Lecture #19. 스크롤링 (2)

2D 게임 프로그래밍

이대현 교수

한국공학대학교
TECH UNIVERSITY OF KOREA

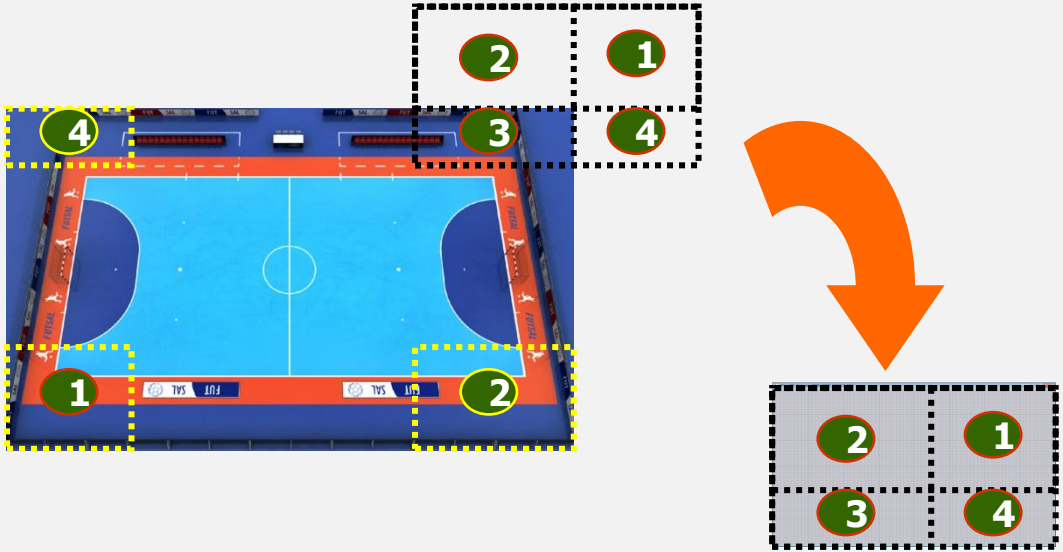
학습 내용

- 무한 스크롤링
- 타일맵 기반 스크롤링
- 시차 스크롤링

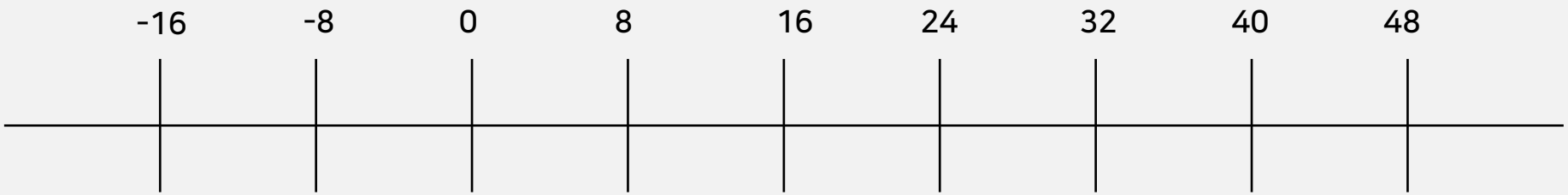
실습

상하좌우 무한 스크롤링

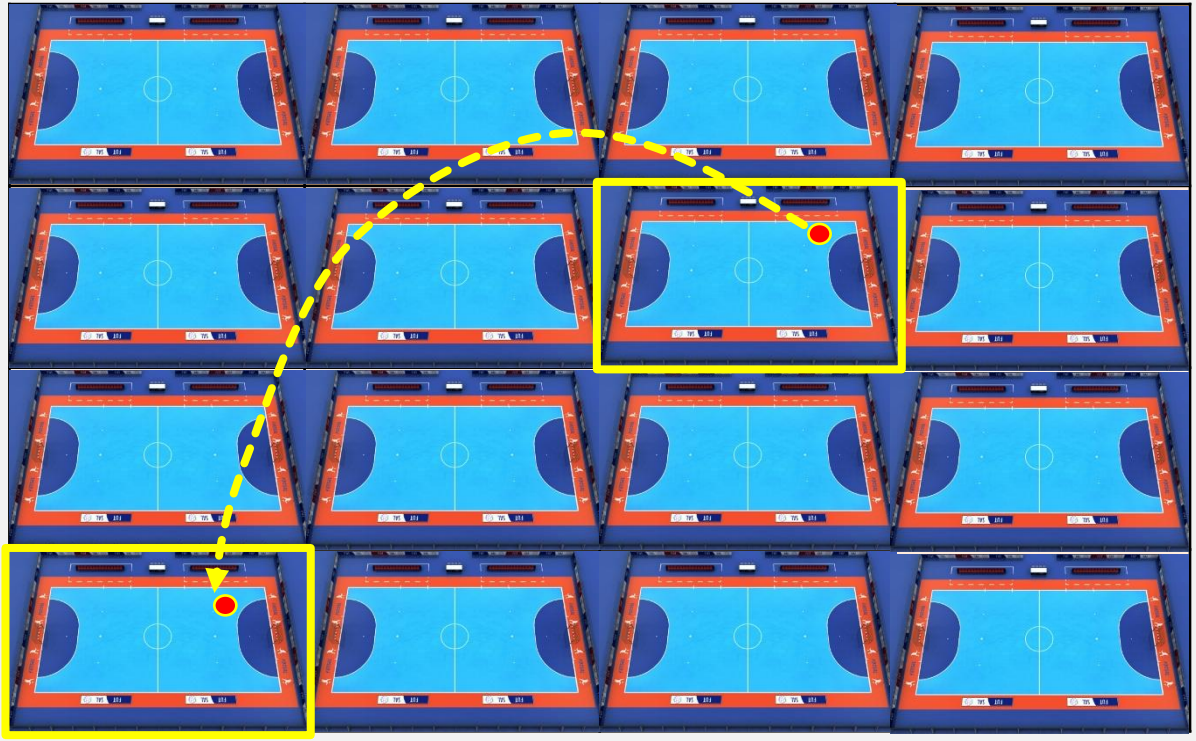
상하좌우 무한스크롤링 공식

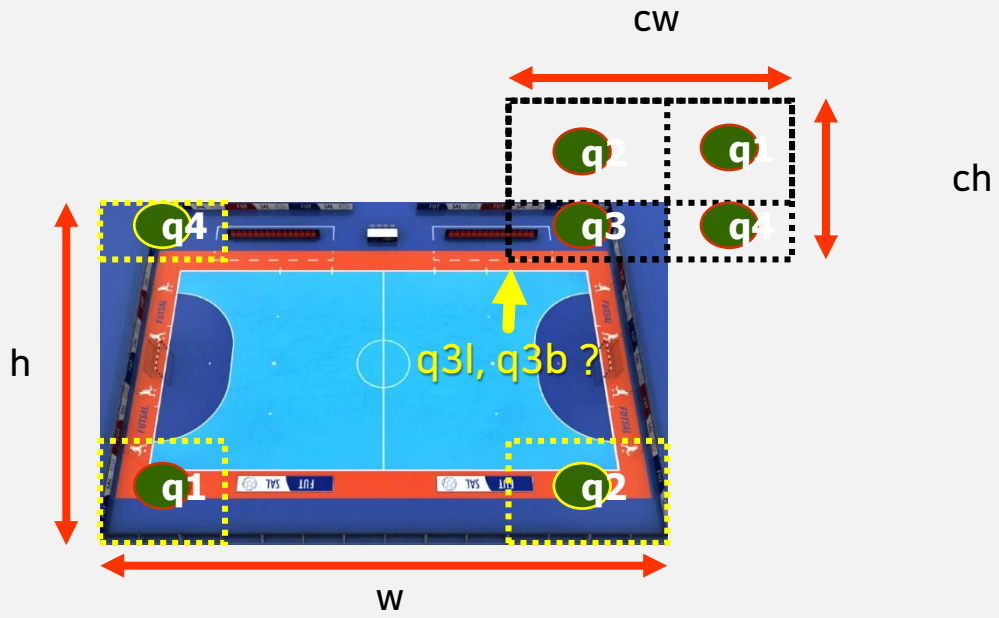


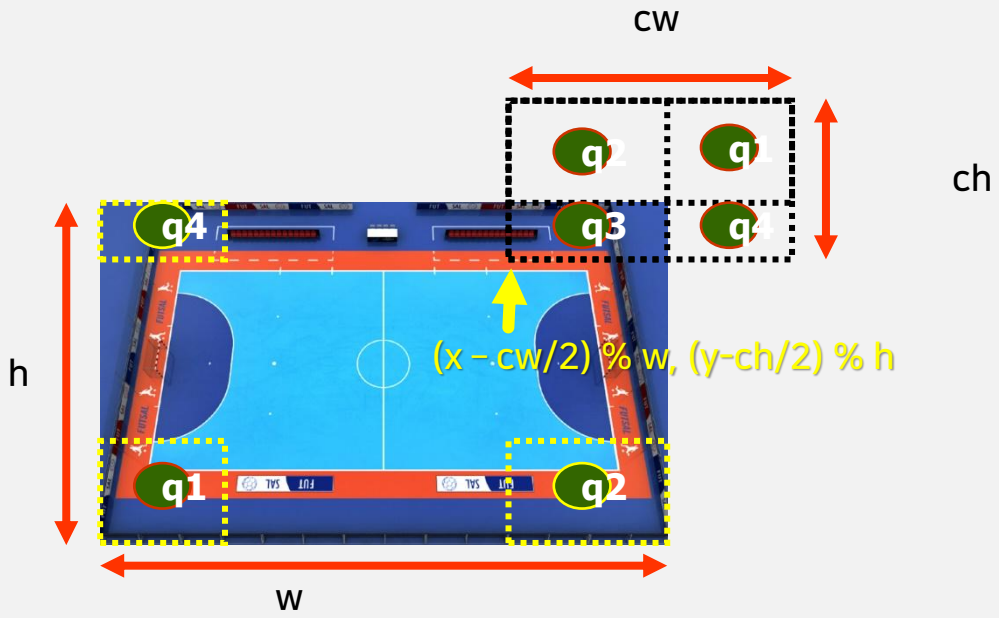
