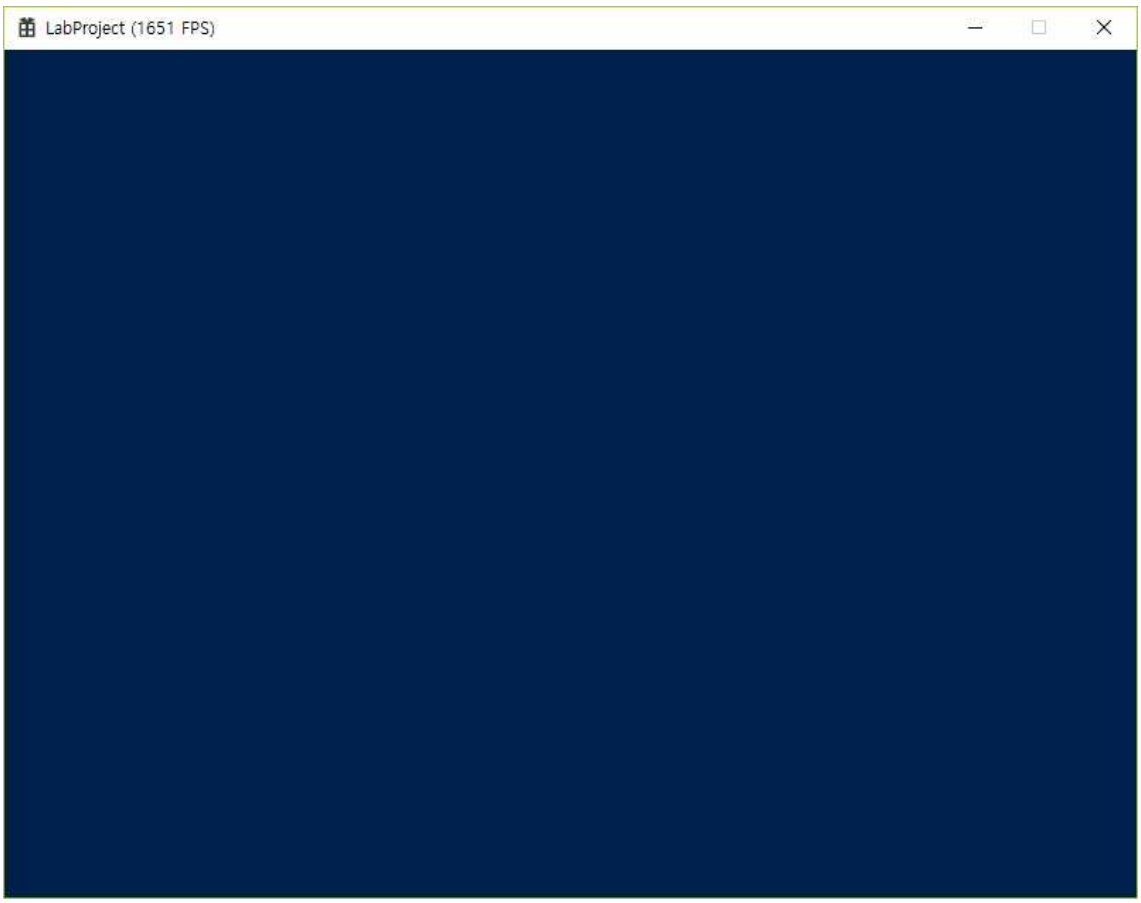
□ 예제 프로그램 4: LabProject03(프레임 레이트 출력하기)

이제 앞에서 작성한 프로그램의 기본 골격(LabProject02 프로젝트)를 기반으로 게임 프레임워크에 게임 타이머를 추가하자. 게임 타이머는 애니메이션 등을 처리할 때 유용하다. 그리고 1초에 몇 번의 프레임을 그리는 가를 나타내는 FPS(Frame Per Second)를 주 윈도우의 윈도우 타이틀(캡션: Caption)에 출력하도록 하자. 그리고 게임의 장면(Scene)을 관리하기 위한 클래스를 추가하도록 하자.



① 새로운 프로젝트의 생성

먼저 새로운 프로젝트 LabProject03을 생성하자. LabProject02를 생성할 때 설명한 것처럼 “LabProjects" 솔루션을 열고 솔루션 탐색기에서 마우스 오른쪽 버튼으로 『솔루션 LabProjects』를 선택하고 메뉴에서 『추가』, 『새 프로젝트』를 차례로 선택한다. 그러면 『새 프로젝트 대화상자』가 나타난다. 그러면 프로젝트 이름 “LabProject03”을 입력하고 『확인』을 선택한다.

