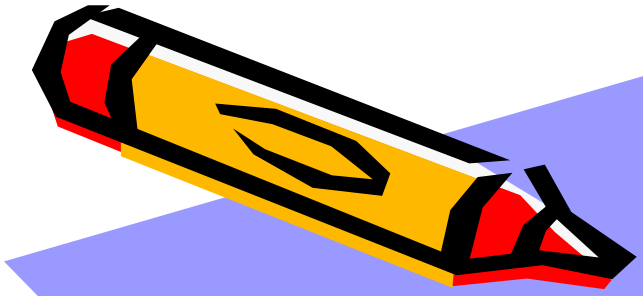


2006년 2학기 윈도우 게임 프로그래밍

제10강 충돌 검사

이대현

한국산업기술대학교

오늘의 학습 내용

- 충돌 검사 개념
- 사각형(바운딩 박스)를 이용한 충돌 검사
- 바운딩 박스를 이용한, 픽셀 단위 정밀도를 가지는 충돌 검사
- 충돌 검사의 실제 적용 방법

충돌 검사(Collision Detection)

- 충돌 검사
  - 게임 상의 오브젝트 간에 충돌이 발생했는지를 검사하는 것.
  - 모든 게임에서 가장 기본적인 물리 계산.
    - 슈팅, 발차기, 펀치, 때리기, 자동차 충돌
  - 기본적으로 시간이 많이 소요되기 때문에, 게임의 오브젝트의 특성에 따라 각종 방법을 통해 최적화해주어야 함.
    - O(N2) 알고리즘

픽셀 단위의 충돌 검사

- 두 개의 오브젝트들의 모든 점들을 일일이 비교.
- 가장 정확함.
- 각 오브젝트들의 픽셀수를 곱한 만큼의 계산 시간이 소요됨.
  - 캐릭터 픽셀 수 x 공 픽셀 수

바운딩 박스(Bounding Box)를 이용한 충돌 검사

- 오브젝트를 감싸는 사각형(바운딩 박스)의 충돌을 비교.
- 사각형의 두개의 교차 여부만 결정하면 되므로 매우 빠름.

오브젝트의 형태가 복잡하면, 충돌 검사 결과가 매우 부정확해짐.

실습

Lecture10-01: 간단한 충돌 검사의 구현

Lecture10-01 프로젝트의 구성

- C++ 소스 파일들
  - ball.cpp – 실습 시간에 작성.
  - collision.cpp – 실습 시간에 작성.
  - gameengine.cpp
  - main.cpp
  - introstate.cpp
  - playstate.cpp – 실습 시간에 작성.
  - sprite.cpp – 실습 시간에 작성.

- C++ 헤더 파일들
  - ball.h – 실습 시간에 작성.
  - collision.h
  - gameengine.h
  - gamestate.h
  - introstate.h
  - playstate.h
  - sprite.h

ball.h

```
#ifndef BALL_H
#define BALL_H

#include <SDL/SDL.h>

class Ball
{
  SDL_Surface* image;
  int px, py;
  int dir; // -1 : left, +1 : right
  SDL_Rect bb;

public:
   Ball();
   ~Ball();

  SDL_Rect& getBoundingBox(void);
  void changeDirection(void);
  void updateFrame(void);
  void draw(void);
};

#endif
```

ball.cpp

```
void Ball::updateFrame(void)
{
  px += (dir * 5);
  if (px < 0 || px > 480 - image->w)
    changeDirection();
}

void Ball::changeDirection(void)
{
  dir *= -1;
}

SDL_Rect& Ball::getBoundingBox(void)
{
  bb.x = px;
  bb.y = py;
  bb.w = image->w;
  bb.h = image->h;
  return bb;
}
```


collision.cpp

```
bool check_collision( SDL_Rect &A, SDL_Rect &B )
{
  int leftA, leftB;
  int rightA, rightB;
  int topA, topB;
  int bottomA, bottomB;

  leftA = A.x;
  rightA = A.x + A.w;
  topA = A.y;
  bottomA = A.y + A.h;

  leftB = B.x;
  rightB = B.x + B.w;
  topB = B.y;
  bottomB = B.y + B.h;

  if( bottomA < topB ) return false;
  if( topA > bottomB ) return false;
  if( rightA < leftB ) return false;
  if( leftA > rightB ) return false;

  return true;
}
```

