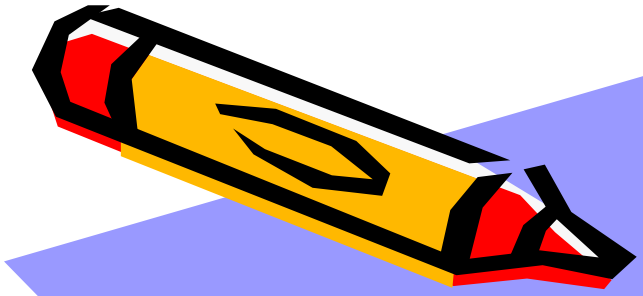


2006년 2학기 윈도우 게임 프로그래밍

제12강 스크롤링과 타일링 (II)

이대현

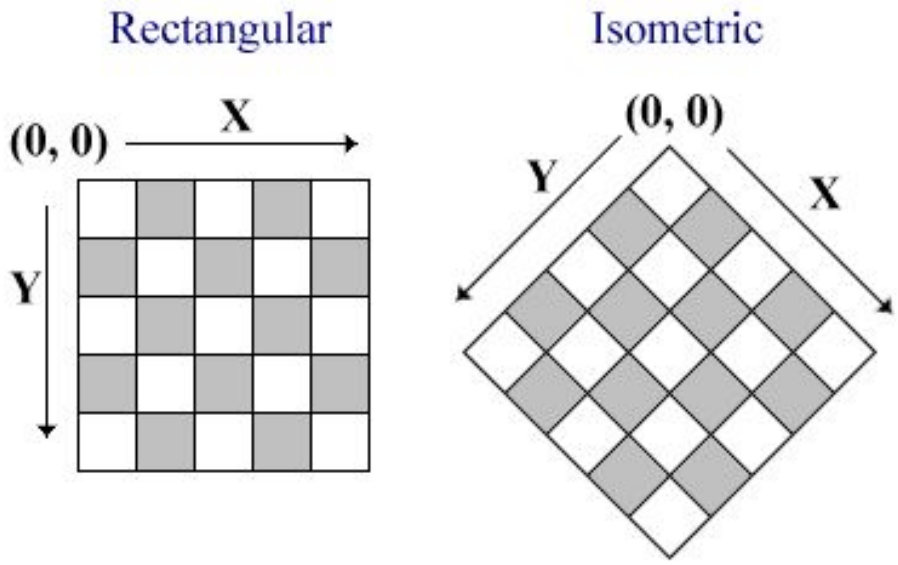
한국산업기술대학교

오늘의 학습 내용

- 동축 타일링(Isometric Tiling)

동축 타일링(Isometric Tiling)

- RTS게임, 전쟁 게임, 시뮬레이션 게임, RPG 게임에서 많이 사용되는 타일링 기법.
- 화면의 깊이감을 표현할 수 있슴.
- 플레이어의 뷰포인트가 하늘위에 있으며, 플레잉 판을 내려다 보는 형식을 띰.



The Sims (2000)

Lecture12-01: 동축 타일링의 구현

Lecture12-01 프로젝트의 구성

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

- C++ 헤더 파일들
  - background.h
  - gameengine.h
  - gamestate.h
  - introstate.h
  - playstate.h
  - sprite.h

background.cpp (1)

```
void Background::putTile(SDL_Surface *screen, int cellid, int
   mapx, int mapy)
{
  SDL_Rect tileRect;
  tileRect.x = cellid * TILE_WIDTH;
  tileRect.y = 0;
  tileRect.w = TILE_WIDTH;
  tileRect.h = TILE_HEIGHT;

  SDL_Rect scrRect;
  scrRect.x = mapx * TILE_WIDTH + (mapy & 1) * (TILE_WIDTH /
   2);
  scrRect.y = mapy * (TILE_HEIGHT / 2);
  scrRect.w = TILE_WIDTH;
  scrRect.h = TILE_HEIGHT;

  scrRect.x -= (TILE_WIDTH / 2);
  scrRect.y -= (TILE_HEIGHT / 2);
  SDL_BlitSurface(tiles, &tileRect, screen, &scrRect);
}
```


background.cpp (2)

```
void Background::blitFrame(SDL_Rect *src, SDL_Surface *screen,
   SDL_Rect *dest)
{
  for (int x = 0; x < src->w; x++) {
    for (int y = 0; y < src->h; y++) {
      int cellid = groundmap[y + src->y][x + src->x];
      if (NULL == dest) {
        putTile(screen, cellid, x, y);
      } else {
        putTile(screen, cellid, x + dest->x, y + dest->y);
      }
    }
  }
}
```

