Lecture #18. 스크롤링 (1)

2D 게임 프로그래밍

이대현 교수

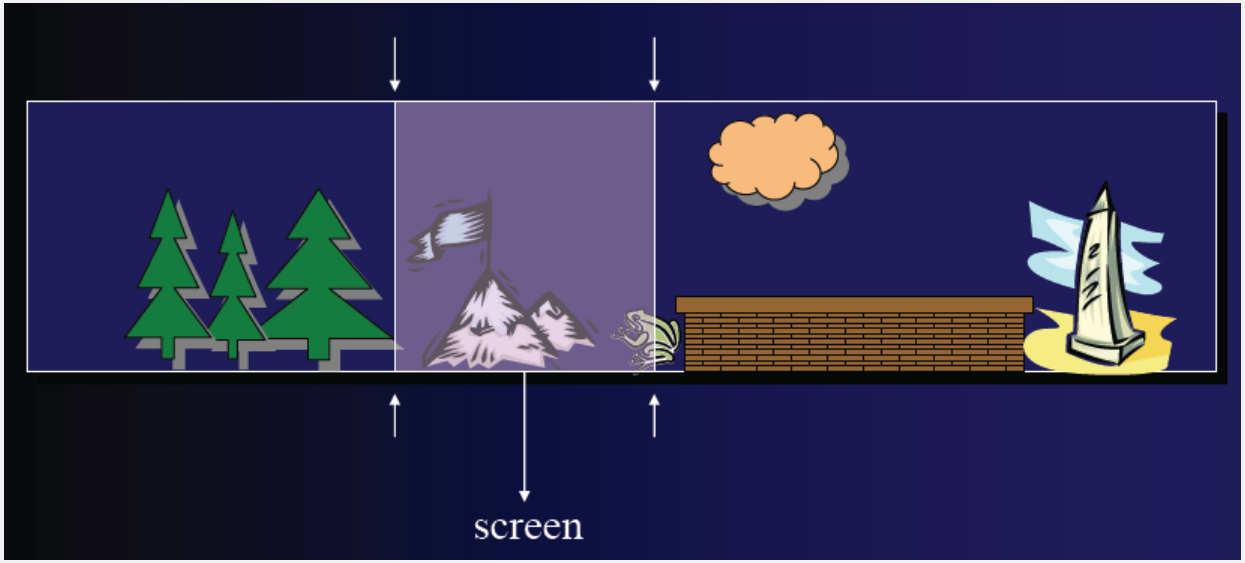
한국공학대학교
TECH UNIVERSITY OF KOREA

학습 내용

