A1: Moving Circle

2017313135 소프트웨어학과 권동민

Data Structure

First, I added more properties and functions into each circle structure

Init Velocity : the original velocity before processing frame and time(for equal speed over computers)

Velocity : this value will be changed in every frames (proportional to time)

Functions are related to physical actions of circles (initial position, wall collision, circle collision, direction and speed after collision)

I used "vector" data structure for processing every circles, in render function, all the circles in this circle vector are updated.

Algorithm

Initial Position of Circles

: when creating circles, each circle gets their own position, velocity, color. But if more than one circle are overlapped, we have to change the position of that circle. So before enqueuing each circle into vector of circles, I compared the circle's position between previous circles(already in the circle vector) and if there is collision, I create new circle until there is not collision with any previous circles. The standard of collision detection is as fallows.

$$radius_a + radius_b < distance_{between\ apos\ and\ bpos}$$

Synchronization circles' movement speeds across different computer

: Let's assume that it took 1 second to run 1frame on computer A and 2 second to run 1frame on computer B, if velocity(per frame) is equal in A and B, in terms of time, users feel that the speeds of circle are different.(in this case, A's speed is two times faster than B). That is, the speed should increase in proportion to time.

So I calculated the execution time and multiply with initial velocity.

$$velocity * (t - bt) * c \ (\ c\ is\ constant\ value)$$

Collision with the wall

: it's very simple. If circle's radius is bigger than the distance between wall and position, reverse the direction of circle(if it's right or left wall, changing velocity-x, otherwise, changing velocity-y)

Collision between circles

: First, we have to detect collision between circles. It is same with setting initial positions. And I used elastic collision for calculating velocity after collision.

$$v'_1 = \frac{(m_1 - m_2)v_1 + 2m_2 v_2}{m_1 + m_2},\ v'_2 = \frac{(m_2 - m_1)v_2 + 2m_1 v_1}{m_1 + m_2}$$

$$or$$

$$v'_1 = v_1,\ v'_2 = v_2$$

But for calculating with above formula, we have to divide velocity vectors into two direction. (x-axis, y-axis for changing them to 1-dimension) but in this case, we have to rotate x-axis and y-axis.(the direction of x-axis is same with line between circle's position) it is same with velocity vector rotation leaving the axes. So first we rotate angle of two velocity vectors by theta and calculate new velocities. Next, we have to reverse that vectors into original axes(rotate by minus theta). I uses go matrix and back matrix for rotating.

$$go = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \qquad back = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

Discussion

In collision between circles, What is the appropriate setting for the mass? If we set the mass to be proportional to the area of the circle(πr^2), the difference becomes too large. It is same although multiplying constant value. So I just set the mass to be proportional to circle's radius in this case.