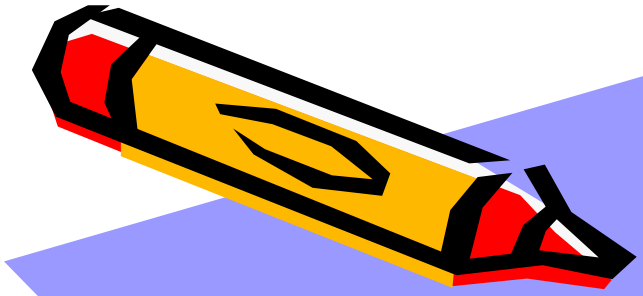


2006년 2학기 윈도우 게임 프로그래밍

제11강 스크롤링과 타일링 (I)

이대현

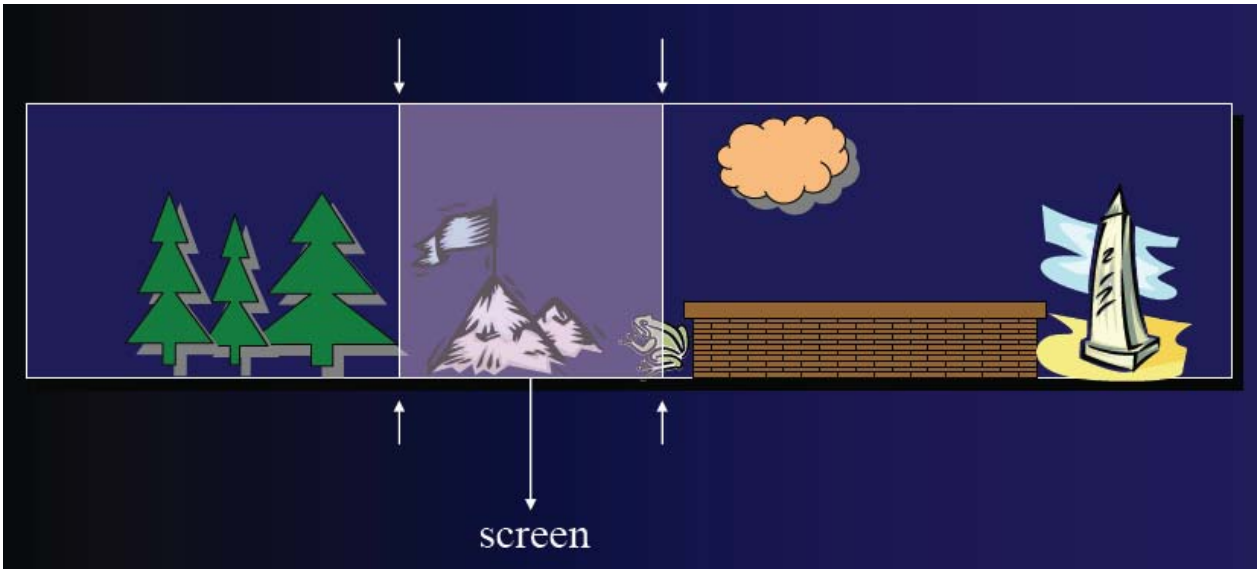
한국산업기술대학교

오늘의 학습 내용

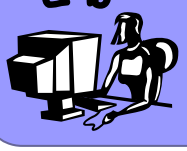
- 스크롤링
- 타일링
- 시차 스크롤링

스크롤링(Scrolling)

- 그림이나 이미지의 일부분을 디스플레이 화면 위에서 상하좌우로 움직이면서 나타내는 기법.
- 슈팅 게임, 고전 RPG 게임에서 주로 사용됨.
- 캐릭터를 직접 이동하는 대신, 배경을 이동함으로써, 캐릭터의 이동을 표현.



실습



Lecture11-01:
무한대 가로 스크롤링의 구현

Lecture10-01 프로젝트의 구성

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

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

playstate.cpp (1)

```
void CPlayState::HandleEvents(CGameEngine* game)
{
… 전략 …
            case SDLK_LEFT:
              player->walkLeft();
              framex = (framex + 960 - 8) % 960;
              break;
            case SDLK_RIGHT:
              player->walkRight();
              framex = (framex + 8) % 960;
              break;
… 후략 …
}
```

playstate.cpp (2)