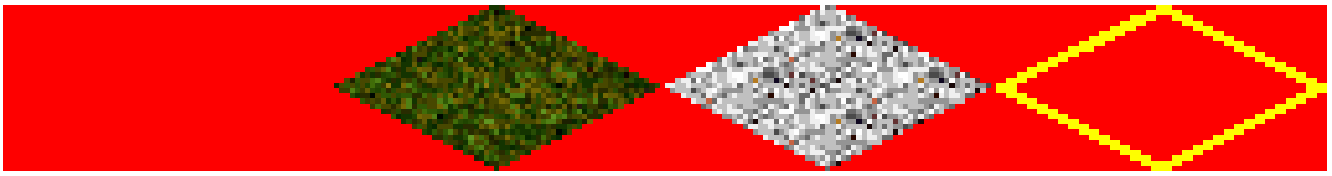
background.cpp (3)

```
void Background::draw(SDL_Surface *screen)
{
  … 전략
  putTile(screen,3,SCREEN_TILE_WIDTH/2,SCREEN_TILE_HEIGHT/2);
}
```


실행 화면

타일의 구성

tiles.bmp

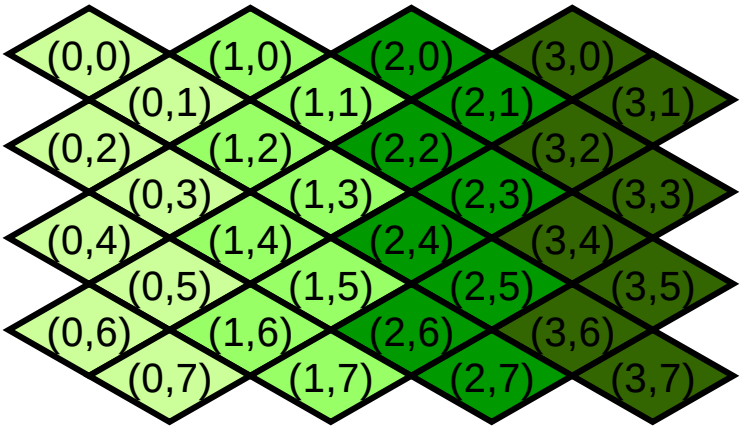


맵의 구성

2차원 배열

(0,0)	(1,0)	(2,0)	(3,0)
(0,1)	(1,1)	(2,1)	(3,1)
(0,2)	(1,2)	(2,2)	(3,2)
(0,3)	(1,3)	(2,3)	(3,3)
(0,4)	(1,4)	(2,4)	(3,4)
(0,5)	(1,5)	(2,5)	(3,5)
(0,6)	(1,6)	(2,6)	(3,6)
(0,7)	(1,7)	(2,7)	(3,7)

