Lecture #17. 스크롤링

2D 게임 프로그래밍

이대현 교수

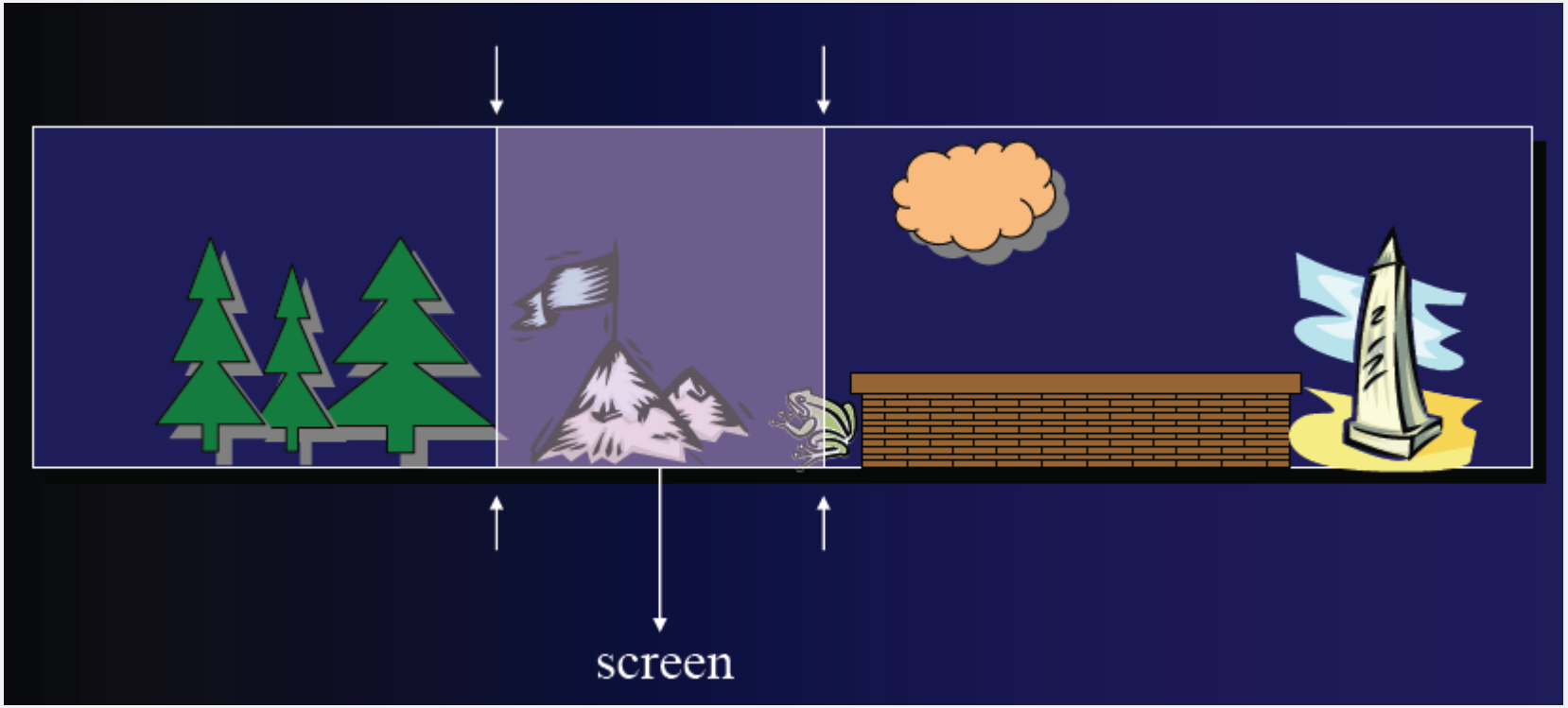
한국산업기술대학교
KOREA POLYTECHNIC UNIVERSITY

학습 내용

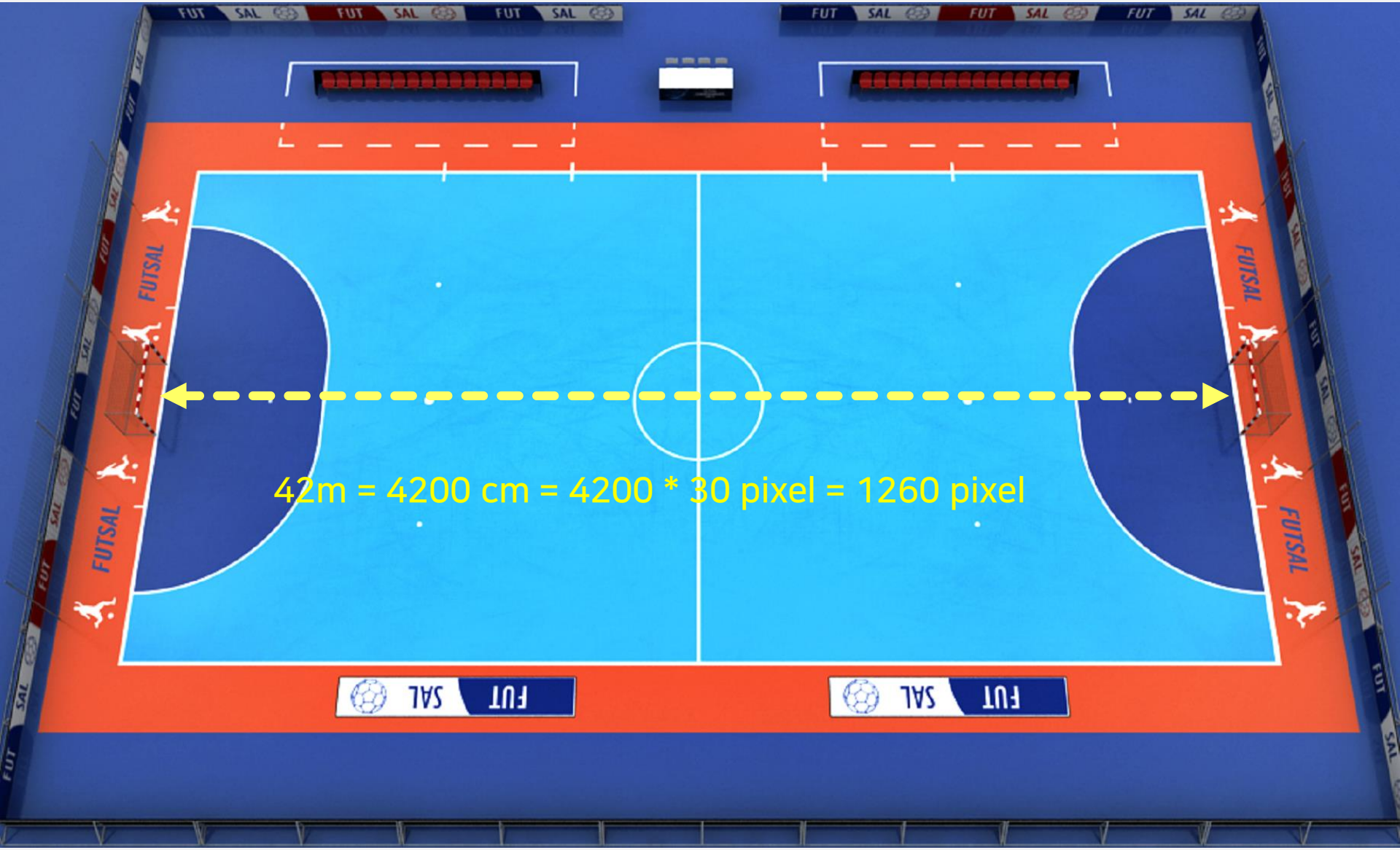
- 스크롤링
- 게임 맵
- 무한 스크롤링
- 시차 스크롤링

스크롤링(Scrolling)

- 그림이나 이미지의 일부분을 디스플레이 화면 위에서 상하좌우로 움직이면서 나타내는 기법.
- 슈팅 게임, 고전 RPG 게임에서 주로 사용됨.