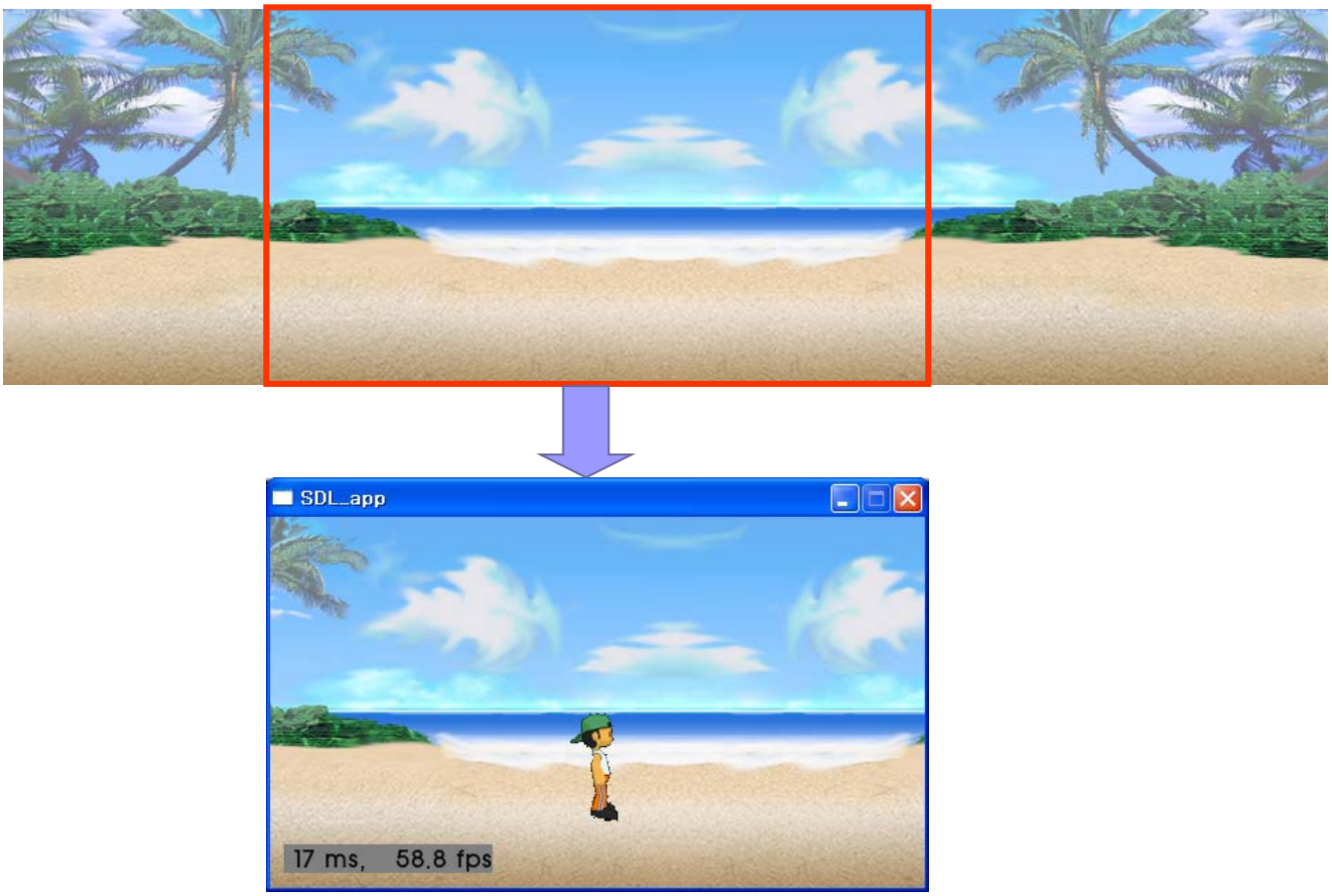
```
void CPlayState::Draw(CGameEngine* game)
{
  SDL_Rect frame, targetFrame;
  if (framex < 480) {
    frame.x = framex; frame.y = 0;
    frame.w = 480; frame.h = 272;
    SDL_BlitSurface(bg, &frame, game->screen, NULL);
  } else {
    frame.x = framex; frame.y = 0;
    frame.w = 960 - framex; frame.h = 272;
    SDL_BlitSurface(bg, &frame, game->screen, NULL);

    frame.x = 0; frame.y = 0;
    frame.w = framex - 480; frame.h = 272;

    targetFrame.x = 960 - framex; targetFrame.y = 0;
    targetFrame.w = framex - 480; targetFrame.h = 272;
    SDL_BlitSurface(bg, &frame, game->screen, &targetFrame);
  }

  player->draw();
}
```


배경 이미지

- 스크롤링이 무한대로 되게 하려면, 배경의 시작과 끝이 매끄럽게 이어져야 한다.

