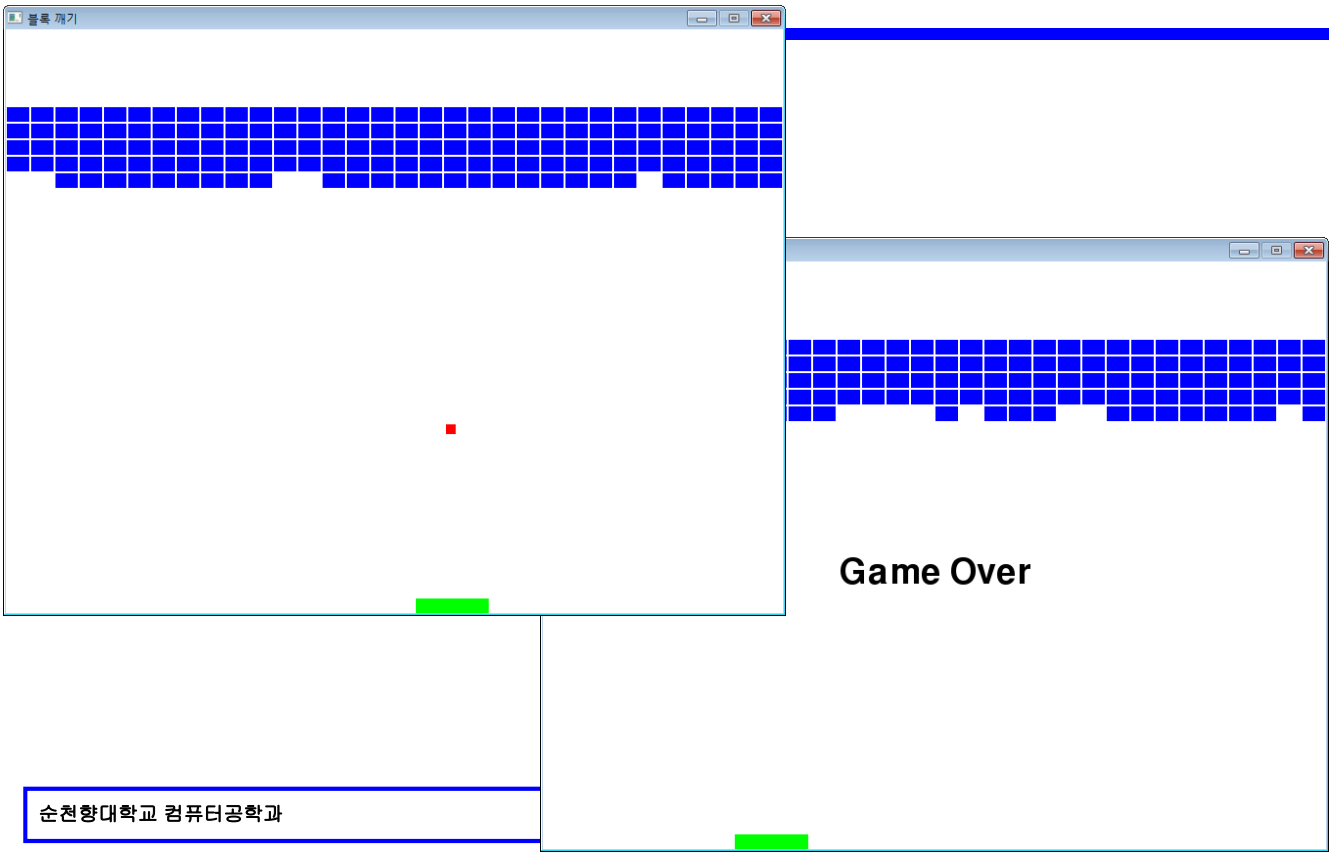
게임 예: 블록 깨기

순천향대학교 컴퓨터공학과
이 상 정

학습 내용

- 블록 깨기 게임 사례 소개
- 객체 지향 프로그래밍
  - 블록 정의 클래스
  - 공 정의 클래스
  - 플레이어 막대 정의 클래스
- 스프라이트 활용
- 삼각함수 이용하여 좌표 및 방향 계산
- 공의 이동
- 공의 충돌
  - 스크린 충돌
  - 플레이어 막대 충돌

블록 깨기 프로그램



클래스 정의 (1)

- 블록 정의 클래스, Block
  - 블록의 이미지, 색, 위치 지정
- 공 정의 클래스, Ball
  - 공의 이미지, 색, 위치 지정
  - 수평면에 공을 되튕기는 함수, bounce() 메소드
  - 공의 위치를 갱신, update() 메소드
- 플레이어 막대 정의 클래스, Player
  - 막대의 이미지, 색, 위치 지정
  - 플레이어 막대의 위치를 갱신, update() 메소드

