A1 - Moving Circles

2016312160 Lee Jun Sung

Circle.h

1. Circle struct : velocity – vector of x difference and y difference.

   m – mass of circle

   checkBoundary() – function that checks collapse between wall and circle
2. checkC() : function which checks collapse and overlaps between walls and other circles.
3. updateRadius() : function which update radius whenever circle is added or deleted.
4. Create_circle() : create and initialize circle.
5. Delete_circle() : delete circle
6. Update() : calls checkBoundary() and moves center of circle based on velocity.

Main.cpp

1. Render() – compare each circle with every circles and update
2. Main() – update and render based on time difference of frame.

Initialization

1. Window size is initialized to 800*450 which is 16/9 aspect ratio. And change the aspect-matrix by ratio of 16/9 which can always maintain 16/9 whenever change the size by mouse.
2. When start program it starts with 30 solid circles which use random radiations, colors(R,G,B), locations and velocities. Random values are created by rand() function.
3. The movements of circles is based on glfwGetTime() function. In main function, when update() and render() functions are called, the circles move. By calling those functions at every fixed time (about 0.015f which is 60fps) which is the time difference between the current and previous frame, it ensures consistent animation speed.
4. Whenever circles are created, checkC() function is called which checks collapse and overlaps between walls and other circles. Also, to avoid infinite looping, update all circle's radius at

the ratio of 1/sqrt(N).

Collision detection

1. checkBoundary() : whenever this function called, it checks if circle overlaps wall. If overlaps, it reverse the velocity opposite the way it collide the wall and moves circles back by the amount it overlaps.

2. When two circles are collapsed or overlapped, reverse the circle's position. By using difference of center positions find vector and normalize it. Until it moves away from other circle, move that circle slowly.

3. By using Wikipedia function which is about physically based elastic collision we can get after-velocity by using before-velocity.

$$\mathbf{v}_1' = \mathbf{v}_1 - \frac{2m_2}{m_1+m_2}\,\frac{\langle \mathbf{v}_1-\mathbf{v}_2,\,\mathbf{x}_1-\mathbf{x}_2\rangle}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}\,(\mathbf{x}_1-\mathbf{x}_2),$$
$$\mathbf{v}_2' = \mathbf{v}_2 - \frac{2m_1}{m_1+m_2}\,\frac{\langle \mathbf{v}_2-\mathbf{v}_1,\,\mathbf{x}_2-\mathbf{x}_1\rangle}{\|\mathbf{x}_2-\mathbf{x}_1\|^2}\,(\mathbf{x}_2-\mathbf{x}_1)$$

C1,2 = circle 1,2 V1,2 = circle1,2 before velocity
V'1,'2 = circle1,2 after velocity X1,2 = circle1,2 center m1,2 = mass of circle1,2

4. To better collision detect, I use finer steps of frame by moving circles slowly and calling update function several times (Moving 100times slowly and update 100times). And use real number measurement to detect collision.