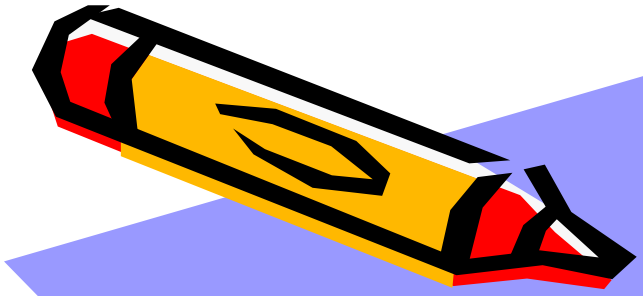


2006년 2학기 윈도우 게임 프로그래밍

제8강 프레임속도의 조절

이대현

한국산업기술대학교

오늘의 학습 내용

- 프레임 속도의 조절
- 30fps 맞추기
- 스프라이트 프레임 속도의 조절

프레임 속도(Frame Rate)

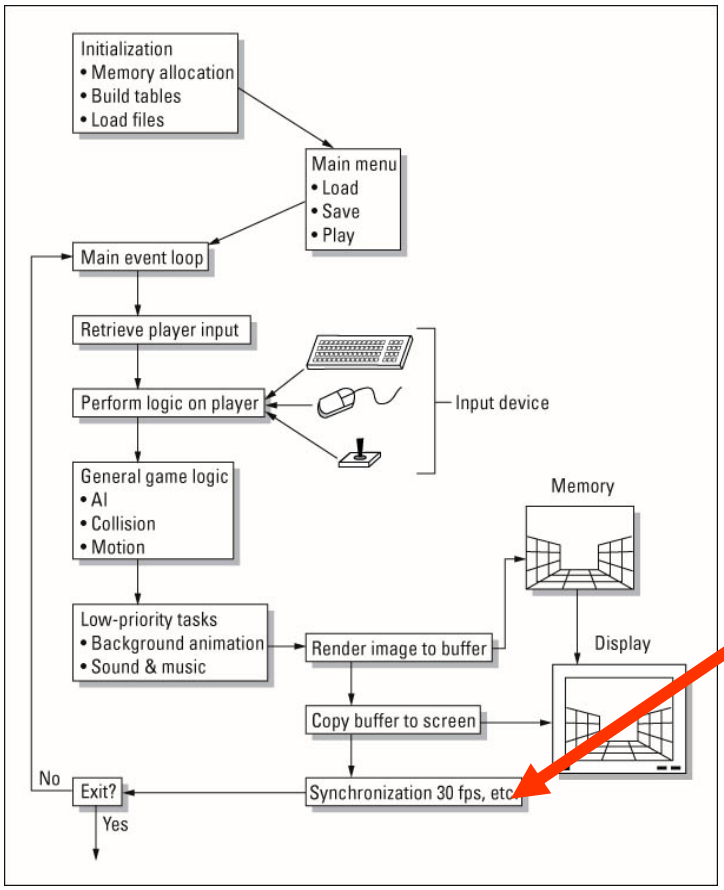
- 프레임 속도란?
  - 얼마나 빨리 프레임(일반적으로 하나의 완성된 화면)을 만들어 낼 수 있는지를 나타내는 척도
  - 일반적으로 초당 프레임 출력 횟수를 많이 사용한다.
    - FPS(Frame Per Sec)
  - 현대의 컴퓨터 게임에서는 일반적으로 최소 25~30 fps 이상이 기준임.

- 프레임 시간(frame time)이란?
  - 한 프레임을 만들어 내는데 걸리는 시간.

- 프레임 시간과 프레임 속도의 관계

Frame per sec = 1 / Frame time

게임 루프에서 프레임 속도의 조절이 필요!



• 일반적으로 게임 루프의 마지막 부분에서 프레임 속도를 동기화시킴.

Lecture08-01: 프레임 속도의 출력

Lecture08-01 프로젝트의 구성

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

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

gameengine.cpp (1)

```
void CGameEngine::Init(const char* title, int width, int height,
                                          int bpp, bool fullscreen)
{
  int flags = 0;
  SDL_Init(SDL_INIT_EVERYTHING);

  if ( fullscreen ) {
    flags = SDL_FULLSCREEN;
  }
  screen = SDL_SetVideoMode(width, height, bpp, flags);

  m_fullscreen = fullscreen;
  m_running = true;

  TTF_Init();
  font = TTF_OpenFont("HYNAMB.ttf", 20);

  curTicks = SDL_GetTicks();
}

void CGameEngine::UpdateScreen(void)
{
  SDL_Flip(screen);
}
```


