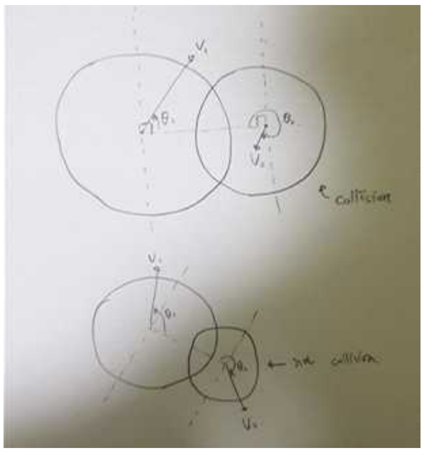
A1 report

2017313107 lee seung tae

1. algorithm

(1) To make initial positions of circles avoid overlaps and collisions with the other circles and walls, I adjust the radius proportional to 1/sqrt(circle number).

(2) In the last requirement I use this following algorithm. Here is the picture to help your understand.



Let's say the location of each center of the circle is x1, x2. And you can create the vector from x1 to x2. Let's say this vector is $\overrightarrow{x}$. And the angle of between $\overrightarrow{x}$ and $\overrightarrow{v_1}$ is θ_1, $\overrightarrow{x}$ and $\overrightarrow{v_2}$ is θ_2. And if both angles are greater than 90 degrees and less than 270 degree, we can determine that the collision did not occur. In other case, we can say collision occur. And we can avoid trembling.

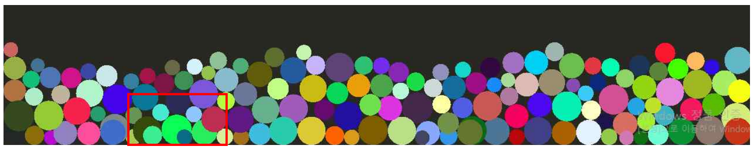
2. other options

(0) When you press ‘r’ reset the circles. (Add for convenience)

(1) First I get an idea on the book of shader.
When you press ‘z’ you can see red component of the circles.
When you press ‘x’ you can see green component of the circles.
When you press ‘c’ you can see blue component of the circles.
When you press ‘v’ you can see gently changing color of the circles.
And if you press these button again, you can see original color of the circles.

(2) Next, I implement a gravity.
When you press ’w‘ you can apply up-gravity.
When you press ’a‘ you can apply left-gravity.
When you press ’s‘ you can apply down-gravity.
When you press ’d‘ you can apply right-gravity.
And if you press these button again, you can see the origin motion(randomly

moving circle)
When gravity works, I implied Coefficient of restitution (0.8). And circles speed are gradually decreases. But When I implied this option I saw a little problem arisen. If I wait and see, the circles overlap. Like this picture.



To avoid this problem, I use many solutions. Among them, when a circle is very slow(maybe stop), the Coefficient of restitution is set to 1. Then circles do not overlap over time.
And I got some ideas to YouTube video.

(3) When you press the ctrl + left_mouse, then the circle at that point becomes smaller. Similarly, the when you press the ctrl + right_mouse, then the circle at that point becomes bigger. And you press the shift + left_mouse, then the circle at that point becomes faster (x2).
 I implemented some function and changed aspect matrix. And it works well even if the window size changes.
 For this implement I understood aspect ratio, how points are mapped to screen and window size.

(4) Finally I implemented gravity that follows the mouse. When you press and drag left mouse, then balls are follow the mouse. As with the (2) above, the same error occurred. I solved it in a similar way (2) (Coefficient of restitution (0.95)). And It can be applied at one screen as shown in (2).
 If you continue to press and hold the mouse, the speed of the circle is close to zero. And you press the shift + left_mouse like (3), then the circle at that point becomes faster (x2). Therefore, we can go back to the previous state. (random move)
And I referred to this website. I learned about how to use cursor.
https://www.glfw.org/docs/3.3/input_guide.html

3. Discussion
 This is my first class and first assignment of computer graphics. But it is very interesting subject. I've been looking at my circles for a long time after finishing my assignment. There was a sense of accomplishment. I wanted to put in more implementations, but I couldn't think of anything more.