UI
시스템
블랙북

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

UI 시스템 블랙북: UI 그래픽스 동작 원리와 핵심 개념

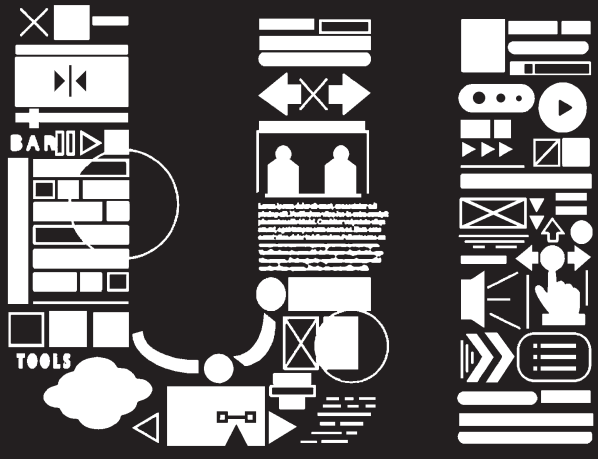
전자책 1쇄 발행 2024년 1월 25일 **지은이** 박춘언 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **편집** 정수진 **등록번호** 제 2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** https://blog.insightbook.co.kr **ISBN** 978-89-6626-436-0

Copyright ⓒ 2024 박춘언, (주)도서출판인사이트

이 책 내용의 일부 또는 전부를 재사용하려면 반드시 저작권자와 (주)도서출판인사이트 양측의 서면에 의한 동의를 얻어야 합니다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

프로그래밍 인사이트



UI 시스템 블랙북

UI 그래픽스 동작 원리와 핵심 개념

박춘언 지음

인사이트

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

무한한 지지를 아끼지 않은 아내와 동료들에게
우리의 미래를 이끌어 줄 독자분들께
오늘도 경험과 지식을 공유하는 전 세계 기여자분들께

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

차례

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

추천의 글

과학을 실생활에 직접적으로 도움이 되는 응용 과학과 그 기반이 되는 기초 과학으로 나눌 수 있는 것처럼, 컴퓨팅 분야도 시대의 필요와 요구에 따라 발전하는 응용 프로그램/서비스와 해당 응용 프로그램/서비스를 가능하게 하는 운영 체제 및 소프트웨어 플랫폼 분야로 나눠볼 수 있습니다. IT 기술은 아주 빠르게 발전할 뿐만 아니라 트렌드에 따라 그때그때 유행하는 컴퓨팅 분야가 있습니다. 최신의 IT 기술 트렌드를 좇아 앱과 서비스에 집중하다 보면 운영 체제나 플랫폼에 대해서는 간과하기 쉽습니다.

다양한 응용 프로그램이 동작할 수 있는 환경을 제공해 주는 소프트웨어 플랫폼은 UI, 멀티미디어, 네트워크, 보안 등 다양한 기술 도메인을 담당하는 부분 시스템들이 모여 구성됩니다. 이 중에서 UI 시스템은 그래픽 사용자 인터페이스(GUI)를 제공하는 응용 프로그램을 뒷받침하는 기술 부분으로, 그래픽스, 폰트, 렌더링과 같은 낮은 단계의 기능부터 UI 컴포넌트 및 테마 지원과 같은 상위 단계의 기능까지 아우르는, 소프트웨어 플랫폼에서 UI에 관련된 모든 기능을 제공하는 소프트웨어의 집합체입니다. UI 시스템을 구성하는 개별 요소에 대한 기술 서적은 많이 있지만, 이러한 각각의 요소 기술들을 조합해서 서로 유기적으로 맞물려 잘 동작하게 하는 전체 시스템에 대한 서적은 찾기 어렵습니다.

이 책은 저자가 십여 년간 소프트웨어 플랫폼과 오픈 소스 분야에서 활동한 경험을 바탕으로 집필한 것으로 UI 시스템에 대한, 특히 그래픽과 관련한 기술 내용 전반을 다루고 있습니다. UI 시스템이 제공해야 하는 기능과 해당 기능을 지원하기 위한 기술을 예제 코드와 함께 자세히 설명하고 있습니다. 소프트웨어 플랫폼 개발을 꿈꾸는 모든 분에게 도움이 될 것입니다. 또한, 응용 프로

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

그램 개발자도 소프트웨어 플랫폼이 동작하는 기본 원리를 이해하게 되어 향후 개발 과정에 많은 도움을 받을 수 있을 것입니다. 소프트웨어 플랫폼 개발을 목표로 하는 분들뿐만 아니라 그래픽 사용자 인터페이스와 관련된 기능이 소프트웨어 플랫폼에서 어떻게 처리되고 제공되는지 그 전반적인 내용을 알고 싶은 모든 분에게 이 책을 추천합니다.