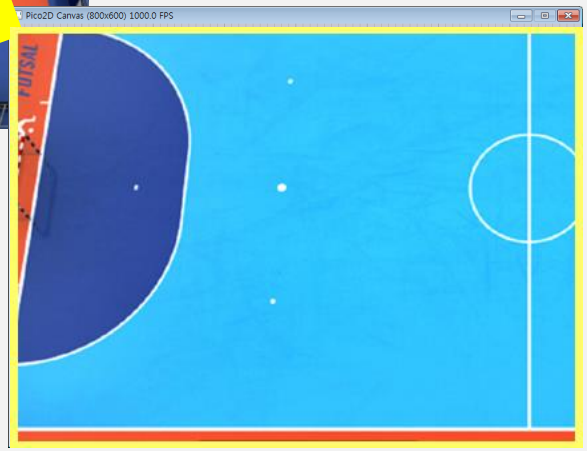


게임 맵은 반드시 실제 물리값으로 크기가 표시되어야 함.



카메라 윈도우를 이용한 스크롤링

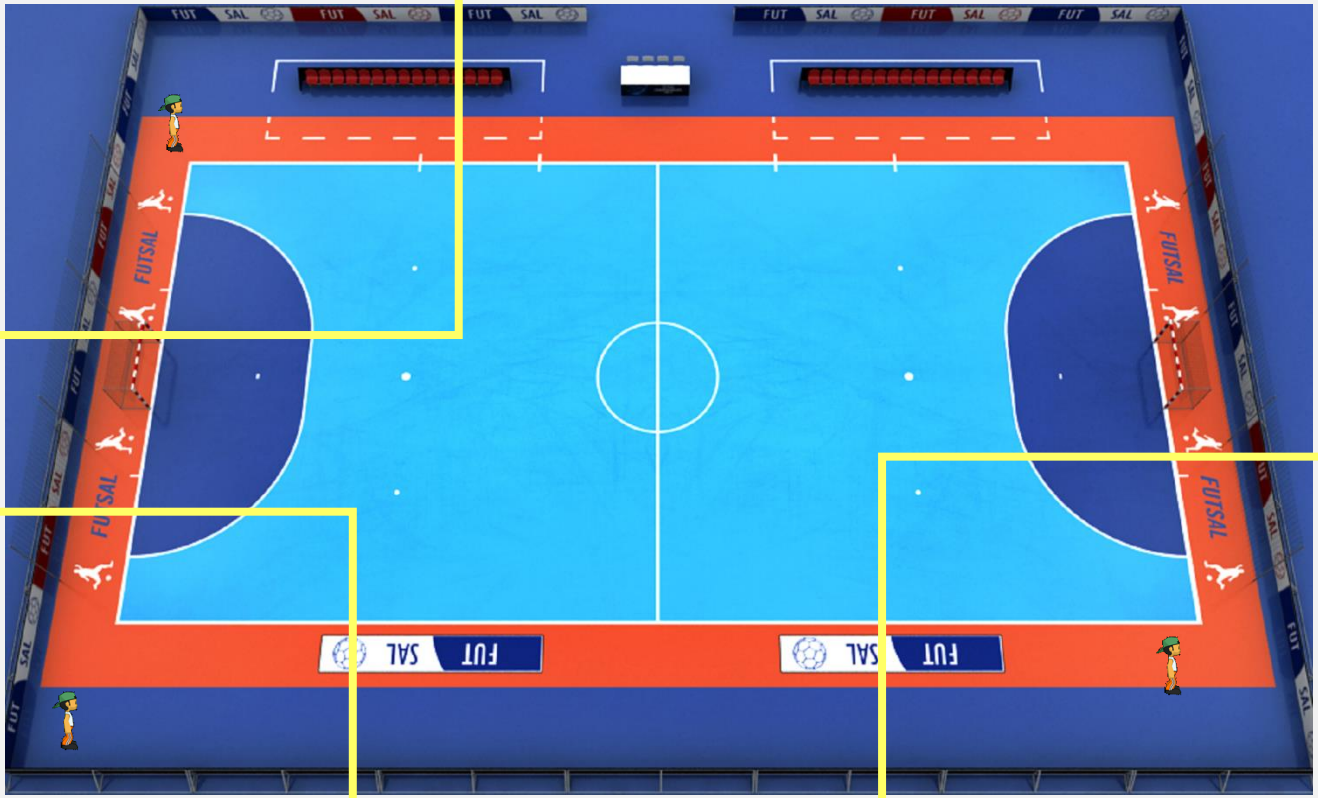
(x-canvas_width//2, y-canvas_height//2)



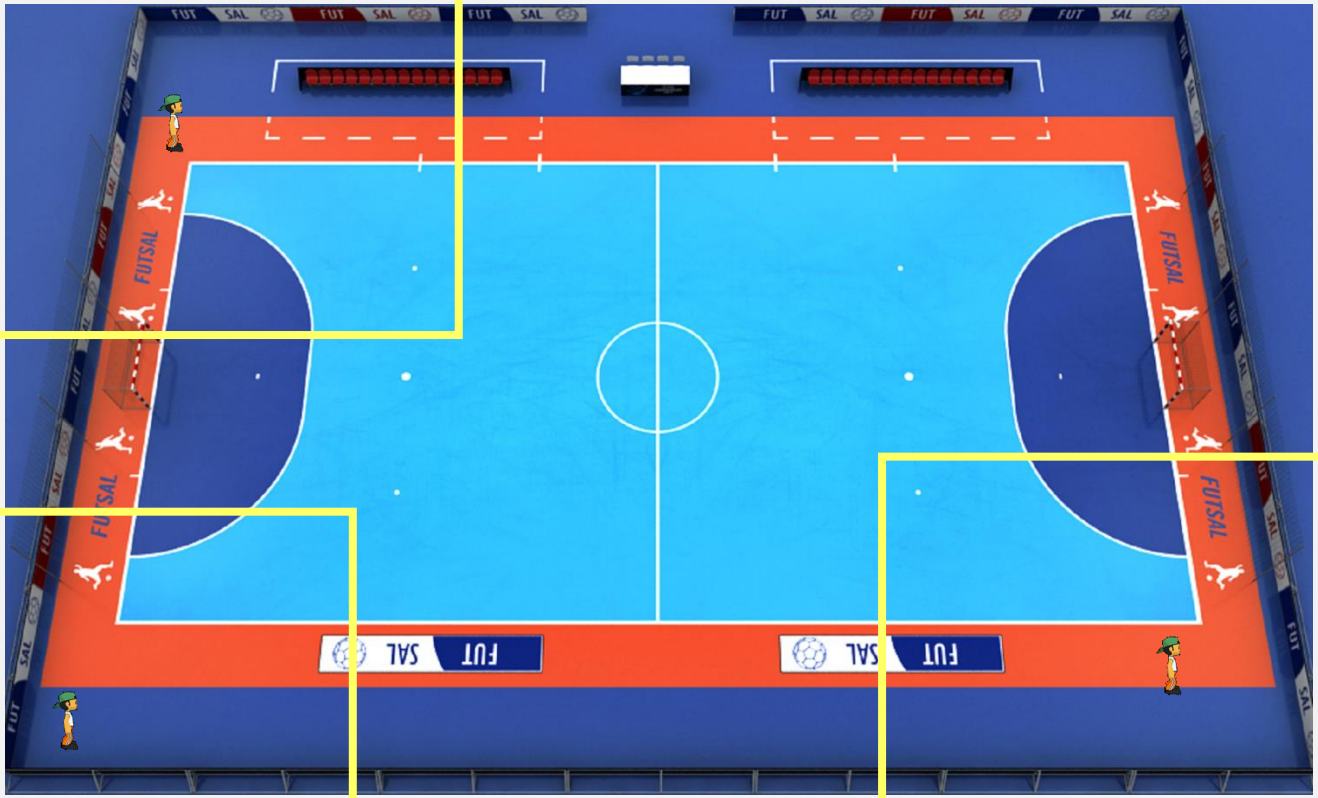
#4. 플레이어를 캔버스에 그림.