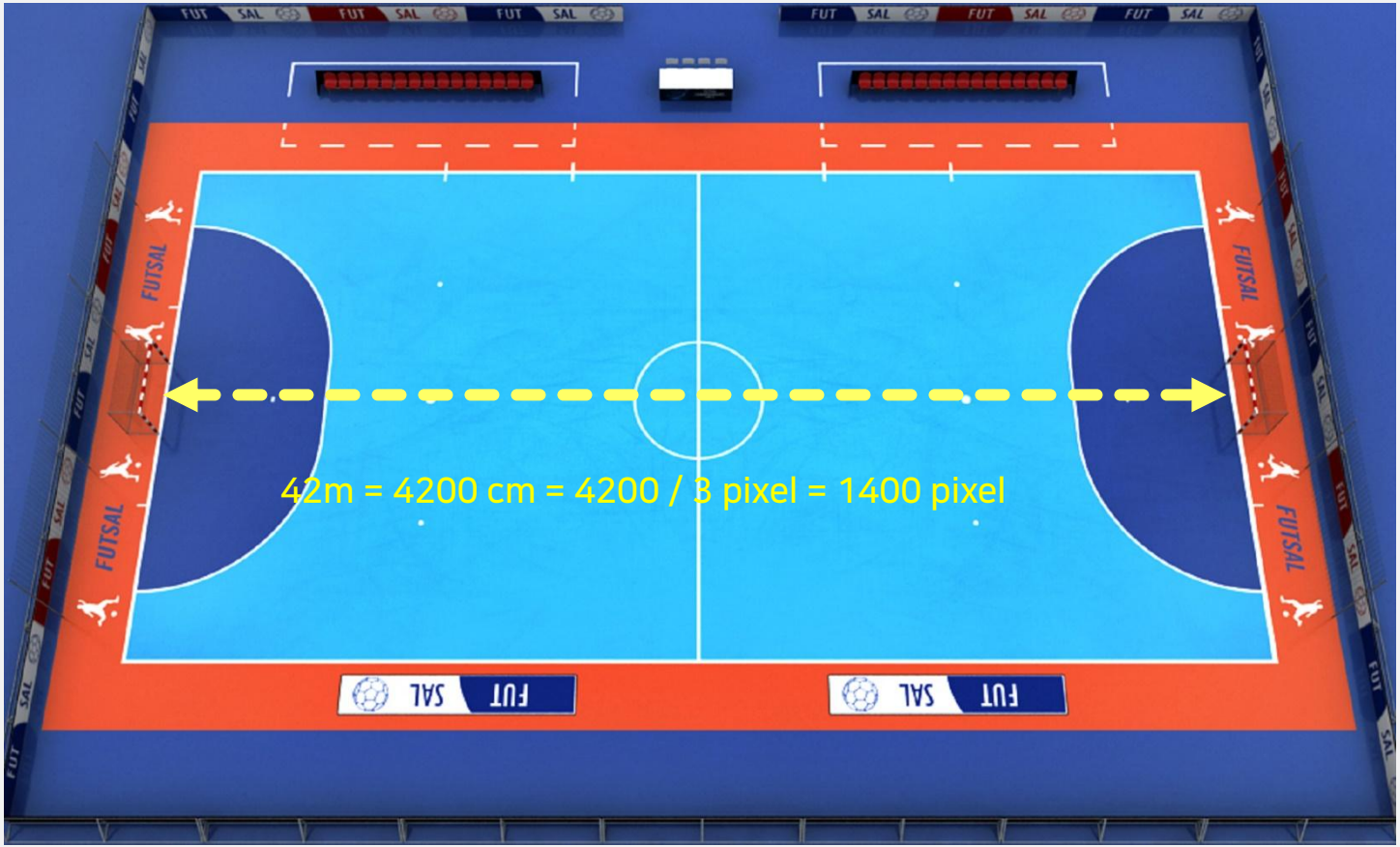
- 스크롤링

스크롤링(Scrolling)

- 그림이나 이미지의 일부분을 디스플레이 화면 위에서 상하좌우로 움직이면서 나타내는 기법.



게임 맵은 반드시 실제 물리값으로 크기가 표시되어야 함.