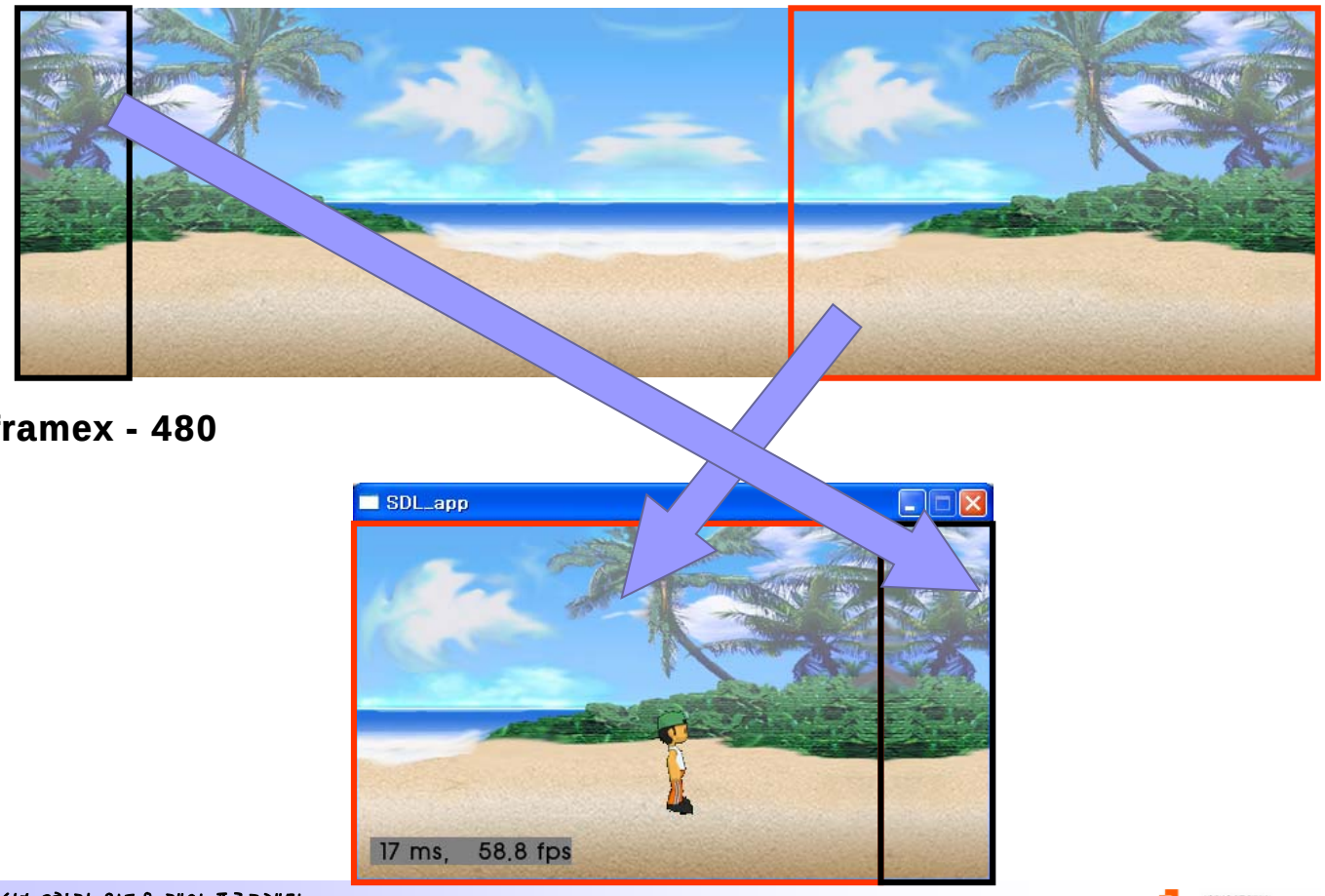
framex < 480 일때,

framex



framex >= 480 일때,

960 - framex

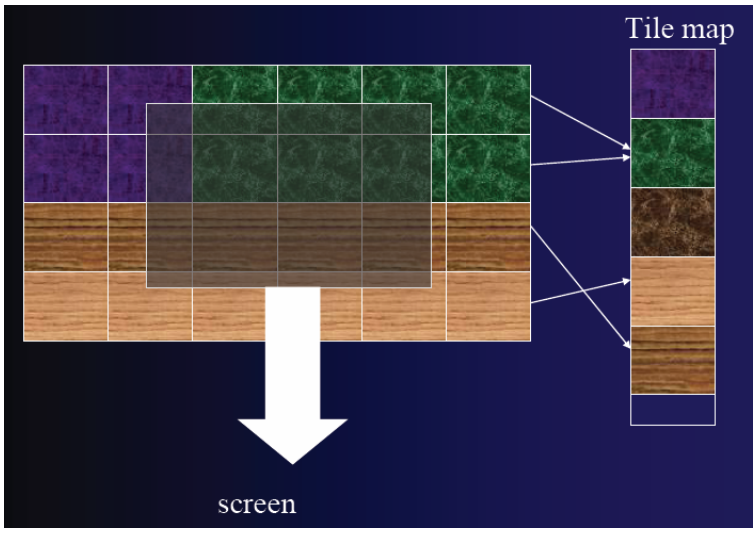


framex의 증가 및 감소

```
case SDLK_LEFT:
  player->walkLeft();
  framex = (framex + 960 - 8) % 960;
  break;


case SDLK_RIGHT:
  player->walkRight();
  framex = (framex + 8) % 960;
  break;
```