클래스 정의 (2)

```
import math
.....
# 블록을 정의하는 클래스
# Sprite 부모 클래스를 상속
class Block(pygame.sprite.Sprite):
    # 생성자, 블록의 색과 위치를 인수로 전달
    def __init__(self,color,x,y):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 블록의 이미지와 색 지정
        self.image = pygame.Surface([block_width, block_height])
        self.image.fill(color)
        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
        # 블록의 위치 지정
        self.rect.x = x
        self.rect.y = y
```

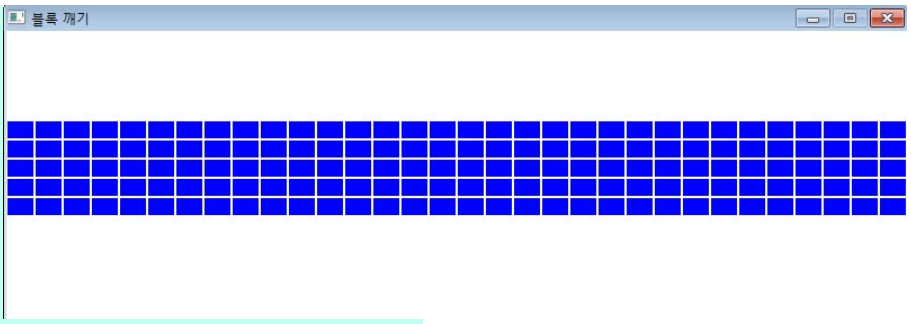


```
import math
.....
# 공을 정의하는 클래스
# Sprite 부모 클래스를 상속
class Ball(pygame.sprite.Sprite):
    .....
    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)

        # 공의 이미지와 색 지정
        self.image = pygame.Surface([self.width, self.height])
        self.image.fill(red)
        # 이미지 크기의 rect 객체 지정
        self.rect =self.image.get_rect()
        # 스크린의 높이와 너비 지정
        self.screenheight = pygame.display.get_surface().get_height()
        self.screenwidth = pygame.display.get_surface().get_width()

    # 수평면에 공을 되튕기는 함수
    def bounce(self,diff):
        .....
    # 공의 위치를 갱신
    def update(self):
        .....
```

```
import math
.....
# 플레이이어 막대를 정의하는 클래스
class Player(pygame.sprite.Sprite):
    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 막대의 이미지와 색 지정
        self.width=75
        self.height=15
        self.image = pygame.Surface([self.width, self.height])
        self.image.fill(green)

        # 이미지 크기의 rect 객체에 위치 지정
        self.rect = self.image.get_rect()
        self.screenheight = pygame.display.get_surface().get_height()
        self.screenwidth = pygame.display.get_surface().get_width()
        self.rect.topleft = (0,self.screenheight-self.height)

    # 플레이어 막대의 위치 갱신
    def update(self):
        .....
```


블록 생성

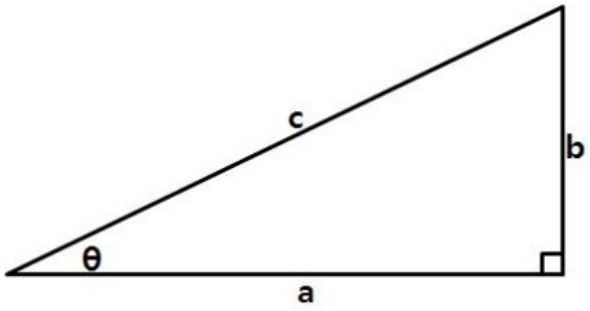
- (0, 80) 위치 부터 5 X 32 블록 생성




```
# 블록의 꼭대기 y 좌표
top =80
# 한 행의 총 블록의 수
blockcount = 32
# 5 x 32 블록 생성
# 5 행의 블록
for row in range(5):
    # 32개 열의 블록
    for column in range(0,blockcount):
        # 블록 생성
        block = Block(blue, column*(block_width+2)+1,top)
        blocks.add(block)
        allsprites.add(block)
    # 아래 행으로 블록 위치 이동
    top += block_height+2
```