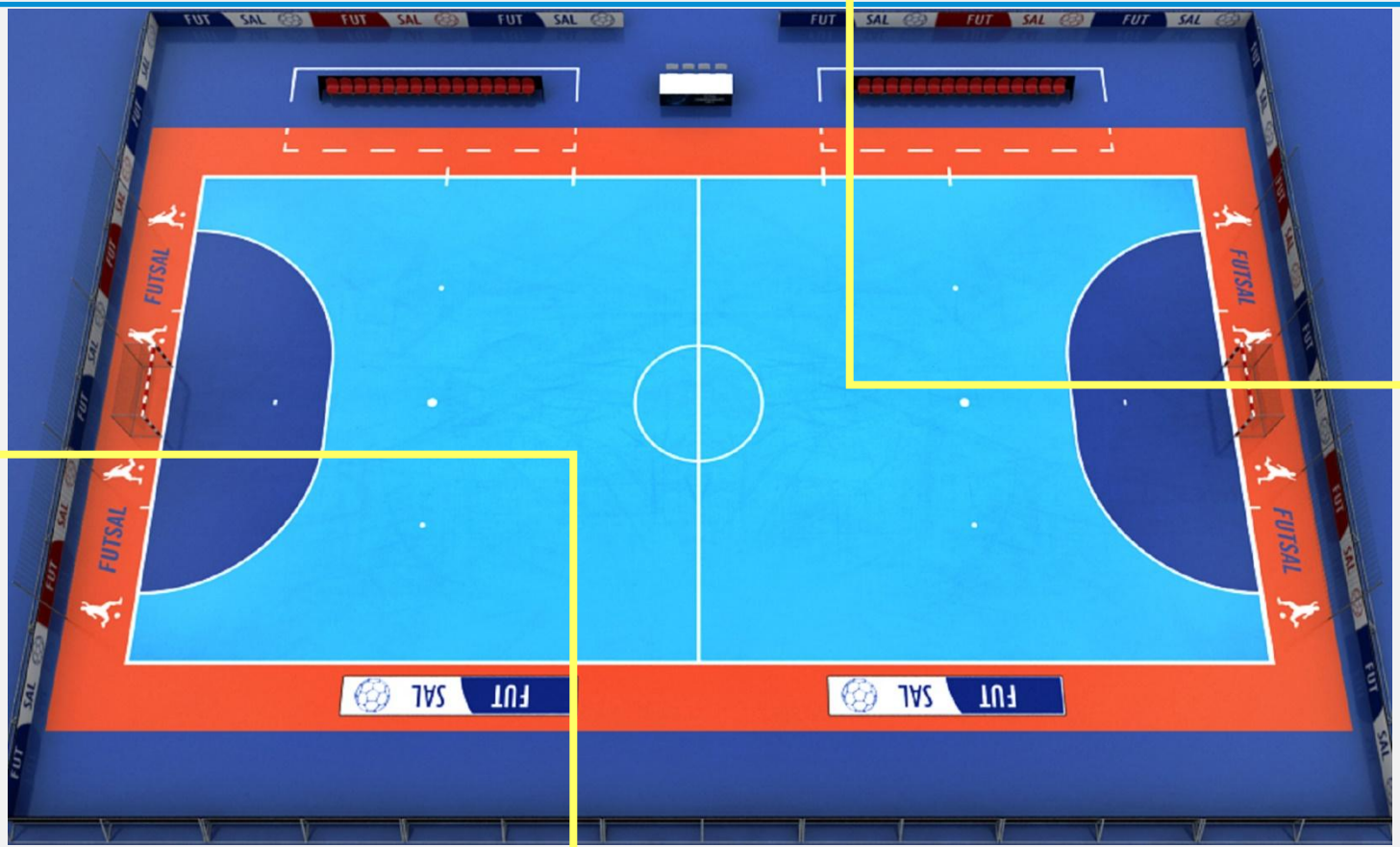
실제 좌표와 화면 좌표를 분리 처리

실제 공간 좌표 – 객체의 실제 좌표 계산할 때,

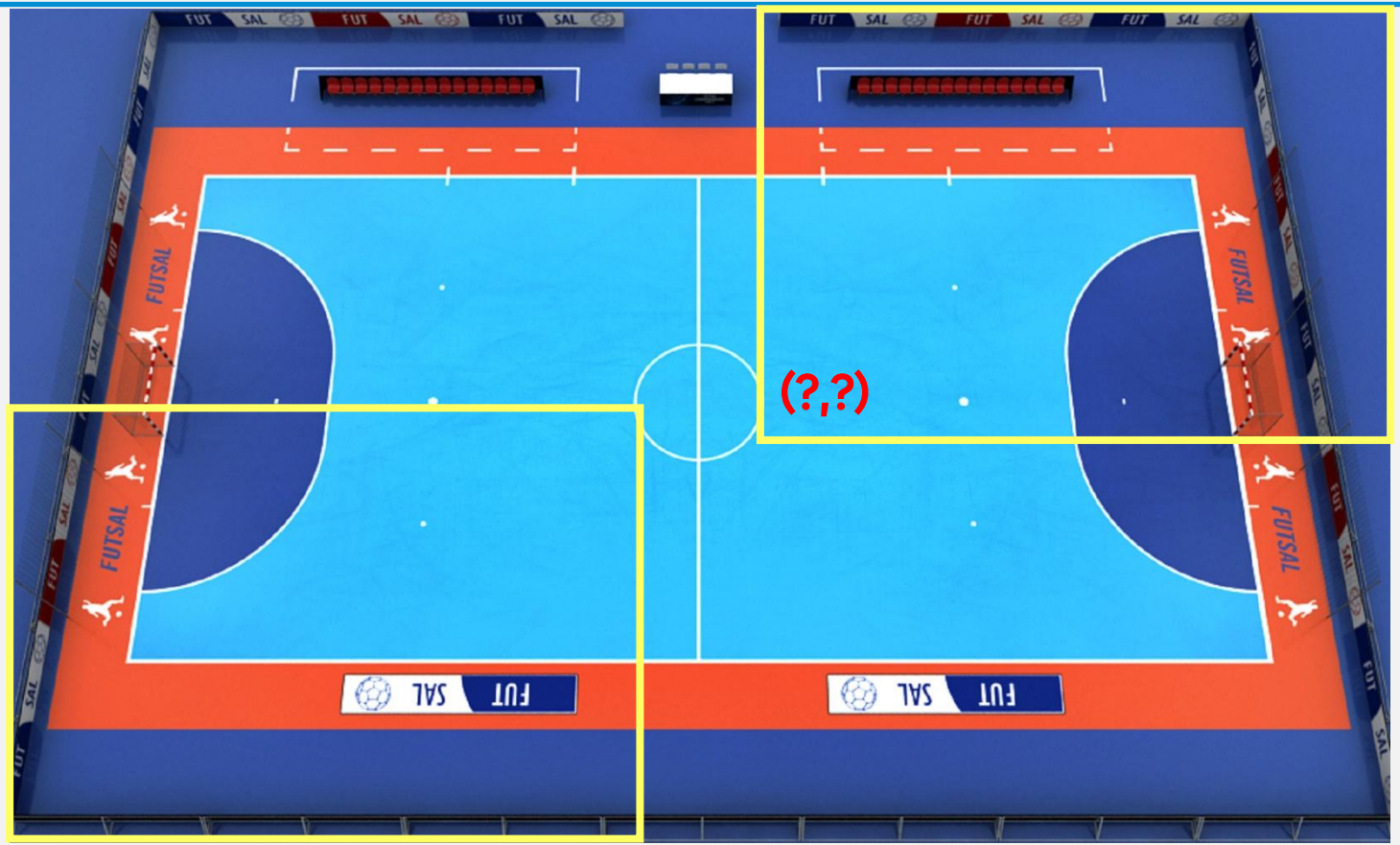