gameengine.cpp (2)

```
void CGameEngine::RegulateFPS()
{
  SDL_Color fgColor={0,0,0}, bgColor = {128, 128, 128};
  char fps[40];

  Uint32 elapsedTicks = SDL_GetTicks() - curTicks;
  curTicks += elapsedTicks;
  sprintf(fps, "%3d ms, %6.1f fps", elapsedTicks, 1000.0 / (float)elapsedTicks);

  SDL_Surface *fpsMsg = TTF_RenderText_Shaded(font, fps, fgColor, bgColor);
  SDL_Rect pos = {10, 240, 0, 0};
  SDL_BlitSurface(fpsMsg, NULL, screen, &pos);
  SDL_FreeSurface(fpsMsg);
}
```

main.cpp

```
int main(int argc, char *argv[])
{
  CGameEngine game;

  game.Init("Engine Test v1.0");

  freopen("CON", "w", stdout);
  freopen("CON", "w", stderr);

  game.ChangeState(&introState);

  while ( game.Running() )
  {
    game.HandleEvents();
    game.Update();
    game.Draw();
     game.RegulateFPS();
     game.UpdateScreen();
  }
  game.Cleanup();

  return 0;
}
```


실행 화면

시간의 계산

Uint32 SDL_GetTicks(void);

```
void CGameEngine::Init(const char* title, int width, int height,
                                                    int bpp, bool fullscreen)
{
  int flags = 0;
  SDL_Init(SDL_INIT_EVERYTHING);
  if ( fullscreen ) {
    flags = SDL_FULLSCREEN;
  }
  screen = SDL_SetVideoMode(width, height, bpp, flags);
  m_fullscreen = fullscreen;
  m_running = true;
  TTF_Init();
  font = TTF_OpenFont("HYNAMB.ttf", 20);

  curTicks = SDL_GetTicks();
}
```

- SDL 라이브러리의 초기화(SDL_Init() 함수 호출)이후 경과된 시간을 얻음.
- ms 단위로 표현.(ms 보다 작은 시간은 표현못함.)

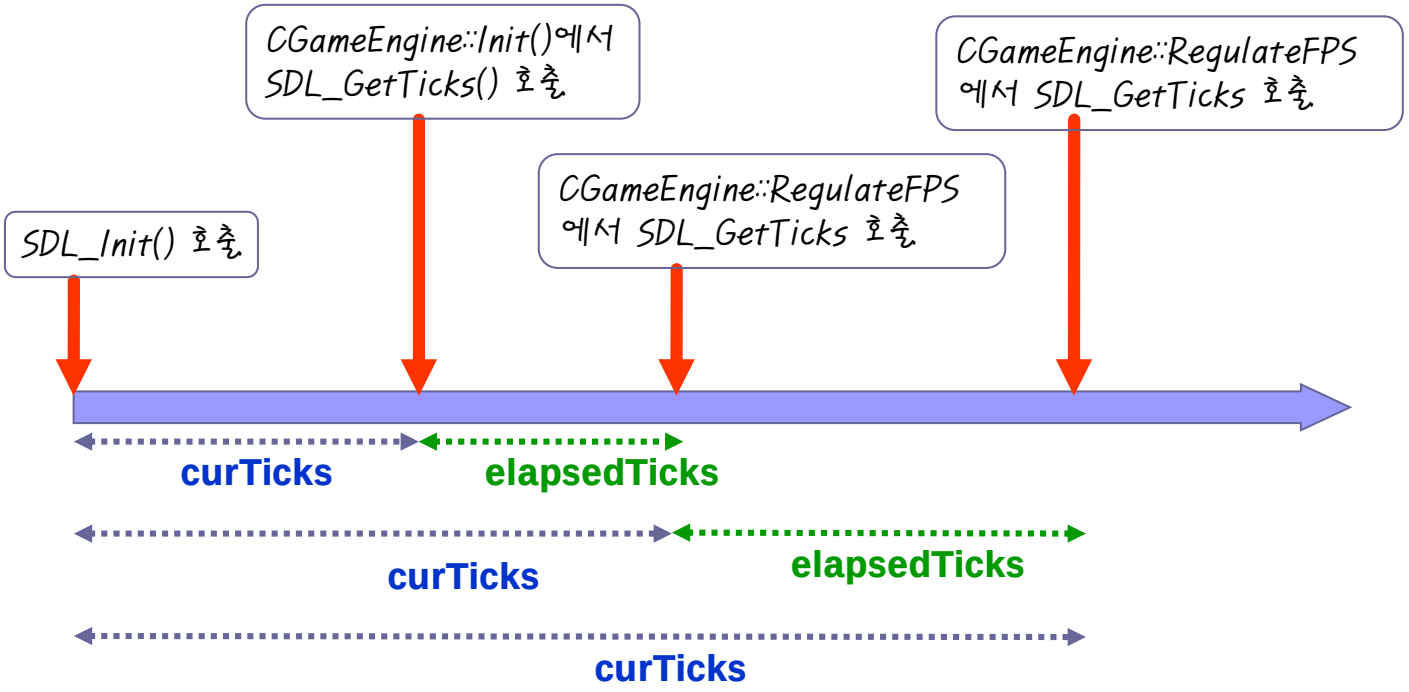
SDL_Flip의 중앙 처리

```
void CGameEngine::UpdateScreen(void)
{
  SDL_Flip(screen);
}
```