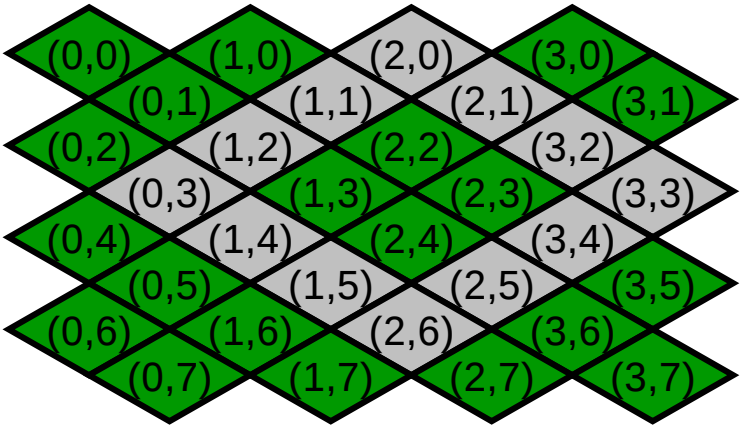
화면 상의 표현



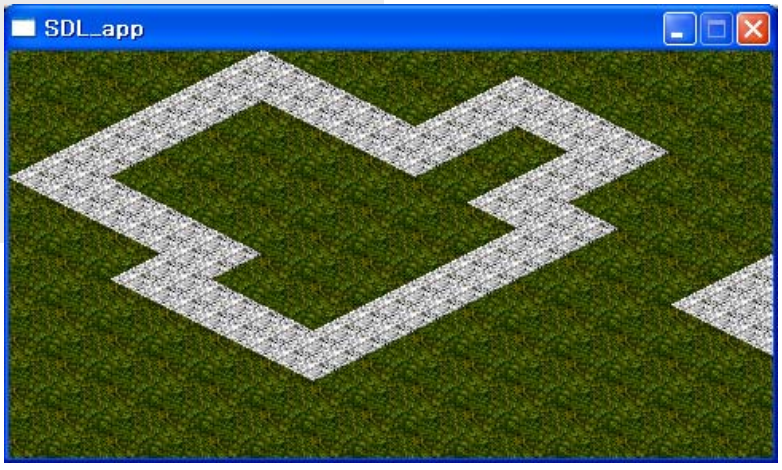
맵 데이터

(0,0)	(1,0)	(2,0)	(3,0)
(0,1)	(1,1)	(2,1)	(3,1)
(0,2)	(1,2)	(2,2)	(3,2)
(0,3)	(1,3)	(2,3)	(3,3)
(0,4)	(1,4)	(2,4)	(3,4)
(0,5)	(1,5)	(2,5)	(3,5)
(0,6)	(1,6)	(2,6)	(3,6)
(0,7)	(1,7)	(2,7)	(3,7)

실제 화면




```
const int groundmap[MAP_TILE_HEIGHT][MAP_TILE_WIDTH] = {
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1},
{1, 2, 1, 2, 2, 2, 1, 1, 1, 1, 1},
{1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1},
{2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1},
{1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1},
{1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1},
{1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1},
{1, 2, 1, 1, 2, 1, 1, 2, 2, 1, 1},
{1, 1, 2, 1, 2, 1, 1, 2, 1, 2, 1},
{1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1},
{1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
};
```