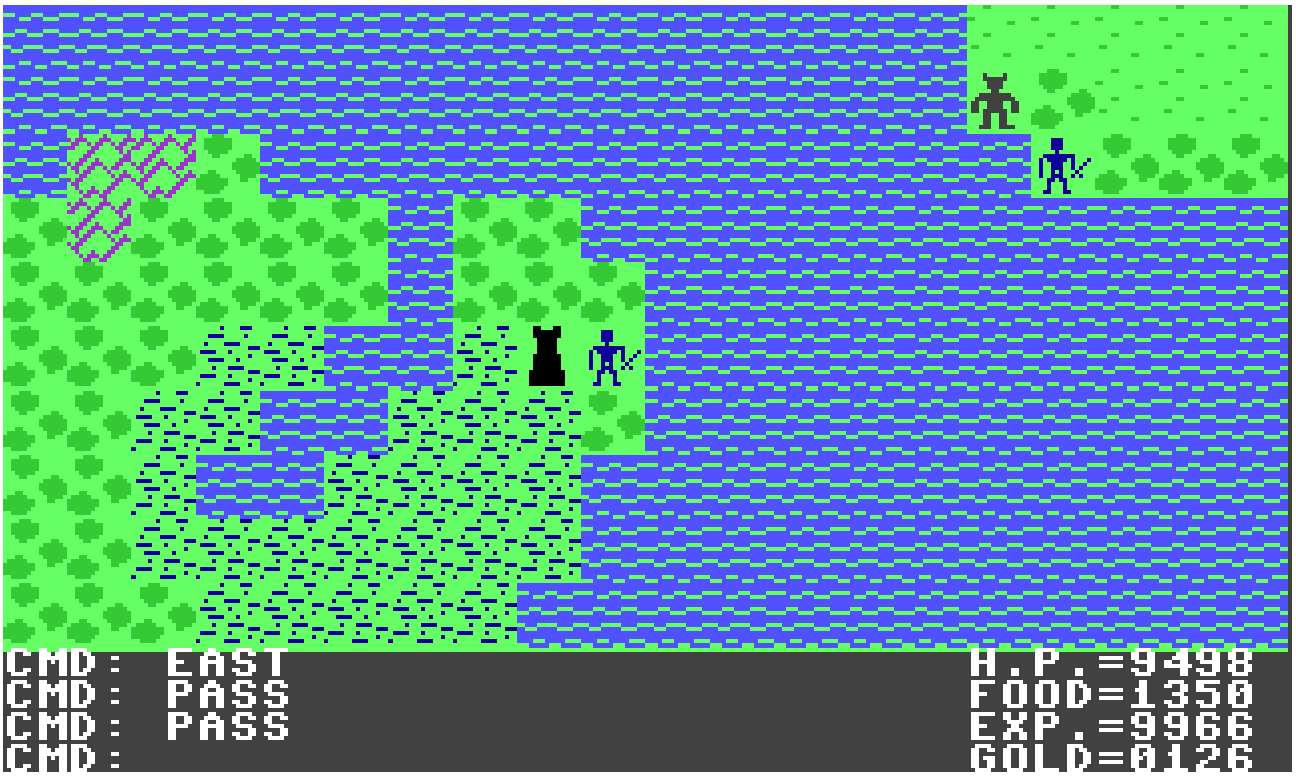

타일링(Tiling)

- 게임의 배경을 단일 이미지로 구성하지 않고, 여러 개의 타일을 이용하여 구성하는 방법. 거의 모든 RPG 게임에 사용됨.