② LabProject03.cpp 파일 수정하기

이제 “LabProject03.cpp" 파일의 내용을 “LabProject02.cpp" 파일의 내용으로 바꾸도록 하자. “LabProject03.cpp" 파일의 내용을 프로젝트 “LabProject02"에서 설명한 것처럼 게임 프레임워크와 관련된 부분으로 직접 변경하여도 좋다. 그러나 더 쉬운 방법은 파일의 내용 전체를 복사하여 붙여넣기를 하는 것이다.

솔루션 탐색기에서 프로젝트 “LabProject03"에서 “LabProject03.cpp" 파일을 연다(마우스 왼쪽 버튼을 더블 클릭한다). 같은 방법으로 프로젝트 “LabProject02"에서 “LabProject02.cpp" 파일을 연다. “LabProject02.cpp" 파일에서 ”Ctrl+a” 키를 누르면 파일의 내용 전체가 선택된다. ”Ctrl+c” 키를 눌러 선택된 파일의 내용을 클립보드(Clipboard)로 복사한다. 이제 “LabProject03.cpp" 파일에서 ”Ctrl+a” 키를 눌러 파일의 내용 전체를 선택하고 ”Ctrl+v” 키를 눌러 클립보드의 내용을 복사한다. 그러면 “LabProject02.cpp" 파일의 내용이 “LabProject03.cpp" 파일로 복사된다.

이제 “LabProject03.cpp" 파일에서 ”Ctrl+h” 키를 눌러 <그림 2>와 같이 “찾기 및 바꾸기” 대화상자가 나타나게 한다. “찾을 내용”에 “LabProject02”를 입력하고 “바꿀 내용”에 “LabProject03”을 입력한다. “찾는 위치”는 “현재 문서”를 선택한다. “모두 바꾸기” 버튼을 눌러 “LabProject02”를 “LabProject03”으로 바꾼다.

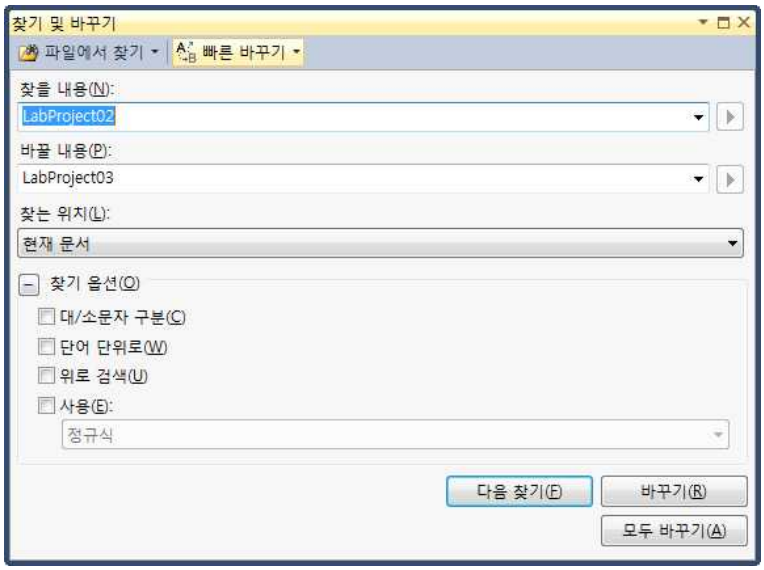


그림 2. 찾기 및 바꾸기 대화상자

다시 “찾을 내용”에 “LABPROJECT02”를 입력하고 “바꿀 내용”에 “LABPROJECT03”을 입력하고 “모두 바꾸기” 버튼을 눌러 “LABPROJECT02”를 “LABPROJECT03”으로 바꾼다.

이렇게 하면 “LabProject03.cpp" 파일의 내용이 게임 프레임워크 클래스와 연동될 수 있도록 변경된다. 앞으로 새로운 프로젝트를 만들 때마다 이 방법으로 “LabProject**xx**.cpp" 파일의 내용을 수정하도록 한다.

③ ”GameFramework.h“와 ”GameFramework.cpp“ 파일 추가하기

프로젝트 “LabProject02"에서와 같이 게임 프레임워크 클래스를 추가하여도 되지만 더 간단한 방법은 이미 존재하는 파일을 복사하고 프로젝트에 추가하는 것이다.