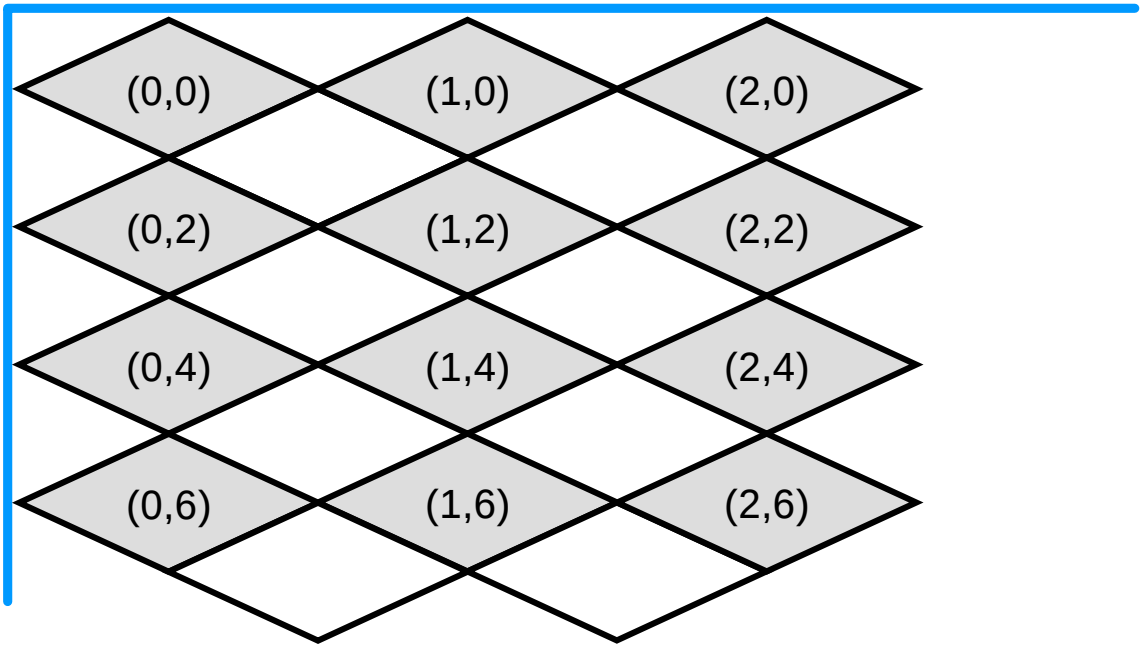


타일의 출력: 맵 데이터의 (mapx,mapy) 위치의 타일 출력

i) mapy가 짝수일 때,

스크린의 x 좌표 = mapx * 타일의 너비

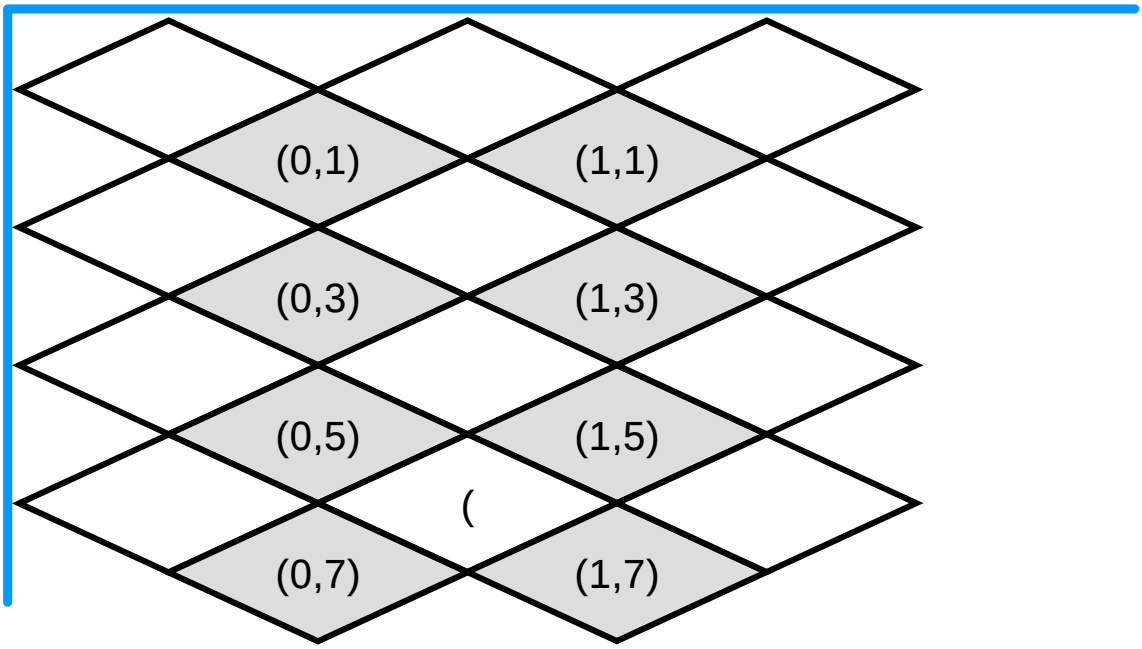
스크린의 y 좌표 = mapy * 타일의 높이 / 2



i) mapy가 홀수일 때,

스크린의 x 좌표 = mapx * 타일의 너비 + 타일의 너비 /2

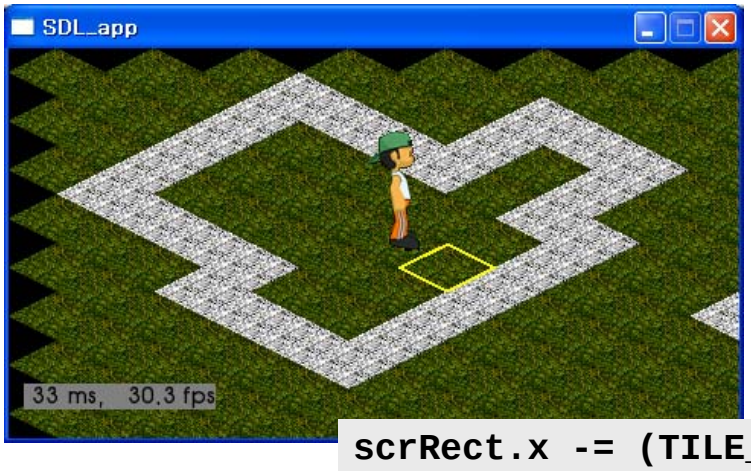
스크린의 y 좌표 = mapy * 타일의 높이 / 2



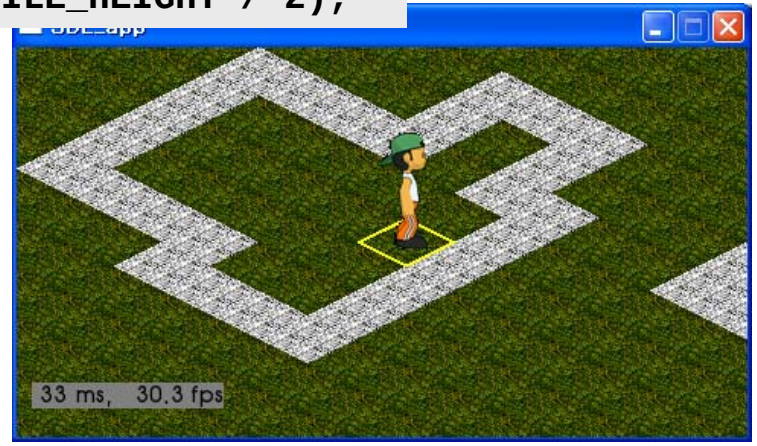
코드

```
…