% 나머지 연산의 기능 – 무한 공간을 유한 범위로 축소

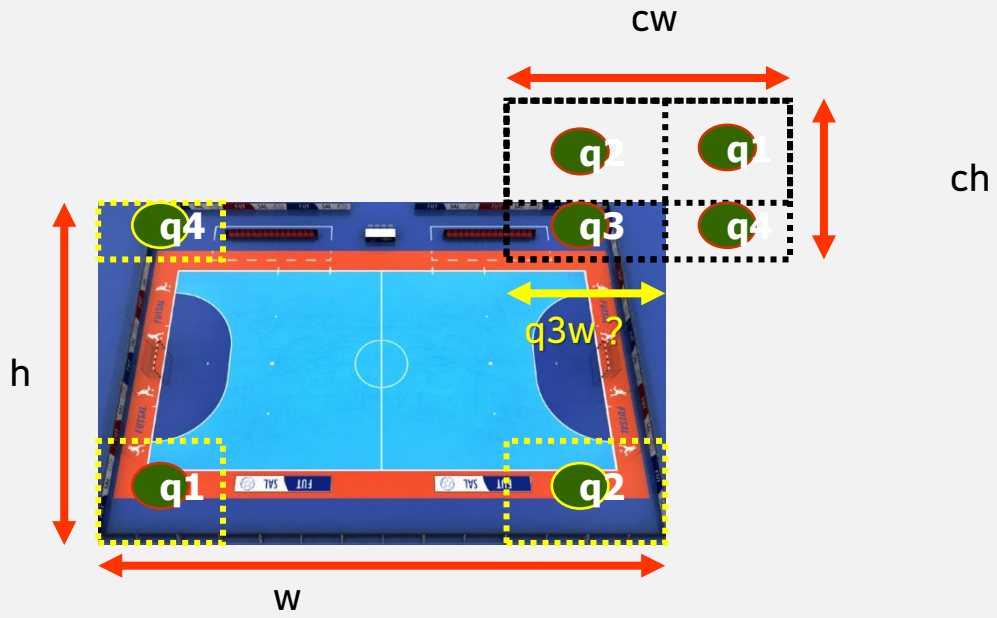


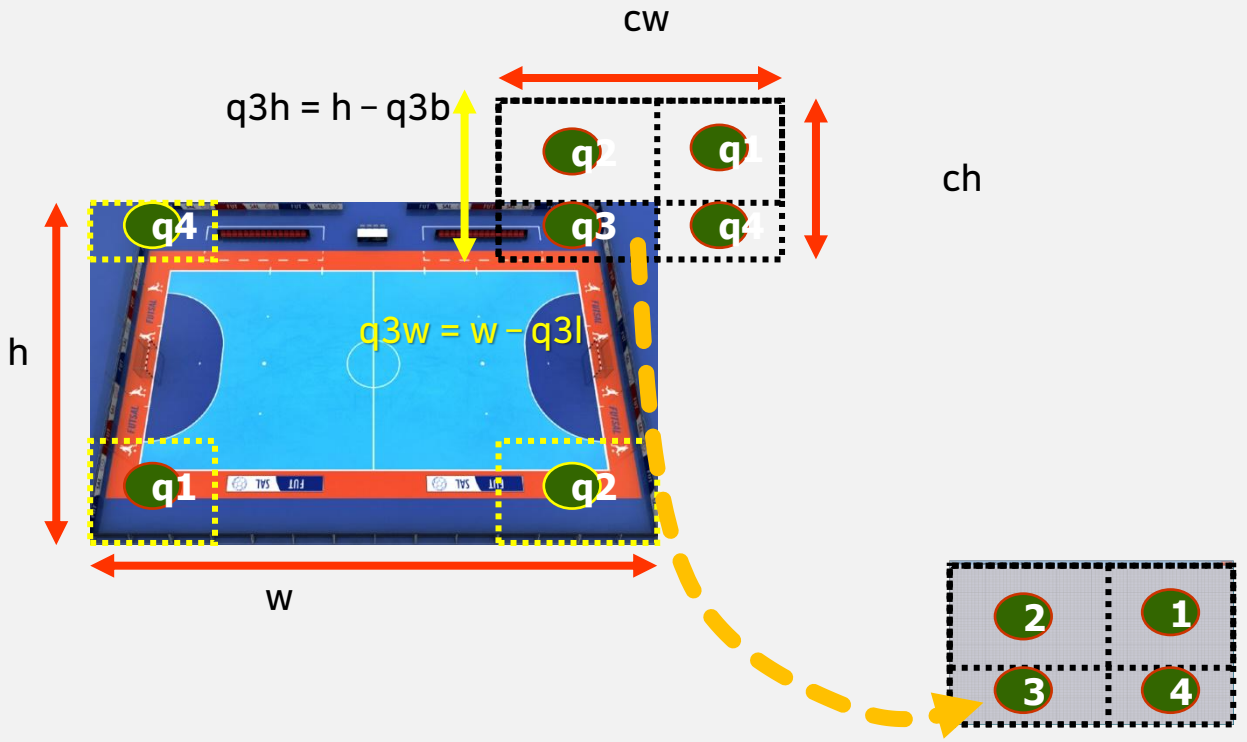
2차원 공간의 모든 점도 % 연산을 통해 특정 사각형 내로 가져올 수 있음.











play_mode.py


```
# from court import Court
# from court import TileCourt as Court
from court import InfiniteCourt as Court
```

boy.py


```
def update(self):

    self.x = clamp(50.0, self.x, server.background.w - 50.0)
    self.y = clamp(50.0, self.y, server.background.h - 50.0)
```

```
def draw(self):
    sx, sy = get_canvas_width() // 2, get_canvas_height() // 2
    self.image.clip_draw(int(self.frame) * 100, self.action * 100, 100, 100, sx, sy)
```

court.py


```
class InfiniteCourt:

    def update(self, frame_time):

        # quadrant 3
        self.q3l = (int(common.boy.x) - self.cw // 2) % self.w
        self.q3b = (int(common.boy.y) - self.ch // 2) % self.h
        self.q3w = clamp(0, self.w - self.q3l, self.cw)
        self.q3h = clamp(0, self.h - self.q3b, self.ch)

        #     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 = ?
```

