DanguDanawa

Crazy Brother
An Jun Young
Lim Chung Ryeol

Contents

Introduction

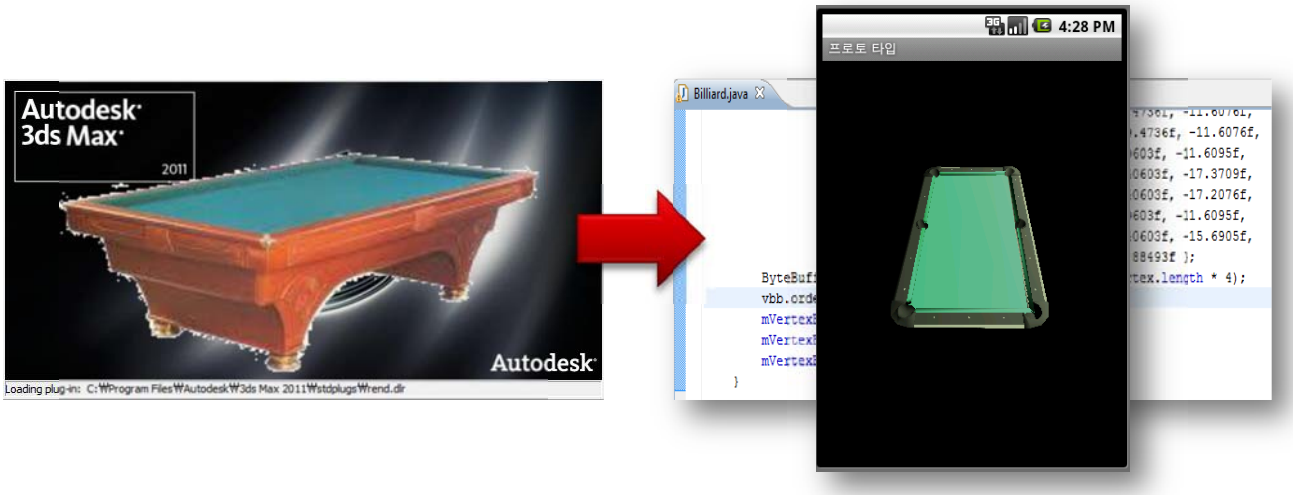
- **Goal**
  - 스마트 폰 기반에서 영상처리를 통한 당구 득점 경로 제공