화면 좌표 – 화면 상에 그릴 때

퀴즈 – 클리핑 영역 계산

클리핑 영역이 물리 공간을 넘어서면?



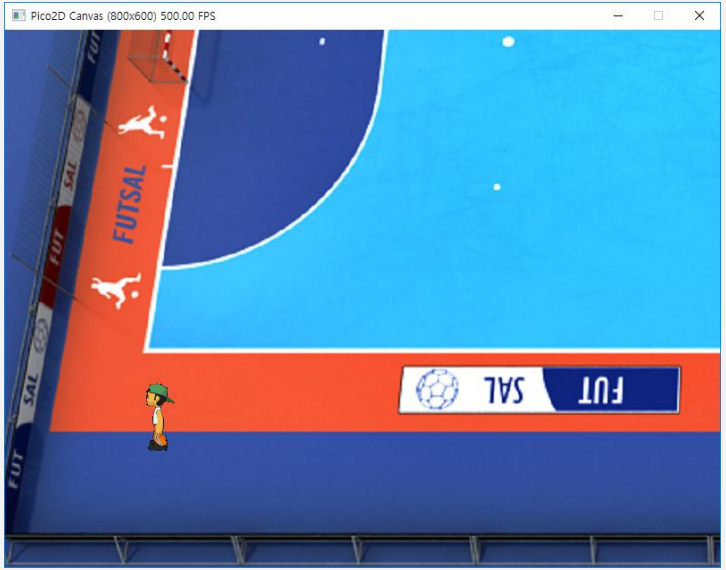
실제 가능한 클리핑 영역은?



