



C++로 시작하는 언리얼 4 게임 프로그래밍

윌리엄 셰리프 지음 / 구진수 옮김

C++로 시작하는
언리얼 4
게임 프로그래밍

윌리엄 셰리프 지음 / 구진수 옮김

C++로 시작하는 언리얼 4 게임 프로그래밍

초판발행 2016년 5월 11일

지은이 윌리엄 웨리프 지음 / 옮긴이 구진수 / 펴낸이 김태헌
펴낸곳 한빛미디어(주) / 주소 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / 팩스 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-801-6 15000 / 정가 24,000원

총괄 전태호 / 책임편집 김창수 / 기획·편집 김상민
디자인 표지/내지 여동일, 조판 금미향
마케팅 박상용, 송경석, 변지영 / 영업 김형진, 김진불, 조유미

이 책에 대한 의견이나 오타자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주세요.
한빛미디어 홈페이지 www.hanbit.co.kr / 이메일 ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

Copyright © Packt Publishing 2015. First published in the English language under the title 'Learning C++ by Creating Games with UE4' (9781784396572). This translation is published and sold by permission of Packt Publishing, which owns or controls all rights to publish and sell the same.

이 책의 저작권은 PACKT와 한빛미디어(주)에 있습니다.

저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금합니다.

지금 하지 않으면 할 수 없는 일이 있습니다.

책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.

한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

지은이_ **윌리엄 웨리프**

윌리엄 웨리프는 프로그래밍 경력 8년 이상의 C++ 프로그래머입니다. 게임 프로그래밍부터 웹 프로그래밍까지 다양한 프로그래밍을 경험했고, 7년 동안 대학교에서 대학 강사로 재직했습니다. 또한, 아이튠즈 스토어에 몇 가지 앱을 출시했습니다. strum과 MARSHALL OF THE ELITE SQUADRON가 대표작입니다.

역자 소개

윤건이_구진수

호서대 컴퓨터공학과를 졸업한 후 게임과 관계 없는 프로그램을 개발하다가 2015년 초에 게임 개발 분야로 이직할 것을 결심하고 현재 모 회사에서 UI 작업을 맡고 있습니다.

- E-mail : paser2@gmail.com

언리얼 세계에 오신 것을 환영합니다. 언리얼 엔진은 꽤나 강력한 툴이며 많은 게임 회사에서 사용하고 있습니다. 예전에는 언리얼 엔진이 유료여서 개인 사용자들이 사용하기엔 부담이 되었지만, 이제는 무료로 전환되었고 많은 개인/소형 팀들이 언리얼 엔진을 사용하여 다양한 게임 제작에 도전하고 있습니다. 아직까지 국내에는 블루프린트로 언리얼 게임을 작성하는 법에 대한 서적만 출시되어 있고, 코드를 사용하는 책은 없습니다. 아마 이 책이 나오게 된다면 처음 나오는 게 아닐까 싶네요.

게임 제작은 힘든 일입니다. 정말입니다. 원래 모든 분야에서 제작이라는 것 자체가 힘든 일이긴 합니다. 그럼에도 제작에 도전하셨다면 자신만의 작품을 끝까지 완성하기를 바랍니다. 포기만 하지 말고 천천히, 꾸준히 가면 됩니다. 우선 간단한 것부터 도전해보는 것도 좋겠네요. 이 책에서 제공하는 예제부터 차근차근 따라 하다 보면 조금씩 익숙해질 것입니다.

여러분의 건투를 빕니다.

_ 구진수

언리얼 엔진 4(UE4)로 게임을 만들어 보고 싶나요? 여기에 언리얼 엔진을 사용해야 하는 몇 가지 중요한 이유가 있습니다.

- UE4는 강력합니다: UE4는 아름답고 실제 같은 광원 효과와 물리 효과를 제공합니다.
- UE4는 다양한 디바이스와 호환합니다: UE4로 작성된 코드는 윈도우 데스크톱 기기, 맥 데스크톱 기기, 안드로이드 기기 및 iOS 기기에서 구동합니다.

따라서 UE4를 사용해 게임의 주요 부분을 완성하고 나면 iOS 마켓(iTunes App 스토어)과 안드로이드 마켓(Google Play 스토어)에 등록할 수 있습니다(물론, 추가적인 과정이 필요하긴 합니다. iOS와 안드로이드 인앱결재는 각각 프로그래밍해야 합니다).

그래서 게임 엔진이 뭔가요?

게임 엔진은 자동차 엔진과 유사합니다. 게임 엔진은 게임을 움직이게 합니다. 여러분이 게임 엔진에 원하는 것을 입력하면 엔진은 (C++ 코드와 UE4 에디터를 사용하여) 여러분이 입력한 것을 실제로 구현하게 합니다.

여러분이 UE4 게임 엔진으로 게임을 만드는 것은 실제 차 엔진에 몸체와 바퀴를 장착하는 방법과 비슷합니다. UE4로 만든 게임을 출시할 때 기본적으로 UE4 엔진을 꾸미고 자신만의 그래픽, 사운드, 코드를 장착할 것입니다.



언리얼 엔진4는 얼마나 하나요? —————

간단하게 말하자면, 언리얼 엔진은 무료입니다.

“뭐라고요?” 라고 말하시겠죠. 무료라고요?

네, 그래요. 무료로 AAA 급의 엔진 및 엔진의 모든 소스까지 다 내려받을 수 있습니다. 다른 엔진들이 라이선스 하나당 500달러에서 1,000달러가 필요하다는 것을 생각해보면 이진 굉장한 겁니다.

다만 언리얼 엔진 4로 만들어 출시한 게임의 분기당 매출이 3,000달러 이상인 경우에는 5%의 로열티를 지급해야 합니다.

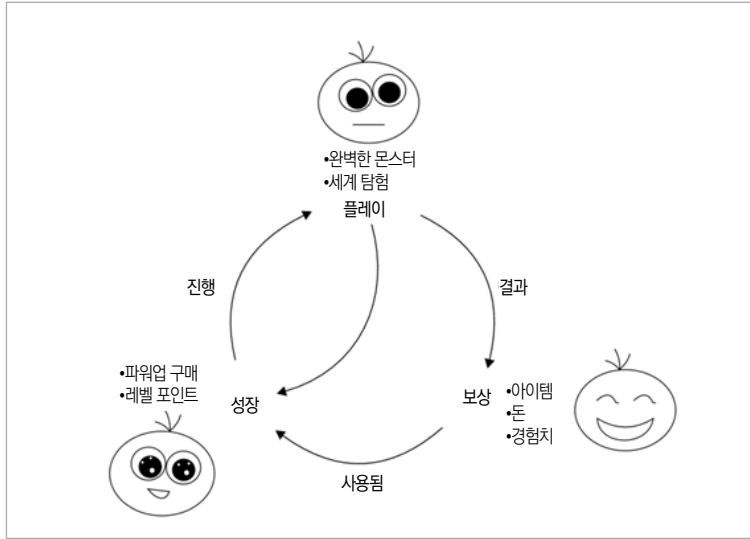
제가 스스로 엔진을 만들어서 5%를 아끼면 안 되나요? —————

아마도 여러분은 정해진 기한 내에 게임을 만들어야 할 상황일 것입니다. 이때 여러분이 게임 개발에만 집중할 수 있게 게임 엔진 부분을 담당할 엔진 전문 프로그래머가 없다면, 과연 여러분은 만들고자 하는 게임 개발에만 집중할 수 있을까요?

게임을 제작할 때 게임 엔진 프로그래밍에 신경 쓸 필요가 없게 된다면, 개발할 게임을 어떻게 만들지에 대해서만 집중할 수 있습니다. 이 말은 게임 엔진의 버그를 수정하거나 유지보수하기 위해서 전혀 시간을 허비할 필요가 없다는 것이죠.