(canvas_width//2, canvas_height//2)

벗어날 경우?



벗어날 경우?



실습

상하좌우 스크롤링

clamp 함수

```
def clamp(minimum, x, maximum):
    return max(minimum, min(x, maximum))
```

main_state.py


```
from boy import Boy
from background import FixedBackground as Background

def enter():
    global boy
    boy = Boy()
    game_world.add_object(boy, 1)

    global background
    background = Background()
    game_world.add_object(background, 0)

    background.set_center_object(boy)
    boy.set_background(background)
```

background.py


```
class FixedBackground:

def set_center_object(self, boy):
    self.center_object = boy


def draw(self):
    self.image.clip_draw_to_origin(
        self.window_left, self.window_bottom,
        self.canvas_width, self.canvas_height,
        0, 0)


def update(self, frame_time):
    self.window_left = clamp(0,
        int(self.center_object.x) - self.canvas_width//2,
        self.w - self.canvas_width)
    self.window_bottom = clamp(0,
        int(self.center_object.y) - self.canvas_height//2,
        self.h - self.canvas_height)
```

boy.py (1)


```
@staticmethod
def do(boy):
    boy.frame = (boy.frame + FRAMES_PER_ACTION * ACTION_PER_TIME *
                             game_framework.frame_time) % FRAMES_PER_ACTION
    boy.x += boy.x_velocity * game_framework.frame_time
    boy.y += boy.y_velocity * game_framework.frame_time

    boy.x = clamp(boy.canvas_width // 2, boy.x, boy.bg.w - boy.canvas_width // 2)
    boy.y = clamp(boy.canvas_height // 2, boy.y, boy.bg.h - boy.canvas_height // 2)
```

boy.py (2)


```
@staticmethod
def draw(boy):
    cx, cy =  boy.canvas_width//2, boy.canvas_height//2
    if boy.x_velocity > 0:
        boy.image.clip_draw(int(boy.frame) * 100, 100, 100, 100, cx, cy)
        boy.dir = 1
    elif boy.x_velocity < 0:
        boy.image.clip_draw(int(boy.frame) * 100, 0, 100, 100, cx, cy)
        boy.dir = -1
    else:
        # if boy x_velocity == 0
        if boy.y_velocity > 0 or boy.y_velocity < 0:
            if boy.dir == 1:
                boy.image.clip_draw(int(boy.frame) * 100, 100, 100, 100, cx, cy)
            else:
                boy.image.clip_draw(int(boy.frame) * 100, 0, 100, 100, cx, cy)
        else:
            # boy is idle
            if boy.dir == 1:
                boy.image.clip_draw(int(boy.frame) * 100, 300, 100, 100, cx, cy)
            else:
                boy.image.clip_draw(int(boy.frame) * 100, 200, 100, 100, cx, cy)
```

main_state.py

```
def enter():
    global boy
    boy = Boy()
    game_world.add_object(boy, 1)

    global background
    background = Background()
    game_world.add_object(background, 0)

    background.set_center_object(boy)
    boy.set_background(background)
```

상호 참조를 설정.

boy는 background 의 정보가 필요.

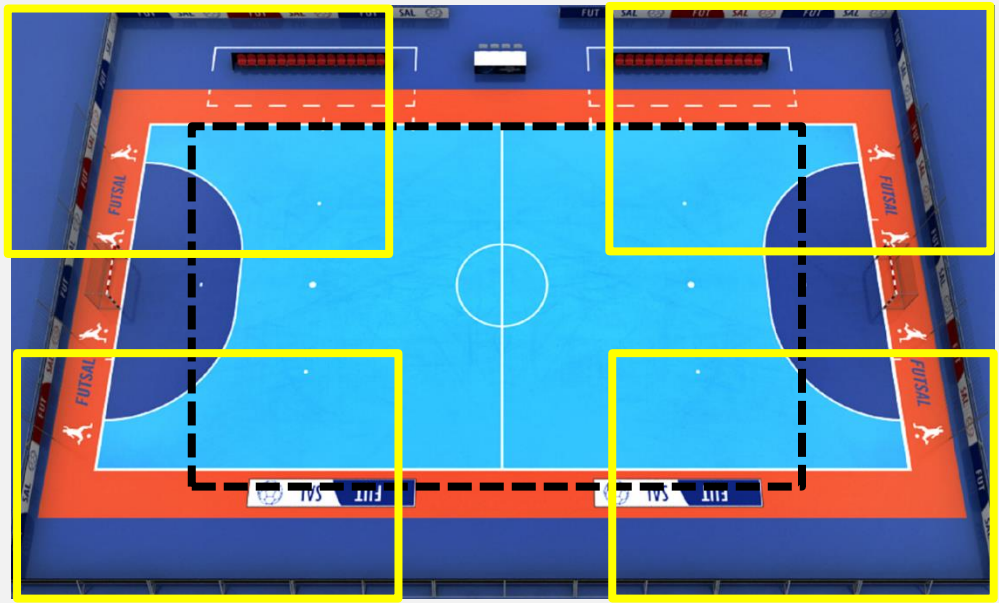
background는 boy의 정보가 필요.

boy.py

boy의 위치는 속도와 시간을 통해서 월드상의 좌표로 계산해야 함.

```
@staticmethod
def do(boy):
    boy.frame = (boy.frame + FRAMES_PER_ACTION * ACTION_PER_TIME *
                             game_framework.frame_time) % FRAMES_PER_ACTION
    boy.x += boy.x_velocity * game_framework.frame_time
    boy.y += boy.y_velocity * game_framework.frame_time

    boy.x = clamp(boy.canvas_width // 2, boy.x, boy.bg.w - boy.canvas_width // 2)
    boy.y = clamp(boy.canvas_height // 2, boy.y, boy.bg.h - boy.canvas_height // 2)
```