court.py

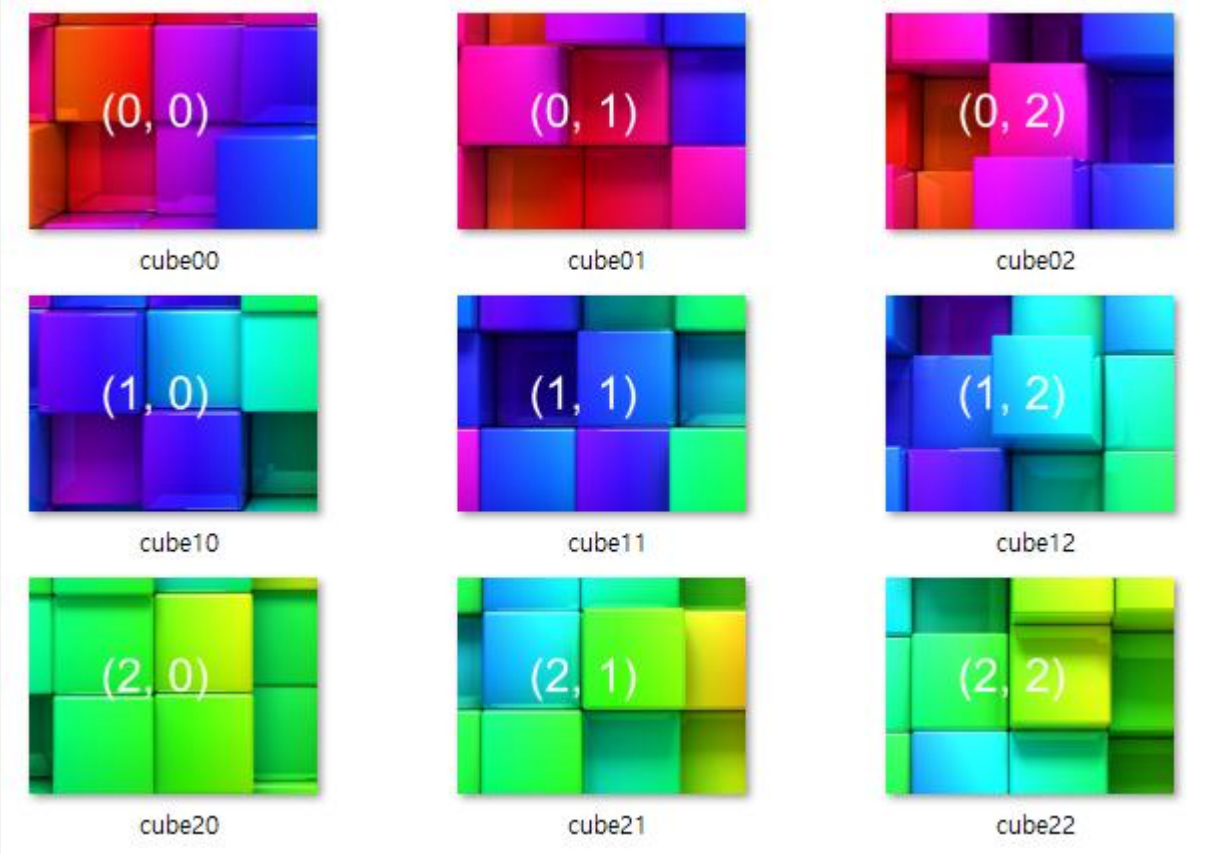

```
class InfiniteCourt:

    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, ?, ?)
```