삼각함수

❑ 삼각함수



- b = c·sin θ
- a = c·cos θ

$$\sin\theta = \frac{b}{c}$$

$$\cos\theta = \frac{a}{c}$$

$$\tan\theta = \frac{b}{a}$$

도를 라디안 단위로 변환

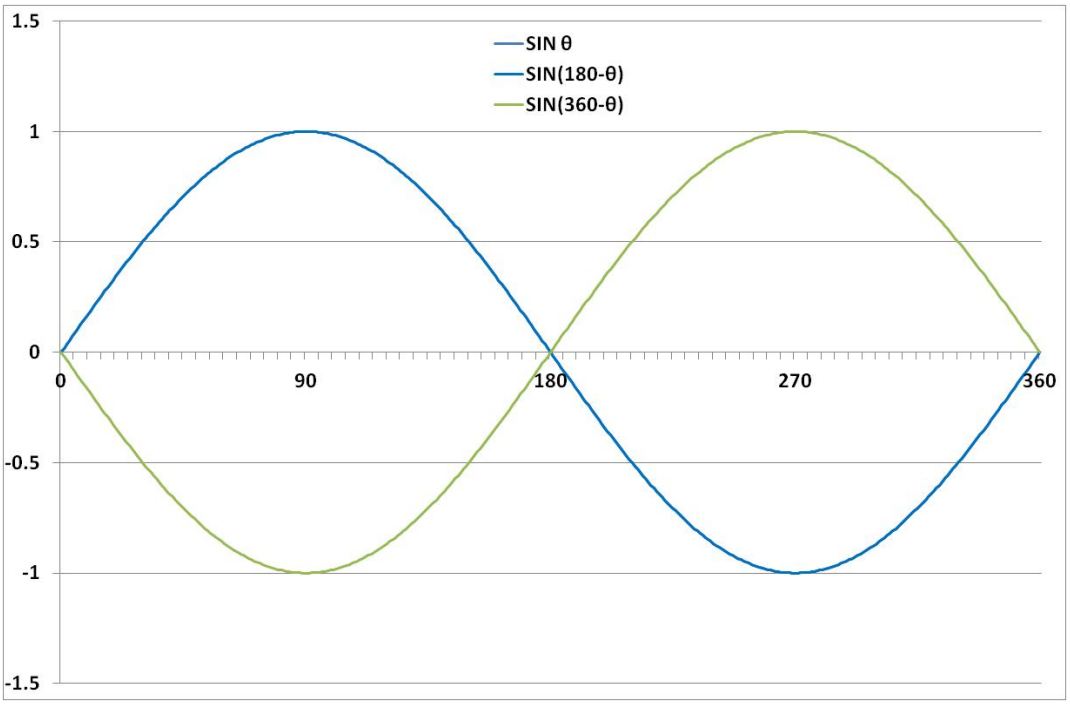
$$도 * \left(\frac{\pi}{180^\circ}\right) = 라디안$$

라디안을 도 단위로 변환

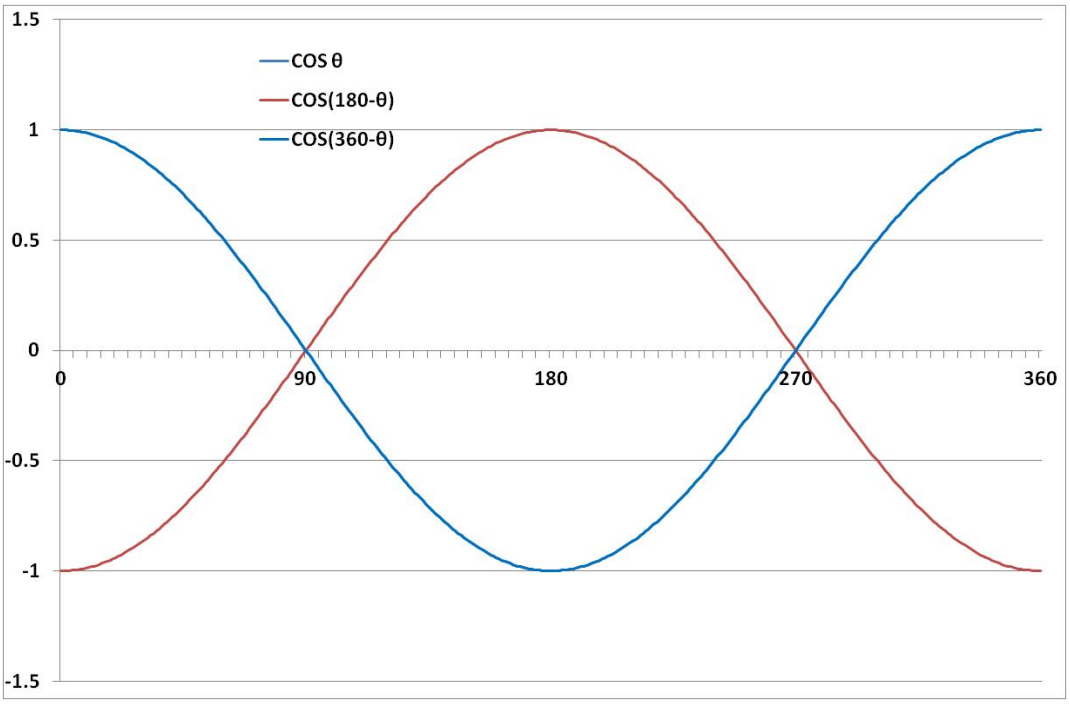
$$라디안 * \left(\frac{180^\circ}{\pi}\right) = 도$$

$$\sin(-\Theta) = -\sin\Theta$$
$$\cos(-\Theta) = \cos\Theta$$
$$\tan(-\Theta) = -\tan\Theta$$

SIN θ, SIN(180−θ), SIN(360−θ)



COS θ, COS(180-θ), COS(360-θ)