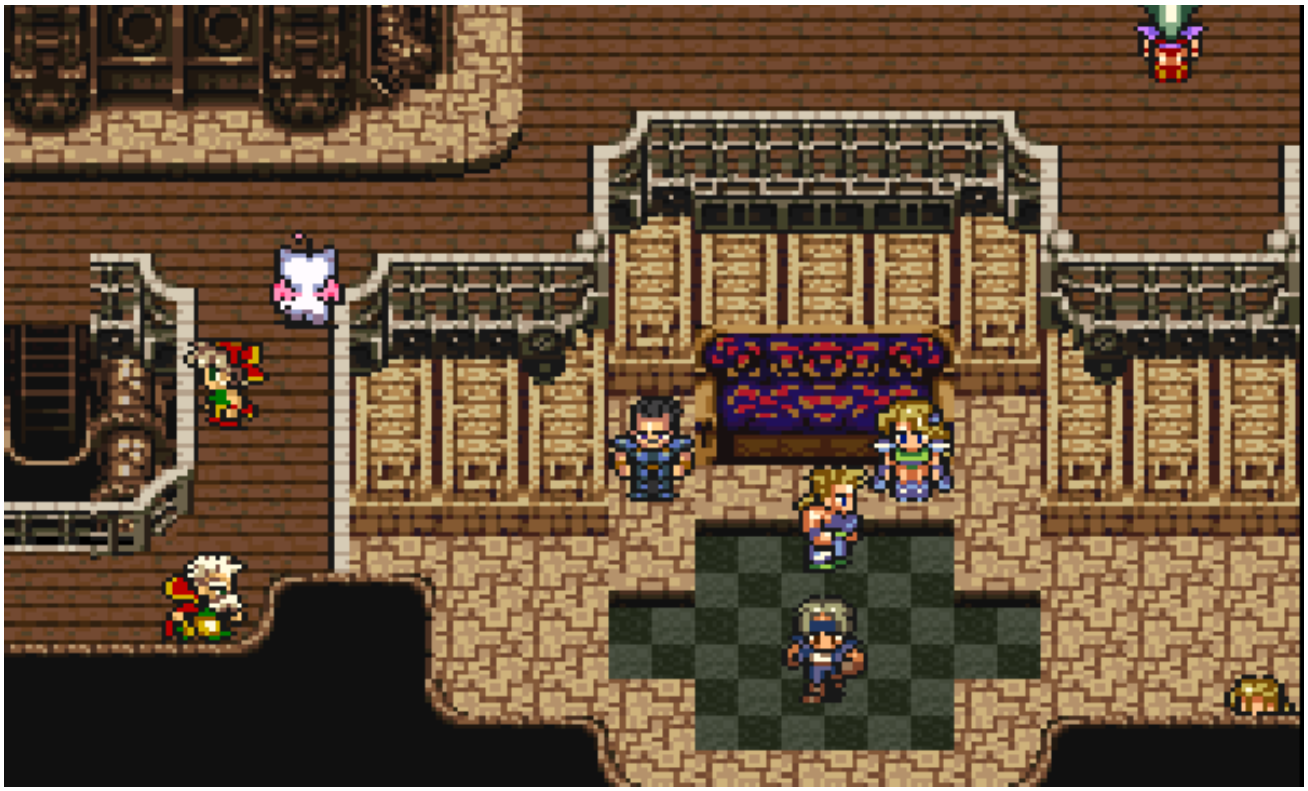


- 장점
  - 맵의 크기를 대폭 줄일 수 있다.
  - 사용자의 구미에 맞게 맵을 편집하여 사용할 수 있다.
- 단점
  - 정교한 배경을 만드는데 많은 시간이 소요되고 어렵다.

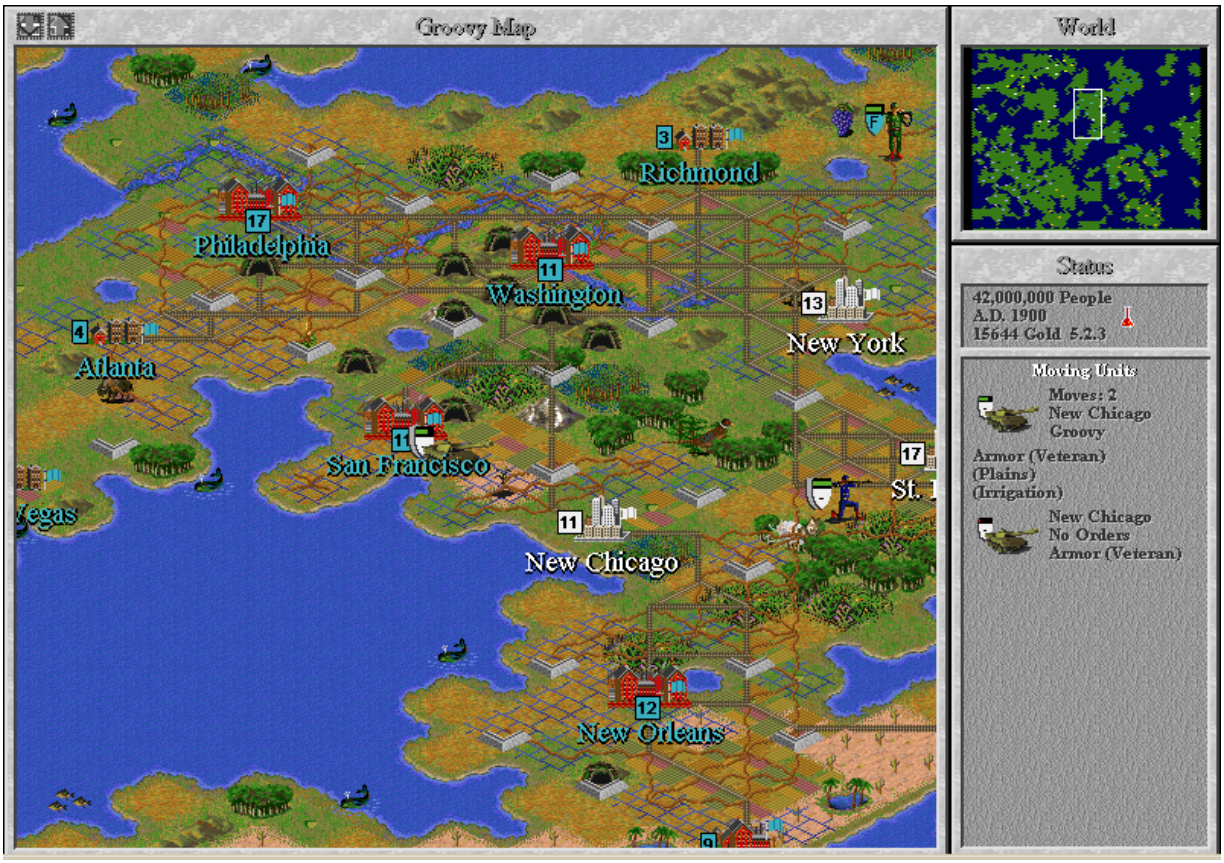
타일 기반 게임의 진화: Ultima2(1982)