System Architecture

User Interface

- FrameLayout + LinearLayout

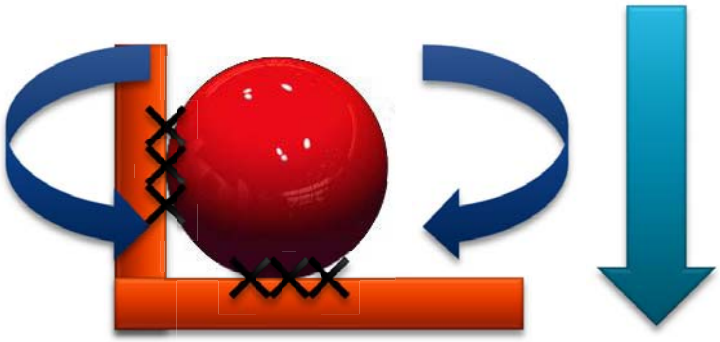
Java code Generator



Java code Generator

- ASE to Java Source

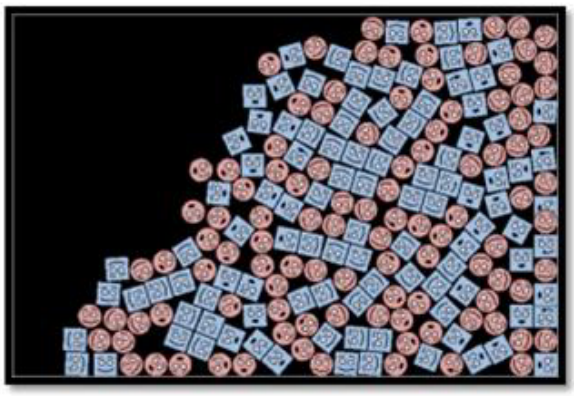
Physics Attribute



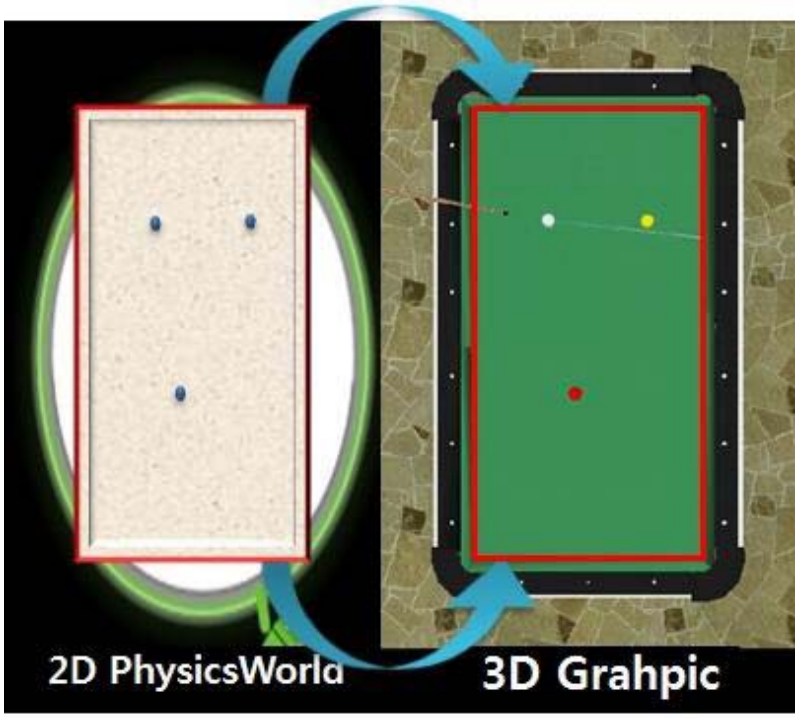
- Attribute
  - Angular Velocity
  - Linear Velocity
  - Gravity (Damping)
  - Density
  - Elasticity
  - Friction

Apply Physics Engine

- AndEngine is a free Android 2D OpenGL Game Engine.



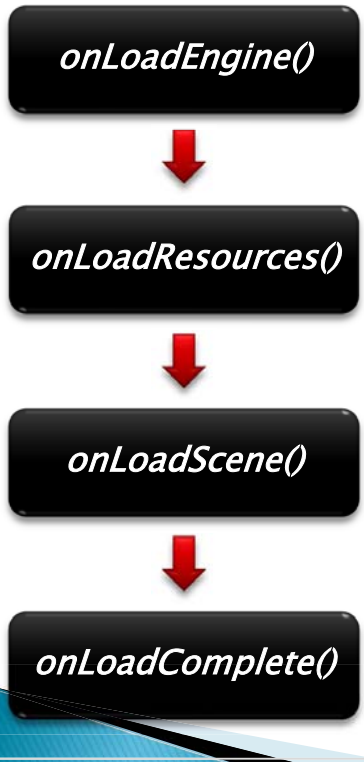
Apply Physics Engine



AndEngine Function

- Drawing Object
- Multi Touch
- Particle Systems
- Network
- Physics
- Text
- Audio
- Backgrounds
- Using Menus, Sub Menu, ZoomCamera, Using ImageFormats(PNG, JPG, GIF, BMP), etc…