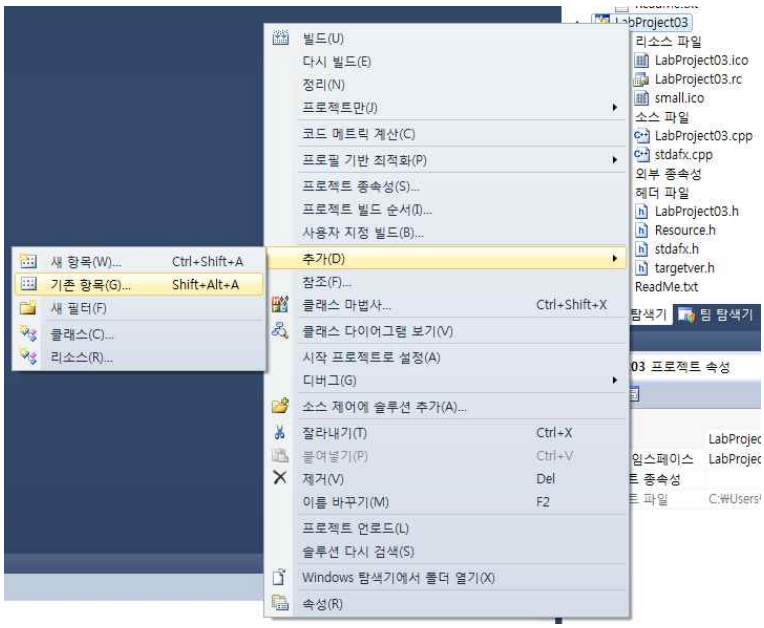


그림 3. 기존 항목 추가

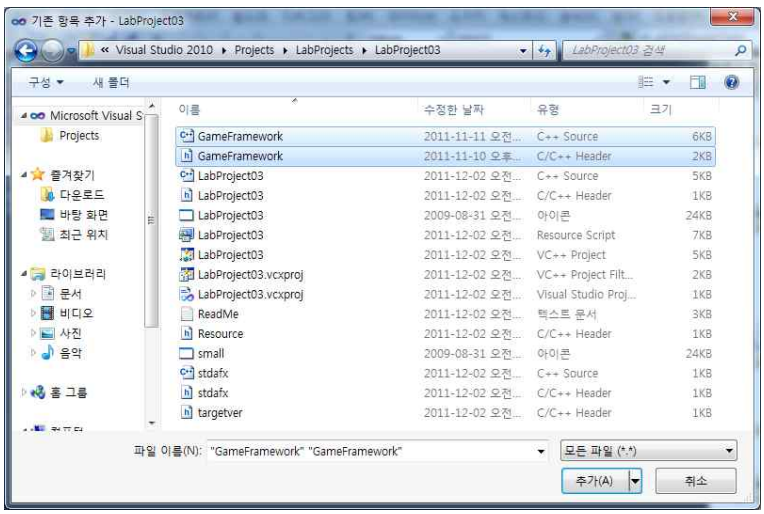


그림 4. 기존 항목 추가 대화상자

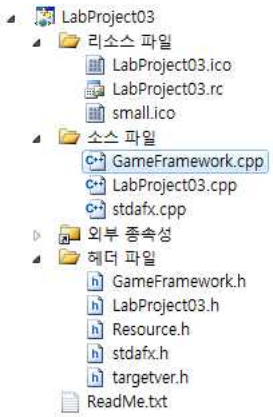


그림 5. 프로젝트

먼저 윈도우의 파일 탐색기에서 프로젝트 “LabProject02" 폴더의 ”GameFramework.h“와 ”GameFramework.cpp“ 파일을 복사하여 프로젝트 “LabProject03"의 폴더에 붙여 넣는다. 그리고 다음 <그림 3>과 같이 Visual Studio 솔루션 탐색기의 프로젝트 “LabProject03"에서 오른쪽 마우스 버튼을 누른다. 『추가』, 『기존 항목』을 차례로 선택한다. 그러면 <그림 4>와 같은 “기존 항목 추가“ 대화상자가 나타난다.