타일 기반 게임의 진화: Final Fantasy VI (1994)



타일 기반 게임의 진화: Civilization II (1996)



타일 기반 게임의 진화: Age of Wonders (2003)

Lecture11-02: 타일맵의 스크롤링

Lecture10-01 프로젝트의 구성

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

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

playstate.cpp

```
void CPlayState::HandleEvents(CGameEngine* game)
{
… 전략 …
            case SDLK_LEFT:
              player->walkLeft();
              bg->scrollRight();
              break;
            case SDLK_RIGHT:
              player->walkRight();
              bg->scrollLeft();
              break;
… 후략 …
}

void CPlayState::Draw(CGameEngine* game)
{
  bg->draw(game->screen);
  player->draw();
}
```


background.cpp (1)

```
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 = bgmap[y + src->y][x + src->x];
      SDL_Rect tileRect;
      tileRect.x = (cellid % 20) * 16;
      tileRect.y = (cellid / 20) * 16;
      tileRect.w = 16; tileRect.h = 16;

      SDL_Rect scrRect;
      if (NULL == dest) {
        scrRect.x = x * 16;
        scrRect.y = y * 16;
        scrRect.w = 16; scrRect.h = 16;
      } else {
        scrRect.x = (x + dest->x) * 16;
        scrRect.y = (y + dest->y) * 16;
        scrRect.w = 16; scrRect.h = 16;
      }
      SDL_BlitSurface(tiles, &tileRect, screen, &scrRect);
    }
  }
}
```

background.cpp (2)

```
void Background::scrollRight(void)
{
  framex = (framex + MAP_TILE_WIDTH - 1) % MAP_TILE_WIDTH;
}

void Background::scrollLeft(void)
{
  framex = (framex + 1) % (MAP_TILE_WIDTH);
}
```