AndEngine Flow Chart

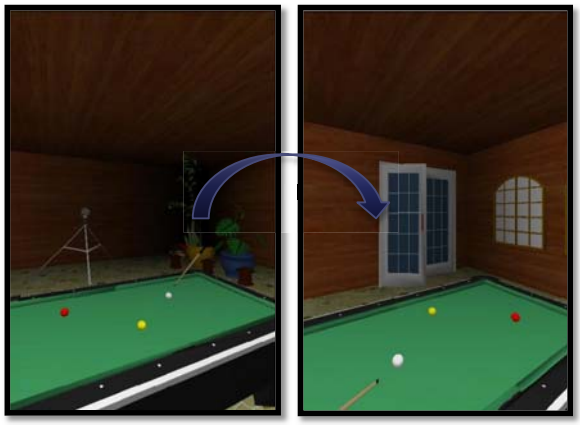


- **Camera** 초기화 : **Zoom** 여부, 사이즈 등
- **Engine** 초기화 : 화면 설정 및 카메라 설정, 멀티 터치 여부 등
- **Sprite Texture** 생성
- **BackGround Texture** 생성
- **Scene** 생성, **UpdateHandler** 등록
- 배경 및 **Static Body(벽)** 등록
- **AndEngine** 초기화 성공 후 사용자 지정 함수 삽입