- 앞선 강의에서는, SDL_Flip의 호출을 각 State의 Draw에서 했으나, 여기서는 gameegine 클래스에서 중앙 처리.
- 프레임속도를 측정하고, 최종적으로 SDL_Flip() 을 호출하여, 화면을 업데이트하기 위해.

```
while ( game.Running() )
 {
   game.HandleEvents();
   game.Update();
   game.Draw();
   game.RegulateFPS();
   game.UpdateScreen();
 }
```

프레임 처리 시간의 계산



경과 시간 elapsedTicks = SDL_GetTicks() - curTicks;

프레임 속도 FPS = 1000 / elapsedTicks;

코드

```
void CGameEngine::RegulateFPS()
{
  SDL_Color fgColor={0,0,0}, bgColor = {128, 128, 128};
  char fps[40];

  Uint32 elapsedTicks = SDL_GetTicks() - curTicks;

  curTicks += elapsedTicks;

  sprintf(fps, "%3d ms, %6.1f fps", elapsedTicks, 1000.0 /
   (float)elapsedTicks);

  SDL_Surface *fpsMsg = TTF_RenderText_Shaded(font, fps,
   fgColor, bgColor);

  SDL_Rect pos = {10, 240, 0, 0};
  SDL_BlitSurface(fpsMsg, NULL, screen, &pos);
  SDL_FreeSurface(fpsMsg);
}
```

- 경과 시간의 계산
- curTicks의 갱신.
- 프레임 시간 및 프레임 속도를 문자열로 변환.
- 사각박스안에 문자 출력.