게임의 개요-플레이-보상-성장 루프 —————

다음 다이어그램을 잘 살펴보기 바랍니다. 많은 초보 개발자가 처음 게임을 만들 때 놓치는 중요한 개념에 대해 설명하고 있습니다. 게임은 사운드 효과, 그래픽, 실제적인 물리 효과로 완성할 수 있지만, 이것만으로는 아직 게임처럼 보이진 않을 겁니다. 왜 그럴까요?



루프 위에서 시작하면, 플레이는 게임을 하는 동안 (몬스터를 물리치는 것과 같은) 어떠한 행동을 하고, 그 결과로 플레이어에게 (골드나 경험치 같은) 보상을 줍니다. 이런 성장은 게임 플레이를 새롭고 흥미로운 방향으로 이끌어 갑니다. 예를 들면, 새로운 무기는 전투 방식을 바꿀 수 있고, 새로운 마법 주문은 다수의 몬스터에게 전혀 다른 방법으로 공격할 수 있도록 하며, 새로운 형태의 이동 수단은 이전에 갈 수 없었던 곳으로 이동하게 해주죠.

이것은 재미있는 게임 플레이를 만들기 위한 기본적인 루프입니다. 가장 중요한 것은 플레이를 통해 어떤 종류의 보상이라도 얻을 수 있어야 한다는 것입니다. 악당을 처리하면 반짝이는 금 조각들이 떨어지는 것을 본 적이 있지요? 바로 그것입니다. 보상에서 중요한 점은, 이것이 게임 플레이 내에서 어떤 식으로든 성장과 관련이 되어야 한다는 점입니다. 롤플레이팅 게임에서 캐릭터의 레벨이 새로운 지역으로 이동할 때 얼마나 중요한 요소로 작용하는지 잘 알고 있겠죠.

플레이만 있는 (보상이나 성장이 없는) 게임은 게임이 주는 흥미에 한계가 있습니다. 단순히 프로토타입으로

생각될 뿐일테니까요. 예를 들면 거대한 세계를 배경으로 하는 비행 시뮬레이션 게임이 있는데 목적지나 목표도 없고 비행기나 무기의 업그레이드도 없다고 생각해보세요. 금방 흥미를 잃게 되겠죠.

하지만 플레이와 보상만 있는 (성장이 없는) 게임은 단순하고 무식하게 보일 겁니다. 사용할 데가 없는 보상을 플레이어가 만족할 일은 없겠죠.

또한 플레이와 성장만 있는 (보상이 없는) 게임은 단지 무의미하게 늘어나는 도전과제로만 보일 겁니다. 플레이어가 업적만을 이루기 좋아하는 사람이 아니라면 말이죠.

세 가지 요소가 갖춰진 게임은 플레이어가 계속 흥미를 느끼게 될 것입니다. 플레이는 보상이라는 결과(아이템과 스토리 진행)로 이어지며, 이 결과물은 게임 세계에서의 성장을 의미합니다. 이것을 게임 기획에 잘 활용한다면 흥미로운 게임을 디자인할 수 있을 것입니다.

NOTE

프로토타입은 게임의 컨셉트만 가지고 제작한 것입니다. 만약 여러분이 특별한 블랙잭을 만들고 싶다고 해봅시다. 먼저 해야 할 것은 어떻게 게임이 돌아갈지 보여줄 수 있는 프로토타입을 만드는 것이겠죠.

수익 창출

게임을 개발할 때 우선 생각해야 되는 요소 중 하나는 바로 수익 전략입니다. 게임으로 어떻게 돈을 벌 것인지에 관한 고민이지요. 만약 회사에서 게임을 개발한다면 수익 요소부터 먼저 생각하는 편이 좋습니다.

제임스타운, 배너 사가, 캐슬 크래셔즈 혹은 네크로댄서의 크리트 같은 게임처럼 게임 자체를 유료로 판매하여 수익을 만들 건가요? 아니면 클래시 오브 클랜즈, 캔디 크러시 사가, 서브웨이 서퍼즈와 같이 무료 플레이에 인앱결제를 도입하실 건가요?

많은 모바일 기기 게임은 (iOS의 건설 게임 같은) 유저가 돈을 내면 중간 단계를 건너뛰고 바로 성장 단계로 진행할 수 있는 방식을 통해서 많은 돈을 벌고 있습니다. 이런 방식이 굉장한 수익을 가져올 때도 있습니다. 많은 유저가 한 게임에 몇백 달러를 쓰기도 하니까요.

왜 C++인가요?

UE4는 C++로 프로그래밍되어 있습니다. UE4에서 코드를 작성하려면 C++를 반드시 알아야만 합니다.

또한 C++은 많은 게임 프로그래머가 사용하는 프로그래밍 언어입니다. 객체 지향 기능과 합쳐져 좋은 성능을 제공하기 때문입니다. 물론 C++ 자체만으로도 매우 강력하면서도 유연한 언어이기도 합니다.

이 책의 구성

1장 C++로 코딩하기에서는 첫 번째 C++ 프로그램을 직접 만들고 실행하는 방법에 대해서 설명합니다.

2장 변수와 메모리에서는 컴퓨터의 메모리에서 변수를 만들고 읽고 쓰는 방법에 대해서 설명합니다.

3장 if, else, switch에서는 코드의 분기에 대해서 설명합니다. 분기라는 것은 프로그램 상태에 따라 다른 부분의 코드가 실행되게 하는 것을 말합니다.

4장 루프에서는 특정 부분의 코드를 원하는 만큼 반복하는 법에 대해서 설명합니다.

5장 함수와 매크로에서는 여러분이 원하는 만큼 호출할 수 있는 코드의 집합인 함수에 대해서 설명합니다.

6장 객체, 클래스, 상속에서는 클래스 정의와 클래스 정의를 기반으로 한 객체의 인스턴스화에 대해서 설명합니다.

7장 동적 메모리 할당에서는 낮은 레벨의 C와 C++ 형식의 배열로 알려진 힙 할당 오브젝트에 대해 설명합니다.

8장 액터와 폰은 실제 UE4 코드를 사용하는 첫 번째 장입니다. 새로운 게임 월드를 생성하고 액터를 넣고 최적화된 액터에서 아바타 클래스를 생성해 볼 것입니다.

9장 템플릿과 자주 사용되는 컨테이너에서는 UE4와 C++ STL의 데이터 목록, 컨테이너에 대해 알아봅니다. 종종 올바른 컨테이너를 사용하는 것만으로도 프로그래밍에서 발생하는 문제가 훨씬 수월하게 해결되는 경우가 있습니다.



10장 인벤토리 시스템과 아이템 습득에서는 인벤토리 시스템과 새로운 아이템을 얻는 능력을 만들어 볼 것입니다.

11장 몬스터에서는 플레이어를 쫓는 몬스터와 무기로 몬스터를 공격하는 법에 대해서 살펴봅니다.

12장 마법 책에서는 주문을 만들고 게임 내에서 만든 주문을 사용하는 법에 대해 살펴봅니다.

준비할 것들

이 책에서는 두 가지 프로그램을 사용합니다. 하나는 통합 개발 환경(IDE)입니다. 다른 하나는 당연히 언리얼 엔진 4입니다.

만약 마이크로소프트 윈도우를 사용하고 있다면 Microsoft Visual Studio Express 2013 for Windows Desktop이 필요합니다. 혹시 맥을 사용하고 있다면 Xcode가 필요합니다. 언리얼 엔진 4는 <https://www.unrealengine.com>에서 내려받으실 수 있습니다.

누구를 위한 책인가요?