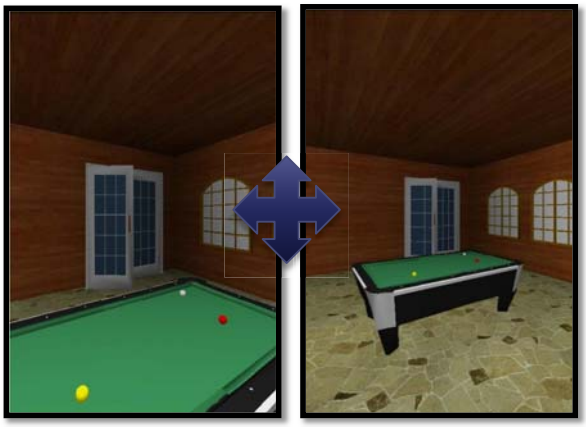
Simulation Function

- Run, Reset, Help

- Rotate

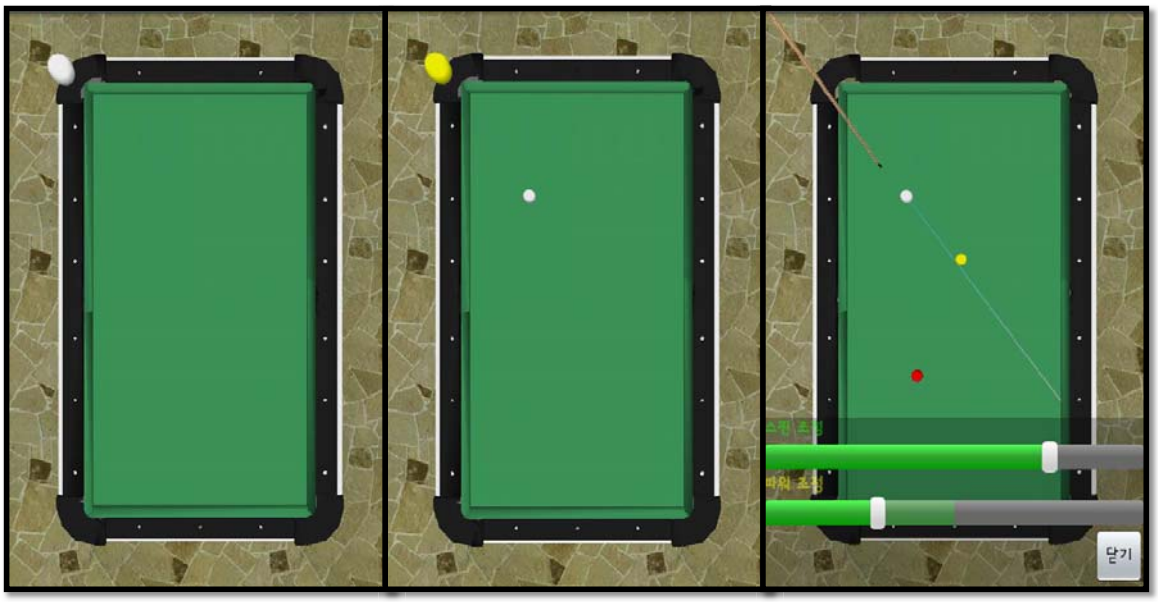


- Zoom in/out

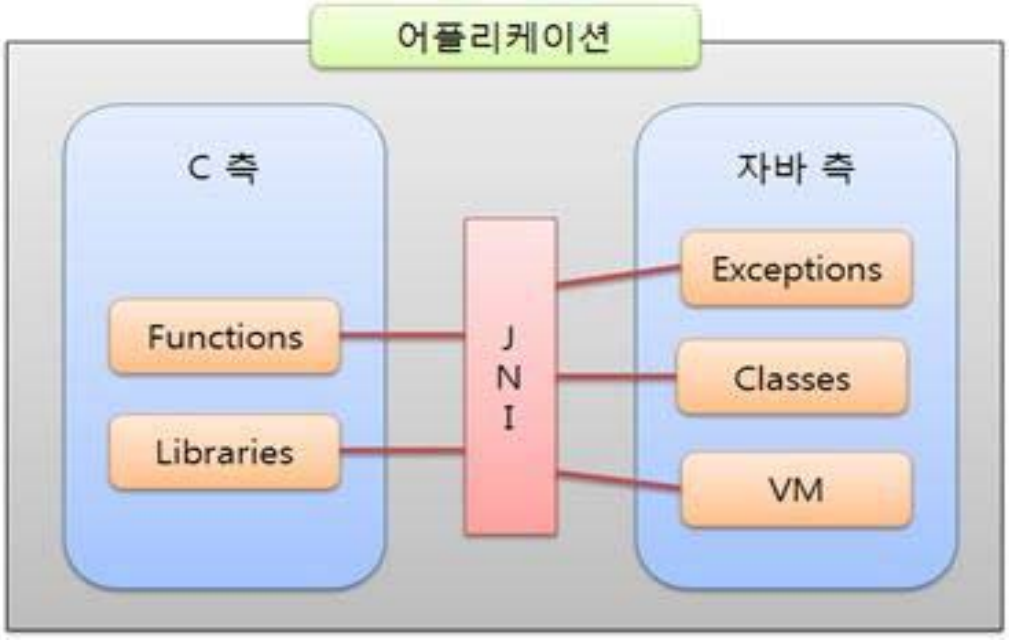


Free Mode Function

- Put a Ball, Control, Shot, Reset, Help



JNI



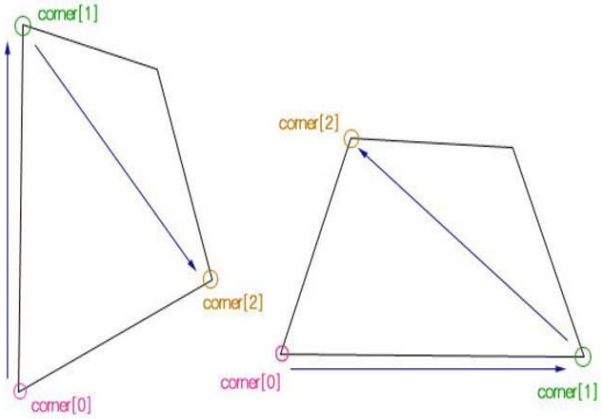
당구대 추출

-색 검출을 통해 필요한 당구대의 영역 확보

-꼭지점 추출을 위한 윤곽선 추출

꼭지점 추출

$$A = \frac{1}{2}(x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + \cdots + x_ny_1 - x_1y_n)$$
$$= \frac{1}{2}\{x_1(y_2 - y_n) + x_2(y_3 - y_1) + \cdots + x_n(y_1 - y_{n-1})\}$$