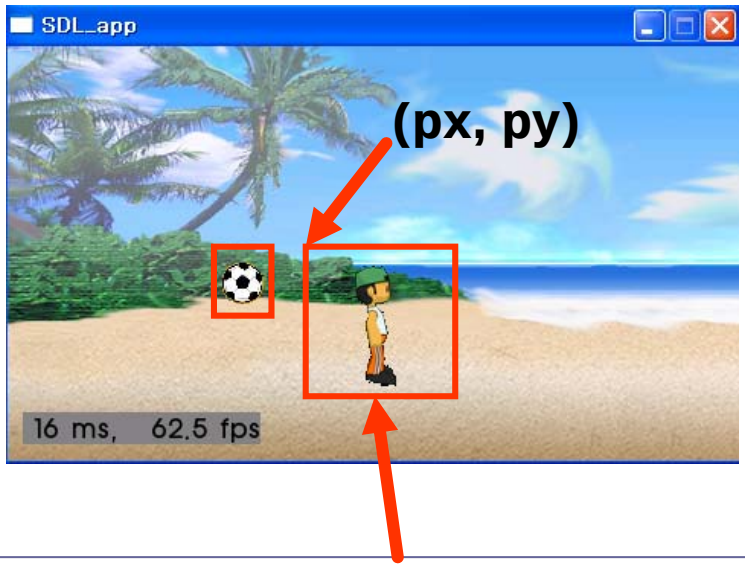
playstate.cpp

```
void CPlayState::Update(CGameEngine* game)
{
  player->updateFrame();
  ball->updateFrame();
  if (check_collision(ball->getBoundingBox(),
                      player->getBoundingBox()))
    ball->changeDirection();
}
```


sprite.cpp

```
SDL_Rect& Sprite::getBoundingBox(void)
{
  bb.x = px;
  bb.y = py;
  bb.w = 100;
  bb.h = 100;
  return bb;
}
```

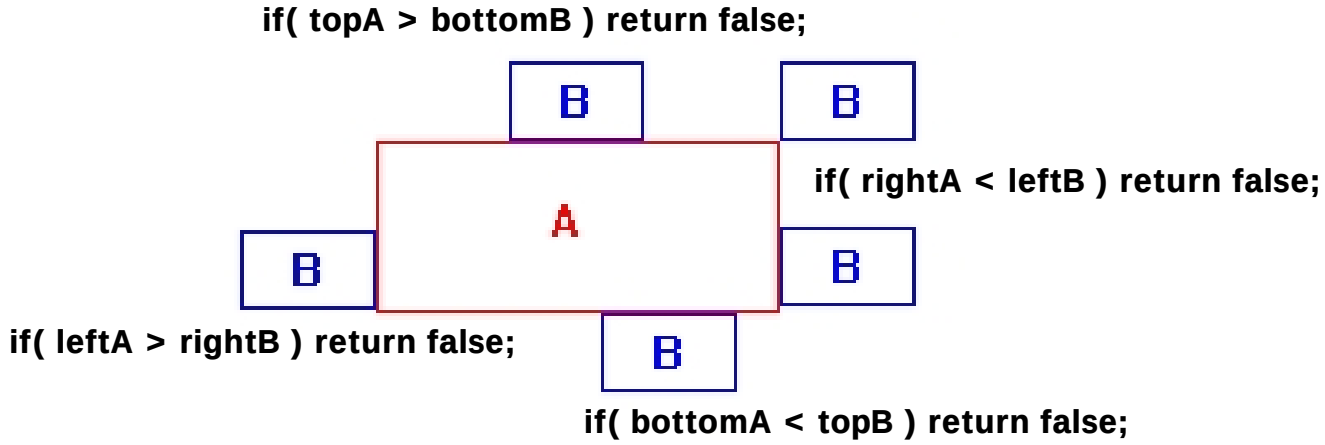
실행 화면



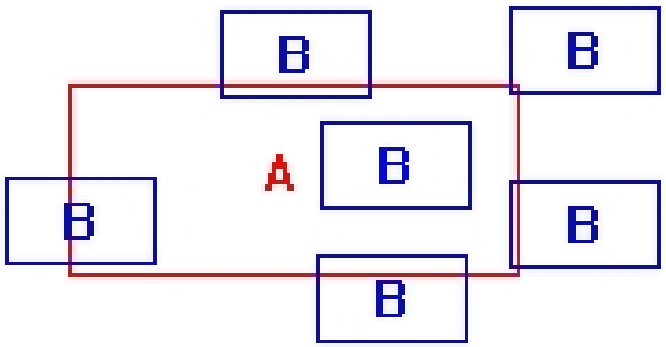
- 실행해보면, 공이 캐릭터와는 떨어진 곳에서 튕긴다.
- 바운딩 박스가 너무 큰 범위로 되어 있기 때문이다(100x100).

바운딩 박스 충돌 체크 알고리즘

충돌이 없는 경우, B의 모든 옆면들이 A의 바깥에 있다.