언리얼 엔진 4로 애플리케이션을 만들고 싶은 독자 모두를 위한 책입니다. 이 책에서는 C++ 애플리케이션을 컴파일하고 구현하는 방법부터 C++ 프로그래밍 언어의 규칙들을 알아볼 겁니다. 그리고 C++ 소개가 끝나면 C++을 사용해 게임을 만들어 볼 것입니다.

예제 소스

이 책의 예제 소스는 다음 URL을 통해 내려받을 수 있습니다.

- <http://www.hanbit.co.kr/exam/2816>

chapter 1 C++로 코딩하기 — 19

| | |
|----------------------------------|----|
| 1.1 새로운 프로젝트 만들기 | 19 |
| 1.1.1 윈도우에서 마이크로소프트 비주얼 C++ 사용하기 | 20 |
| 1.1.2 맥에서 XCode 사용하기 | 23 |
| 1.2 첫 번째 C++ 프로그램 만들기 | 26 |
| 1.2.1 세미콜론 | 30 |
| 1.2.2 에러 처리하기 | 30 |
| 1.2.3 주의 | 31 |
| 1.3 빌드와 컴파일은 무엇입니까? | 32 |
| 1.3.1 스크립트 | 32 |
| 1.4 요약 | 33 |

chapter 2 변수와 메모리 — 35

| | |
|-------------------------|----|
| 2.1 변수 | 36 |
| 2.1.1 변수 선언 - 실리콘을 건드린다 | 37 |
| 2.1.2 숫자가 전부입니다 | 38 |
| 2.1.3 변수에 대해서 | 39 |
| 2.1.4 C++에서의 수학 | 41 |
| 2.1.5 일반적인 변수 구문 | 43 |
| 2.1.6 기본 형식 | 43 |
| 2.1.7 객체 종류 | 44 |
| 2.1.8 포인터 | 48 |
| 2.1.9 포인터는 무엇을 하나요? | 49 |
| 2.1.10 주소 연산자 & | 49 |
| 2.2 정리 | 54 |

chapter 3 If, Else, Switch — 55

| | |
|--------|----|
| 3.1 분기 | 55 |
|--------|----|

| | | |
|-------|------------------------|----|
| 3.2 | 프로그램의 흐름 조절하기 | 56 |
| 3.2.1 | == 연산자 | 57 |
| 3.2.2 | if문 작성하기 | 57 |
| 3.2.3 | else문 작성하기 | 59 |
| 3.2.4 | 다른 비교 연산자를 사용하여 부등식 검사 | 60 |
| 3.3 | 논리 연산자 사용하기 | 61 |
| 3.3.1 | 부정 연산자 | 61 |
| 3.3.2 | 논리곱 연산자 | 63 |
| 3.3.3 | 논리합 연산자 | 63 |
| 3.4 | 언리얼 엔진을 사용한 첫 번째 예제 | 64 |
| 3.4.1 | 두 방향 이상으로 코드 분기하기 | 70 |
| 3.4.2 | else if 상태문 | 70 |
| 3.4.3 | switch 상태문 | 73 |
| 3.5 | 요약 | 78 |

chapter 4 루프 79

| | | |
|-------|-----------------|----|
| 4.1 | while 루프 | 79 |
| 4.1.1 | 무한 루프 | 81 |
| 4.2 | do/while 루프 | 84 |
| 4.3 | for 루프 | 84 |
| 4.4 | 언리얼 엔진에서 루프 만들기 | 87 |
| 4.5 | 요약 | 88 |

chapter 5 함수와 매크로 89

| | | |
|-------|------------------------------|----|
| 5.1 | 함수 | 89 |
| 5.2 | <cmath> 라이브러리 함수 예제 - sqrt() | 90 |
| 5.3 | 함수를 만들어 봅시다 | 92 |
| 5.3.1 | 예제 프로그램 분석 | 93 |

| | |
|---|-----|
| 5.4 인자를 사용하는 함수 | 96 |
| 5.5 값을 반환하는 함수 | 97 |
| 5.6 변수 다시 보기 | 100 |
| 5.6.1 전역 변수 | 100 |
| 5.6.2 지역 변수 | 101 |
| 5.6.3 변수의 범위 | 101 |
| 5.6.4 정적 지역 변수 | 103 |
| 5.6.5 const 변수 | 104 |
| 5.6.6 함수 프로토타입 | 104 |
| 5.6.7 h와 .cpp 파일 | 105 |
| 5.6.8 prototypes.h 내용 | 106 |
| 5.6.9 funcs.cpp 내용 | 106 |
| 5.6.10 main.cpp 내용 | 107 |
| 5.6.11 외부 변수 | 107 |
| 5.7 매크로 | 108 |
| 5.7.1 조건 - 가능하면 const 변수를 사용하세요 | 109 |
| 5.8 인자를 가진 매크로 | 109 |
| 5.8.1 조건 - 인자를 가진 매크로 대신에 인라인 함수를 사용하세요 | 110 |
| 5.9 정리 | 111 |

chapter 6 객체, 클래스, 상속 113

| | |
|---------------------------------|-----|
| 6.1 구조체 객체 | 114 |
| 6.1.1 멤버 함수 | 114 |
| 6.1.2 문자열은 객체인가요? | 116 |
| 6.1.3 멤버 함수 실행 | 116 |
| 6.1.4 private와 캡슐화 | 119 |
| 6.1.5 public으로 하기 좋아하는 몇몇 사람들 | 121 |
| 6.2 클래스 대 구조체 | 121 |
| 6.3 getter와 setter | 122 |
| 6.3.1 Getter | 123 |
| 6.3.2 Setter | 124 |
| 6.3.3 get/set 함수의 중요한 점이 무엇입니까? | 124 |
| 6.4 생성자와 소멸자 | 126 |
| 6.5 클래스 상속 | 127 |
| 6.5.1 파생 클래스 | 127 |
| 6.5.2 is-a 관계 | 132 |

| | |
|------------------------|-----|
| 6.5.3 protected 변수 | 133 |
| 6.5.4 가상 함수 | 133 |
| 6.5.5 순수 가상 함수와 추상 클래스 | 134 |
| 6.6 다중 상속 | 135 |
| 6.6.1 private 상속 | 135 |
| 6.7 클래스를 헤더로 넣어봅시다 | 136 |
| 6.7.1 .h와 .cpp | 139 |
| 6.8 정리 | 141 |

chapter 7 동적 메모리 할당 143

| | |
|---------------------------------------|-----|
| 7.1 동적 메모리 할당 | 144 |
| 7.1.1 delete 키워드 | 145 |
| 7.1.2 메모리 누출 | 145 |
| 7.2 일반적인 배열 | 147 |
| 7.2.1 배열 구문 | 147 |
| 7.3 C++ 형식의 동적 크기 배열(new[]와 delete[]) | 149 |
| 7.4 C 형식의 동적 배열 | 151 |
| 7.5 정리 | 152 |

chapter 8 액터와 폰 153

| | |
|----------------------------------|-----|
| 8.1 액터 대 폰 | 153 |
| 8.2 액터를 넣을 월드 생성 | 154 |
| 8.3 UE4 에디터 | 157 |
| 8.3.1 에디터 조작법 | 157 |
| 8.3.2 플레이 모드 조작법 | 157 |
| 8.3.3 씬에 객체 추가하기 | 158 |
| 8.4 처음부터 시작하기 | 160 |
| 8.4.1 광원 추가 | 161 |
| 8.4.2 콜리전 볼륨 | 162 |
| 8.5 씬에 액터 추가 | 165 |
| 8.6 플레이어 본체 생성 | 165 |
| 8.6.1 UE4 GameFramework 클래스 상속받기 | 165 |
| 8.6.2 메시 로딩하기 | 169 |