이종민, 타이젠 플랫폼 수석 아키텍트(삼성전자)

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

서문

저자의 말

삼성전자에서 근무하던 당시, 타이젠 플랫폼 팀에서 UI 시스템 기술 개발을 담당했었다. 당시에 필자는 팀에 새로 합류한 친구들이 필수 역량을 빠르게 습득할 수 있는 가이드라인, 특히 웹, 안드로이드, iOS 프로그래밍과 같이 특정 플랫폼에 국한되지 않는 범용한 기술에 대한 학습 자료가 필요하다고 생각했다. 그 시절에는 (지금도 그렇지만) UI 시스템과 관련된 전공서는 찾아볼 수 없었고, 목마른 자가 우물을 판다는 심정으로 집필을 시작하게 되었다.

UI 시스템과 관련된 주제는 산업 표준이 아니라 플랫폼, 대상 제품의 특성, 그리고 사용자 경험과 디자인에 크게 의존하기 때문에 공통적으로 널리 쓰이는 기술을 다루기 어려운 점이 있다. 그럼에도 불구하고 UI 시스템에 대한 독자들의 흥미를 고려하여 공통적으로 사용하는 기술을 다룰 주제를 찾고자 노력했다. 그 결과 UI 시스템의 핵심 기술 개념에 중점을 두고, 이 분야에서 틀을 세우고 내용을 정리하면 의미가 있을 것이라 생각했다. 또한, 상대적으로 개발자들의 관심이 높은 그래픽스라는 주제에 비중을 두면서도 IT 분야에서 종사하는 개발자뿐만 아니라 학생부터 다양한 분야의 전문 소프트웨어 엔지니어들까지 부담 없이 책을 읽을 수 있도록 집필하려고 노력했다.

결과적으로 이 책은 완성된 기능을 보여주기보다는 기본 원리를 이해하고 습득하는 데 중점을 두게 되었다. 기술적으로 지나치게 어려운 책은 독자들에게 부담이 될 수 있으며 전문가 수준의 해결책은 다양한 상황에 적용하기 어렵다고 생각한다. 그 대신, 기본 원리와 이를 뒷받침할 수 있는 지식을 제공한다면 독자가 자신만의 경험과 지식을 더하여 문제를 해결할 수 있을 것이다. 원래 계획과 달리 이 책에서 제외되거나 정리가 미흡한 주제가 있어 아쉽지만 이는 추후에 보완할 수 있기를 기대해 본다.

정리하면, 이 책은 UI 시스템 관점에서 필요한 컴퓨터 공학 지식, 그래픽스,

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

프로그래밍 역량을 학습하는 데 초점을 맞추고 있다. UI 도메인 기술을 중점으로 자료 구조 및 알고리즘, 객체 지향 설계, 그래픽스, 프로그래밍, 설계 기법 등 소프트웨어 개발의 필수 지식을 다루며, 특정 프로그래밍 언어를 선택하지 않고 가상의 객체 지향 언어를 활용하여 문법에 너무 지나치게 얽매이지 않고 핵심 내용에 집중할 수 있도록 집필했다. 이 책에서 다루는 프로그래밍 언어는 서문의 코드 규칙에서 가볍게 학습할 수 있을 것이다.

이 책이 완성될 때까지 도움을 주신 모든 분들께 감사의 말씀을 전한다. 책의 기술적 완성도를 향상시키기 위해 기여해주신 모든 리뷰어들과 함께 책을 출판할 수 있도록 기회를 제공해 주신 인사이트 출판사에 깊은 감사를 표한다. 마지막으로, 본 서적의 완성도를 높이는 데 끝까지 힘써주신 정수진 편집자님께도 감사 인사를 남긴다.