"Ctrl" 키를 누른 상태에서 마우스로 "GameFramework.h"와 "GameFramework.cpp" 파일을 선택한다. "추가" 버튼을 누르면 선택된 두 파일이 프로젝트 "LabProject03"에 추가된다. 그리고 파일 탐색기에서 프로젝트 "LabProject02"의 폴더의 "stdafx.h" 파일도 복사하여 프로젝트 "LabProject03"의 폴더에 붙여 넣는다.

④ "CGameTimer" 클래스 추가하기

프로젝트 "LabProject02"에서와 같은 방법으로 프로젝트 "LabProject03"에 새로운 클래스 "CGameTimer"를 추가하자. "CGameTimer" 클래스는 게임 프레임워크의 애니메이션 처리와 프레임 레이트를 계산하기 위한 타이머 관리 클래스이다(파일의 이름은 <그림 6>과 같이 Timer.h와 Timer.cpp로 설정한다).

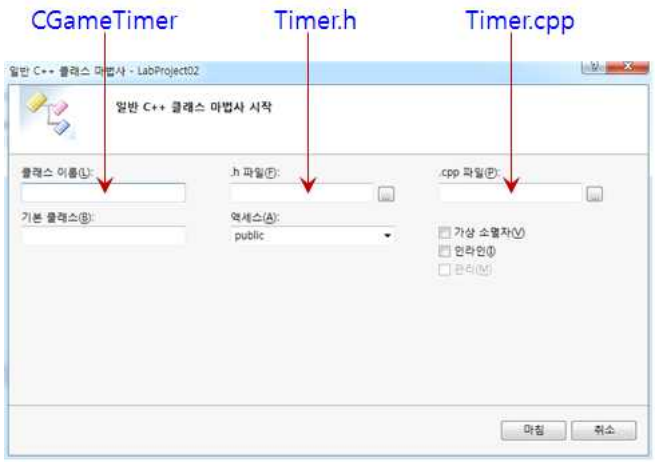


그림 6. CGameTimer 클래스 추가

⑤ "Timer.h" 파일의 내용을 다음과 같이 변경하자.

❶ 프레임 레이트(Frame Rate)의 계산은 일정한 횟수 동안 각 프레임을 렌더링하는 시간을 평균하여 계산한다. 이 횟수를 나타내는 상수를 정의한다.

```
const ULONG MAX_SAMPLE_COUNT = 50; // 50회의 프레임 처리시간을 누적하여 평균한다.

class CGameTimer
{
public:
   CGameTimer();
   virtual ~CGameTimer();
```

❷ 클래스의 멤버 함수를 다음과 같이 선언한다.

```
   void Tick(float fLockFPS = 0.0f);
   void Start();
   void Stop();
   void Reset();
   unsigned long GetFrameRate(LPTSTR lpszString = NULL, int nCharacters=0);
   float GetTimeElapsed();
   float GetTotalTime();
```

함수	설명
Tick	타이머의 시간을 갱신한다.
GetFrameRate	프레임 레이트를 반환한다.
GetTimeElapsed	프레임의 평균 경과 시간을 반환한다.

❸ 클래스의 멤버 변수를 다음과 같이 선언한다.

```
private:
   bool m_bHardwareHasPerformanceCounter;
   float m_fTimeScale;
   float m_fTimeElapsed;

   __int64 m_nCurrentTime;
   __int64 m_nLastTime;

   __int64 m_nBaseTime;
   __int64 m_nPausedTime;
   __int64 m_nStopTime;

   __int64 m_nPerformanceFrequency;

   float m_fFrameTime[MAX_SAMPLE_COUNT];
   ULONG m_nSampleCount;

   unsigned long m_nCurrentFrameRate;
   unsigned long m_nFramesPerSecond;
   float m_fFPSTimeElapsed;

   bool m_bStopped;
};
```

변수	설명
m_bHardwareHasPerformanceCounter	컴퓨터가 Performance Counter를 가지고 있는 가
m_fTimeScale	Scale Counter의 양
m_fTimeElapsed	마지막 프레임 이후 지나간 시간
m_nCurrentTime	현재의 시간
m_nLastTime	마지막 프레임의 시간
m_nPerformanceFrequency	컴퓨터의 Performance Frequency
m_fFrameTime	프레임 시간을 누적하기 위한 배열
m_nSampleCount	누적된 프레임 횟수
m_nCurrentFrameRate	현재의 프레임 레이트
m_nFramesPerSecond	초당 프레임 수
m_fFPSTimeElapsed	프레임 레이트 계산 소요 시간

⑥ "Timer.cpp" 파일의 내용을 다음과 같이 변경하자.