| | |
|---------------------------------|-----|
| 8.7 게임 캐릭터를 조작하기 위한 C++ 코드 작성 | 177 |
| 8.7.1 아바타 클래스의 인스턴스를 플레이어로 설정하기 | 177 |
| 8.7.2 컨트롤러 입력 설정 | 179 |
| 8.7.3 요(Yaw)와 피치(Pitch) | 183 |
| 8.8 논-플레이어 캐릭터 엔티티 생성 | 184 |
| 8.9 각 NPC의 대화상자에 인용구 표시하기 | 188 |
| 8.9.1 HUD에서 메시지 출력하기 | 188 |
| 8.9.2 TArray<Message> 사용 | 192 |
| 8.9.3 NPC에 접근했을 때 이벤트 실행하기 | 195 |
| 8.10 정리 | 200 |

chapter 9 템플릿과 자주 사용되는 컨테이너들 201

| | |
|-------------------------------|-----|
| 9.1 UE4에서 결과 디버깅하기 | 201 |
| 9.2 UE4의 TArray<T> | 202 |
| 9.2.1 TArray<T>를 사용하는 예제 | 203 |
| 9.2.2 TArray 순차 접근 | 204 |
| 9.2.3 TArray에 요소가 있는지 찾기 | 206 |
| 9.3 TSet<T> | 207 |
| 9.3.1 TSet 순차 접근 | 207 |
| 9.3.2 TSet 교차하기 | 208 |
| 9.3.3 TSet 결합하기 | 208 |
| 9.3.4 TSet 검색하기 | 209 |
| 9.4 TMap<T, S> | 209 |
| 9.4.1 플레이어 인벤토리의 아이템 목록 | 209 |
| 9.4.2 TMap 순차 접근 | 210 |
| 9.5 C++ STL 버전의 자주 사용되는 컨테이너들 | 210 |
| 9.5.1 C++ STL set | 211 |
| 9.5.2 C++ STL map | 213 |
| 9.6 정리 | 215 |

chapter 10 인벤토리 시스템과 아이템 줌기 217

| | |
|---------------|-----|
| 10.1 백팩 선언하기 | 217 |
| 10.1.1 전방 선언 | 218 |
| 10.1.2 예셋 임포팅 | 219 |

| | |
|------------------------------|-----|
| 10.1.3 액션 매핑을 키와 연결하기 | 222 |
| 10.2 기본 클래스 PickupItem | 224 |
| 10.2.1 루트 컴포넌트 | 226 |
| 10.3 플레이어 인벤토리 그리기 | 230 |
| 10.3.1 HUD::DrawTexture() 사용 | 230 |
| 10.3.2 인벤토리 아이템 클릭 확인 | 234 |
| 10.4 요약 | 238 |

chapter 11 몬스터 — 239

| | |
|--------------------|-----|
| 11.1 배경 | 240 |
| 11.1.1 배경 꾸미기 | 242 |
| 11.2 몬스터 | 243 |
| 11.2.1 기본 몬스터 지능 | 247 |
| 11.3 몬스터가 플레이어를 공격 | 254 |
| 11.3.1 근접 공격 | 254 |
| 11.3.2 소켓 | 260 |
| 11.3.3 원거리 공격 | 278 |
| 11.3.4 플레이어 낙백 | 285 |
| 11.4 정리 | 286 |

chapter 12 마법 책 — 287

| | |
|---------------------------------------|-----|
| 12.1 파티클 시스템 | 288 |
| 12.1.1 파티클 속성 변환 | 290 |
| 12.1.2 블리자드 주문 설정 | 292 |
| 12.2 Spell 클래스 액터 | 297 |
| 12.2.1 주문 블루프린트 만들기 | 300 |
| 12.2.2 주문 줄기 | 301 |
| 12.3 오른쪽 마우스 클릭을 주문 시전과 연결하기 | 304 |
| 12.3.1 아바타의 CastSpell 함수 작성하기 | 305 |
| 12.3.2 AMYHUD::MouseRightClicked() 작성 | 307 |
| 12.4 다른 주문 작성 | 310 |
| 12.4.1 불 주문 | 310 |
| 12.5 정리 | 312 |

여러분은 아마도 초급 프로그래머일 것입니다. 배울 것이 많습니다. 어쩌면 학교에서 이론적인 프로그래밍 개념을 배웠을 수도 있습니다. 그런 건 학교에 있는 분들께 맡겨두겠습니다. 이 책에서는 이론적인 부분만을 다루지는 않을 겁니다. 먼저 C++ 개념을 설명한 뒤 이론적인 부분을 다루고 궁극적으로는 여러분만의 게임을 구현할 수 있도록 돕겠습니다. 들어가기에 앞서 여러분께 권하고 싶은 사항은 연습해보라는 것입니다. 읽기만 해서는 프로그래밍을 배울 수 없습니다. 이론과 연습을 함께해야 합니다.

이제 C++로 아주 단순한 프로그램을 만들어 보겠습니다. 여러분은 지금 당장 게임을 완성하고 완성된 게임을 해보고 싶겠죠. 그러나 여러분이 게임을 완성하려면 처음부터 시작해야 합니다(처음부터 보기가 싫다면, 12장으로 넘어가거나 책 중간의 아무 예제나 열어서 살펴보세요).

이 장에서는 다음의 주제에 대해 알아볼 것입니다.

- 새로운 프로젝트를 만듭니다(비주얼 스튜디오 및 Xcode)
- 첫 번째 C++ 프로젝트
- 에러 처리하는 방법
- 빌드와 컴파일은 무엇인가요?

1.1 새로운 프로젝트 만들기

첫 번째 C++ 프로그램은 UE4로 작성하지 않습니다. 시작하기 전에 Xcode와 비주얼 스튜디오 2013을 학습할 것입니다. 하지만 1.2부터는 윈도우를 쓰는 맥을 쓰는 상관없이 C++ 코드에 대해서만 설명할 겁니다.

1.1.1 윈도우에서 마이크로소프트 비주얼 C++ 사용하기

윈도우용 코드 편집기인 마이크로소프트 비주얼 스튜디오를 설치합니다. 맥을 사용한다면 1.1.2로 넘어가 주세요.

NOTE

비주얼 스튜디오 익스프레스 에디션은 마이크로소프트가 제공하는 비주얼 스튜디오의 무료 버전입니다. <https://www.visualstudio.com/downloads/download-visual-studio-vs>로 가서 왼쪽 하단의 Visual Studio 2013을 클릭하여 생성되는 항목 중 Express 2013 for Desktop을 클릭하여 설치를 시작하세요.

시작하려면 Express 2013 for Windows Desktop을 내려받아서 설치하세요. 아이콘은 다음과 같습니다.



NOTE

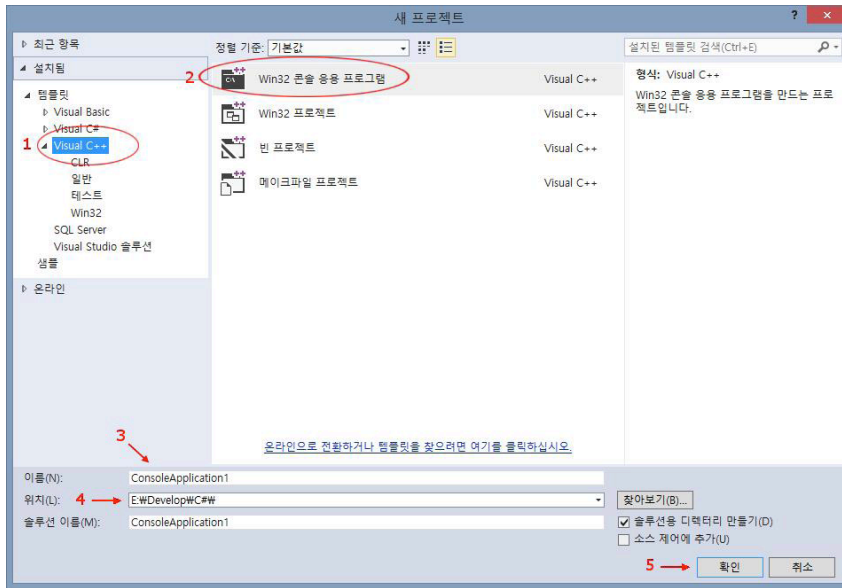
Express 2013 for Windows는 설치하지 마세요. 다른 패키지입니다. 따라서 이 책의 쓰임과 약간 다르게 사용됩니다.