-임의의 점에서 가장 먼 점을 첫 번째 꼭지점

-같은 방법으로 두 번째 꼭지점

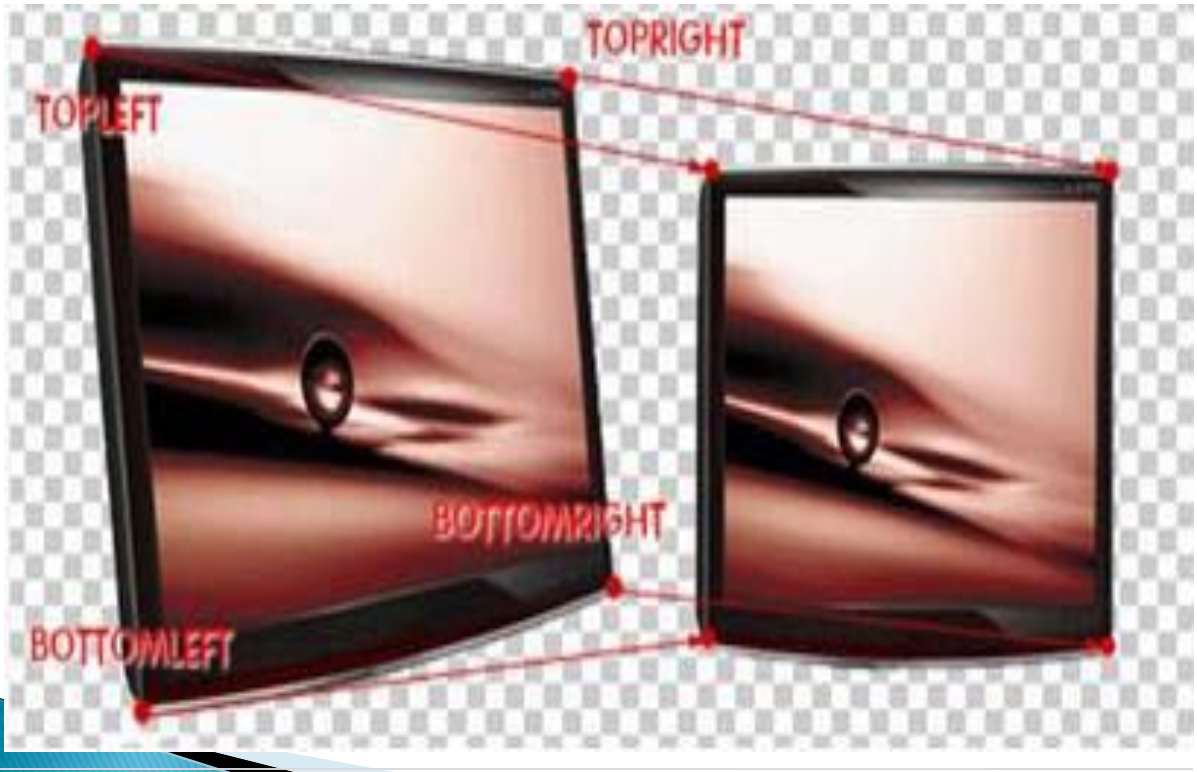
-첫 번째와 두 번째 점에서 가장 먼 점이 세 번째 꼭지점

-사각형의 넓이 정보로 네 번째 꼭지점

영상 보정

실제 당구대를 데이터로 활용하기 위해서 발생하는 문제점

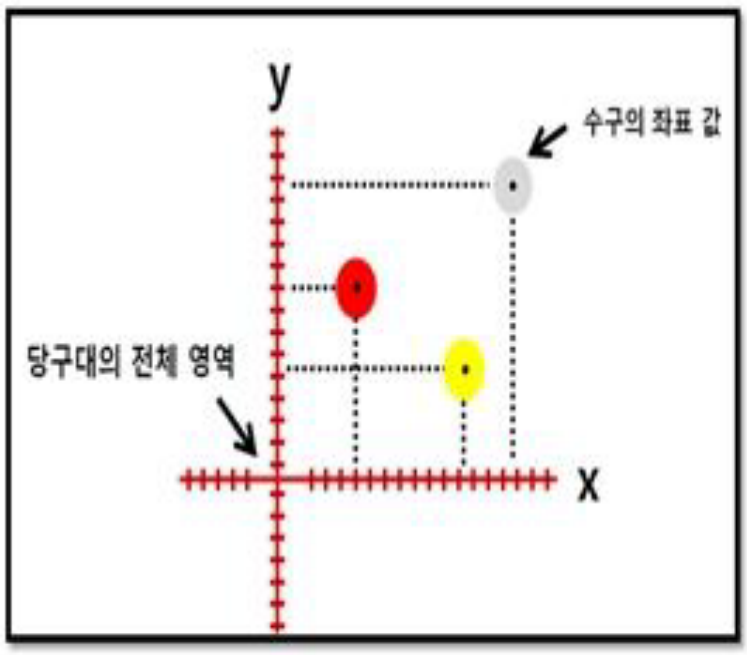
영상 보정



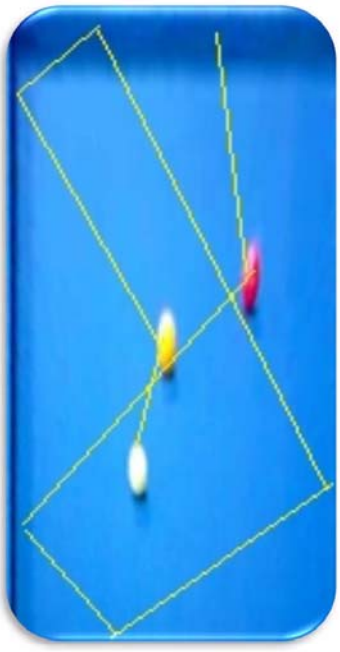
영상 보정

-추출된 당구대를 실제 당구대 비율과 동일한 변환 매트릭스를 만들어 원본이미지를 구하고자 하는 형태의 이미지로 매칭

수구의 좌표 값 전달



수구의 득점 경로 표시



-영상보정의 반대 방법으로 수구의 득점 경로를 입력하여 원 영상으로 병합

Billiards Algorithm

- 목표
  - 당구공의 위치를 이용하여 그 위치에서 **3**쿠션이 되는 최적의 길을 찾아 주는 것

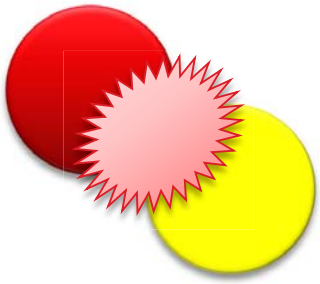
- 제약사항
  - 당구대의 크기는 당구장에서 가장 많이 사용하는 일반적인 당구대의 크기인

