Computer Graphics: Assignment#1

소프트웨어학과

2018310520

김세란

Data structure

2차원에 존재하는 원 c에 대해서 물리적 상태를 포함하는 객체를 struct로 설정하였다.

원 c는 우선 랜덤한 값의 색깔과 일정 범위 내에서 랜덤한 값의 반지름을 가진다.

또한 물리량으로는 위치 벡터(center), 속도 벡터(velocity)를 가지며, 충돌에 의한 속도 변화량을 저장하기 위해 속도 변화량 벡터(delta_v)를 선언하였다.

Algorithm

한 frame 당 실행되는 알고리즘은 다음과 같다.

0. delta_t를 구한다.

   delta_t는 프레임 간 시간(시간 간격)이다.

1. Delta_t에 따른 circle들의 position변화를 적용한 후 화면에 그린다.

   Position += delta_t*velocity

2. circle들의 Velocity를 구한다.

   2.1 충돌에 의한 속도 변화량 delta_v를 구한다.

   2.2 velocity += delta_v

2.1 에서 충돌에 따른 속도 변화량, delta_v를 구하는 방법은 다음과 같다.

충돌은 원과 원의 충돌, 벽과 원의 충돌이 있으며, 충돌이라고 판정하는 방식은 다음과 같다.

① 현재 위치에서 원이 충돌했다. ② 겹치는 정도(음수 가능)가 이전 위치 보다 증가했다.

(*②를 보면, 겹치는 정도가 증가하지 않는다면, 충돌이 아니며 속도 변화량에 영향을 주지 않는다.)

충돌이라고 판정되었다면, 충돌에 따른 속도 변화량을 구할 수 있다. 이때 사용한 식은 2차원에서의 탄성 충돌에 적용되는 식과 같다. 구현은 vec2자료형을 이용하여 아래의 벡터 연산을 구현하였다.

Advanced Features

운동하는 원을 클릭할 때, 해당 원이 정지하고 마치 벽처럼 질량이 다른 원에 비해 매우 커서 질량 비를 무시할 수 있게끔 처리하였다. 따라서 한 번 클릭이 된 원의 경우 절대 움직이지 않으며, 다른 원들은 평소와 같이 운동을 진행한다.

Discussion

처음에 원의 충돌까지 완성을 하고 코드를 실행했을 때, 운동에너지가 보존되지 못하고 시간이 지남에 따라 기하급수적으로 전체 원들의 속도가 빨라지는 일이 발생하였다.

이는 분명 충돌에 따른 원의 속도 변화를 구하는 부분에 문제가 있는 것인데, 물리 법칙을 올바르게 적용했는데 문제가 발생하는 것은 float의 정밀도 때문에 오차가 발생하는 것이라고 판단했다. 하지만 정밀도가 훨씬 높은 double로 자료형을 모두 바꾸어 구현해 보아도 결과는 똑같았다.

사실 원인은 for문을 두 번 도는 동안 반복문 안에서 velocity에 변화를 주어 서로 충돌한 원 중 for문에서 먼저 접근한 원의 속도 변화가 두 번째로 접근한 원의 속도 변화에 영향을 미친 것 이였다. 결과적으로 delta_v를 이중 포문 안에서 별도로 구하고, 이후에 velocity에 delta_v 값을 더해주는 것으로 순서를 바꾸어 문제를 해결할 수 있었다.