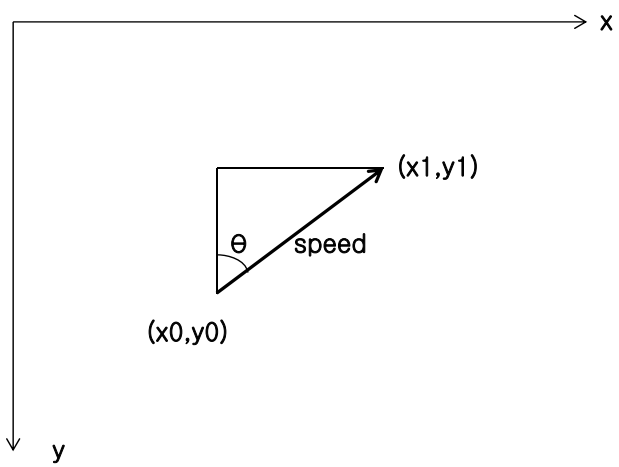


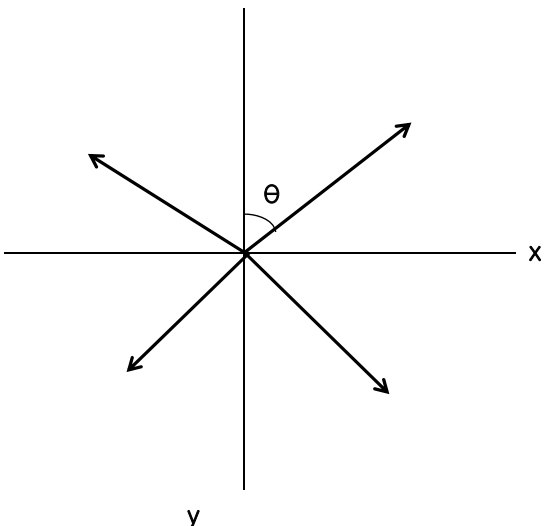
공의 이동 (1)

- 공의 이동
  - 공의 이동 속도와 방향(각도)에 따라 새로운 이동 좌표 계산

$x1 = x0 + speed \cdot \sin \theta$

$y1 = y0 - speed \cdot \cos \theta$

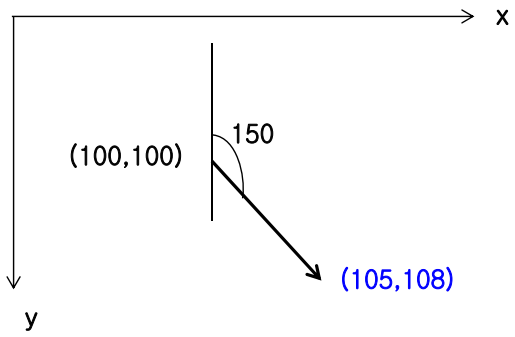




공의 이동 (2)

```
import math
.....
class Ball(pygame.sprite.Sprite):
    # 이동 속도 (픽셀/사이클)
    speed =10.0
    # 공의 위치를 표현하는 실수, 초기값
    x = 0.0
    y = 180.0
    # 공의 방향 (도,degrees)
    direction = 200
    ......
    # 공의 위치를 갱신
    def update(self):
        # 각도를 라디안으로 변환
        direction_radians = math.radians(self.direction)
        # 이동 속도와 방향에 따라 위치를 결정
        self.x +=self.speed * math.sin(direction_radians)
        self.y -=self.speed * math.cos(direction_radians)
        .....
```

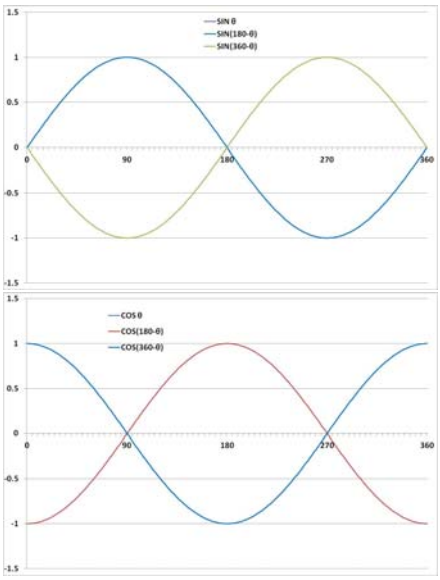
- 예
  - (x0, y0) = (100,100)
  - speed = 10
  - direction (θ) = 150도
    - 라디안 = 150 x pi/180 = 2.62
  - x1 = x0 + 10 x sin 150
    = 100 + 10 x 0.5 = 105
    y1 = y0 – 10 x cos 150
    = 100 – 10 x (–0.87) = 108.7



게임 예: 볼록 깨기

공의 충돌: 스크린 충돌 (1)

- 공이 왼쪽, 오른쪽 스크린에 충돌
  - θ = (360– θ)로 지정하여 sin 부호 반전
    - sin θ = –sin(360–θ)
    - x 방향이 왼쪽(오른쪽)에서 오른쪽(왼쪽)으로 반전
- 공이 위쪽 스크린에 충돌
  - θ = (180– θ)로 지정하여 cos 부호 반전
    - cos θ = –cos(180–θ)
    - y 방향이 위쪽에서 아래쪽으로 반전
- 공이 아래쪽 스크린에 충돌
  - 게임 종료