SDL_Rect scrRect;
scrRect.x = mapx * TILE_WIDTH + (mapy & 1) * (TILE_WIDTH / 2);
scrRect.y = mapy * (TILE_HEIGHT / 2);
…
```


출력 위치의 보정

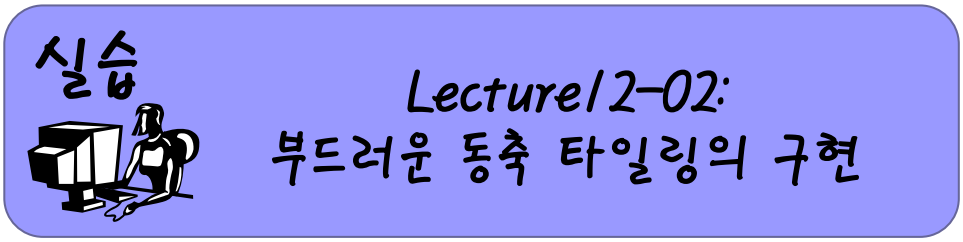



```
scrRect.x -= (TILE_WIDTH / 2);
scrRect.y -= (TILE_HEIGHT / 2);
```



플레이어 바닥 타일의 표시

```
void Background::draw(SDL_Surface *screen)
{
  … 전략
  putTile(screen,3,SCREEN_TILE_WIDTH/2,SCREEN_TILE_HEIGHT/2);
}
```