책 소개

이 책은 UI 시스템 관점에서 필요한 컴퓨터 공학 지식, 그래픽스, 그리고 프로그래밍 능력을 학습하는 데 초점을 맞춘다. UI 시스템은 시스템 도메인 및 디바이스 환경에 따라 정의와 기능 범주가 다를 수 있지만, UI 시스템의 핵심 기능은 크게 UI 툴킷, 엔진, 그리고 렌더링 세 부분으로 나눌 수 있다. 이 책은 이 세 가지 기능을 중심으로 다루며, 특히 독자들의 관심이 큰 렌더링(그래픽스) 기술에 많은 부분을 할애한다. 또한, 사용자 관점뿐만 아니라 엔진 내부 기술 구현을 직접 조망함으로써 어려운 프로그래밍 기술과 그 적용 사례를 학습하고 원리를 이해할 수 있도록 구성했다.

1장에서는 UI 툴킷 개발에 중점을 둔다. UI의 핵심 구성 요소를 정리하고, 이러한 요소를 UI 앱 관점에서 구현하는 사례를 통해 현대 UI 앱의 작동 원리를 파악한다. 더불어 스케일러블 UI의 다양한 예와 테마 커스터마이징과 같은 UI 툴킷의 주요 기능을 탐구한다. 이 과정을 통해 UI 툴킷이 필요로 하는 주요 기능을 이해하고 시스템 설계 관점에서 UI 툴킷과 시스템을 학습할 수 있다.

2장에서는 UI 렌더링에 사용되는 캔버스 기능을 개발한다. 캔버스는 렌더링 기능을 UI 시스템에 맞게 추상화하는 역할을 수행하며, 렌더링 엔진과 시스템

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

간의 출력 동작을 조율하고 드로잉 인터페이스를 통해 UI 앱에서 요청한 비주얼 요소를 화면에 렌더링한다. 이 과정을 통해 캔버스를 탐구함으로써 렌더링 엔진 설계 단계 및 다양한 렌더링 기법과 메모리 관리 기술을 학습한다.

3장에서는 렌더링 엔진을 구성하는 중요한 기술 중 하나인 벡터 그래픽 기술을 탐구한다. 벡터 그래픽은 다양한 해상도를 지원하면서도 품질을 유지하고 모션 그래픽을 구현하는 핵심 기술이다. 도형, 선, 색상 칠하기 기능 등을 구현하면서 벡터 그래픽 엔진을 직접 개발하고, 래스터 기술을 통해 더 효과적으로 출력하는 방법을 학습한다.

4장에서는 이미지 데이터 처리 기술을 학습한다. 이미지는 벡터 그래픽스와 함께 UI를 더욱 풍부하게 만드는 중요한 역할을 한다. 주요 이미지 포맷 중 하나인 PNG 파일을 예로 들어 이미지 파일의 특성부터 화면에 출력하는 과정을 살펴보고 크기 변환, 회전, 블렌딩과 같은 대표적인 이미지 후처리 기술도 함께 학습한다.

5장에서는 폰트와 텍스트 기술을 학습한다. 텍스트는 UI 앱의 기능 요소를 직접 전달하는 중요한 구성 요소이며, 폰트는 텍스트의 특성을 형성하는 필수 구성 요소이다. OTF와 TTF와 같은 산업 표준 포맷을 기반으로 폰트의 특성을 이용하여 화면에 출력할 글리프를 생성하고 텍스트를 구성하는 과정을 따라가면서 폰트와 텍스트의 작동 원리를 이해한다.

마지막 장인 6장에서는 사용자 상호 작용에 대한 UI 효과를 구현하는 방법과 관련된 기술을 다룬다. 시스템 통합 관점에서 애니메이션을 구동하는 다양한 기술과 기법을 탐구하고, 디바이스에서 발생한 입력 신호를 어떻게 활용하여 애니메이션과 사용자 입력을 처리하는지를 구현 예시를 통해 살펴본다. 또한 애니메이션을 더 효과적으로 다루기 위한 병렬화 기법도 함께 학습한다.

용어 정리

다음은 이 책에서 언급하는 컴퓨터 프로그램 용어를 정리한 것이다.

- 시스템: 소프트웨어의 조합으로 구성된 복잡한 기술적 엔터프라이즈를 말한다. 시스템은 특정 목적을 위해 설계되며, 데이터 처리 및 문제 해결을 위

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

한 다양한 컴포넌트와 하위 시스템으로 구성할 수 있다. UI 시스템은 UI 앱을 구동하기 위한 필요 기능들의 집합이라 할 수 있다.