❶ CGameTimer 클래스의 생성자와 소멸자를 다음과 같이 정의한다.
컴퓨터가 성능 카운터(Performance Counter) 하드웨어를 가지고 있으면 성능 카운터와 성능

주파수(Performance Frequency)를 사용하여 시간 단위를 설정한다. 성능 주파수를 사용할 수 없으면 멀티미디어 타이머를 사용한다. 이 경우 시간 단위는 0.001초(Millisecond)이다. 성능 카운터에 대한 자세한 설명은 윈도우 API 도움말을 참조하라.

```
CGameTimer::CGameTimer()
{
   if (::QueryPerformanceFrequency((LARGE_INTEGER *)&m_nPerformanceFrequency))
   {
      m_bHardwareHasPerformanceCounter = TRUE;
      ::QueryPerformanceCounter((LARGE_INTEGER *)&m_nLastTime);
      m_fTimeScale = 1.0f / m_nPerformanceFrequency;
   }
   else
   {
      m_bHardwareHasPerformanceCounter = FALSE;
      m_nLastTime= ::timeGetTime();
      m_fTimeScale = 0.001f;
   }

   m_nSampleCount = 0;
   m_nCurrentFrameRate = 0;
   m_nFramesPerSecond = 0;
   m_fFPSTimeElapsed = 0.0f;
}

CGameTimer::~CGameTimer()
{
}
```

❷ CGameTimer::Tick() 멤버 함수

컴퓨터(CPU)가 성능 카운터(Performance Counter) 하드웨어를 가지고 있으면 성능 카운터와 성능 주파수(Performance Frequency)를 사용하여 현재 시간을 갱신하고 없으면 멀티미디어 타이머를 사용한다.

```
void CGameTimer::Tick(float fLockFPS)
{
   if (m_bHardwareHasPerformanceCounter)
   {
      ::QueryPerformanceCounter((LARGE_INTEGER *)&m_nCurrentTime);
   }
   else
   {
      m_nCurrentTime = ::timeGetTime();
   }

//마지막으로 이 함수를 호출한 이후 경과한 시간을 계산한다.
   float fTimeElapsed = (m_nCurrentTime - m_nLastTime) * m_fTimeScale;

   if (fLockFPS > 0.0f)
   {
//이 함수의 파라메터(fLockFPS)가 0보다 크면 이 시간만큼 호출한 함수를 기다리게 한다.
      while (fTimeElapsed < (1.0f / fLockFPS))
      {
         if (m_bHardwareHasPerformanceCounter)
         {
```

```
            ::QueryPerformanceCounter((LARGE_INTEGER *)&m_nCurrentTime);
         }
         else
         {
            m_nCurrentTime = ::timeGetTime();
         }
//마지막으로 이 함수를 호출한 이후 경과한 시간을 계산한다.
         fTimeElapsed = (m_nCurrentTime - m_nLastTime) * m_fTimeScale;
      }
   }

//현재 시간을 m_nLastTime에 저장한다.
   m_nLastTime = m_nCurrentTime;

/* 마지막 프레임 처리 시간과 현재 프레임 처리 시간의 차이가 1초보다 작으면 현재 프레임 처리 시간
을 m_fFrameTime[0]에 저장한다. */
   if (fabsf(fTimeElapsed - m_fTimeElapsed) < 1.0f)
   {
      ::memmove(&m_fFrameTime[1], m_fFrameTime, (MAX_SAMPLE_COUNT - 1) *
sizeof(float));
      m_fFrameTime[0] = fTimeElapsed;
      if (m_nSampleCount < MAX_SAMPLE_COUNT) m_nSampleCount++;
   }

//초당 프레임 수를 1 증가시키고 현재 프레임 처리 시간을 누적하여 저장한다.
   m_nFramesPerSecond++;
   m_fFPSTimeElapsed += fTimeElapsed;
   if (m_fFPSTimeElapsed > 1.0f)
   {
      m_nCurrentFrameRate = m_nFramesPerSecond;
      m_nFramesPerSecond = 0;
      m_fFPSTimeElapsed = 0.0f;
   }

//누적된 프레임 처리 시간의 평균을 구하여 프레임 처리 시간을 구한다.
   m_fTimeElapsed = 0.0f;
   for (ULONG i = 0; i < m_nSampleCount; i++) m_fTimeElapsed += m_fFrameTime[i];
   if (m_nSampleCount > 0) m_fTimeElapsed /= m_nSampleCount;
}
```