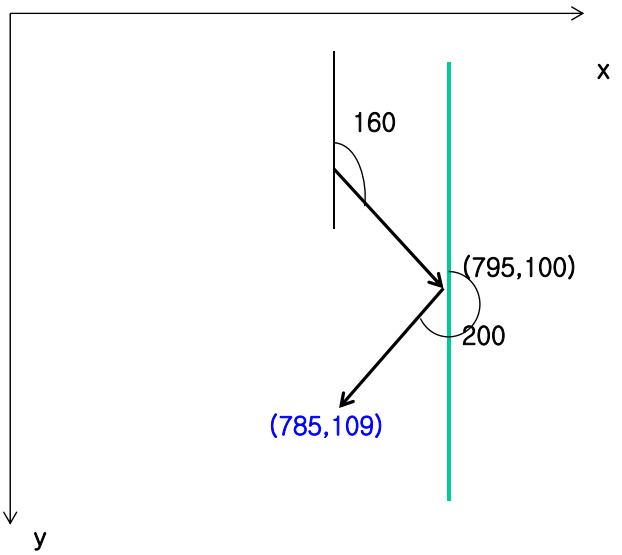


공의 충돌: 스크린 충돌 (2)

```
import math
.....
class Ball(pygame.sprite.Sprite):
    ......
    # 공의 위치를 갱신
    def update(self):
        ......
        self.x +=self.speed * math.sin(direction_radians)
        self.y -=self.speed * math.cos(direction_radians)
        .....
        # 스크린 위에서 충돌 후 되튕기기
        if self.y <=0:
            self.direction = (180-self.direction)%360
                            # cos 부호 반전, y 방향 반전
            self.y = 1

        # 스크린 왼쪽에서 충돌 후 되튕기기
        if self.x <=0:
            self.direction =(360-self.direction)%360
                            # sin 부호 반전,  x 방향 반전
            self.x = 1

        # 스크린 오른쪽에서 충돌 후 되튕기기
        if self.x > self.screenwidth-self.width:
            self.direction = (360-self.direction)%360
                            # sin 부호 반전 ,  x 방향 반전
            self.x = self.screenwidth-self.width-1

        # 스크린 아래로 떨어지는 여부 조사
        if self.y >600:
            return True
        else:
            return False
```


공의 충돌: 스크린 충돌 (3)

- 예
  - 스크린 크기 800x600, 공 크기 10 x 10
  - 충돌 전 좌표, 방향
    - (x0, y0) = (795,100)
    - speed = 10
    - direction (θ) = 160도
  - 충돌 후 이동 좌표, 방향
    - direction (θ) = 360−160 = 200도
    - x0 = screenwidth−self.width−1
      = 800−10−1 = 789
    - x1 = 789 + 10 x sin 200
      = 789 + 10 x (−0.34) = 785.6
      y1 = 100 − 10 x cos 200
      = 100 − 10 x (−0.94) = 109.4



공의 충돌: 플레이어 막대 충돌 (1)

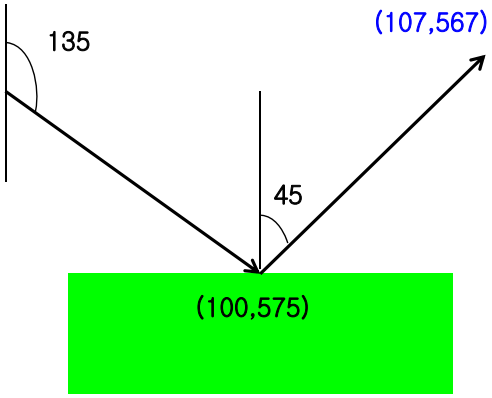
- 막대의 중앙 에서 공이 충돌한 위치의 차이 diff 계산
  - 중앙 왼쪽 충돌 +, 오른쪽 충돌 – diff 값
  - diff = (player.rect.left +player.width/2) – (ball.rect.left+ball.width/2)
    - player.width = 75, ball.width = 10
    - diff = 0~42
- 공의 y 위치를 막대 경계 값으로 조정
  - ball.rect.top =screen.get_height() – player.rect.height – ball.rect.height –1
  - (x, 600–15–10) = (x, 575)
- θ = (180– θ)로 지정하여 cos 부호 반전
  - y 방향이 아래 쪽에서 위 쪽으로 반전
  - cos θ = –cos(180–θ)
- 각도 direction 값에 diff를 빼줌
  - direction = direction – diff

공의 충돌: 플레이어 막대 충돌 (2)

```
.....
class Ball(pygame.sprite.Sprite):
    .....
    # 수평면에 공을 되튕기는 함수
    def bounce(self,diff):
        self.direction = (180-self.direction)%360
                        # cos 부호 반전, y 방향 반전
        self.direction -= diff

        .....
 # 플레이어 막대와 공이 충돌하면
    if pygame.sprite.spritecollide(player, balls, False):
        # 막대의 중앙 에서 공이 충돌한 위치의 차이 계산, 중앙 왼쪽 충돌 +, 오른쪽 충돌 -
        diff = (player.rect.left +player.width/2) -(ball.rect.left+ball.width/2)
        # 공의 y 위치를 막대 경계 값으로 조정
        ball.rect.top =screen.get_height() - player.rect.height - ball.rect.height -1
        ball.bounce(diff)
     .....
```