    단 쿠션방향 : **1224mm**

    장 쿠션방향 : **2448mm**
  - 당구공의 사이즈는 **3**구 공의 크기가 아닌 **4**구 공의 크기인 **65.5m**

Billiards Algorithm

- 당구에서 충돌의 종류
  - 공과 공



  - 공과 큐

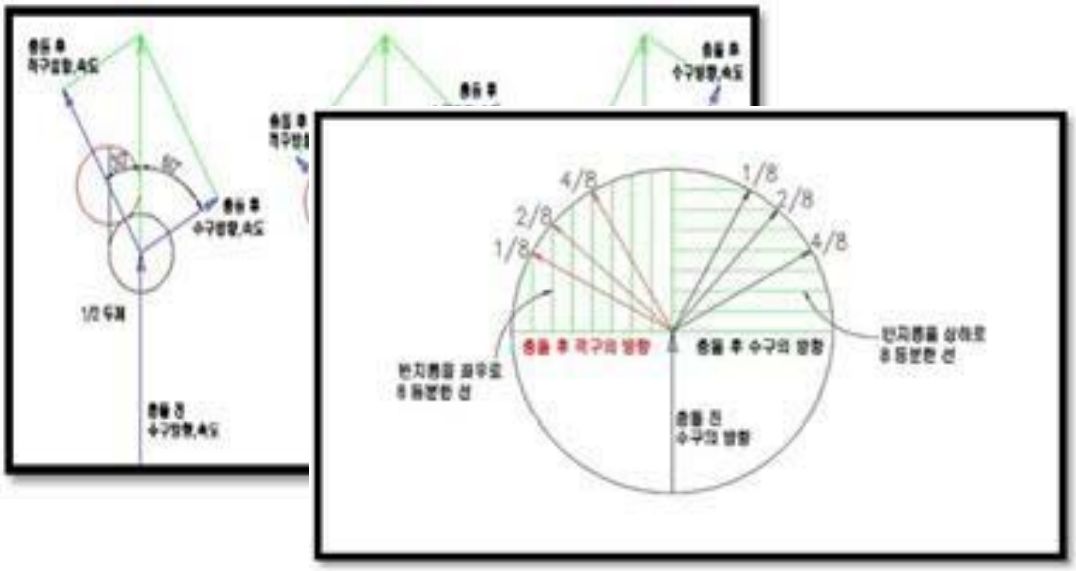
  - 벽과 공

Billiards Algorithm

- 발생하는 물리현상
  - 쿠션의 역할
    - 당구공의 운동에너지를 잠시 담았다 반대 방향으로 되돌려 줌
    - 당구공에 회전이 있으면, 회전반향과 충돌방향이 같으면 반사각이 더 작아지고 반대의 경우이면, 커짐
  - 큐의 역할
    - 운동에너지를 가해서 당구공을 움직이게 하는 요소
  - 당구대의 역할
    - 마찰력을 발생시켜 공을 멈추게 하는 역할