- 엔진: 엔진은 특정 작업을 자동화하거나 실행하기 위한 소프트웨어 핵심 구성 요소이다. 예를 들어 그래픽 엔진은 그래픽 처리를 관리하고, UI 툴킷 엔진은 UI 요소를 생성하고 이들 기능 동작을 수행하는 데 사용된다.
- 프레임워크: 소프트웨어 개발을 위한 구조나 기반을 제공하는 추상화된 템플릿 또는 라이브러리 집합을 가리킨다. 프레임워크는 개발자가 특정 작업을 보다 쉽게 수행하도록 도와주며, 코드의 재사용성과 일관성을 증가시킨다. 예를 들어, 반복적으로 작성해야 할 코드를 하나의 기능 호출로 대신할 수 있게 하거나, 여러 모듈을 조합해 완성할 수 있는 복잡한 기능 구현을 직관적이면서도 간단한 인터페이스 호출로 대신할 수 있게 도와준다.
- 라이브러리: 다양한 프로그래밍 언어에서 사용 가능한 코드 모음을 말한다. 이는 기능 재사용을 촉진하며 개발 시간을 단축시킨다. 일반적으로 라이브러리에는 특정 작업을 수행하기 위한 함수, 클래스 및 기능이 포함되며, 개발자는 라이브러리를 활용하여 소프트웨어를 개발할 수 있다. 라이브러리 사용 방식은 동적 라이브러리와 정적 라이브러리로 구분한다. 동적 라이브러리는 프로그램과 라이브러리가 런타임에 연동되는 방식을 사용하며, 정적 라이브러리는 프로그램 패키징(컴파일) 시점에 프로그램과 라이브러리가 연동되는 방식을 채택한다.
- 모듈: 소프트웨어의 작은 독립 단위로, 특정 작업 또는 기능을 수행하는 코드 조각을 나타낸다. 코드의 재사용을 촉진하고 유지·보수를 단순화하는 데 도움을 준다. 모듈은 주로 라이브러리(또는 파일) 단위로 구성하여 사용한다. 예를 들어 하나의 엔진은 여러 작은 모듈로 구성될 수 있으며, 모듈을 교체함으로써 엔진의 기능을 달리 할 수 있다.
- 인터페이스: 두 개체 또는 시스템 간의 상호 작용을 가능하게 하는 규격 또는 계약을 말한다. 이는 코드의 분리와 상호 운용성을 제공하며, 다른 시스템이나 모듈과 통신하는 데 사용할 수 있다. 인터페이스는 클래스나 모듈 외부와의 상호 작용을 정의하며 어떤 메서드 또는 함수를 호출하고 어떤 데

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

이터를 주고받을 수 있는지 규정한다. UI 앱은 API(인터페이스)를 통해 UI 시스템의 기능을 호출할 수 있다.