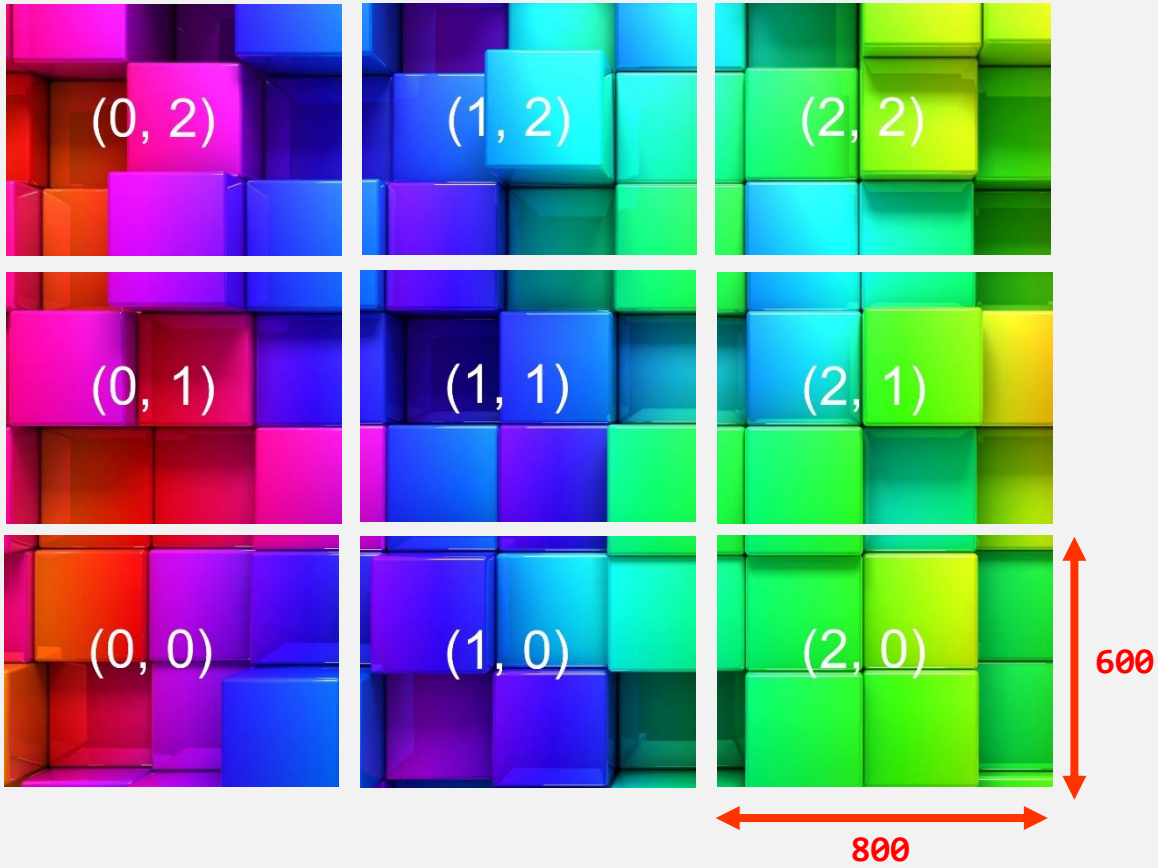
실습

상하좌우 스크롤링 (타일링 배경)

Tile image



타일맵 구조



play_mode.py


```
from boy import Boy
# from court import Court
from court import TileCourt as Court
```

court.py (1)


```
class TileCourt:
    def __init__(self):
        self.cw = get_canvas_width()
        self.ch = get_canvas_height()
        self.w = 800 * 3
        self.h = 600 * 3

        # fill here
        self.tiles = [ [ load_image('cube%d%d.png' % (x, y)) for x in range(3) ] for y in range(3) ]
```

court.py (2)

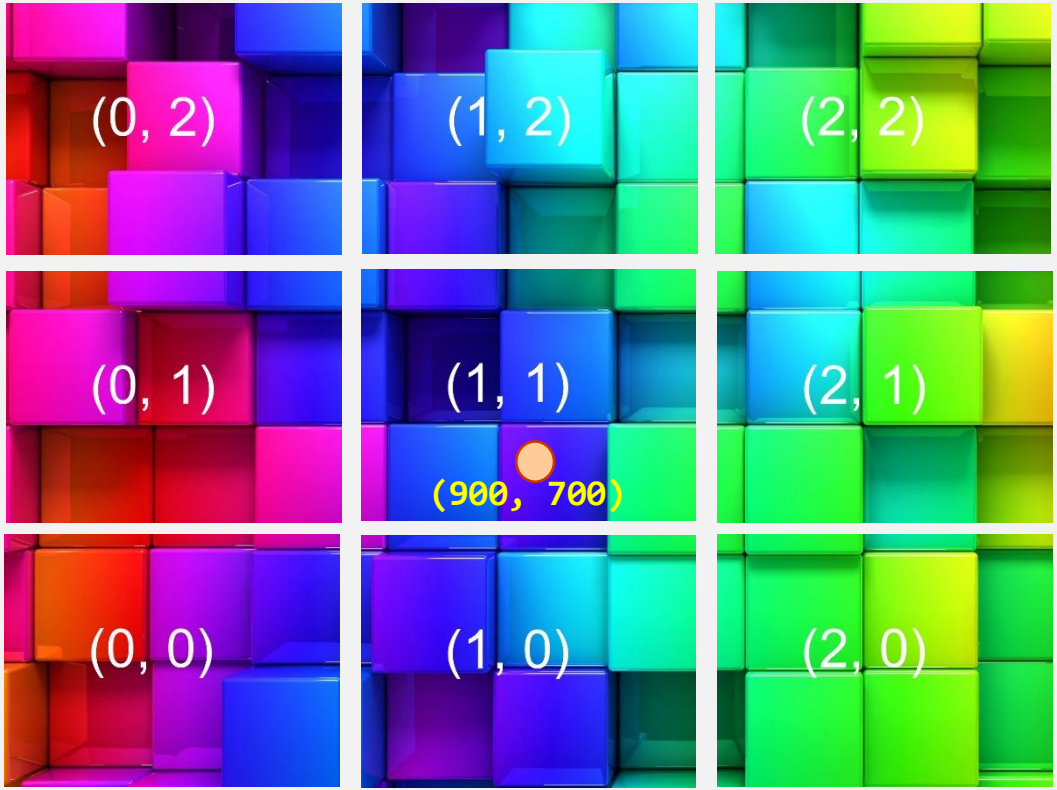

```
def draw(self):
    self.window_left = clamp(0, int(common.boy.x) - self.cw // 2, self.w - self.cw - 1)
    self.window_bottom = clamp(0, int(common.boy.y) - self.ch // 2, self.h - self.ch - 1)

    tile_left = self.window_left // 800
    tile_right = (self.window_left + self.cw) // 800
    left_offset = self.window_left % 800

    tile_bottom = self.window_bottom // 600
    tile_top = (self.window_bottom + self.ch) // 600
    bottom_offset = self.window_bottom % 600

    for ty in range(tile_bottom, tile_top+1):
        for tx in range(tile_left, tile_right+1):
            self.tiles[ty][tx].draw_to_origin(-left_offset + (tx-tile_left)*800, -bottom_offset+(ty-tile_bottom)*600)
```

