncing of LNS.

Solution:

The following is a pseudo-code.

We need to distinguish between 2 cases:

1 - The Circle might hit the body of LNS first (2 sub-cases = 2 sides (normal's side and opposite normal's side))

2 - The circle might hit one of the edges (end points) of LNS.

MovingCircleVsStaticLine()

```
{           //N is normalized
if(N̂.Bs - N̂.P0 <= -R) //Here we consider we have an imaginary line LNS1, distant by –R (opposite N̂ direction)
        //Check if the velocity vector V⃗ is within the end points of LNS1
        //M⃗ is the outward normal to Velocity V⃗. Compute P0' and P1'
        P0' = P0 – R*N̂   and   P1' = P1 – R*N̂       //To simulate LNS1 line edge points
        if(M⃗.BsP0' * M⃗.BsP1' < 0)
                Ti = (N̂.P0 - N̂.Bs - R) / (N̂.V⃗)  //We are sure N̂.V⃗ != 0
                if(0 <= Ti <= 1)
                        Bi = Bs + V⃗*(Ti)
                        B'e = ApplyReflection(-N̂, BiBe)  //Normal of reflection is -N̂
        else
                CheckMovingCircleToLineEdge(false)
else if(N̂.Bs - N̂.P0 >= R) //Here we consider we have an imaginary line LNS2 distant by +R (Same N̂ direction)
        //Check if the velocity vector V⃗ is within the end points of LNS2
        //M⃗ is the outward normal to Velocity V⃗. Compute P0' and P1'
        P0' = P0 + R*N̂   and   P1' = P1 + R*N̂       //To simulate LNS2 line edge points
        if(M⃗.BsP0' * M⃗.BsP1' < 0)
                Ti = (N̂.P0 - N̂.Bs + R) / (N̂.V⃗)  //We are sure N̂.V⃗ != 0
                if(0 <= Ti <= 1)
                        Bi = Bs + V⃗*(Ti)
                        B'e = ApplyReflection(N̂, BiBe)  //Normal of reflection is N̂
        else
                CheckMovingCircleToLineEdge(false)
else //The circle's starting position Bs, is between both lines LNS1 and LNS2.
        CheckMovingCircleToLineEdge(true)
}
```

Point2D ApplyReflection(Vector2D normal, Vector2D penetration)

{

return $\mathbf{B_i}$ + **penetration** - 2(**penetration** . **normal**) * **normal**;

}