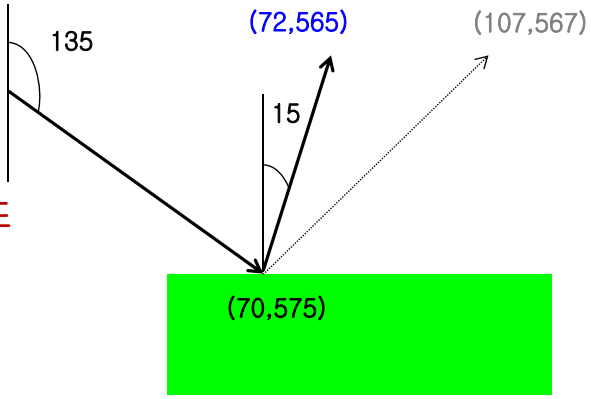
공의 충돌: 플레이어 막대 충돌 (3)

- 공이 막대 중앙(x=100)에 충돌한 경우
  - 충돌 시 좌표, 방향
    - (x0, y0) = (100, 575)
    - speed = 10
    - direction (θ) = 135도
  - 충돌 후 좌표, 방향
    - diff = 0
    - direction (θ) = 180−θ−diff
      = 180−135−0 = 45도
    - x1 = 100 + 10 x sin 45
      = 100 + 10 x 0.71 = 107.1
      y1 = 575 − 10 x cos 45
      = 575 − 10 x 0.71 = 567.9



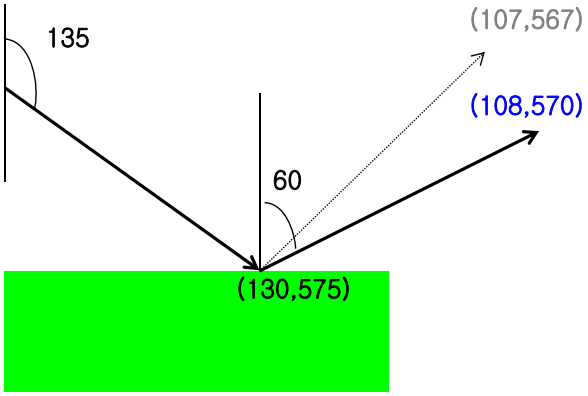
공의 충돌: 플레이어 막대 충돌 (4)

- 공이 막대 중앙에서 왼쪽에 충돌한 경우
  - 충돌 시 좌표, 방향
    - (x0, y0) = (70, 575)
    - speed = 10
    - direction (θ) = 135도
  - 충돌 후 좌표, 방향
    - diff = 100−70 = 30
    - direction (θ) = 180−θ−diff
      = 180−135−30 = 15도
    - x1 = 70 + 10 x sin 15
      = 70 + 10 x 0.26 = 72.6
      y1 = 575 − 10 x cos 15
      = 575 − 10 x 0.97 = 565.3



공의 충돌: 플레이어 막대 충돌 (5)

- 공이 막대 중앙에서 오른쪽에 충돌한 경우
  - 충돌 시 좌표, 방향
    - (x0, y0) = (130, 575)
    - speed = 10
    - direction (θ) = 135도
  - 충돌 후 좌표, 방향
    - diff = 100−130 = −30
    - direction (θ) = 180−θ−diff
      = 180−135+30 = 60도
    - x1 = 130 + 10 x sin 60
      = 130 + 10 x 0.87 = 108.7
      y1 = 575 − 10 x cos 60
      = 575 − 10 x 0.5 = 570



공과의 충돌: 블록과 충돌

- 공이 블록과 충돌하면 블록은 제거
  - 스프라이트 충돌 함수 이용
  - deadblocks = pygame.sprite.spritecollide(ball, blocks, True)
    - 세번째 인수가 참이면 그룹에서 제거
- 공의 y 방향이 위쪽에서 아래쪽으로 반전
  - θ = (180− θ)로 지정하여 cos 부호 반전
  - ball.bounce(0)

```
# 공과 블록의 충돌 조사
deadblocks = pygame.sprite.spritecollide(ball, blocks, True)

# 공이 블록에 충돌했으면 되튕기기
if len(deadblocks) > 0:
    ball.bounce(0)

    # 모든 블록을 깼으면 게임 종료
    if len(blocks) ==0:
        game_over =True
```