설치가 끝났으면 실행해보세요. 다음 단계를 따라 하면서 실제로 코드를 입력해보겠습니다.

1. 파일 메뉴에서 새 프로젝트를 선택하세요.



2. 다음과 같은 대화창을 볼 수 있습니다.



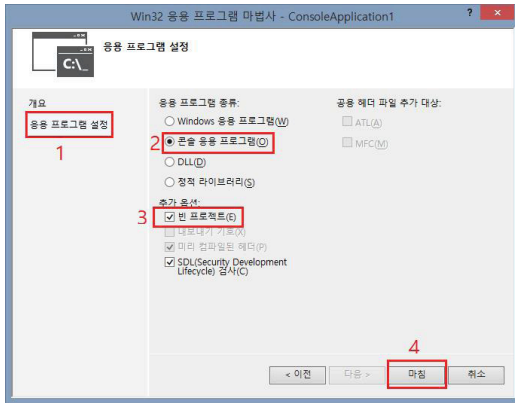
NOTE

솔루션 이름 옆에 작은 박스가 있는 것이 보이나요? 일반적으로 비주얼 스튜디오 솔루션은 많은 프로젝트를 가지고 있을 수 있습니다. 이 책에서는 하나의 프로젝트만을 사용하지만, 경험이 쌓이다 보면 하나의 솔루션에서 다양한 프로젝트를 통합해서 사용하는 것이 얼마나 편리한지 알게 될 겁니다.

3. 다음의 5가지 사항을 따라 해보세요.

1. 왼쪽 사이드 패널에서 Visual C++을 선택하세요.
2. 오른쪽 사이드 패널에서 Win32 콘솔 응용 프로그램을 선택하세요.
3. 이름을 정하세요(여기서는 MyFirstApp이라고 하겠습니다).
4. 코드를 저장할 폴더를 선택하세요.
5. 확인 버튼을 누르세요.

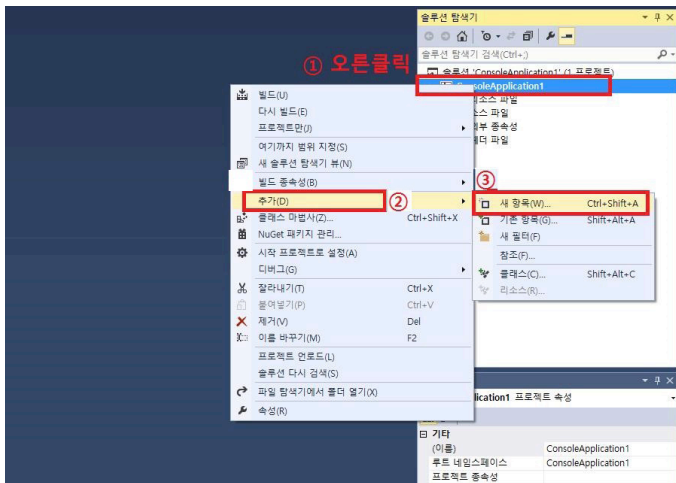
4. 다음 스크린샷과 같은 응용 프로그램 마법사 대화창이 뜹니다.



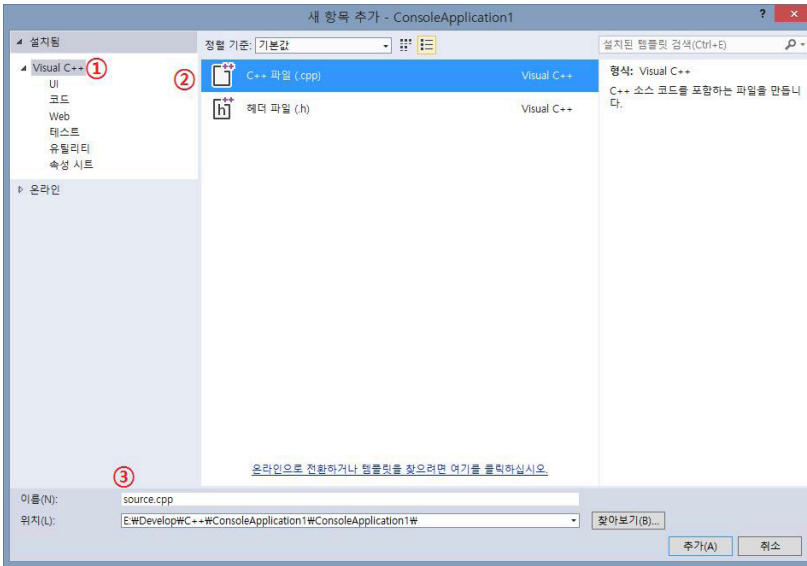
5. 이 대화창에서는 다음 4가지를 실행합니다.

1. 왼쪽 패널의 응용 프로그램 설정을 선택합니다.
2. 콘솔 응용 프로그램이 선택되었는지 확인하세요.
3. 빈 프로젝트를 선택하세요.
4. 마침 버튼을 클릭하세요.

드디어 설치가 끝나고 본격적으로 비주얼 스튜디오 2013에서 개발할 수 있게 되었습니다. 이곳이 바로 여러분의 작업장입니다. 먼저 코드를 작성할 파일이 필요합니다. 다음 스크린샷을 참고해서 프로젝트에 C++ 코드를 추가합니다.



다음 스크린샷을 참고해서 새로운 소스 코드 파일을 추가하세요.



이제 Source.cpp를 편집합니다. 첫 번째 C++ 프로그램 부분으로 넘어가 코드를 입력하세요.

1.1.2 맥에서 XCode 사용하기

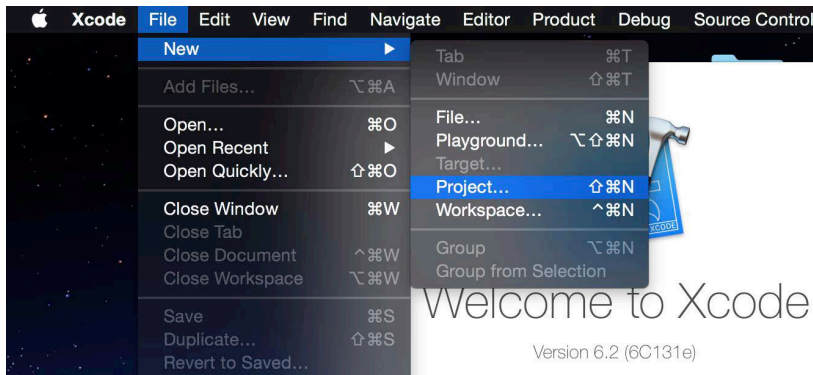
여기서는 맥에서 Xcode를 설치하는 방법에 대해서 설명할 것입니다.

XCode는 모든 맥 기기에서 사용할 수 있습니다. 다음에 보이는 것처럼 애플 앱스토어에서 XCode를 내려받을 수 있습니다(무료입니다).