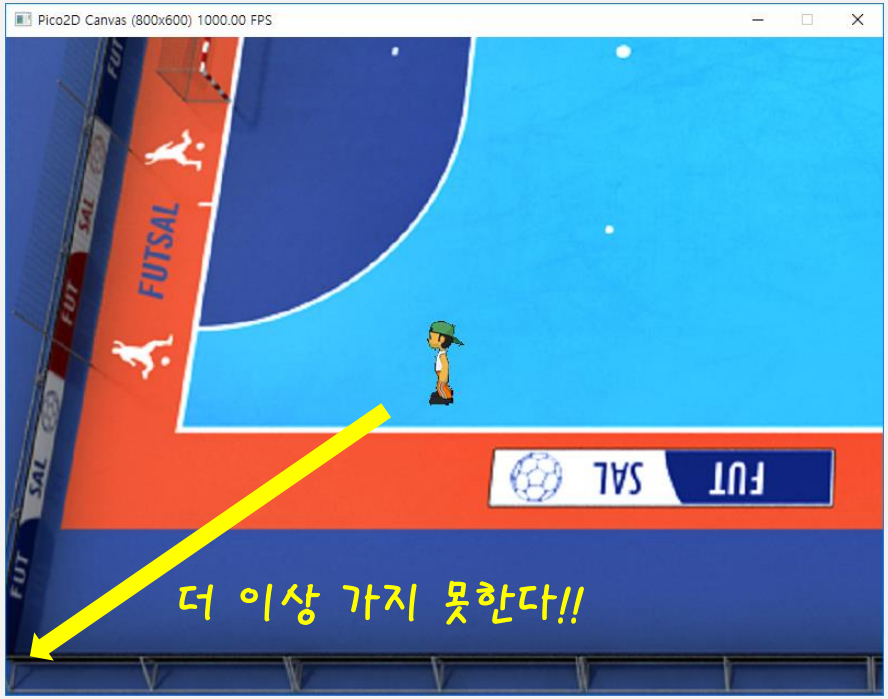
월드(구장이미지) 상에서, 검정 점선영역안으로만 이동 가능



background.py

```
class FixedBackground:

def set_center_object(self, boy):
    self.center_object = boy


def draw(self):
    self.image.clip_draw_to_origin(
        self.window_left, self.window_bottom,
        self.canvas_width, self.canvas_height,
        0, 0)


def update(self, frame_time):
    self.window_left = clamp(0,
        int(self.center_object.x) - self.canvas_width//2,
        self.w - self.canvas_width)
    self.window_bottom = clamp(0,
        int(self.center_object.y) - self.canvas_height//2,
        self.h - self.canvas_height)
```

window의 left x 좌표의 최대값은, 이미지의 너비에서 화면의 너비를 뺀 값.



플레이어의 화면상의 좌표 계산

boy.py

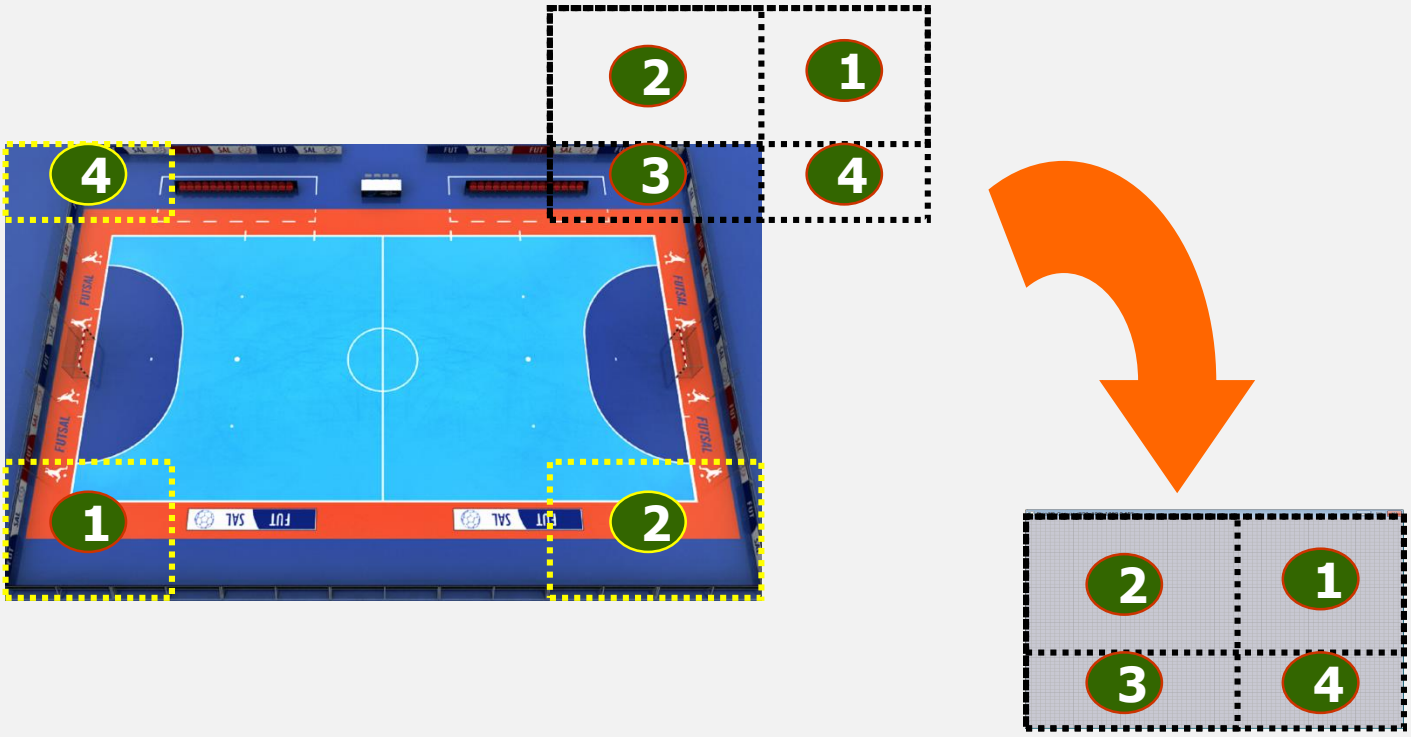

```
@staticmethod
def do(boy):
    boy.frame = (boy.frame + FRAMES_PER_ACTION * ACTION_PER_TIME *
                             game_framework.frame_time) % FRAMES_PER_ACTION
    boy.x += boy.x_velocity * game_framework.frame_time
    boy.y += boy.y_velocity * game_framework.frame_time

    boy.x = clamp(0, boy.x, boy.bg.w)
    boy.y = clamp(0, boy.y, boy.bg.h)
```