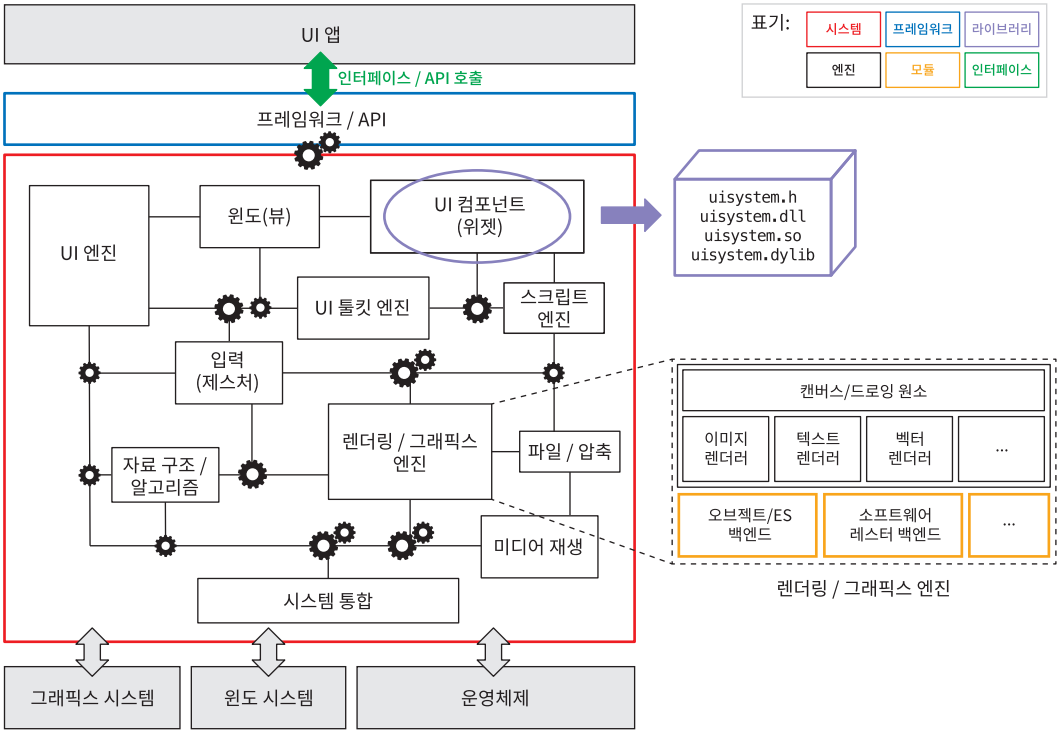


그림 0.1 컴퓨터 프로그램 용어 정리

코드 규칙

다음은 이 책에서 활용한 코드 예제의 프로그래밍 문법과 작성 규칙이다.

1. 코드는 페이지 공간 제약을 최소화하기 위해 두 자의 들여쓰기 규칙을 적용한다. 또한 각 구절은 브래킷({,})으로 구분하는 대신 들여쓰기로 이를 구분한다. 기능은 도메인을 구분하기 위해 클래스(class), 메서드(method)로 구현하고, 그렇지 않은 경우에는 함수(function)로도 구현할 수 있다. 최종 선언은 콜론(:)으로 한다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

코드 0.1 클래스, 메서드, 조건문

```
ClassA:       // 클래스 선언
  methodA():  // 메서드 선언
    // 이하 methodA() 구현부
    ...
    // repeat가 참인 동안 while 구문 실행
    while repeat
      if conditionA or conditionB
        // 이하 코드는 conditionA 또는 condition B가 참인 구문에 해당
        ...
        break  // while 구문 종료
      else
        // 이하 코드는 conditionA와 conditionB가 거짓인 구문에 해당
    // while() 종료
  // methodA() 종료
// ClassA 종료

// 함수 선언
funcA():
  ...
// funcA 종료
```

2. 예제에서 클래스 상속(inheritance)을 구현한다. 확장 클래스는 기능을 확장(extends)하거나 인터페이스를 구현(implements)할 수 있다. 이때 오버라이드(override) 키워드를 활용하여 메서드를 구현 또는 재정의한다. 부모 클래스의 기능을 호출할 때는 super 키워드를 활용한다.

코드 0.2 인터페이스 구현과 클래스 확장

```
// 인터페이스 선언. update() 동작 정의
interface UIObject:
  update()

// UIObject 인터페이스를 구현한 UIBaseBody
UIBaseBody implements UIObject:
  // update() 동작 구현
  override update():
    ...

// UIBaseBody를 확장하여 UIShape 정의
UIShape extends UIBaseBody
  override update():
```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

```
    // UIBaseBody의 update() 호출
    super.update()
    ...
```

3. 내장 데이터 타입으로 var, bool, string 세 가지를 제공한다. var 형은 정수, 실수와 문자를 수용하고 bool 형은 참(true), 거짓(false) 값을 표현한다. string은 문자열 값을 표현한다. 타입 선언은 생략할 수 있고 선언은 행을 종료함으로써 완성한다.

코드 0.3 변수 선언 및 초기화

```
var x, y          // 정수 내지 실수 데이터 선언
a = 0, b = -20.2  // 선언과 동시에 값을 초기화함으로써 타입 생략

// 이하 불(boolean) 변수 선언
bool repeat = true
condition = false

name = "ClassA" // string 타입
```

4. 이 책에서는 배열과 리스트를 구분하지 않는다. 프로그래밍보다는 내용의 본질에 집중하기 위해 모든 복수 데이터를 리스트 하나로 취급한다. 단, 변수에 브래킷([])을 선언함으로써 본 데이터가 리스트임을 명시할 수 있다.