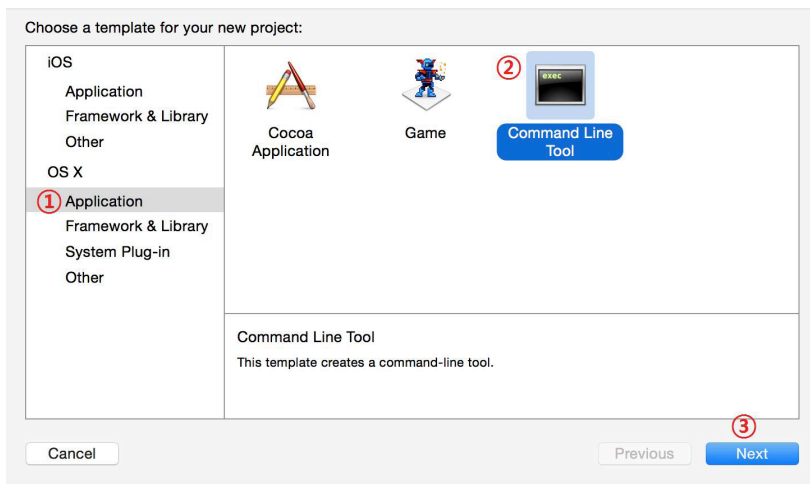


1. Xcode를 설치했다면 실행해주세요. 그다음 화면 상단에 있는 시스템 메뉴 바에서 File | New |

Project... 를 선택해주세요. 다음 스크린샷처럼 말이죠.



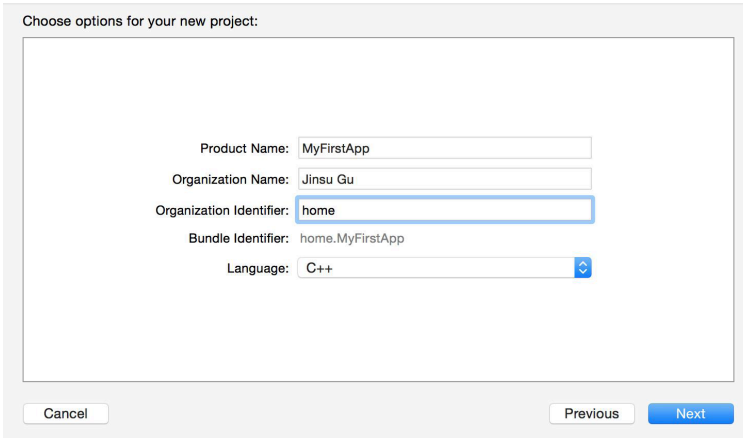
2. New Project 대화상자에서 왼쪽 패널의 OS X 아래에 있는 Application을 선택해주세요. 그리고 오른쪽 패널에서 Command Line Tool을 선택하고 Next를 눌러주세요.



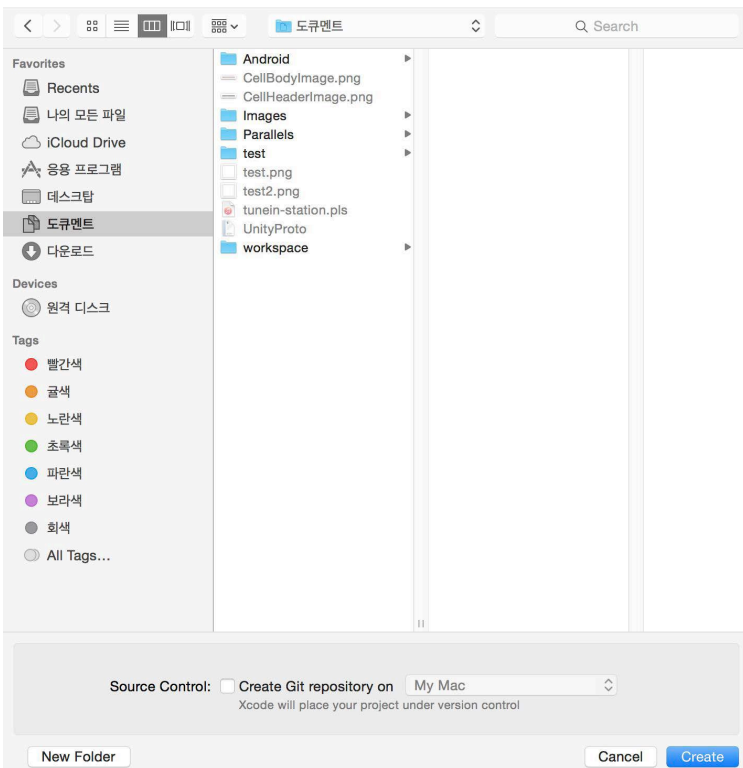
NOTE

Game 아이콘을 누르고 싶겠지만 그러지 마세요.

3. 다음 대화상자에서 프로젝트명을 지어주세요. 필드를 채우지 않으면 다음 단계로 나아갈 수 없습니다. Type은 꼭 C++로 설정하고 Next 버튼을 눌러주세요.



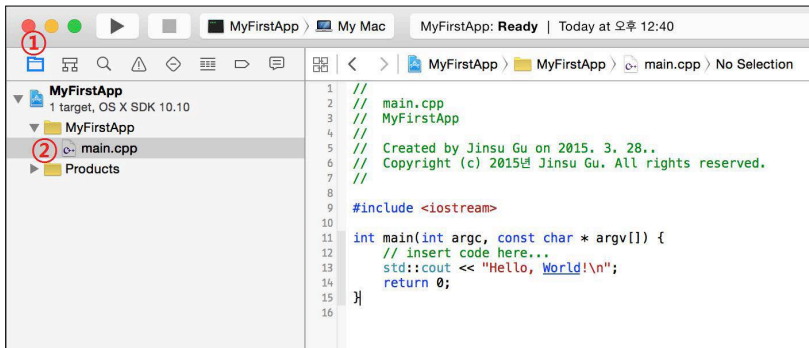
4. 다음 팝업창은 프로젝트를 저장할 경로를 묻습니다. 저장할 경로를 선택하세요. Xcode는 기본적으로 여러분이 만드는 모든 프로젝트마다 Git 저장소를 만듭니다. 하단의 Create git repository 체크는 해제합니다. 이 책에서는 Git을 다루지는 않습니다.



NOTE

Git은 버전 관리 시스템 (Version control system)입니다. Git은 여러분의 프로젝트 내 모든 코드의 스냅샷을 (여러분이 보관소에 커밋할 때마다) 보관합니다. 다른 유명한 소스 제어 관리 도구 (source control management, SCM)로는 머큐리얼, 퍼포스, 서브버전이 있습니다. 많은 사람이 같은 프로젝트에 협력할 때, SCM 도구는 다른 사람들이 바꾼 코드를 자동적으로 합쳐 저장소에서 여러분의 로컬 작업소로 바꾼 결과를 복제합니다.

자, 이제 완료되었습니다. 왼쪽 패널에서 main.cpp 파일을 선택하세요. 파일이 나타나지 않으면 왼쪽 상단의 폴더 아이콘이 눌러져 있는지 확인해보세요.



1.2 첫 번째 C++ 프로그램 만들기

이제 C++ 소스 코드를 작성해 볼 것입니다. 작성할 코드를 소스 코드라고 부르는 이유는 이 파일이 실제 실행되는 바이너리 코드를 빌드하는 데 사용되는 소스이기 때문이죠. 같은 C++ 소스는 맥, 윈도우, iOS와 같은 다른 플랫폼에서 빌드가 가능하고, 이론상으로 실행 코드는 각각의 플랫폼에서 같은 결과가 출력됩니다.

C와 C++이 소개되기 전에 프로그래머들은 목표로 하는 각각의 머신마다 따로 코드를 작성해야 했습니다. 어셈블리라고 불리는 언어를 사용해서 코드를 작성했죠. 그러나 지금은 C와 C++을 사용해서 프로그래머가 코드를 한 번만 작성하면, 다른 컴파일러로 보내는 것만으로도 다양한 머신에서 구동할 수 있습니다.