Lecture12-02 프로젝트의 구성

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

- C++ 헤더 파일들
  - background.h
  - gameengine.h
  - gamestate.h
  - introstate.h
  - playstate.h
  - sprite.h

background.cpp

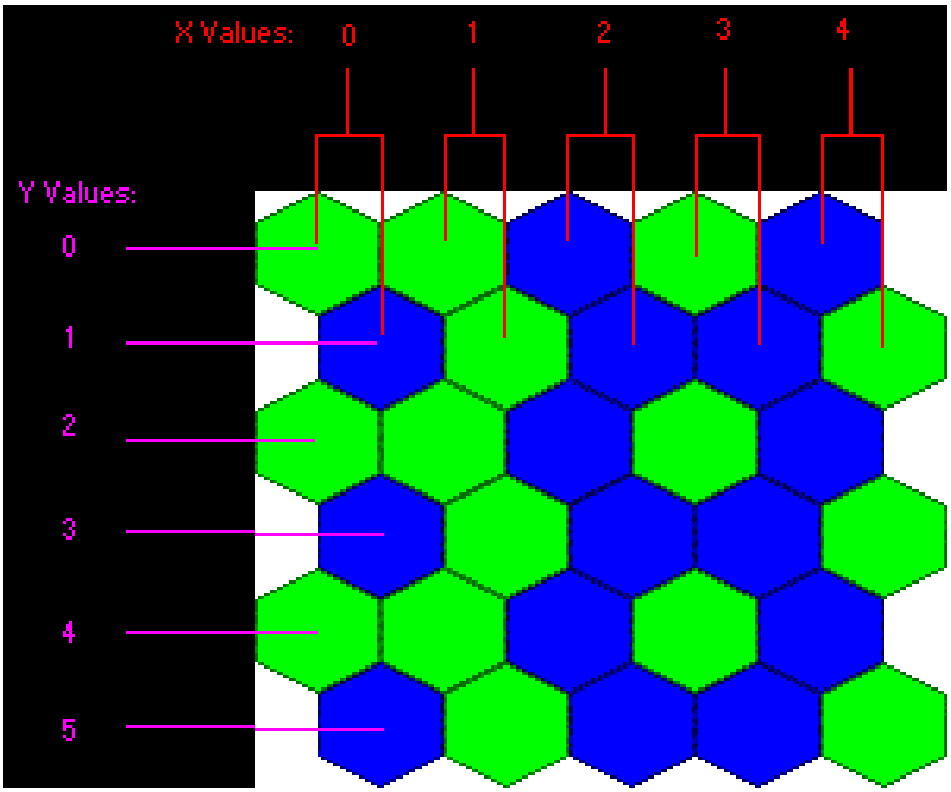
```
void Background::putTile(SDL_Surface *screen, int cellid, int
   mapx, int mapy)
{
… 전략
 scrRect.x -= offset;
 SDL_BlitSurface(tiles, &tileRect, screen, &scrRect);
}
void Background::scrollRight(void)
{
  leftx = (leftx + (TILE_WIDTH * MAP_TILE_WIDTH) - 4) %
   (TILE_WIDTH * MAP_TILE_WIDTH);
  framex = (leftx / TILE_WIDTH) % (MAP_TILE_WIDTH);
  offset = leftx % TILE_WIDTH;
}
void Background::scrollLeft(void)
{
  leftx = (leftx + 4) % (TILE_WIDTH * MAP_TILE_WIDTH);
  framex = (leftx / TILE_WIDTH) % (MAP_TILE_WIDTH);
  offset = leftx % TILE_WIDTH;
}
```

부드러운 스크롤링 알고리즘

```
void Background::scrollRight(void)
{
  leftx = (leftx + (TILE_WIDTH * MAP_TILE_WIDTH) - 4) %
   (TILE_WIDTH * MAP_TILE_WIDTH);

  framex = leftx / TILE_WIDTH;

  offset = leftx % TILE_WIDTH;
}

void Background::putTile(SDL_Surface *screen, int cellid, int
   mapx, int mapy)
{
… 전략
 scrRect.x -= offset;
```

- 화면에 뿌릴 프레임의 시작 x좌표를, 도트 단위로 환산하여 계산함.
- 위에서 계산한 프레임시작x좌표를 맵 좌표로 변환함.
- 화면 출력시, 조절해야할 오프셋을 계산.
- 오프셋만큼, 왼쪽으로 화면을 밀어서 출력함.

육각타일을 이용한 맵의 구성



실습 과제 #10

- 동축 타일맵을 배경으로 한, 상하좌우 스크롤링의 구현
  - 좌우 스크롤뿐만 아니라, 상하스크롤도 되게 실습 12-02를 수정함.