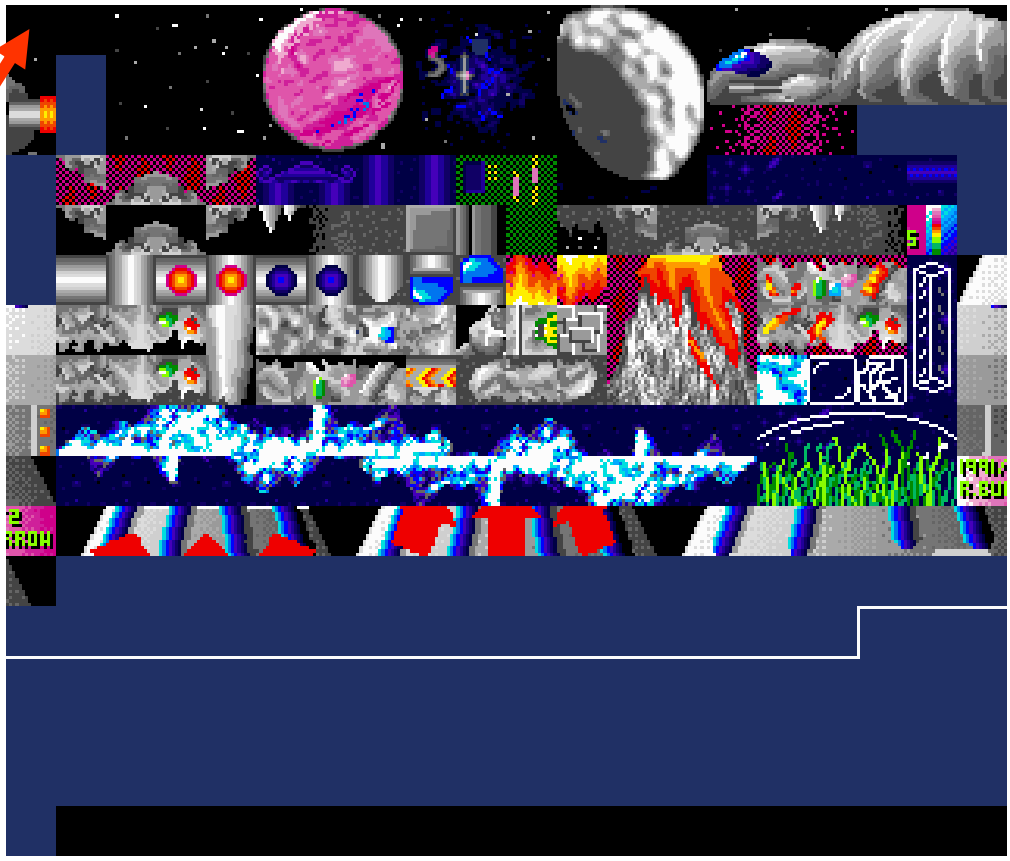

background.cpp (3)

```
void Background::draw(SDL_Surface *screen)
{
  SDL_Rect frame, targetFrame;
  if (framex < MAP_TILE_WIDTH - SCREEN_TILE_WIDTH) {
    frame.x = framex; frame.y = 0;
    frame.w = SCREEN_TILE_WIDTH; frame.h = SCREEN_TILE_HEIGHT;
    blitFrame(&frame, screen, NULL);
  } else {
    frame.x = framex; frame.y = 0;
    frame.w = MAP_TILE_WIDTH - framex; frame.h = SCREEN_TILE_WIDTH;
    blitFrame(&frame, screen, NULL);

    frame.x = 0; frame.y = 0;
    frame.w = framex - (MAP_TILE_WIDTH - SCREEN_TILE_WIDTH);
    frame.h = SCREEN_TILE_HEIGHT;

    targetFrame.x = MAP_TILE_WIDTH - framex;
    targetFrame.y = 0;
    targetFrame.w = framex - (MAP_TILE_WIDTH - SCREEN_TILE_WIDTH);
    targetFrame.h = SCREEN_TILE_HEIGHT;
    blitFrame(&frame, screen, &targetFrame);
  }
}
```

실행 화면

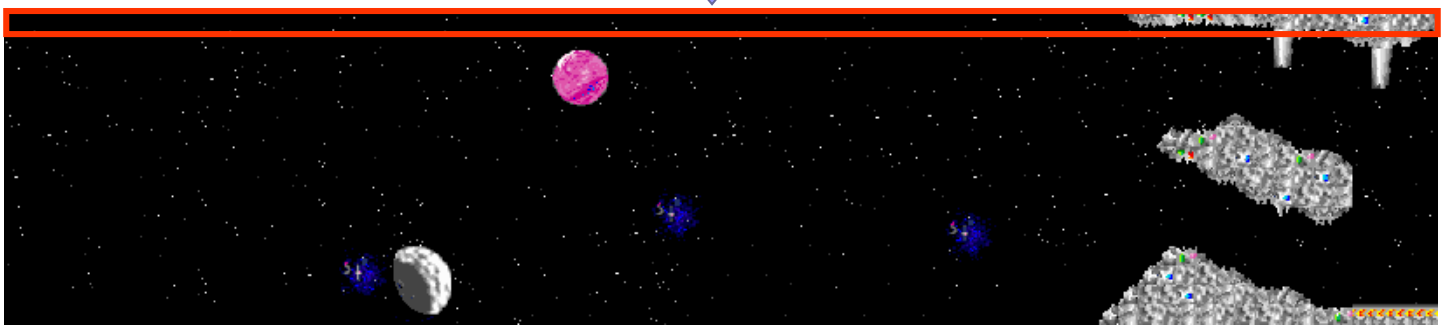
타일 이미지

- #0: 투명 타일.



맵의 구성

```
const int bgmap[16][MAP_TILE_WIDTH] = {
{ 2, 3, 1, 1, 1, 23, 2, 42, 1, 1, 22, 1, 1, 1, 22, 2,
1, 43, 1, 1, 1, 1, 44, 43, 1, 42, 1, 23, 1, 23, 1, 4,
1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 3, 1, 24, 1,
1, 2, 44, 1, 1, 24, 1, 4, 3, 81, 121, 122, 123, 123, 122, 122,
121, 128, 125, 126, 126, 127, 128, 125, 125, 126, 130, 102, 125, 126, 127, 125,
126, 125, 102, 102, 102, 69, 70, 90, 90, 90, 90, 102, 125, 128, 125, 128,
127, 125, 128, 131, 125, 126, 126, 128, 125, 126, 127, 126, 126, 121, 123, 122,
141, 142, 142, 143, 141, 141, 142, 95, 87, 87, 87, 87, 87, 87, 87, 97,
23, 23, 22, 102, 42, 3, 44, 23, 2, 102, 2, 23, 1, 1, 1, 23,
```