전체 맵 좌표로부터, 타일맵 좌표의 계산



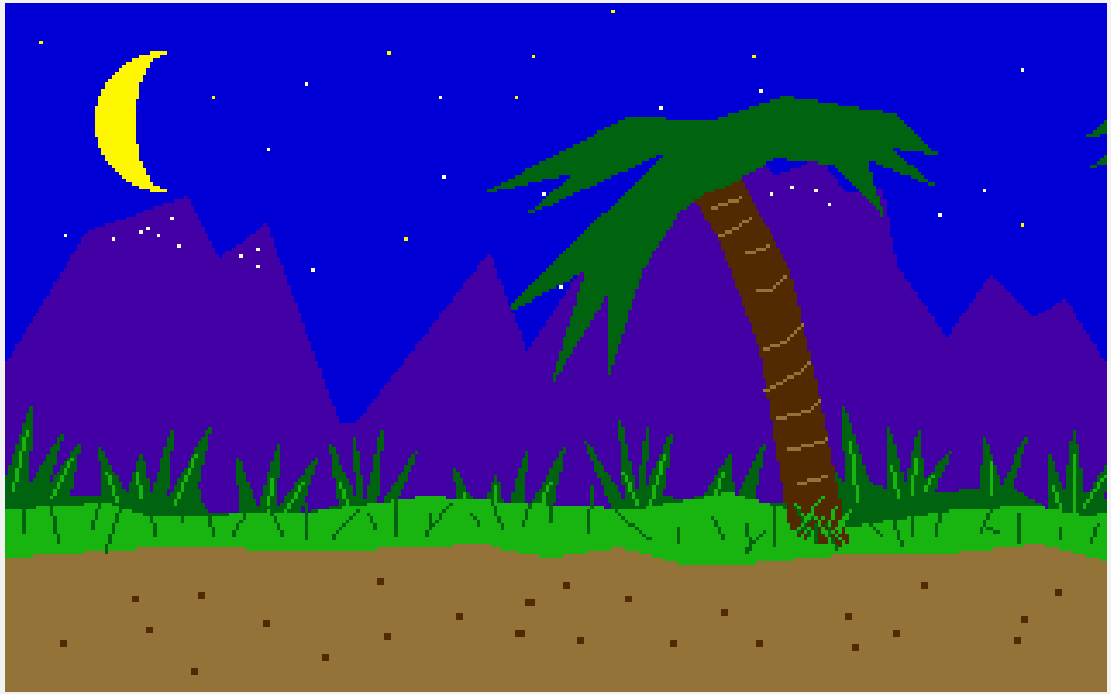

```
tx = 900 // 800
ty = 700 // 600
```

시차(視差) 스크롤링(Parallax Scrolling)