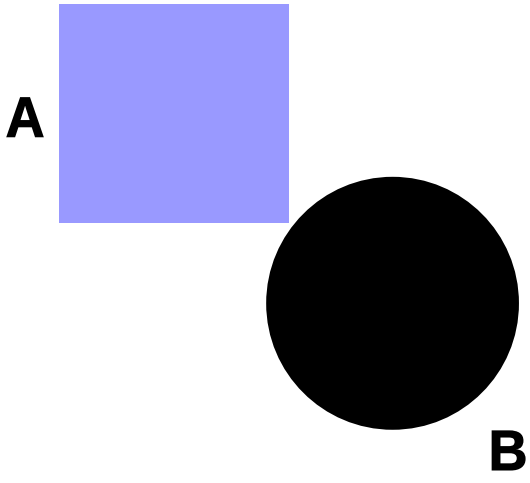


충돌이 있는 경우, B의 옆면들 중 적어도 하나는 A의 안쪽에 있다.

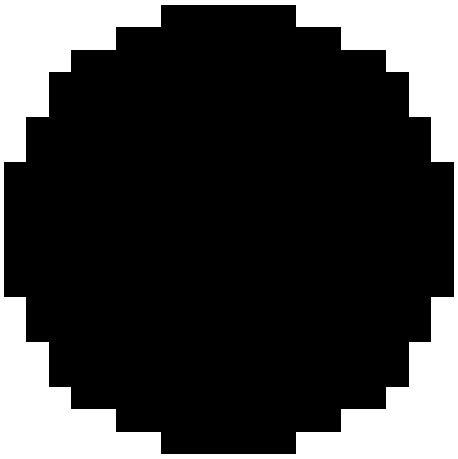


바운딩 박스를 이용한, 픽셀 단위 정밀도를 가지는 충돌 검사

- 예) 사각형 A와 원 B의 충돌 검사
  - 픽셀 단위로 일일이 비교하면, A의 픽셀수 x B의 픽셀수 만큼의 비교가 필요.

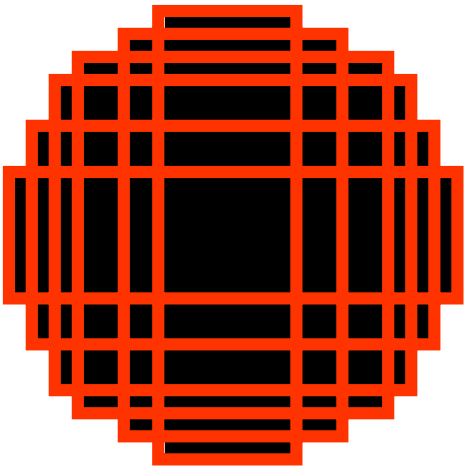


원을 확대하면



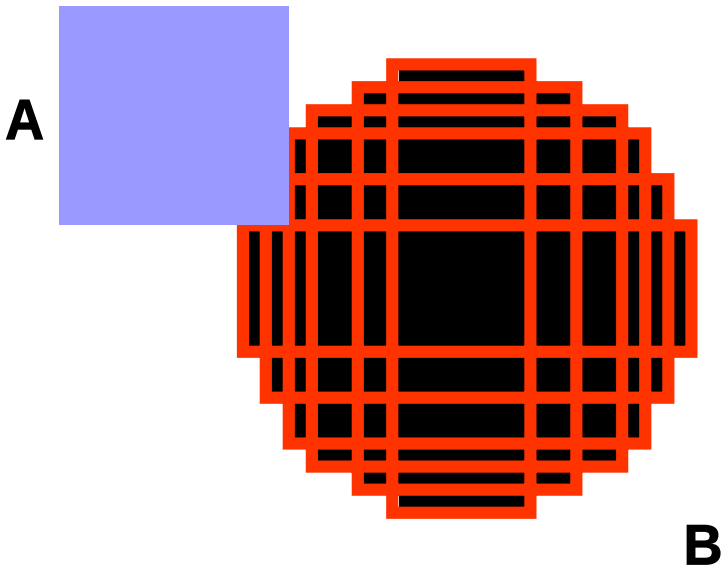
B

원 이미지를 6개의 사각형으로 나타낼 수 있다!!



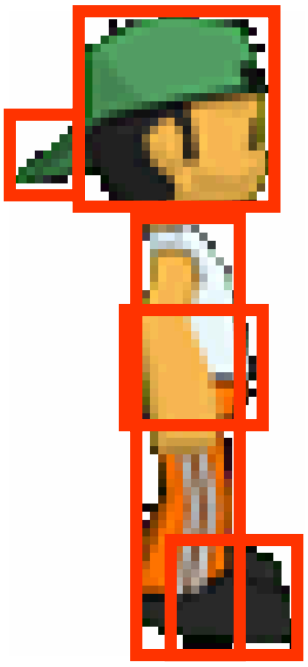
B

사각형 A와 B를 구성하는 여섯개의 사각형을 비교하면 된다.