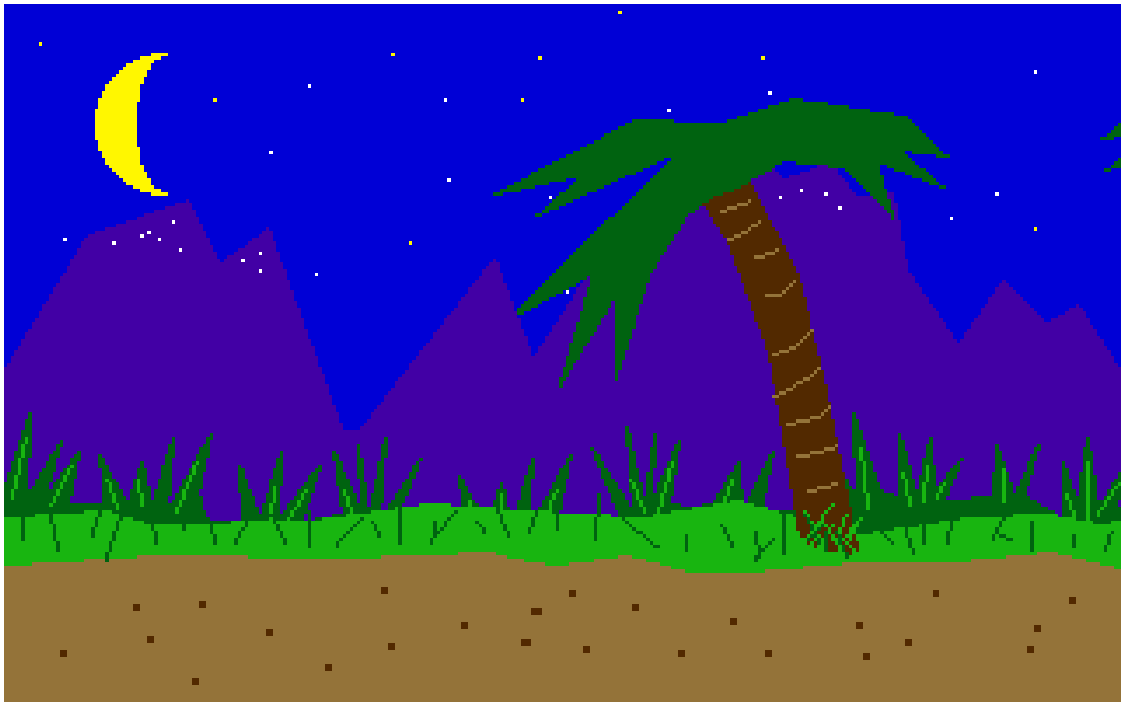


시차(視差) 스크롤링(Parallax Scrolling)

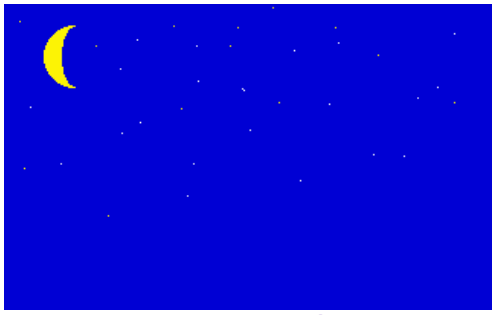
- 물체와 눈의 거리에 따라, 물체의 이동속도가 달라보이는 효과를 이용하여, 3차원 배경을 흉내내는 기법.
  - 예) http://joesparks.shockwave.com/production/productionnote_02.htm

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

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



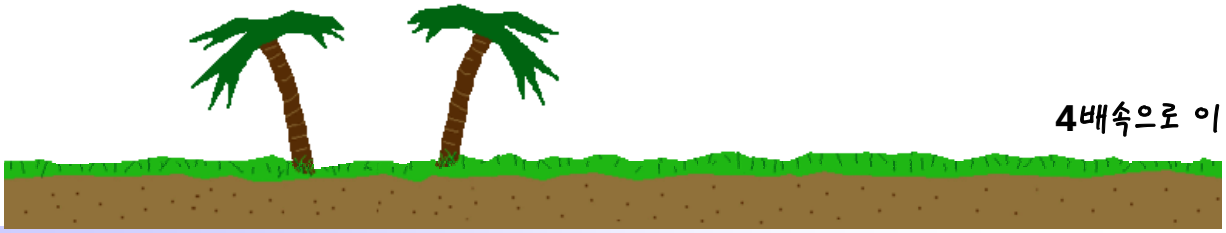
시차 스크롤링 방법



1배속으로 이동

2배속으로 이동

3배속으로 이동



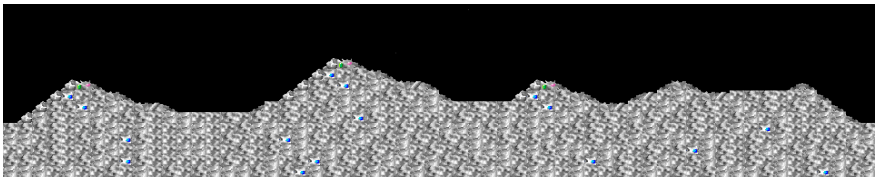
4배속으로 이동

실습 과제 #9 및 과제 #3

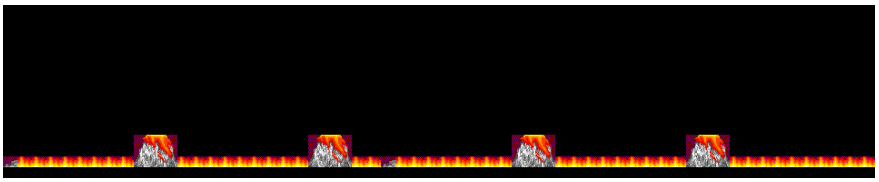
- 시차 스크롤링의 구현
  - lecture11-01-dist.zip을 사용.
  - Bg1이 맨뒤로, 그위에 bg2, bg3, bg4 가 놓이고 각기 다른 속도로 움직임.
  - 캐릭터는 좌우로 움직일 수 있음.

bg1map

bg2map



bg3map



bg4map