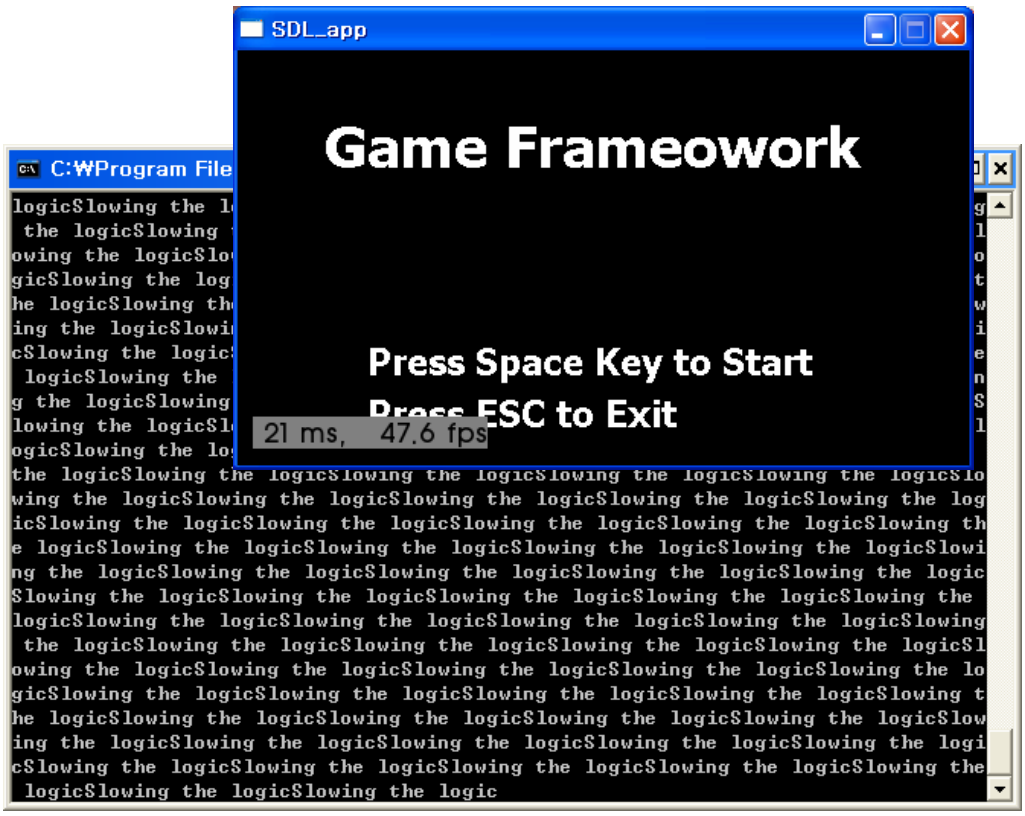
Lecture08-02: 프레임 속도의 조절

프레임 속도를 낮추는 무식한 방법

```
int main(int argc, char *argv[])
{
  CGameEngine game;
  game.Init("Engine Test v1.0");
  freopen("CON", "w", stdout);
  freopen("CON", "w", stderr);
  game.ChangeState(&introState);

  while ( game.Running() )
  {
    game.HandleEvents();
    game.Update();
    game.Draw();

    for (int i = 0; i < 100; i++)
      printf("Slowing the logic");

    game.RegulateFPS();
    game.UpdateScreen();
  }
  game.Cleanup();
  return 0;
}
```

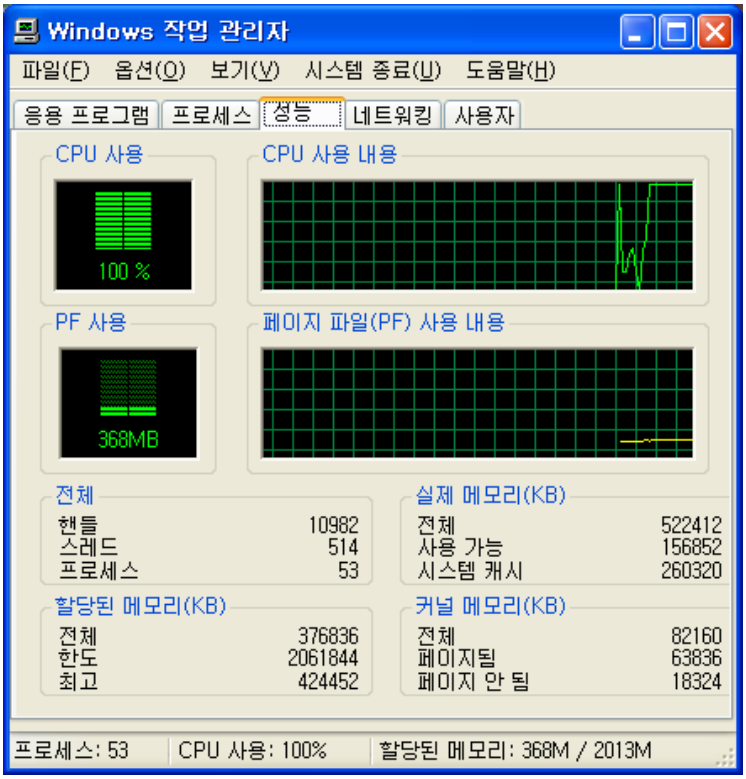
- 강제로 쓸데없는 반복을 돌린다.

실행 결과: fps가 낮아진다!



왜 무식한 방법인가?

- 일단, PC의 성능에 따라, fps가 달라진다.
- 더 심각한 것은, 운영체제의 시간 자원을 모두 다 잡아먹는다.



똑똑한 방법: gameengine.cpp

```
#define TARGET_FPS 30

void CGameEngine::RegulateFPS()
{
  SDL_Color fgColor={0,0,0}, bgColor = {128, 128, 128};
  char fps[40];

  Uint32 elapsedTicks = SDL_GetTicks() - curTicks;
  if (1000 / elapsedTicks > TARGET_FPS)
    SDL_Delay(1000 / TARGET_FPS - elapsedTicks);

  elapsedTicks = SDL_GetTicks() - curTicks;
  curTicks += elapsedTicks;
  sprintf(fps, "%3d ms, %6.1f fps", elapsedTicks, 1000.0 / (float)elapsedTicks);

  SDL_Surface *fpsMsg = TTF_RenderText_Shaded(font, fps, fgColor, bgColor);
  SDL_Rect pos = {10, 240, 0, 0};
  SDL_BlitSurface(fpsMsg, NULL, screen, &pos);
  SDL_FreeSurface(fpsMsg);
}
```