NOTE

실제로는 비주얼 스튜디오와 Xcode의 C++은 약간의 차이가 있습니다. 그러나 이 차이는 템플릿 같은 고급 C++ 개념을 사용할 때만 발생합니다. UE4가 유용한 이유 중 하나가 바로 윈도우와 맥 간의 차이를 없애주는 것입니다. UE4 개발진은 같은 코드가 윈도우나 맥에서 똑같이 돌아갈 수 있도록 많은 노력을 기울였습니다.

NOTE 실제 사용 팁

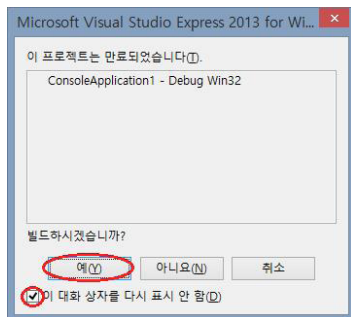
모든 기기에서 같은 결과가 나오도록 코드가 실행되는 것은 아주 중요합니다. 특히 네트워크 게임이나 게임에서 리플레이 영상이 필요할 때 같은 경우에 말이지. 이것은 표준을 사용하여 해결할 수 있습니다. 예를 들어 IEEE 부동소수점 표준은 모든 C++ 컴파일러에서 소수점 연산을 구현하는 데 사용됩니다. $200 * 3.14159$ 같은 식을 계산한 결과가 모든 기기에서 동일해야 한다는 것입니다.

비주얼 스튜디오나 Xcode에서 다음 코드를 작성하세요.

```
#include <iostream> // 입출력 라이브러리를 가져옵니다
using namespace std; // std::cout 대신에
                        // cout을 쓸 수 있도록 합니다

int main()
{
    cout << "Hello, world" << endl;
    cout << "이제부터 C++ 프로그래밍을 합니다." << endl;
    return 0; // 운영체제로 돌아갑니다
}
```

비주얼 스튜디오에서 Ctrl + F5를 누르거나 Xcode에서 command + R을 눌러 앞의 코드를 실행하세요. 비주얼 스튜디오에서 처음 Ctrl + F5를 눌렀다면 다음 대화 상자를 볼 수 있습니다.



<이 대화 상자를 다시 표시 안 함>을 체크하고 예 버튼을 선택해주세요. 이렇게 하면 나중에 발생할지도 모르는 문제를 미연에 방지할 수 있습니다.

아마도 일반적인 텍스트에서 해시 기호(#)와 중괄호({})를 많이 접하지는 못했을 겁니다(트위터를 사용하면 #은 접했을 수도 있겠군요). 하지만 C++ 코드를 사용하려면 이런 기호에 익숙해져야 합니다.

자, 처음 줄부터 이 프로그램을 해석해봅시다. 첫 번째 줄은 이렇습니다.

```
#include <iostream> // 입출력 라이브러리를 가져옵니다
```

이 줄에서는 깊고 넘어가야 할 두 가지 중요한 점이 있습니다.

1. 첫 번째는 #include입니다. C++에 <iostream>이라고 부르는 또 다른 C++ 소스 파일의 내용을 복사해 작성 중인 코드로 곧바로 가져오도록 요청하고 있습니다. <iostream>은 표준 C++ 라이브러리로, 화면에 텍스트를 출력하는 코드를 다룹니다.
2. 두 번째는 // 주석입니다. C++은 이중 슬래시(//) 뒤부터 줄 끝까지의 텍스트를 전부 무시합니다. 주석은 코드가 무엇을 하는지 설명할 때 사용합니다. 어쩌면 C의 주석 타입인 /* */을 본 적이 있을 겁니다. C나 C++에서 슬래시-별(/*)과 별-슬래시(*/)로 둘러싸인 텍스트는 컴파일러가 그 부분의 코드를 제거하도록 지시합니다.

다음 줄의 코드는 이렇습니다.

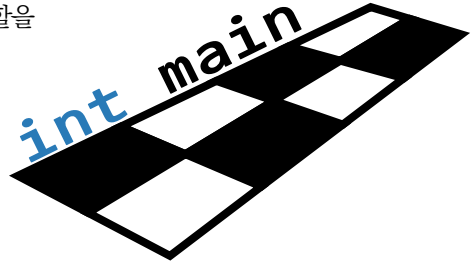
```
using namespace std; // std::cout 대신에  
// cout을 쓸 수 있도록 합니다
```

이 줄의 주석은 using 명령이 무엇을 하는지에 대해 설명하고 있습니다. C++ 코드 명령의 전체를 쓰는 대신에 짧게 써도 실행이 된다는 내용입니다. 예를 들자면 std::cout 대신에 cout만 작성해도 똑같은 명령이 되는 것이죠. using namespace std;라고 작성하는 걸 좋아하지 않을 수도 있습니다. cout를 사용하고 싶을 때 짧게 줄이는 대신에 std::cout와 같이 길게 늘어 쓸 수도 있죠. 하지만 여기서는 using namespace std;처럼 간결하게 작성하겠습니다.

그다음 줄입니다.

```
int main()
```

이 줄은 응용 프로그램의 시작점입니다. main은 경주의 출발선이라고 생각해도 됩니다. int main() 명령은 C++ 프로그램이 어디서 시작할지 가르쳐주는 역할을 합니다. 옆의 그림을 봐주세요.



int main() 프로그램 마커를 가지고 있지 않거나 main의 철자가 틀렸다면, 여러분의 프로그램은 제대로 동작하지 않을 겁니다. 어디서 시작할지 모르기 때문이죠.

다음 줄은 여러분이 자주 보지 못한 글자입니다.

```
{
```

이 { 글자는 콧수염을 돌려놓은 것이 아닙니다. 중괄호라고 부릅니다. 바로 본격적으로 프로그램이 시작되는 지점이지요.

다음 두 줄은 화면에 글자를 출력합니다.

```
cout << "Hello, world" << endl;  
cout << "이제부터 C++ 프로그래밍을 합니다." << endl;
```

cout 명령은 콘솔 출력을 수행합니다. 큰따옴표 사이에 있는 글자가 그대로 콘솔에 출력될 것입니다. 큰따옴표 사이에 큰따옴표를 제외한 어떤 내용이라도 작성할 수 있으며, 작성한 내용은 고스란히 콘솔에 출력됩니다.

NOTE

큰따옴표 사이에 큰따옴표를 넣으려면 큰따옴표를 넣고 싶은 부분 바로 앞에 백슬래시(\)를 넣어야 합니다. 다음 코드와 같이 말이죠.

```
cout << "존은 동굴에게 \" 안녕! \" 이라고 소리쳤다. 동굴이 울렸다."
```

\ 기호는 이스케이프 시퀀스의 한 종류입니다. 다른 이스케이프 시퀀스도 있습니다. 여러분이 가장 자주 쓰게 될 것은 \n이며 다음 줄로 이동하는 데 사용되죠.

프로그램의 마지막 줄은 `return` 명령입니다.

```
return 0;
```

이 코드는 C++ 프로그램이 종료된다는 것을 뜻합니다. 혹은 `return` 명령을 운영체제로 돌아가는 것이라고 생각해도 됩니다.

마지막으로 프로그램의 끝에서는 중괄호를 닫습니다. 콧수염을 반대로 돌려놓은 것처럼 생겼죠.

```
}
```

1.2.1 세미콜론

세미콜론(;)은 C++ 프로그램에서 중요한 역할을 합니다. 앞 예제에서 행 끝 대부분에 세미콜론이 있는 것을 눈치챘나요? 행 마지막에 세미콜론을 넣지 않으면 코드가 컴파일되지 않을 겁니다. 그렇게 된다면 여러분이 짠 코드는 무용지물이 되겠지요.