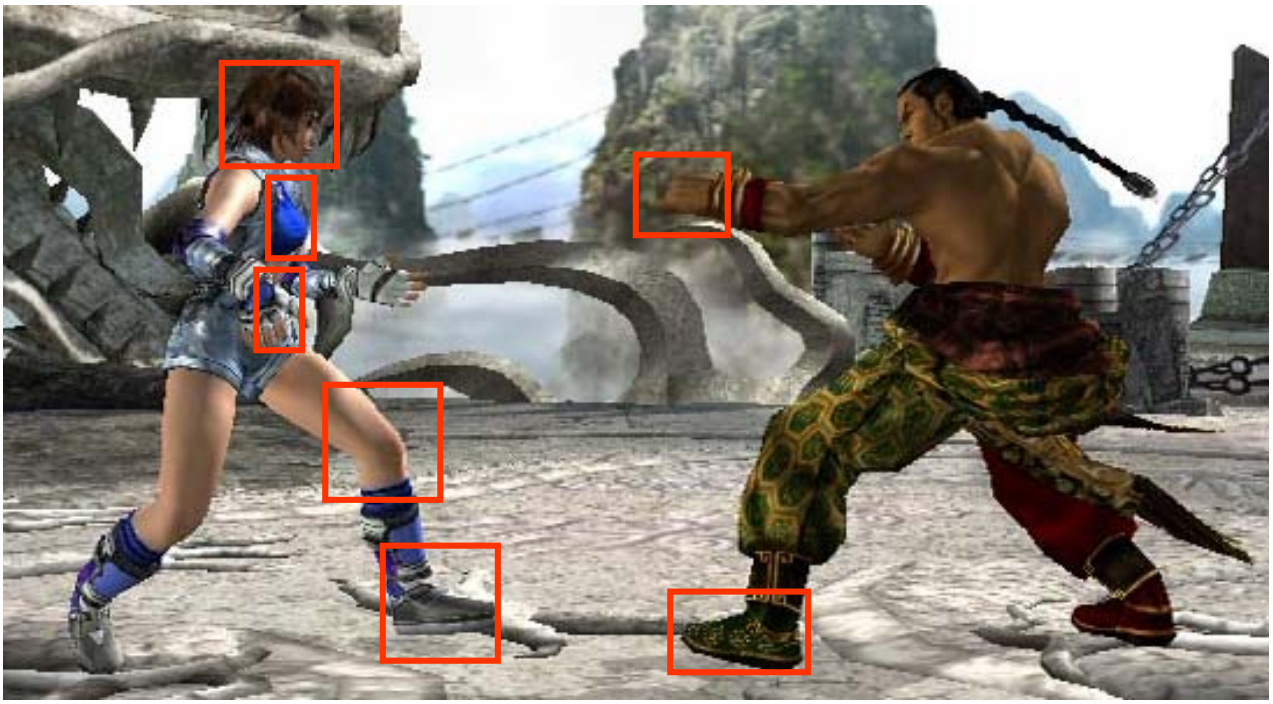
충돌 검사의 실제 적용 방법 (1)

- 정확도를 높이면 한편, 속도 측면에서도 효율적으로 하기 위해, 오브젝트를 적절한 개수의 바운딩 박스로 나눈다.
- 잘게 나누면 나눌수록, 정확도는 높아진다.



충돌 검사의 실제 적용 방법 (2)

- 게임의 특성에 따라 필요한 부분만 바운딩 박스를 적용한다.
  - 격투 대전 게임에서 가격에 사용되는 손 또는 발 부분, 가격이 가해지는 머리, 복부, 배 부분만을 바운딩 박스로 적용.



실습 과제 #8 및 과제 #2

- 공이 캐릭터에 부딪히는 부위에 따라, 기술이 달라짐.
  - lecture10-01-dist.zip을 사용.
  - 캐릭터는 상하좌우로 움직일 수 있음.
  - 공이 맞는 부위에 따라, 화면 우측 상단에 기술의 이름을 표시
    - 이마 부분: Heading을 표시
    - 가슴 부분: Trapping을 표시
    - 발 부분: Kick을 표시
    - 그외: NOP을 표시.
  - 최대한 정확한 충돌 처리 요구.
  - 공의 좌우 떨림 현상 방지.