SDL_Delay() 함수의 사용

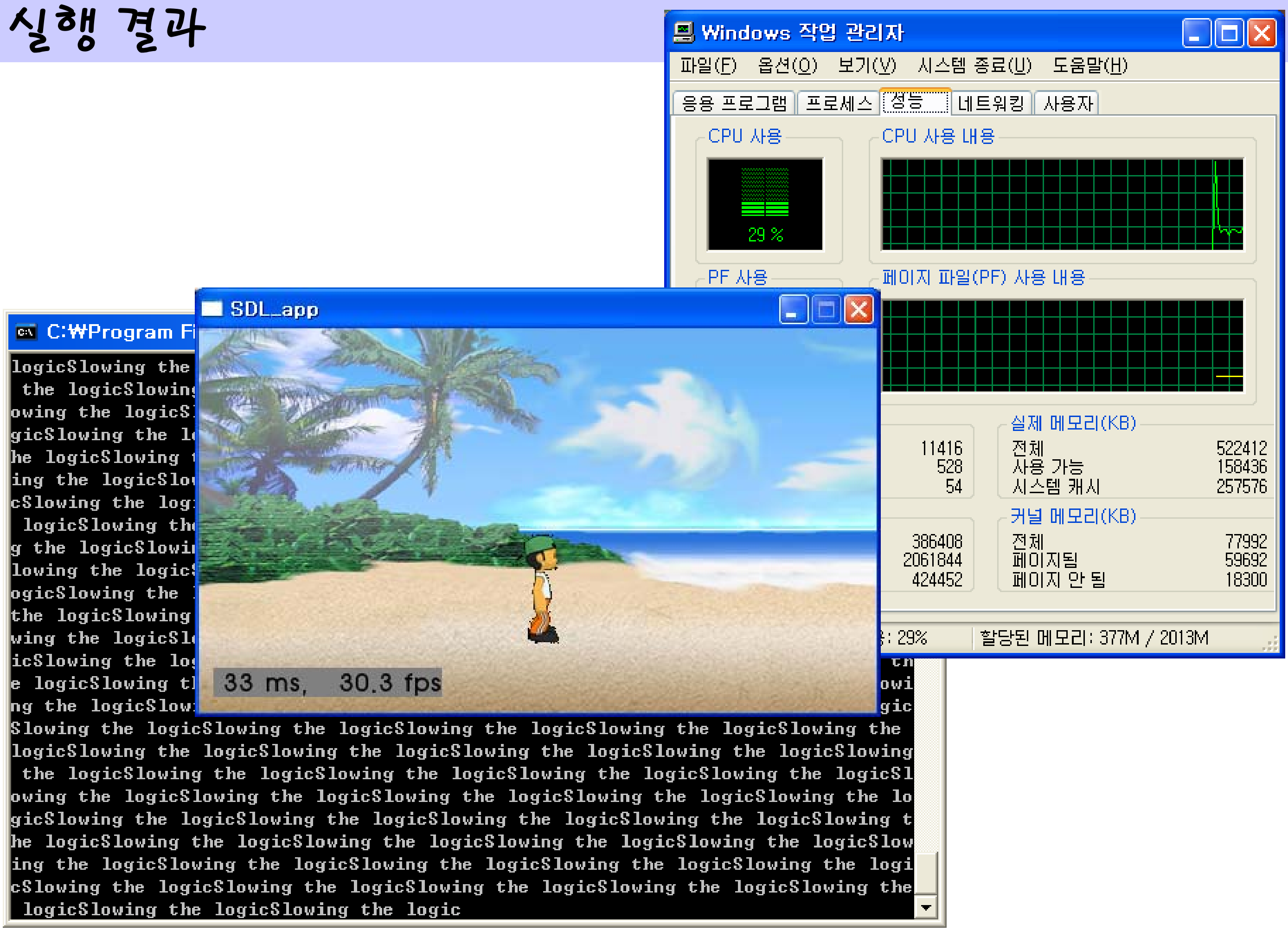
void SDL_Delay(Uint32 ms);

```
Uint32 elapsedTicks = SDL_GetTicks() - curTicks;

if (1000 / elapsedTicks > TARGET_FPS)

    SDL_Delay(1000 / TARGET_FPS - elapsedTicks);
```

현재 fps가 목표 fps보다 높으면

- 목표 fps를 맞추기 위해, 필요한 시간만큼 지연시킨다.
- SDL_Delay() 함수는 단순히 지연을 시킬 뿐만 아니라, 운영체제의 다른 프로세스에게 기회를 준다.

실행 결과



실습

Lecture08-03:
캐릭터 프레임 속도의 조절

sprite.cpp

```
void Sprite::updateFrame(void)
{
  static int count = 0;

  if (count % 3 == 0)
    frame = (frame + 1) % 8;

  count = (count + 1) % 3;
}
```

- 스프라이트 애니메이션의 초당 프레임 수를 1/3로 낮춘다.