코드 0.4 리스트와 반복문 활용

```
// 크기가 10인 var 타입의 값을 원소로 갖는 리스트 선언
data[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

sum = 0

// data를 순회하며 각 원소를 val로 전달
// for()가 끝난 시점에 sum 값은 55
for val : data
  sum += val

// 크기가 10x10인 var 타입의 값을 원소로 갖는 이차원 리스트
var data2[10, 10]

```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

```
// for(i = 0; i < data.size; ++i) 동일
for i : data2.size
  for j : data2[].size
    data2[i][j] = data[j]

// for(i = 0; i < 10; i+=2) 동일
for i : [0 ~ 10], i+=2
  data[i] /= 2

// 크기가 명시되지 않은 리스트 선언
var list[]

list.add(10)  // list[0] = 10
list.add(20)  // list[1] = 20
list.add(30)  // list[2] = 30

sum = 0

// list를 순회하며 각 원소를 val로 전달
// for가 끝난 시점에 sum 값은 60
for val : list
  sum += val
```

5. 클래스는 생성자(constructor)와 소멸자(destructor)를 지원하며 이들은 객체의 생성과 소멸 시점에 자동 호출된다.

코드 0.5 생성자와 소멸자

```
ClassA:
  x = 0, y = 0

  constructor(x, y):
    .x = x, .y = y

  destructor():
    ...

// example() 종료 시점에 ClassA.destructor() 호출
example():
  a = ClassA(10, 20)  // ClassA.constructor(x, y) 호출
```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

6. 메서드의 파라미터(parameter)와 반환(return) 타입 선언은 생략한다. 필요시 주석(@p)으로 설명을 보충하고 내용 설명에서 중요하지 않은 파라미터는 ...으로 생략한다. 인스턴스 자신의 멤버 변수 내지 메서드를 가리키기 위해서는 점(.) 또는 self 키워드를 활용한다.

코드 0.6 메서드 인자, 반환 값 활용

```
ClassA:
  var x, y

  // @p x: var, 이동할 좌표 X 값
  // @p y: var, 이동할 좌표 Y 값
  move(x, y):
    newPos = Vector2(x, y)

    // 메서드 인자 x, y를 멤버 변수 x, y 값에 기록
    .x = x
    self.y = y

    // 메서드 종료 시 true 값 반환
    return true
```

7. 메서드 반환 시 단일 또는 복수의 값을 반환할 수 있으며 필요하면 조건을 덧붙일 수 있다.

코드 0.7 반환 조건 및 복수 값 반환 방식

```
calle(x):
  // x가 0 보다 작으며, 0, 0을 반환
  return 0, 0 if x < 0
  // x가 0 보다 크거나 같으면 3, 2를 반환
  return x + 2, x * 2

caller():
  // a와 b의 값은 각각 5, 6
  a, b = calle(3)
```

8. 멤버 변수를 갱신하는 방법은 여러 가지이지만 이 책에서는 독자들에게 효과적으로 의미를 전달하는 데에만 집중한다. 따라서 이 책에서는 데이터 가

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

시성(visibility)과 데이터에 접근하기 위한 인터페이스는 따로 언급하지 않는다. 다만 코드 0.8은 멤버 변수를 갱신하는 과정을 확인하기 위한 예시다.

코드 0.8 멤버 변수 값 갱신

```
Vector2:
  x = 0, y = 0

  x(x):
    .x = x

  Vector2(x, y):
    .x = x
    .y = y

ClassA:
  Vector2 pos

  constructor(x, y):
    .pos(x, y)        // ClassA.pos(x, y) 호출

  pos(x, y):
    .pos.x = x        // Vector2.x(x) 호출
    .pos.y = y        // Vector2.y(y) 호출

example():
  a = ClassA(10, 20)  // ClassA.constructor(x, y) 호출
  a.pos(20, 30)       // ClassA.pos(x, y) 호출
  a.pos = {20, 30}    // Vector2(x, y) 호출
  a.pos.x = 20        // Vector2.x(x) 호출
  a.pos.y = 30        // Vector2.y에 직접 접근 기록
```

9. UI를 쉽고 빠르게 작성하기 위해 선언형 UI와 친숙한 프로그래밍 스타일을 적용한다. 가령, 객체 생성 후 객체의 속성(property)을 설정하거나 이를 기반으로 다른 UI 객체를 합성(composing)할 경우 콜론 후 들여쓰기와 함께 작성할 수 있다. 합성 시에는 `.contain()`을 이용한다.