블록 깨기 코드

```
import math
import pygame

black = (0,0,0)
white = (255,255,255)
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)

# 블록의 크기
block_width=23
block_height=15

# 블록을 정의하는 클래스
# Sprite 부모 클래스를 상속
class Block(pygame.sprite.Sprite):
    # 생성자, 블록의 색과 위치를 인수로 전달
    def __init__(self,color,x,y):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)

        # 블록의 이미지와 색 지정
        self.image = pygame.Surface([block_width,
    block_height])
        self.image.fill(color)
        # 이미지 크기의 rect 객체 지정
        self.rect = self.image.get_rect()
        # 블록의 위치 지정
        self.rect.x = x
        self.rect.y = y

# 공을 정의하는 클래스
# Sprite 부모 클래스를 상속
class Ball(pygame.sprite.Sprite):
    # 이동 속도 (픽셀/사이클)
    speed =10.0
    # 공의 위치를 표현하는 실수
    x = 0.0
    y = 180.0
    # 공의 방향 (도,degrees)
    direction = 200

    # 공의 크기
    width=10
    height=10

    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)

        # 공의 이미지와 색 지정
        self.image = pygame.Surface([self.width,
      self.height])
        self.image.fill(red)
        # 이미지 크기의 rect 객체 지정
        self.rect =self.image.get_rect()
        # 스크린의 높이와 너비 지정
        self.screenheight =
      pygame.display.get_surface().get_height()
        self.screenwidth =
      pygame.display.get_surface().get_width()

    # 수평면에 공을 되튕기는 함수
    def bounce(self,diff):
        self.direction = (180-self.direction)%360
                    # cos 부호 반전, y 방향 반전
        self.direction -= diff

    # 공의 위치를 갱신
    def update(self):
        # 각도를 라디안으로 변환
        direction_radians = math.radians(self.direction)
        # 이동 속도와 방향에 따라 위치를 결정
        self.x +=self.speed * math.sin(direction_radians)
        self.y -=self.speed * math.cos(direction_radians)
        # 이미지를 이동
        self.rect.x = self.x
        self.rect.y = self.y

        # 스크린 위에서 충돌 후 되튕기기
        if self.y <=0:
            self.bounce(0)
            self.y=1
        # 스크린 왼쪽에서 충돌 후 되튕기기
        if self.x <=0:
            self.direction =(360-self.direction)%360
                            # sin 부호 반전, x 방향 반전
            self.x = 1
        # 스크린 오른쪽에서 충돌 후 되튕기기
        if self.x > self.screenwidth-self.width:
            self.direction = (360-self.direction)%360
                            # sin 부호 반전, x 방향 반전
            self.x = self.screenwidth-self.width-1
```

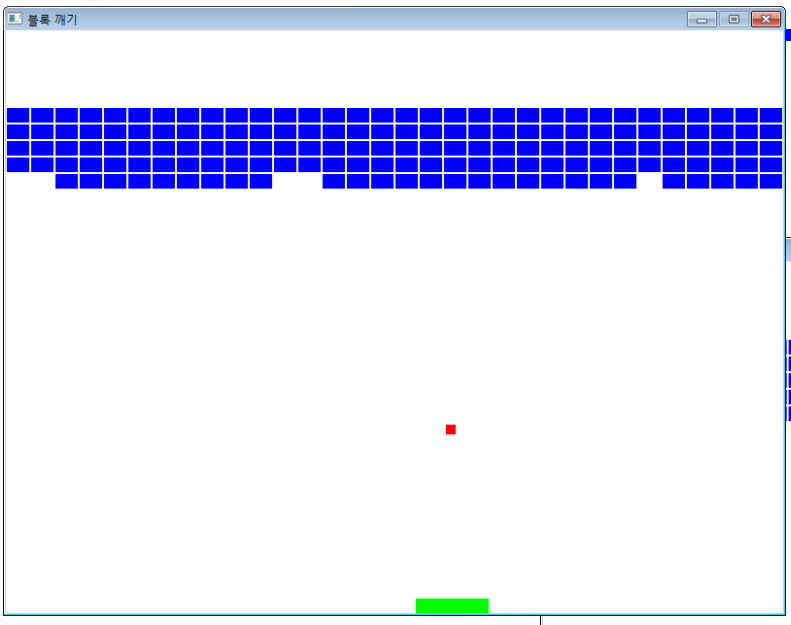
```
        # 스크린 아래로 떨어지는 여부 조사
        if self.y >600:
            return True
        else:
            return False

# 플레이이어 막대를 정의하는 클래스
class Player(pygame.sprite.Sprite):
    # 생성자
    def __init__(self):
        # 부모 클래스 생성자 호출
        pygame.sprite.Sprite.__init__(self)
        # 막대의 이미지와 색 지정
        self.width=75
        self.height=15
        self.image = pygame.Surface([self.width,
      self.height])
        self.image.fill(green)
        # 이미지 크기의 rect 객체에 위치 지정
        self.rect =self.image.get_rect()
        self.screenheight =
      pygame.display.get_surface().get_height()
        self.screenwidth =
              pygame.display.get_surface().get_width()
        self.rect.topleft =
              (0,self.screenheight-self.height)
    # 플레이어 막대의 위치 갱신
    def update(self):
        # 마우스 현재 위치에 막대 위치
        pos =pygame.mouse.get_pos()
        self.rect.left = pos[0]
        # 스크린 오른쪽 경계를 막대가 넘지 않도록 조정
        if self.rect.left > self.screenwidth -self.width:
            self.rect.left =self.screenwidth -self.width

pygame.init()

# 윈도우 설정
screen =pygame.display.set_mode([800, 600])
pygame.display.set_caption('블록 깨기')

# 마우스 포인트 감춤
pygame.mouse.set_visible(0)

# 텍스트 폰트 생성
font = pygame.font.Font('freesansbold.ttf', 36)

# 스프라이트 리스트 생성
blocks = pygame.sprite.Group()
balls = pygame.sprite.Group()
allsprites = pygame.sprite.Group()
```