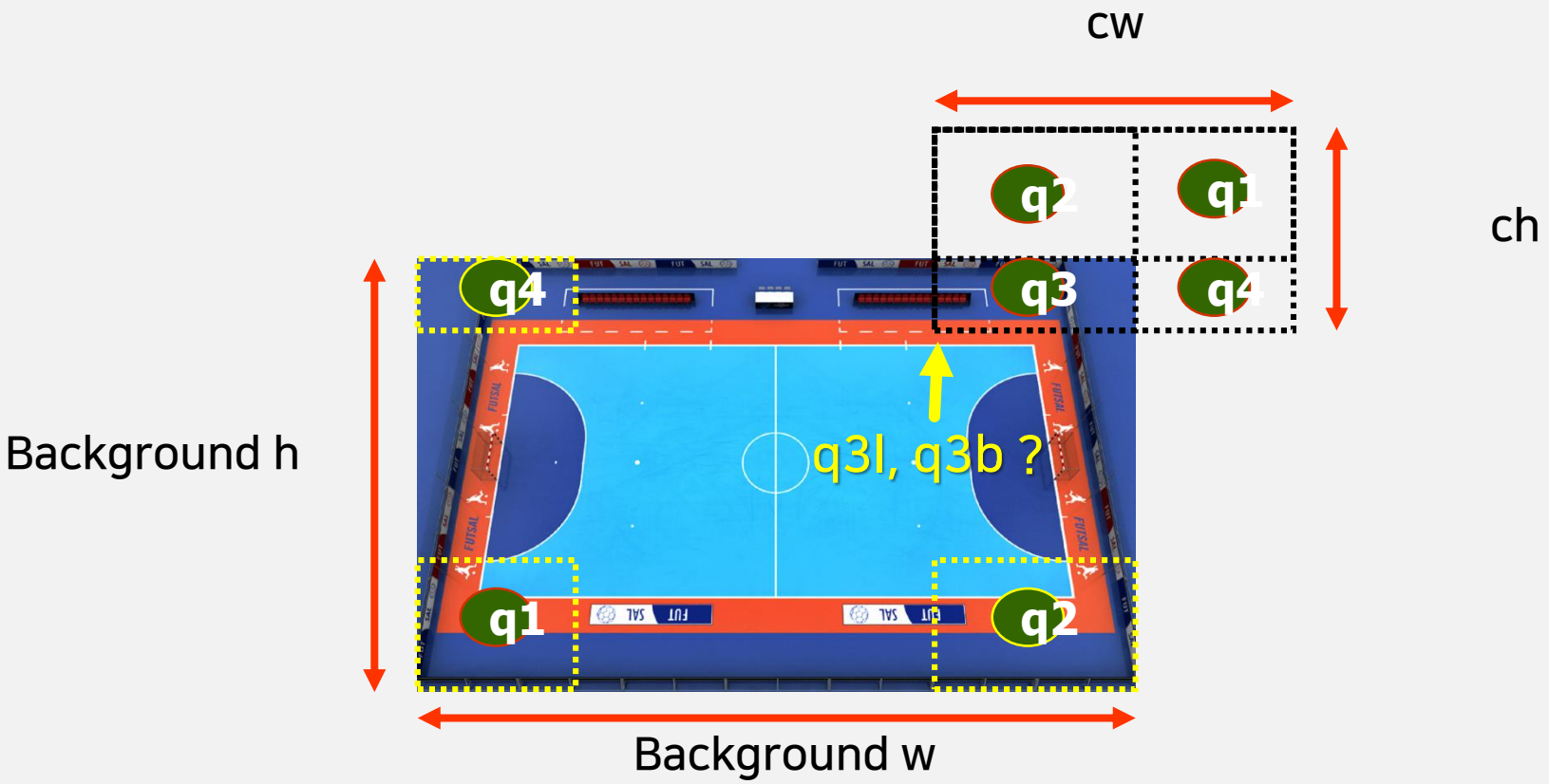
```
@staticmethod
def draw(boy):
    cx, cy = boy.x - boy.bg.window_left, boy.y - boy.bg.window_bottom
    if boy.x_velocity > 0:
```