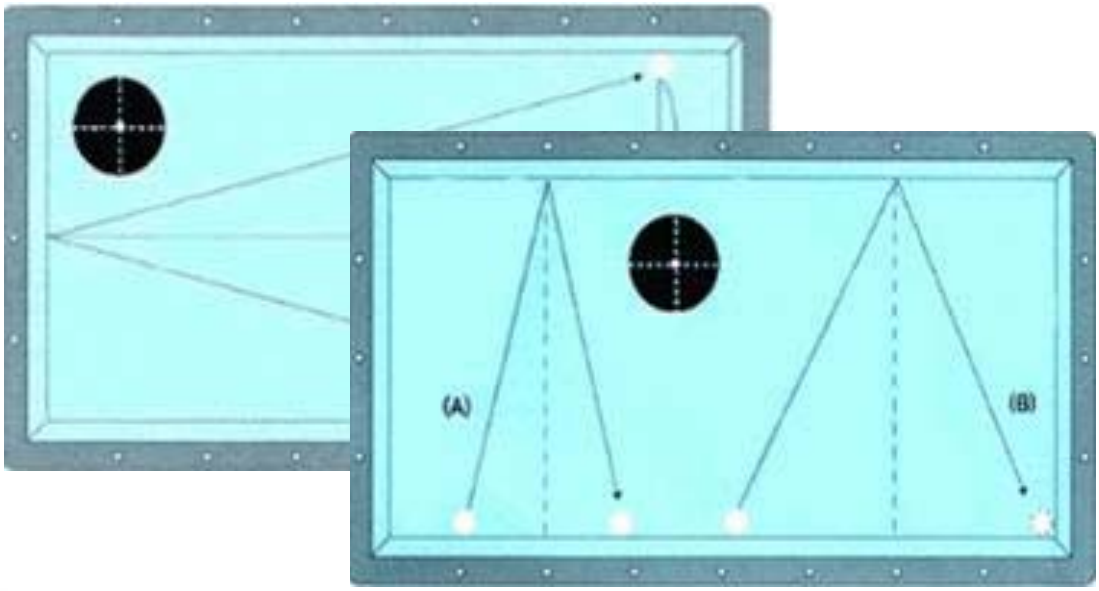
Billiards Algorithm

- 물리 법칙
  - 입사각과 반사각 법칙



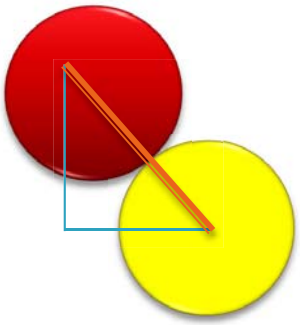
Billiards Algorithm

- 물리 법칙
  - 입사각과 반사각 법칙



Physical Engine

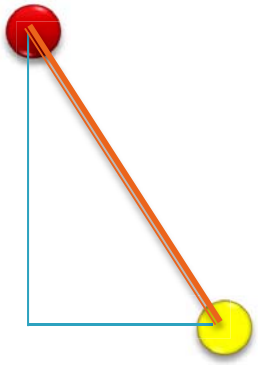
- 수구와 적구의 충돌 확인
  - 피타고라스의 정리를 이용하여 충돌 확인
  - **BALLSIZE**가 빗변의 길이보다 같아지게 되면 충돌 확인
  - $(X2 - X1)^2 + (Y2 - Y1)^2 <= BALLSIZE * BALLSIZE$



Physical Engine

- 수구와 적구 사이의 각도
  - 피타고라스의 정리를 이용하여 밑변과 높이를 이용하여 빗변을 구하고 그것의 호도각을 이용
  - 빗변을 **c**, 밑변을 **a**라고, 각도를 **angle**라고 하면

    cos(angle) = a/c 즉 **angle = arccos(a/c)**
  - **atan2(y2 – y1, x2 – x1) * 180 / *PI***



Physical Engine

- 완전 탄성 충돌에서 사용 가능한 법칙
  - **1.** 운동에너지 보존법칙

$$\frac{1}{2}m_1v_1^2+\frac{1}{2}m_2v_2^2=\frac{1}{2}m_1v'^2_1+\frac{1}{2}m_2v'^2_2$$

  - **2.** 운동량 보존법칙

$$m_1v_1+m_2v_2=m_1v'_1+m_2v'_2$$

  - **3.** 반발계수 **= 1**

$$1 = -\frac{(v'_1-v'_2)}{(v_1-v_2)}$$

Physical Engine

- 반발계수 식에서

  $v_1 - v_2 = -(v'_1 - v'_2)$, $v'_2 = v_1 - v_2 + v'_1$ 을

- 운동량 보존법칙에 대입

  $m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 (v_1 - v_2 + v'_1)$

- **v'**$_1$ 에 대해 정리하면,

  $(m_1 + m_2) v'_1 = m_1 v_1 - m_2 v_1 + 2 m_2 v_2$

- 따라서

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

Billiards Algorithm

- 각도를 이용하여 충돌할 곳을 예측 하여 길을 찾는 방법
  - 장점 : 충돌 지점만을 검사하면 예측이 가능하기 때문에 알고리즘을 계산하는 속도가 빠름
  - 단점 : 키스를 예측하기 힘들고 키스와 모든 상황을 고려하면 장점이 단점으로 변함

- 수구를 시뮬레이션처럼 미리 굴려보고 길을 찾는 방법
  - 장점 : 키스와 같은 모든 상황을 대처할 수 있고 키스를 이용한 길 또한 찾을 수 있음
  - 단점 : 공의 움직이는 모든 경우를 예측해야 하기 때문에 알고리즘을 계산하는 속도가 느림