1.2.2 에러 처리하기

코드를 입력하다가 실수를 했다면 문법적 오류가 나타납니다. 문법적 오류가 발생하면 C++은 엄청나게 핀잔을 줄 것이고, 여러분의 프로그램은 컴파일조차 되지 않을 것입니다. 물론 실행도 되지 않겠죠. 앞의 코드에 몇 가지 에러를 넣어봅시다.

```
#include <iostreams>
```

```

Using namespace std;
int mains(
{
count << "Hello, world << ender;
count << "I am nowb a C++ programmer." << ender
}

```

조심하세요. 앞의 코드는 에러를 가지고 있습니다.

에러를 찾아서 고쳐보는 것은 좋은 연습이 됩니다. 연습 삼아 앞의 코드에 있는 모든 에러를 찾아서 고쳐 봅시다.

NOTE

C++을 처음 사용한다면 조금 어려울지도 모릅니다. 이 예제는 C++ 코드를 작성할 때 얼마나 주의해야 하는지에 대해 알려주는 것이 목적이니 조금 어렵다고 느껴져도 걱정 마세요.

컴파일 오류를 고치는 작업은 따분할 수 있습니다. 하지만 이 코드를 코드 편집기에 넣고 컴파일하려고 하면, 컴파일러가 즉시 에러를 보고할 것입니다. 에러를 하나씩 고친 뒤에 다시 컴파일을 시도하세요. 다음과 같이 새로운 에러가 나타나거나 에러가 없다면 정상적으로 프로그램이 동작할 겁니다.

```

1 #include <iostream>
2 using namespace std;
3
4 int maind(
5 {
6     count << "Hello, world << ender;
7     count << "I am nowb a C++ programmer." << ender;
8 }

```

여러분이 컴파일하려고 할 때 Xcode가 에러를 보여줍니다.

이런 예제 프로그램을 보여주는 이유는, C++에 익숙해지기 전까지는 다음 순서를 따라가는 게 좋기 때문입니다.

1. 언제든지 C++ 예제 코드로 시작하세요. 첫 번째 C++ 프로그램을 기반으로 새로운 C++ 프로그램을 작성할 수 있습니다.
2. 조금씩 코드를 수정하세요. 아직 처음이라면 새로운 코드 한 줄을 쓸 때마다 곧바로 컴파일하세요. 한두 시간 동안 계속 코드만 입력하다가 한 번에 컴파일하지 마세요.

3. 여러분이 처음 코드를 작성할 때 예상했던 대로 실행되는 코드를 작성하기까지 몇 달이 걸릴 수도 있습니다. 실망하지 마세요. 코드 배우기는 재미있습니다.

1.2.3 주의

컴파일러는 명확한 에러 외에 의심이 가는 내용도 표시합니다. 이것은 컴파일러가 주의라고 표시하는 또 다른 부류입니다. 주의를 여러분이 굳이 고칠 필요는 없지만 컴파일러가 고치기를 권장하는 문제점입니다. 주의를 완벽하지는 못한 코드를 표시하며, 주의를 수정하는 것 역시 좋은 연습이 됩니다.

모든 주위가 코드에서 문제를 일으키지는 않습니다. 몇몇 프로그래머는 주의 표시를 해제하여 그 문제에 대해 신경 쓰지 않는 편을 선호하기도 합니다(예를 들면 4018 주위는 부호의 불일치인데, 나중에 많이 보게 될 겁니다).

1.3 빌드와 컴파일은 무엇입니까?

컴파일이라고 불리는 컴퓨터 프로세스 용어를 들어본 적이 있을 겁니다. 컴파일은 C++ 프로그램은 CPU에서 돌아가는 코드로 변환하는 과정입니다. 소스 코드를 빌드한다는 것은 컴파일한다는 것과 같은 의미입니다. 소스 코드 .cpp 파일은 곧바로 컴퓨터에서 돌아가지 않습니다. 실행하려면 컴파일을 해야 합니다.

여기서 마이크로소프트 비주얼 스튜디오 익스프레스나 Xcode를 사용하는 중요한 이유가 나옵니다. 두 프로그램은 모두 컴파일러입니다. C++ 소스 코드는 메모장 같은 텍스트 편집 프로그램에서 작성할 수 있지만, 실행하려면 반드시 컴파일러가 필요합니다.

모든 운영체제에는 C++ 코드를 컴파일할 수 있는 하나 이상의 C++ 컴파일러가 있습니다. 윈도우에는 비주얼 스튜디오와 인텔 C++ 스튜디오 컴파일러가 있습니다. 맥에는 Xcode가 있고, 윈도우와 맥, 리눅스에서 사용할 수 있는 GNU 컴파일러 모음(GCC)이 있습니다.

여기서 작성한 C++ 소스 코드는 각각의 운영체제에서 다른 컴파일러를 이용하여 컴파일됩니다. 이론적으로는 모두 다 같은 결과가 나와야 합니다. 다른 플랫폼에서 동일한 코드를 컴파일하는 기술을 이식성이라고 합니다. 이식성이 좋은 언어가 활용하기 편리하겠죠.

1.3.1 스크립트

스크립트 언어라는 프로그램 언어가 있습니다. 여기에 해당하는 언어는 PHP, 파이썬, 액션스크립트 등이 있습니다. 스크립트 언어는 컴파일되지 않습니다. 자바스크립트, PHP, 액션스크립트에는 컴파일 단계가 없습니다. 대신에, 프로그램이 실행될 때 소스 자체에서 해석됩니다. 따라서 스크립트 언어의 가장 큰 장점은 플랫폼 독립적이라는 것입니다. 인터프리터라는 것이 플랫폼 독립적이게끔 아주 세심하게 디자인되기 때문입니다. 스크립트 언어에 대해 궁금하다면 관련된 도서나 인터넷 검색을 통해 궁금증을 해결하세요.

연습 - 아스키 아트

게임 프로그래머는 아스키 아트를 사랑합니다. 그 이유는 텍스트만으로 그림을 그릴 수 있기 때문이죠. 다음 예제는 아스키 아트를 사용한 미로 예제입니다.

```
cout << " * * * * * " << endl;
cout << " * ..... * . * " << endl;
cout << " * . * . * * * * * * . * . * " << endl;
cout << " * . * . * ..... * " << endl;
cout << " * . * . * * * * * * * * " << endl;
cout << " * * * . * * * ..... * " << endl;
```

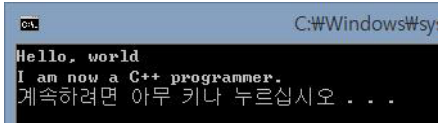
앞의 예제처럼 C++에서 텍스트만 사용하여 자신만의 미로를 작성해 보세요.

1.4 요약

이 장에서는 통합 개발 환경(IDE, 비주얼 스튜디오나 Xcode)에서 C++로 프로그램을 작성하는 법에 대해 배워보았습니다. 간단한 프로그램이지만, 첫 번째 프로그램을 컴파일하고, 첫 번째 성취감을 맛보았습니다. 다음 장에서는 더 복잡한 프로그램에 대해 알아보고 자신만의 게임 개발을 위해 언리얼 엔진을 사용해 보도록 합시다.

```
1 #include <iostream> // 입출력 라이브러리를 포함합니다
2 using namespace std; // std::cout 대신에
3 // cout을 쓸 수 있도록 합니다
4
5 int main()
6 {
7     cout << "Hello, world" << endl;
8     cout << "I am now a C++ programmer." << endl;
9     return 0;
10 }
11
```

앞의 스크린샷은 여러분의 첫 번째 C++ 프로그램이고, 다음은 그 결과입니다.



C:\ Windows#sy
Hello, world
I am now a C++ programmer.
계속하려면 아무 키나 누르십시오 . . .