스크린 윈도우를 이용한 스크롤링

window_left, window_bottom

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



상하좌우 스크롤링 #1

clamp 함수

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

화면의 정중앙에 캐릭터를 그림


```
def draw(self):
    sx, sy = get_canvas_width() // 2, get_canvas_height() // 2
    self.boy.font.draw(sx - 100, sy + 60, f'({self.boy.x:5.5}, {self.boy.y:5.5})', (255, 255, 0))

    if self.boy.face_dir == 1:  # right
        self.boy.image.clip_draw(int(self.boy.frame) * 100, 300, 100, 100, sx, sy)
    else:  # face_dir == -1: # left
        self.boy.image.clip_draw(int(self.boy.frame) * 100, 200, 100, 100, sx, sy)
```

```
def draw(self):
    sx, sy = get_canvas_width() // 2, get_canvas_height() // 2
    self.boy.font.draw(sx - 100, sy + 60, f'({self.boy.x:5.5}, {self.boy.y:5.5})', (255, 255, 0))

    if self.boy.xdir == 0: # 위 아래로 움직이는 경우
        if self.boy.face_dir == 1: # right
            self.boy.image.clip_draw(int(self.boy.frame) * 100, 100, 100, 100, sx, sy)
        else:
            self.boy.image.clip_draw(int(self.boy.frame) * 100, 0, 100, 100, sx, sy)
    elif self.boy.xdir == 1:
        self.boy.image.clip_draw(int(self.boy.frame) * 100, 100, 100, 100, sx, sy)
    else:
        self.boy.image.clip_draw(int(self.boy.frame) * 100, 0, 100, 100, sx, sy)
```

boy.py - 물리 좌표계와 화면 좌표의 분리


```python
def __init__(self):
    self.x, self.y = get_canvas_width() / 2, get_canvas_height() / 2

    # 물리 좌표계로 바꿔야 함.
    self.x, self.y = common.court.w / 2, commont.court.h / 2
```

```python
def update(self):
    self.state_machine.update()
    self.x = clamp(50.0, self.x, get_canvas_width()-50.0)
    self.y = clamp(50.0, self.y, get_canvas_height()-50.0)

    # 물리 좌표계로 바꿔야 함.
    self.x = clamp(get_canvas_width()/2, self.x, common.court.w - get_canvas_width()/2)
    self.y = clamp(get_canvas_height()/2, self.y, common.court.h - get_canvas_height()/2)
```

court.py


```python
class Court:
    def __init__(self):
        self.image = load_image('futsal_court.png')
        self.cw = get_canvas_width()
        self.ch = get_canvas_height()
        self.w = self.image.w
        self.h = self.image.h

    def update(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)

    def draw(self):
        self.image.clip_draw_to_origin(self.window_left, self.window_bottom, self.cw, self.ch, 0, 0)
```

court.py

```
def update(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)
```

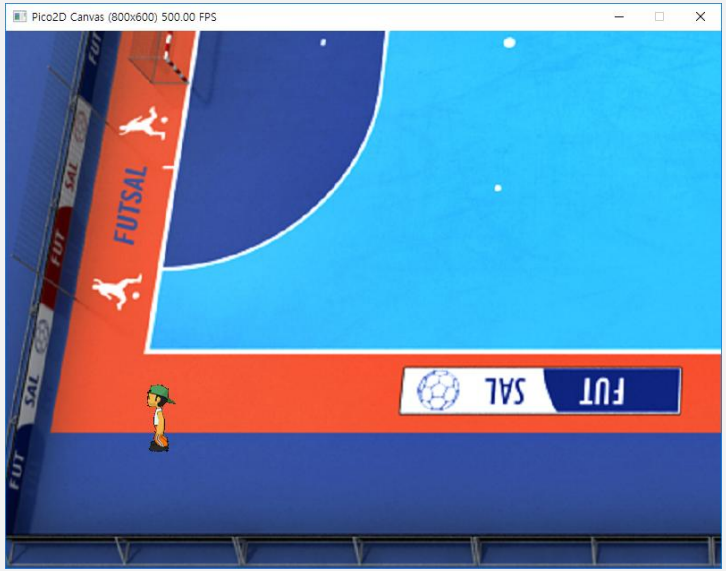
window의 left x 좌표의 최대값은,전체 배경 너비에서 화면의 너비를 뺀 값.

```
def draw(self):
    self.image.clip_draw_to_origin(self.window_left, self.window_bottom, self.cw, self.ch, 0, 0)
```

피봇(중심)을 무시하고, 왼쪽 아래 원점을 피봇으로 간주.

상하좌우 스크롤링 #2

```
sx = self.boy.x - common.court.window_left
sy = self.boy.y - common.court.window_bottom
```



boy.py (1)


```
def update(self):
    self.state_machine.update()

    self.x = clamp(50.0, self.x, common.court.w - 50.0)
    self.y = clamp(50.0, self.y, common.court.h - 50.0)
```