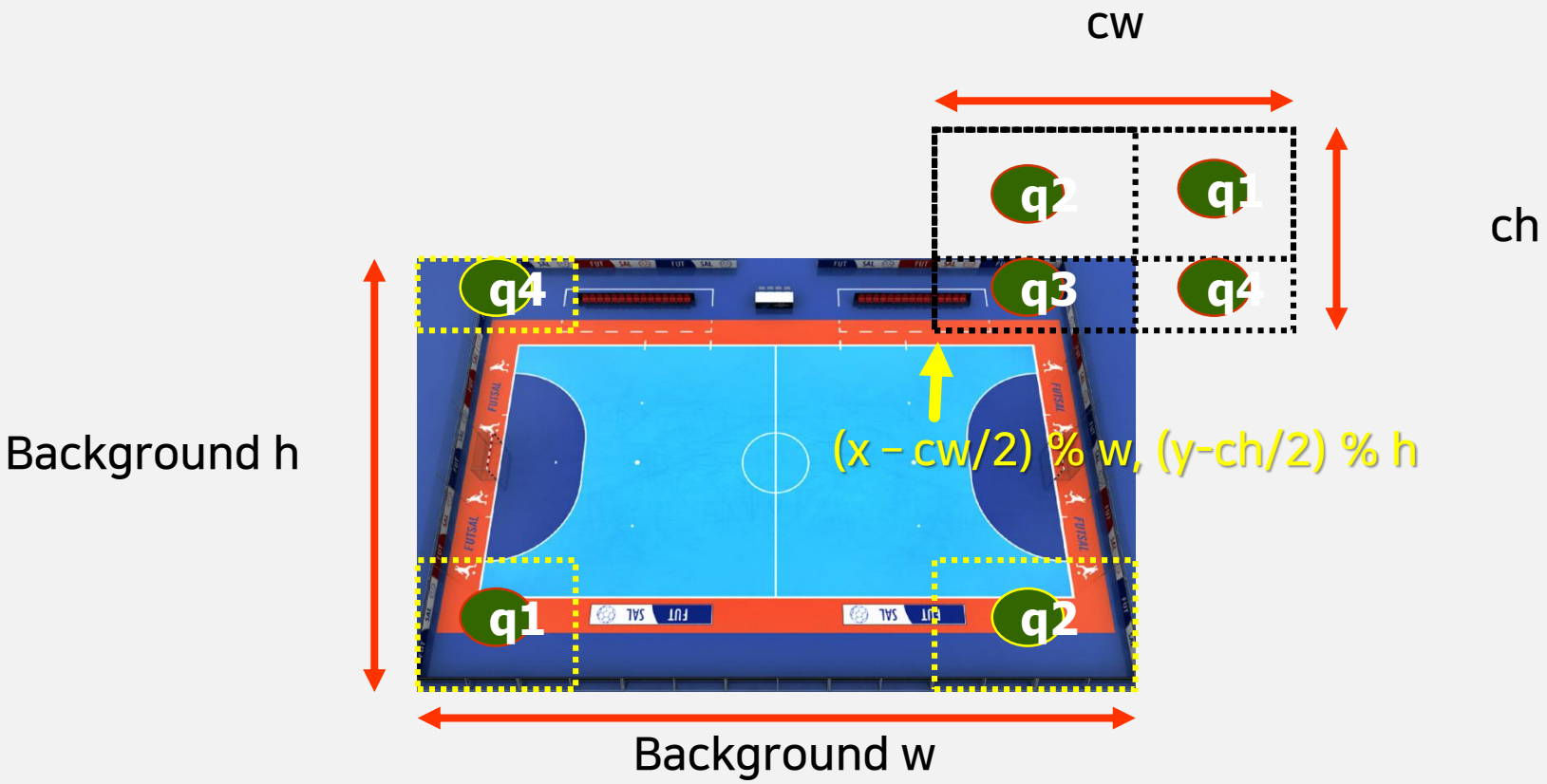
실습

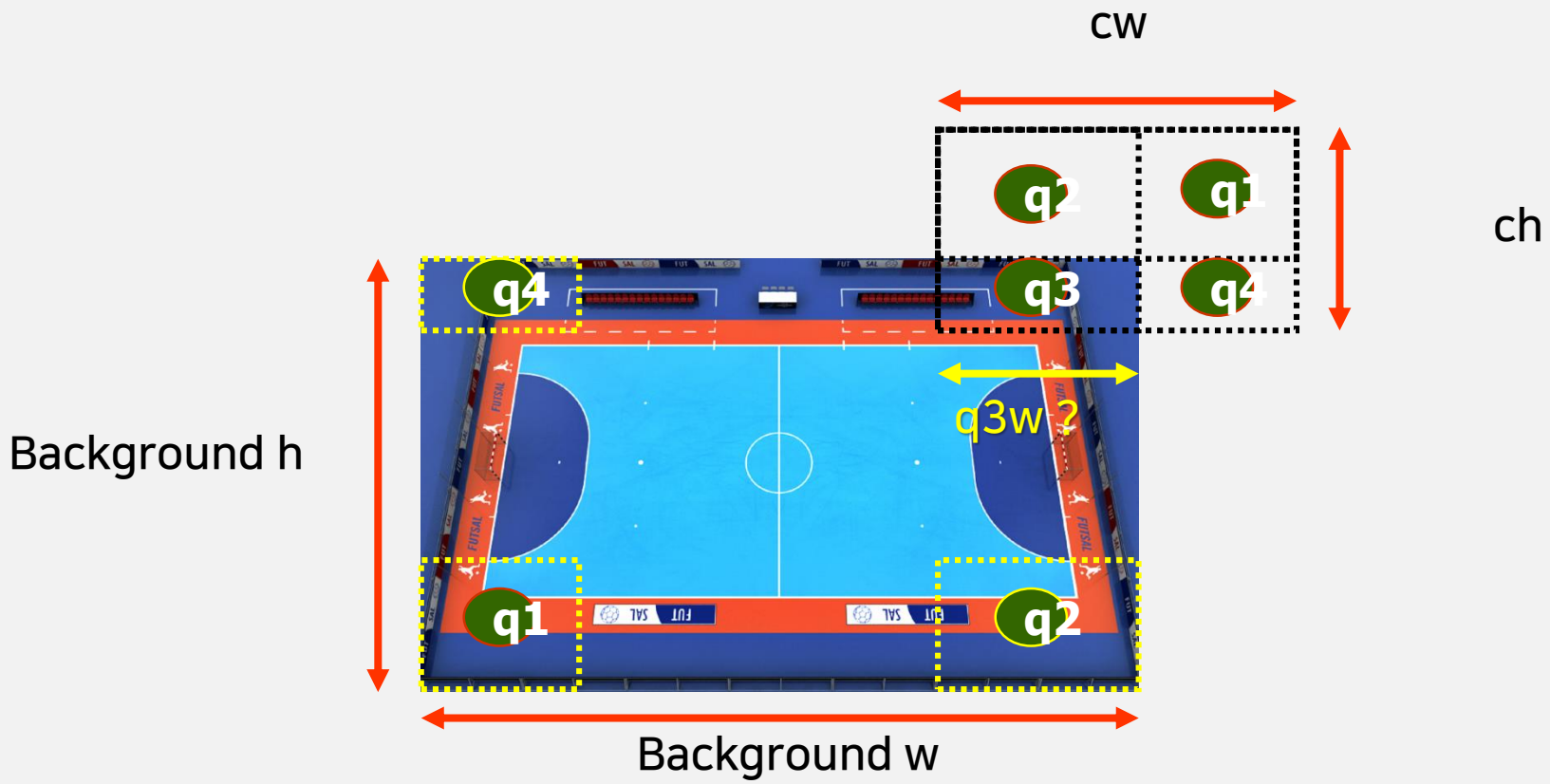
상하좌우 무한 스크롤링

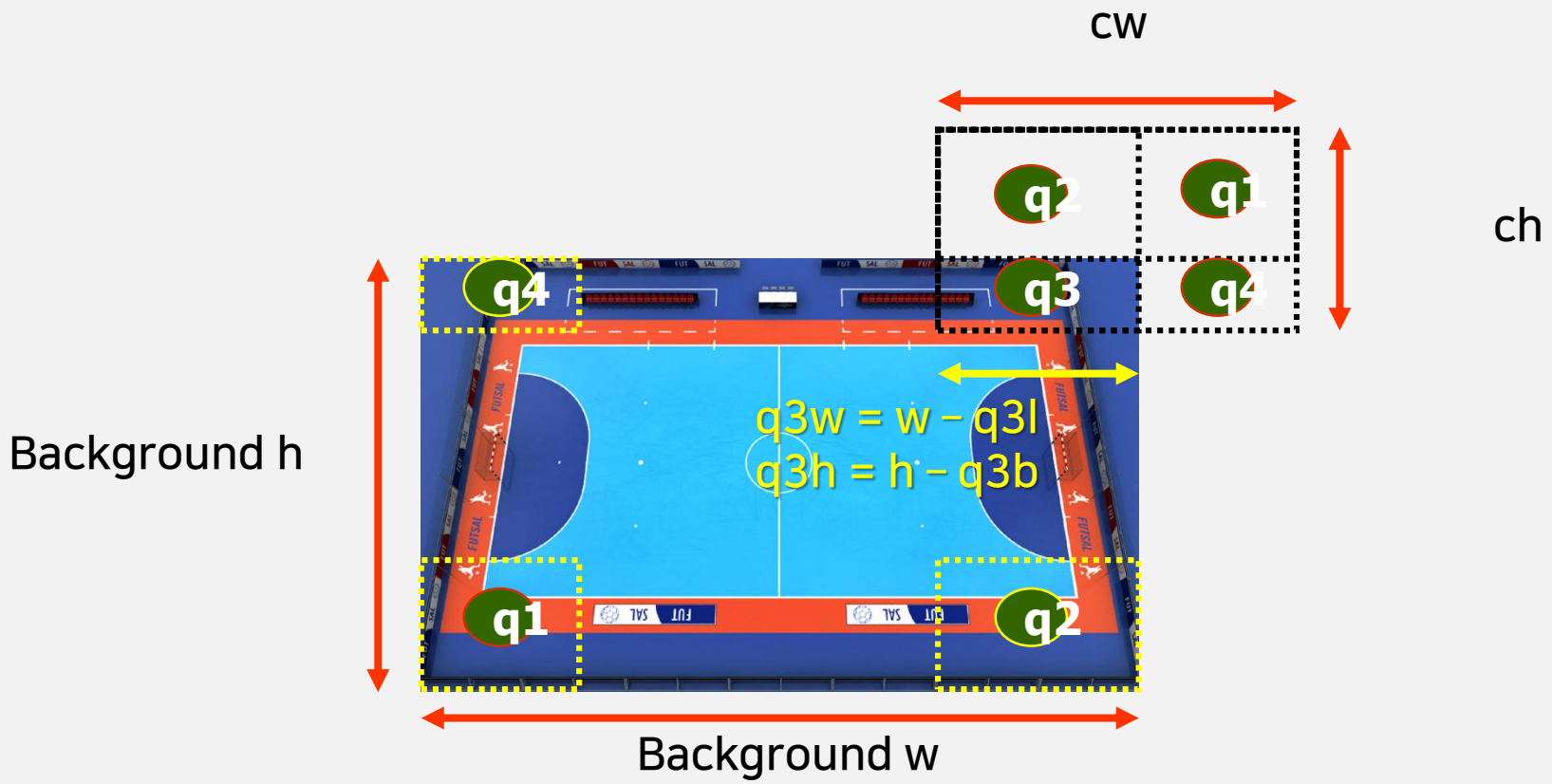
상하좌우 무한스크롤링 공식











main_state.py


```
#from background import FixedBackground as Background
from background import InfiniteBackground as Background
```