코드 0.9 메서드 인자, 반환 값 활용

```
// 버튼 객체 생성
myBtn = UIButton():
  .text = "My Button"              // 버튼 텍스트
```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

```
  .geometry = {110, 80, 200, 130}  // 버튼 위치 및 크기 지정

// 수직 선형 레이아웃 생성 후 두 개의 버튼 추가
myLayout = UIVerticalLinearLayout():
  .contain():
    UIButton():
      .text = "Button 1"
    UIButton():
      .text = "Button 2"
```

10. 예제에서는 이벤트 주도 방식의 프로그래밍을 유도하며 많은 경우 콜백 함수를 선언하고 구현한다. 이때 콜백 함수는 람다(lambda)를 이용하여 작성한다. 대부분의 람다 함수는 자신을 호출한 객체(obj)를 인자로 전달한다.

코드 0.10 람다 함수 이용한 이벤트 구현

```
myBtn = UIButton():                // 버튼 생성
  .text = "My Button"              // 버튼 텍스트
  .geometry = {110, 80, 200, 130}  // 버튼 위치 및 크기

/*
 * 버튼 클릭 이벤트를 람다 함수로 구현
 * Clicked 이벤트 발생 시 func() 수행
 * obj 인자로 myBtn 인스턴스 전달
*/
func(UIButton obj):
  obj.text = "Button Pressed!" // 버튼 레이블 변경

myBtn.EventClicked += func     // 클릭 이벤트 수행 함수 func() 등록
```

독자들이 코드를 지면에서 분석하기 어려울 수 있다는 점을 고려하여, 이 책의 모든 예제 코드를 필자의 깃허브(*https://github.com/hermet/ui-system-blackbook*)에서 볼 수 있다. 필요한 경우 깃허브에서 코드를 다운로드하여 확인하길 바란다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

1장

UI System Blackbook

UI 툴킷과 앱

UI 앱에서 사용자 경험은 매우 중요하다. 데스크톱과 모바일 환경의 경계가 사라진 지금 앱 스토어에 등록된 앱은 셀 수 없을 정도로 다양하고 기능이 유사하거나 목적이 같은 소프트웨어도 무수하다. 이 때문에 세련된 디자인은 물론 사용하기 편한 앱을 사용자가 더 주목하는 시대이다. 다시 말해, 같은 기능을 제공하는 앱이라면 사용자 경험이 더욱 뛰어난 소프트웨어가 사용자에게 매력을 호소할 수 있다.

이러한 이유로 앱 개발자는 차별화된 UI 앱을 구현하기 위해 디자인 확장이 유연하면서도 개발이 쉽고 성능이 뛰어난 UI 시스템을 선호한다. UI 시스템은 정교하게 설계된 프로그래밍 인터페이스는 물론, 다양한 환경에서도 동일한 동작을 보장할 수 있는 호환성을 갖춰야 한다. 뿐만 아니라 시스템 내부 동작과 UI 앱의 기능 로직을 연결하기 위한 프레임워크, UI 컴포넌트, 그래픽스 엔진과 같은 핵심 기능 로직을 라이브러리로 제공해야 한다. 이러한 제반 기능 요소와 함께 툴킷(Toolkit)을 완성된 형태로 제공함으로써 앱 개발자가 쉽고 빠르게 앱 UI를 완성할 수 있도록 도와주어야 한다. 여기서 툴킷은 UI를 구동하는 프로그램(라이브러리)뿐만 아니라 UI를 작성하기 위한 도구(에디터)를 포함한다. 앱 개발자는 UI 에디터를 통해 화면에 UI 요소를 빠르게 배치하고, 스크립트와 API를 통해 사용자와 앱 사이의 상호 작용 로직을 구현한다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

이번 장에서는 UI 툴킷의 주요 기능을 학습하면서 UI 컴포넌트와 앱 개발에 필요한 기반 지식을 배우게 된다. 여러분이 UI 개발 경험이 충분치 않다면 이번 장은 여러분에게 멋진 도입부가 될 것이다.

☑ 학습 목표

이번 장을 통해 다음 사항을 학습한다.

- UI 구성 요소와 이를 활용하는 기본 메커니즘
- 스케일러블 UI 개념과 여러 응용 사례
- 메인 루프 개념과 동작 원리
- 수명 주기를 기반으로 프레임워크를 구축하는 과정
- 뷰와 이를 관리하는 메커니즘
- 테마 지원 기술 구현 방안

1.1 UI 기능 이해

1.1.1 그래픽 요소

본격적인 시작에 앞서 UI의 그래픽 요소를 먼저 살펴볼 것이다. UI의 그래픽 요소는 앱 UI를 구성하기 위한 최소 필수 기능 조건이다. 다음과 같은 앱 UI를 구성하기 위해서 어떤 리소스가 필요할까?