❸ CGameTimer::GetFrameRate() 멤버 함수

마지막 프레임 레이트를 문자열과 정수형으로 반환한다.

```
unsigned long CGameTimer::GetFrameRate(LPTSTR lpszString, int nCharacters)
{
//현재 프레임 레이트를 문자열로 변환하여 lpszString 버퍼에 쓰고 " FPS"와 결합한다.
   if (lpszString)
   {
      _itow_s(m_nCurrentFrameRate, lpszString, nCharacters, 10);
      wcscat_s(lpszString, nCharacters, _T(" FPS)"));
   }

   return(m_nCurrentFrameRate);
}
```

❹ CGameTimer::GetTimeElapsed() 멤버 함수

```
float CGameTimer::GetTimeElapsed()
{
   return(m_fTimeElapsed);
}
```

❺ CGameTimer::Reset() 멤버 함수

```
void CGameTimer::Reset()
{
   __int64 nPerformanceCounter;
   ::QueryPerformanceCounter((LARGE_INTEGER*)&nPerformanceCounter);

   m_nBaseTime = nPerformanceCounter;
   m_nLastTime = nPerformanceCounter;
   m_nStopTime = 0;
   m_bStopped = false;
}
```

❻ CGameTimer::Start() 멤버 함수

```
void CGameTimer::Start()
{
   __int64 nPerformanceCounter;
   ::QueryPerformanceCounter((LARGE_INTEGER *)&nPerformanceCounter);
   if (m_bStopped)
   {
      m_nPausedTime += (nPerformanceCounter - m_nStopTime);
      m_nLastTime = nPerformanceCounter;
      m_nStopTime = 0;
      m_bStopped = false;
   }
}
```

❼ CGameTimer::Stop() 멤버 함수

```
void CGameTimer::Stop()
{
   if (!m_bStopped)
   {
      ::QueryPerformanceCounter((LARGE_INTEGER *)&m_nStopTime);
      m_bStopped = true;
   }
}
```

❽ CGameTimer::GetTotalTime() 멤버 함수

```
float CGameTimer::GetTotalTime()
{
   if (m_bStopped) return(float(((m_nStopTime - m_nPausedTime) - m_nBaseTime) *
m_fTimeScale));
   return(float(((m_nCurrentTime - m_nPausedTime) - m_nBaseTime) *
m_fTimeScale));
}
```

⑦ "GameFramework.h" 파일의 내용을 다음과 같이 변경한다.

❶ "GameFramework.h" 파일의 앞쪽에 다음을 추가한다.

```
#include "Timer.h"
```

❷ "CGameFramework" 클래스에 다음 멤버 변수를 추가한다.

```
private:
//다음은 게임 프레임워크에서 사용할 타이머이다.
   CGameTimer m_GameTimer;

//다음은 프레임 레이트를 주 윈도우의 캡션에 출력하기 위한 문자열이다.
   _TCHAR m_pszFrameRate[50];
```

⑧ "GameFramework.cpp" 파일의 내용을 다음과 같이 변경한다.

❶ "CGameFramework" 클래스의 생성자에 다음을 추가한다.

```
   _tcscpy_s(m_pszFrameRate, _T("LapProject ("));
```

❷ "CGameFramework" 클래스의 FrameAdvance() 멤버 함수를 다음과 같이 변경한다. 타이머의 시간이 갱신되도록 하고 프레임 레이트를 계산한다.

```
void CGameFramework::FrameAdvance()
{
//타이머의 시간이 갱신되도록 하고 프레임 레이트를 계산한다.
   m_GameTimer.Tick(0.0f);

   ProcessInput();

   ...

   m_pDXGISwapChain->Present(0, 0);

/*현재의 프레임 레이트를 문자열로 가져와서 주 윈도우의 타이틀로 출력한다. m_pszBuffer 문자열이 "LapProject ("으로 초기화되었으므로 (m_pszFrameRate+12)에서부터 프레임 레이트를 문자열로 출력하여 " FPS)" 문자열과 합친다.
   ::_itow_s(m_nCurrentFrameRate, (m_pszFrameRate+12), 37, 10);
   ::wcscat_s((m_pszFrameRate+12), 37, _T(" FPS)"));
*/
   m_GameTimer.GetFrameRate(m_pszFrameRate + 12, 37);
   ::SetWindowText(m_hWnd, m_pszFrameRate);
}
```