boy.py

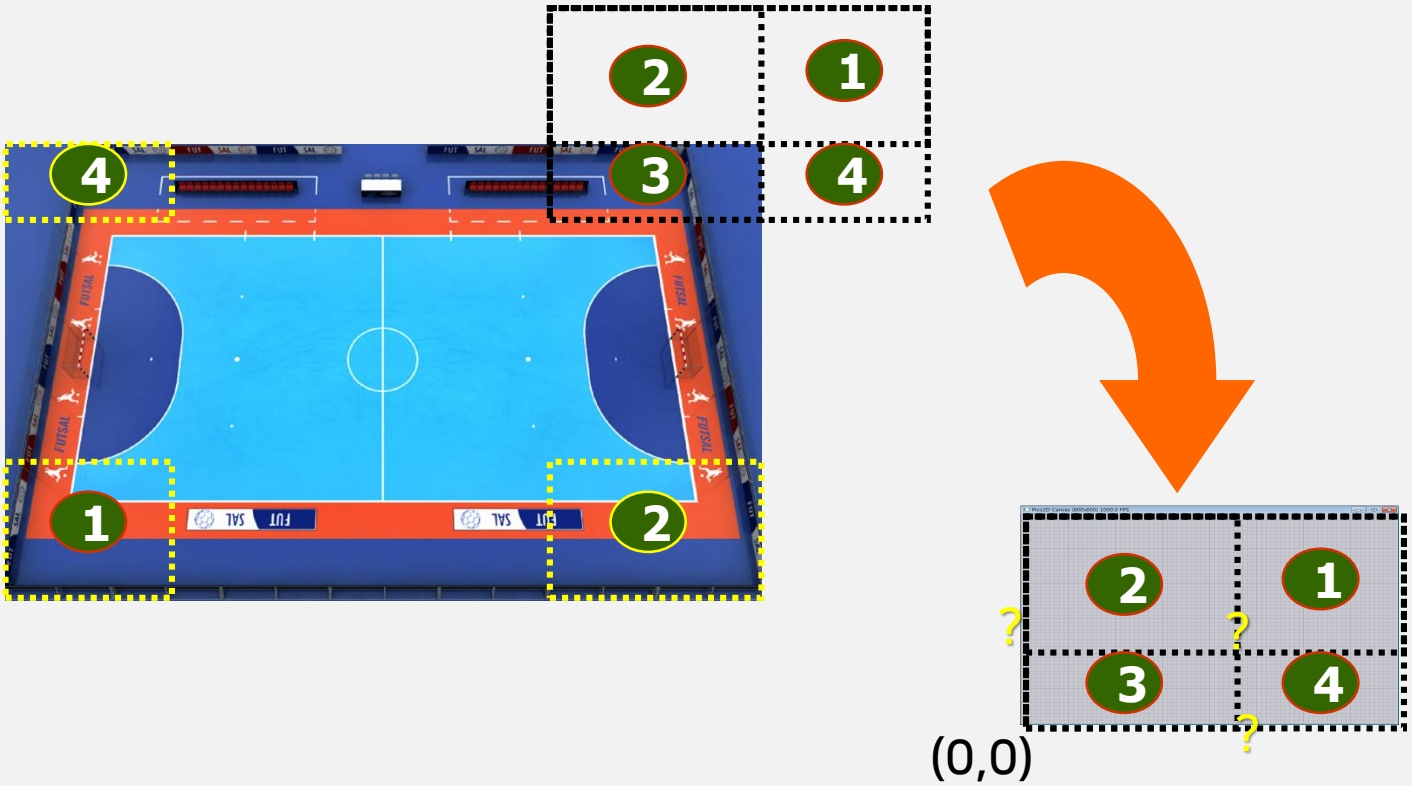

```
@staticmethod
def do(boy):
    boy.frame = (boy.frame + FRAMES_PER_ACTION * ACTION_PER_TIME *
                             game_framework.frame_time) % FRAMES_PER_ACTION
    boy.x += boy.x_velocity * game_framework.frame_time
    boy.y += boy.y_velocity * game_framework.frame_time

    boy.x = clamp(0, boy.x, boy.bg.w)
    boy.y = clamp(0, boy.y, boy.bg.h)
```

```
@staticmethod
def draw(boy):
    cx, cy =  boy.canvas_width//2, boy.canvas_height//2
    if boy.x_velocity > 0:
```

background.py


```
class InfiniteBackground:

    def update(self, frame_time):
        #     quadrant 3
        self.q3l = (int(self.center_object.x) - self.canvas_width // 2) % self.w
        self.q3b = (int(self.center_object.y) - self.canvas_height // 2) % self.h
        self.q3w = clamp(0, self.w - self.q3l, self.w)
        self.q3h = clamp(0, self.h - self.q3b, self.h)

        #     quadrant 2
        self.q2l = ?
        self.q2b = ?
        self.q2w = ?
        self.q2h = ?

        #     quadrant 4
        self.q4l = ?
        self.q4b = ?
        self.q4w = ?
        self.q4h = ?

        #     quadrant 1
        self.q1l = ?
        self.q1b = ?
        self.q1w = ?
        self.q1h = ?
```

