0. 솔루션 파트 초안

 # Detail_AR 프로젝트에서 중요한 부분을 담당하는 "당구 시스템 솔루션을 해결하기 위해 간단히 정리해 보았다.

1. 시뮬레이션 클래스

 # 기능별로 시뮬레이션 클래스의 맴버변수와 메소드를 분리해서 적어 보았다.

1) 당구공의 상태를 관리하는 부분 (쳐야하는 공은 하얀공, 노란공 둘다 되지만 설명을 위해 노란공을 기준으로 한다)

– 맴버변수

1) 쳐야하는 당구공의 색깔 = 노란공
2) 노란당구공의 위치
3) 노란공의 물리 상태
  * 회전량, 진행방향, 속도 등
4) 다른 당구공들의 위치

– 메소드

1) 각각 당구공의 색깔, 위치를 초기화하고, 반환하는 setter getter 메소드

2) 시뮬레이션 부분 (물리)

– 맴버변수

1) 당구대 크기, 모서리 높이, 마찰계수 등 자동 초기화.
2) 다른 공들(빨간공, 흰공)의 위치가 변하지 않는다고 단순히 가정하면, 다른 공들의 위치를 미리 초기화 한다.
3) 노란 당구공이 다른 물체와 부딪힌 횟수 (오랜 반복에도 빨간공을 맞출 수 없다면 멈추기 위해)
4) 빨간공1, 2와 부딪힌 횟수 (모두 부딪혔다면 즉시 종료)

– 메소드

1) 처음 큐대에서 노란공을 쳤을때 물리변화 (큐대 각도, 큐대 강도 등)에 따른 치는 당구공의 물리 값 출력 메소드
* input : 큐대의 강도, 큐대의 각도(360), 에임(가운데 칠것인가, 회전을 위해 옆을 칠것인가)
* output : 노란공의 진행 방향과 물리 상태

2) 노란공의 물리값을 토대로 노란공의 경로 계산 메소드 (단, 당구대 벽이나 다른공에 부딪힐 때 까지의 경로이다.)
* input : 현재 노란공의 물리값(진행 방향, 회전상태 등), 현재 노란공의 위치
* output : 나중 노란공의 위치 및 물리 값, 나중 노란공의 위치, 어떤 물체와 부딪혔는지
* 설명 : 노란공이 가지고 있는 물리값(진행 방향, 회전 등)을 토대로 노란공이 나아가는 경로를 계산한다.
회전이 없다면 직선일 것이고, 회전이 있다면 곡선일 것이다.
노란공이 진행하는 도중, 다른 물체와 부딪히는지 지속적인 검사를 한다.
다른 물체와 부딪혔다면 그 자리에서 즉시 멈추고
부딪히기 직전 위치와 변한 물리값(마찰로 인해 회전이 줄었을 것이다.)을 반환한다.

3) 노란공과 다른공과 부딪혔을때 물리값 계산 메소드
* input : 현재 노란공의 물리 값, 현재 노란공의 위치, (다른 공들의 위치)
* output : 변화된 나중 노란공의 물리값(다른공과 부딪혀 변화된 노란공의 진행 방향벡터나 회전량)
* 설명 : 노란공과 다른공이 부딪혔을때 노란공이 어떻게 변하는지 계산한다.
현재 노란공의 물리값(입사각, 회전 등)을 토대로 다른공과 부딪혔을때 변하는 노란공의 물리값(반사각, 회전)등을 계산하여 반환한다.

4) 노란공과 당구벽과 부딪혔을때 물리값 계산 메소드
* input : 현재 노란공의 물리 값, 현재 노란공의 위치, (다른 공들의 위치)
* output : 변화된 나중 노란공의 물리값(당구벽과 부딪혀 변화된 노란공의 진행 방향 벡터나 회전량)
* 설명 : 3번과 비슷하다.

5) 솔루션 반환 메소드
* 설명 :
맴버변수 4번의 노란당구공이 다른 물체와 부딪힌 횟수가 너무 많으면 솔루션이 없다고 판단한다.
빨간공1, 2와 모두 부딪혔으면 시뮬레이션을 멈추고 솔류션을 반환한다.

3) 경로 저장 파트

– **맴버변수**
1) 노란 당구공이 어떤 물체와 부딪히는 이벤트가 발생했을때 그 위치를 저장한다.
2) 특히 노란 당구공과 빨간공이 부딪혔을때 위치를 구별해서 저장한다.

– **메소드**
1) 노란공이 지나왔던 경로를 반환하는 메소드
2) 노란공과 빨간 당구공이 부딪혔을때 부딪힌 두께를 계산하는 메소드

2. 어떤 경로가 좋은 경로인가?

 # 시뮬레이션을 통해 반환된 여러 솔루션중 좋은 경로로 판단되는 솔루션을 필터링 해야 한다.

1) 쉬운 방법이 좋은 경로이다.
– 쿠션과 부딪히는 횟수가 적은 경로
– 회전이 들어가지 않은 경로

2) 빨간공이 최대한 정중앙에 맞는 경로가 좋은 경로이다.
– 빨간공이 스쳐지나가는 솔루션은 맞추기가 어려울 것이다.
– 경로계산에서 빨간공과 노란공 사이의 두께도 미리 계산 및 저장해두자.

3) 기타.

3. 알고리즘 흐름 정리

시뮬레이션 클래스와, 좋은 경로를 탐색하는 알고리즘을 이용해 전체적인 흐름을 도식화 하였다.

void Solution (Input : 탐지된 당구공들의 위치를 불러온다, output : 계산된 경로를 반환한다)
{

① 시뮬레이션 클래스 선언 : 당구공 위치값 초기화 1)부분 참고

② for (큐대를 이용해 모든 정수의 수로 노란 당구공을 친다){

물리값 변수 A = (방향  , 리턴  , 강도 등)

2)-1> 큐대로 노란공을 쳤을때 물리변화 (input : A , output B)

for (노란공이 빨간공과 부딪힐때까지 반복, 너무 반복이 많으면 break) {

B : 노란공의 진행 방향과 물리 상태가 담겨있음

2)-2> B를 큐대를 노란공의 경로 계산 (input : B , output C, D)

C : 어떤 물체와 부딪혔는지 알려주는 변수

D : 부딪히기 직전 노란공의 물리상태 및 위치

if (부딪혔다면){

3)-1)-2> 결과저장!

if (C = 다른공){

2)-3> 노란공이 다른공과 부딪혔을때 물리값 반환 (input : D , output : E)

E : 반환 노란공의 물리값

}

else{

2)-4> 노란공이 쿠션과 부딪혔을때 물리값 반환 (input : D , output : E)

E : 변환 노란공의 물리값

}

B = E : 노란공의 물리 상태 업데이트 및 반복!

}

}

}