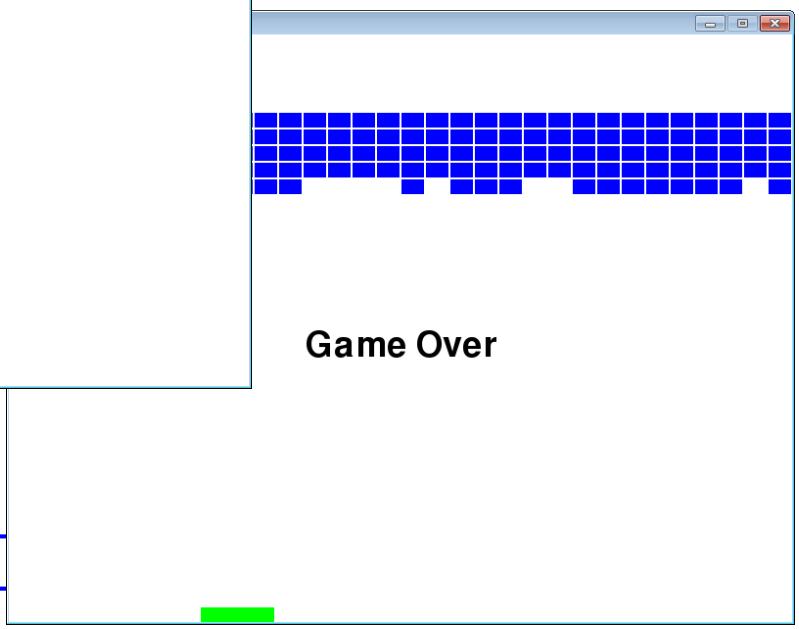
```
# 플레이어 막대 생성
player = Player()
allsprites.add(player)

# 공 생성
ball = Ball()
allsprites.add(ball)
balls.add(ball)

# 블록의 꼭대기 y 좌표
top =80
# 한 행의 총 블록의 수
blockcount =32

# 5 x 32 블록 생성
# 5 행의 블록
for row in range(5):
    # 32개 열의 블록
    for column in range(0,blockcount):
        # 블록 생성
        block = Block(blue,column*(block_width+2)
                      +1,top)
        blocks.add(block)
        allsprites.add(block)
    # 아래 행으로 블록 위치 이동
    top +=block_height+2

clock =pygame.time.Clock()
game_over =False
exit_program =False

while exit_program !=True:
    # 30 프레임/초
    clock.tick(30)

    # 스크린 클리어
    screen.fill(white)

    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            exit_program =True

    # 공과 플레이어 위치 갱신
    if not game_over:
        player.update()
        game_over = ball.update()

    if game_over:     # 게임 종료 이면 Game Over 출력
        text=font.render("Game Over", 1, black)
        textpos
            = text.get_rect(centerx=screen.get_width()/2)
        screen.blit(text, textpos)
```

```
    # 플레이어 막대와 공이 충돌하면
    if pygame.sprite.spritecollide(player, balls, False):
        # 막대의 중앙 에서 공이 충돌한 위치의 차이 계산,
        # 중앙 왼쪽 충돌 +, 오른쪽 충돌 -
        diff =(player.rect.left +player.width/2) -
                                (ball.rect.left+ball.width/2)

        # 공의 y 위치를 막대 경계 값으로 조정
        ball.rect.top =screen.get_height() -
                player.rect.height - ball.rect.height -1
        ball.bounce(diff)

    # 공과 블록의 충돌 조사
    deadblocks = pygame.sprite.spritecollide(ball, blocks, True)

    # 공이 블록에 충돌했으면 되튕기기
    if len(deadblocks) > 0:
        ball.bounce(0)

        # 모든 블록을 깼으면 게임 종료
        if len(blocks) ==0:
            game_over =True

    # 모든 스프라이트 그리기
    allsprites.draw(screen)

        # 스크린 갱신하여 디스플레이
        pygame.display.flip()

    pygame.quit()
```

게임 예: 볼록 깨기

시험주행





과제

1. 지금까지 배운 내용을 활용한 텀 프로젝트 부분 프로그램 작성
   - 이 전 텀 프로젝트 부분 과제에 추가된 내용 설명
   - 지금까지 배운 내용 활용한 부분 프로그램 소스
   - 실행 결과