상하좌우 무한스크롤링 공식



background.py

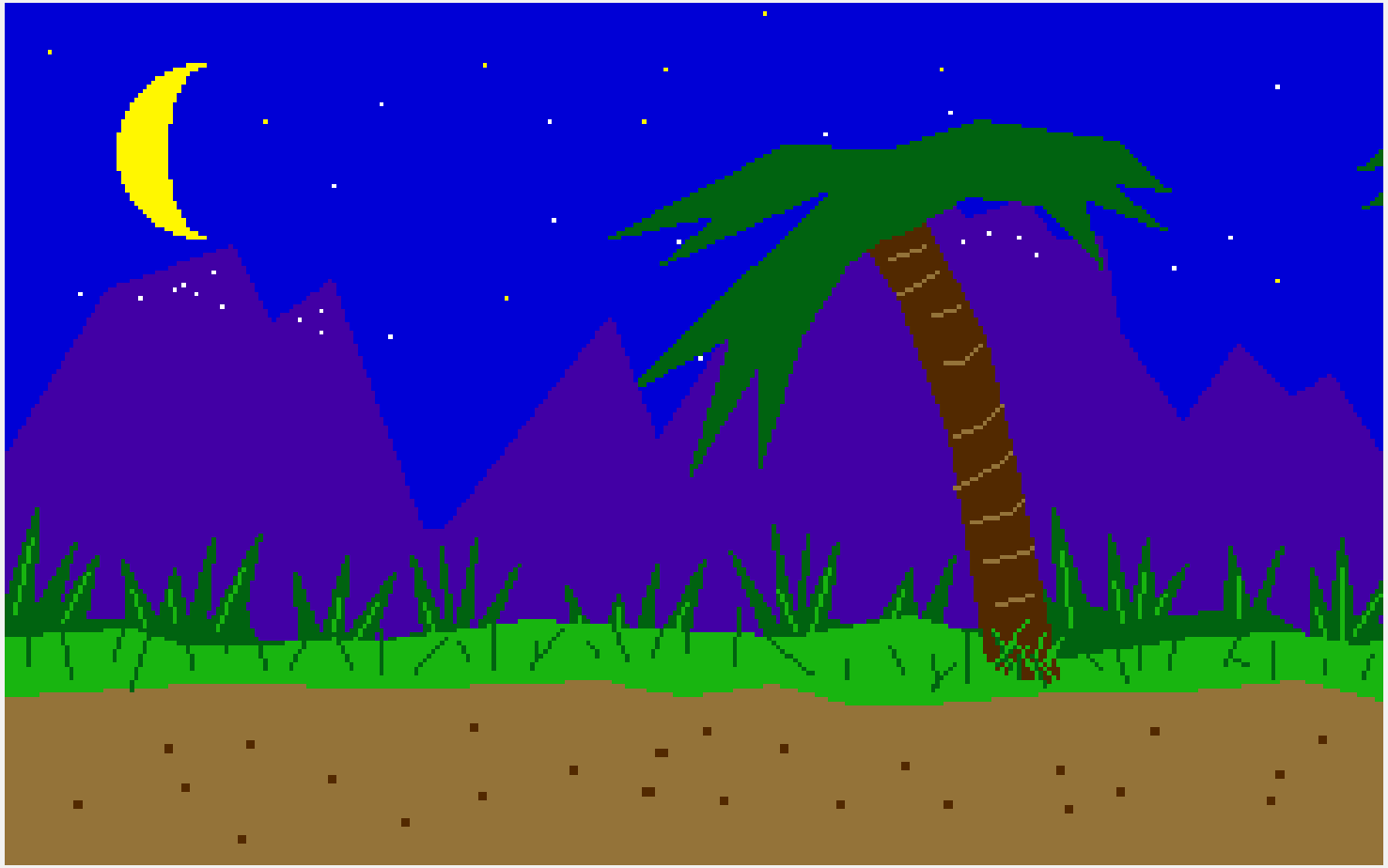

```
class InfiniteBackground:

    def draw(self):
        self.image.clip_draw_to_origin(self.q3l, self.q3b, self.q3w, self.q3h, 0, 0)
        self.image.clip_draw_to_origin(self.q2l, self.q2b, self.q2w, self.q2h, ?, ?)
        self.image.clip_draw_to_origin(self.q4l, self.q4b, self.q4w, self.q4h, ?, ?)
        self.image.clip_draw_to_origin(self.q1l, self.q1b, self.q1w, self.q1h, ?, ?)
```

시차(視差) 스크롤링(Parallax Scrolling)

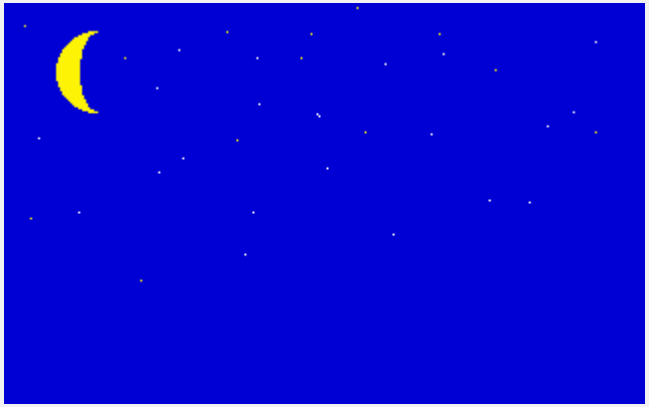
- 물체와 눈의 거리에 따라, 물체의 이동속도가 달라보이는 효과를 이용하여, 3차원 배경을 흉내내는 기법.

- 1982년 “Moon Patrol”이라는 게임에서 세계 최초로 사용됨.

• 밤하늘, 뒷산, 앞산의 스크롤링 속도를 다르게 함으로써, 3차원적인 깊이 효과를 구현.



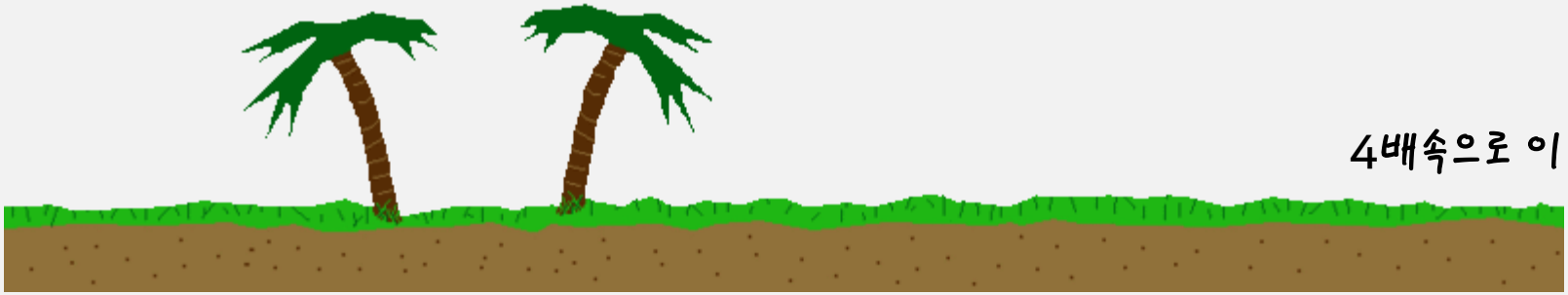
시차 스크롤링 방법



1배속으로 이동

2배속으로 이동

3배속으로 이동



4배속으로 이동

정답

```
def draw(self):
    # fill here
    self.image.clip_draw_to_origin(self.q3l, self.q3b, self.q3w, self.q3h, 0, 0)                       # quadrant 3
    self.image.clip_draw_to_origin(self.q2l, self.q2b, self.q2w, self.q2h, 0, self.q3h)                # quadrant 2
    self.image.clip_draw_to_origin(self.q4l, self.q4b, self.q4w, self.q4h, self.q3w, 0)                # quadrant 4
    self.image.clip_draw_to_origin(self.q1l, self.q1b, self.q1w, self.q1h, self.q3w, self.q3h)         # quadrant 1
```

```
def update(self):

    # quadrant 3
    self.q3l = (int(self.center_object.x) - self.canvas_width // 2) % self.w
    self.q3b = (int(self.center_object.y) - self.canvas_height // 2) % self.h
    self.q3w = clamp(0, self.w - self.q3l, self.w)
    self.q3h = clamp(0, self.h - self.q3b, self.h)


    # quadrant 2
    self.q2l = self.q3l
    self.q2b = 0
    self.q2w = self.q3w
    self.q2h = self.canvas_height - self.q3h


    # quadrand 4
    self.q4l = 0
    self.q4b = self.q3b
    self.q4w = self.canvas_width - self.q3w
    self.q4h = self.q3h


    # quadrand 1
    self.q1l = 0
    self.q1b = 0
    self.q1w = self.q4w
    self.q1h = self.q2h
```