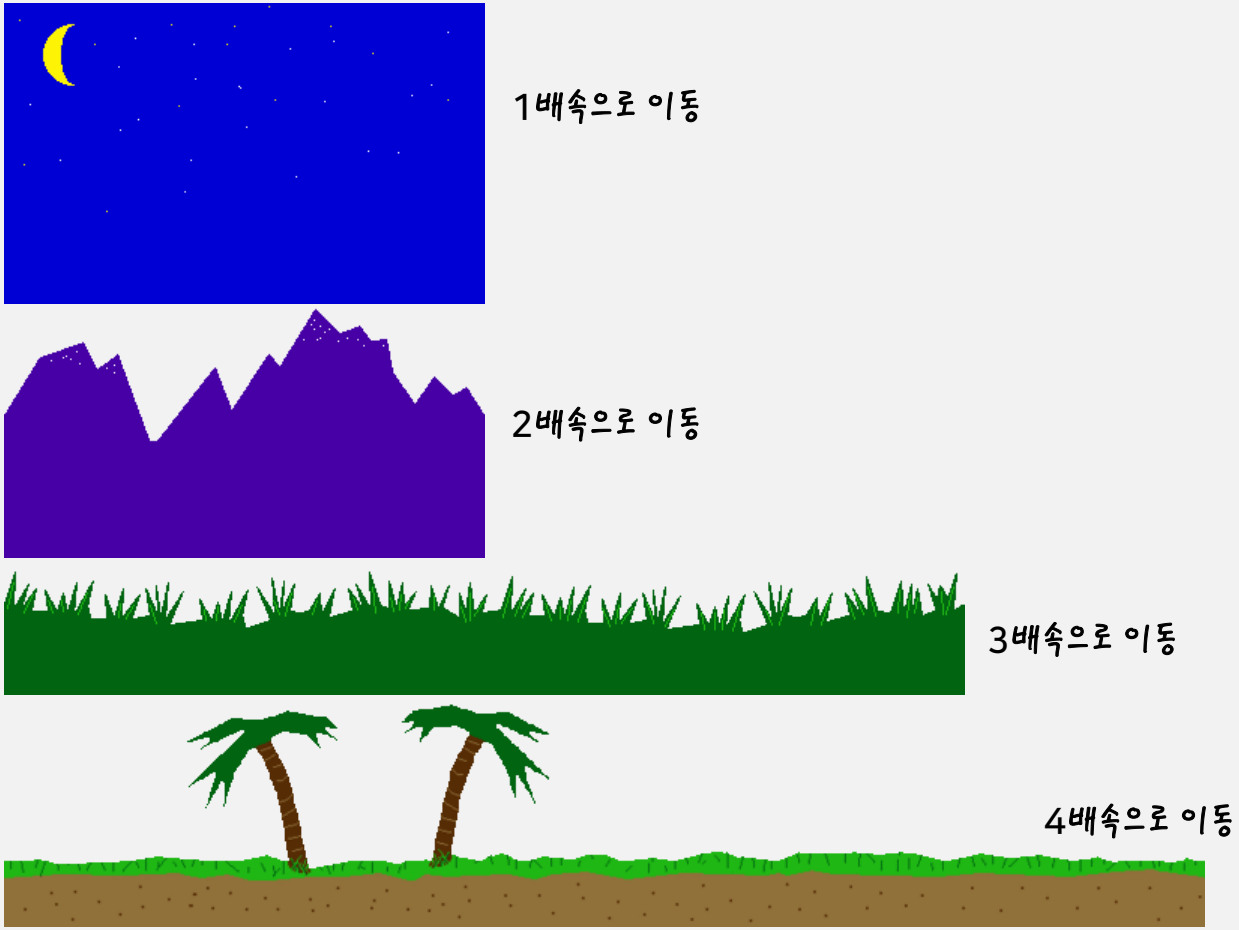
- 물체와 눈의 거리에 따라, 물체의 이동속도가 달라보이는 효과를 이용하여, 3차원 배경을 흉내 내는 기법.

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

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



시차 스크롤링 방법



court.py

```
def draw(self):
    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(common.boy.x) - self.cw // 2) % self.w
    self.q3b = (int(common.boy.y) - self.ch // 2) % self.h
    self.q3w = clamp(0, self.w - self.q3l, self.cw)
    self.q3h = clamp(0, self.h - self.q3b, self.ch)
    # quadrant 2
    self.q2l = self.q3l
    self.q2b = 0
    self.q2w = self.q3w
    self.q2h = self.ch - self.q3h
    # quadrand 4
    self.q4l = 0
    self.q4b = self.q3b
    self.q4w = self.cw - self.q3w
    self.q4h = self.q3h
    # quadrand 1
    self.q1l = 0
    self.q1b = 0
    self.q1w = self.q4w
    self.q1h = self.q2h
```