그림 1.1은 우리가 잘 알고 있는 크롬 브라우저(Chrome browser)의 구글 페이지 화면이다. 크롬 역시 하나의 UI 앱으로 간주할 수 있다. 얼핏 보면 UI가 복잡해 보이지만 UI를 조목조목 따져 보면 결국에는 이미지와 텍스트 두 가지 그래픽 요소로 구성되어 있음을 확인할 수 있다.

결국 앱의 화면을 구성하는 기본 그래픽 요소(primitive graphical element)는 화면을 화려하게 장식해 주는 이미지와 문맥 정보를 전달하는 텍스트 두 개로 축약할 수 있다. 이 두 기능만 있으면 원시적인 형태일지라도 어떠한 종류의 앱 화면도 정확히 구현할 수 있다. 코드 1.1에서 그림 1.1의 검색 상자(Search Box)를 어떻게 구현할 수 있는지 보여준다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

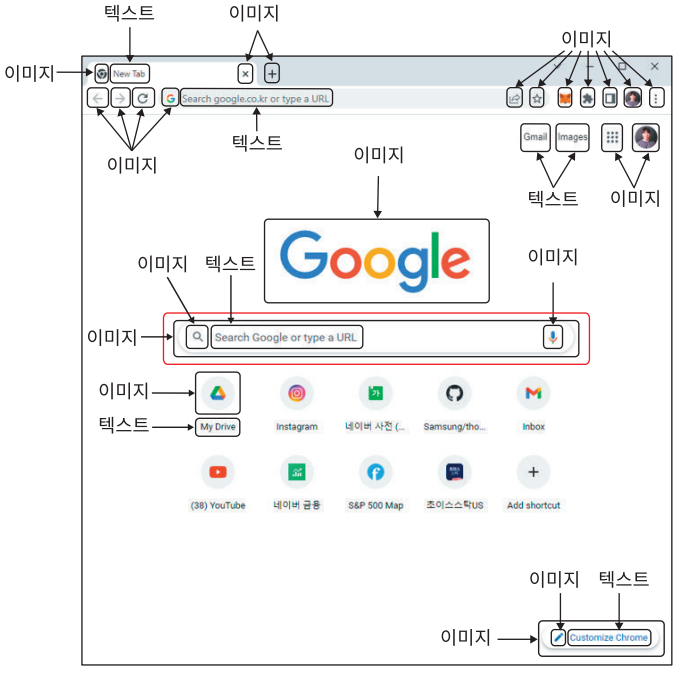


그림 1.1 UI 앱 화면을 구성하는 그래픽 요소(Google Chrome)

코드 1.1 이미지와 텍스트를 이용한 앱 UI 구현

```
// 검색 상자
searchBox = UIImage():                  // 이미지 생성
  .path = "./res/SearchBox.png"         // 이미지 리소스
  .geometry = {140, 400, 620, 55}       // 이미지 위치 및 크기

// 검색 상자 검색 아이콘
searchIcon = UIImage():                 // 이미지 생성
  .path = "./res/SearchIcon.png"        // 이미지 리소스
  .geometry = {160, 410, 35, 35}        // 이미지 위치 및 크기

// 검색 상자 가이드 텍스트
guideText = UIText():                           // 텍스트 생성
  .text = "Search Google or type a URL"         // 텍스트 설정
  .color = "lightgray"                          // 텍스트 색상
  .geometry = {200, 410, 235, 35}               // 텍스트 위치 및 크기

// 검색 상자 음성 아이콘
voiceIcon = UIImage():                  // 이미지 생성
  .path = "./res/VoiceIcon.png"         // 이미지 리소스
  .geometry = {700, 410, 35, 35}        // 이미지 위치 및 크기
```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

코드 1.1과 같이 이미지와 텍스트를 이용하면 그림 1.1의 다른 UI도 동일한 방식으로 구현할 수 있다. 이때 도형 출력 기능을 이용할 수 있다면 별도의 이미지 데이터를 사용하지 않고도 UI 화면을 구현할 수 있다. 코드 1.2는 코드 1.1의 검색 상자 이미지를 도형을 이용한 방식으로 구현한다.

코드 1.2 도형을 이용한 UI 화면 구성

```
// 검색 상자
searchBox = UIRoundRect():          // 모서리를 둥글린 사각형 생성
  .geometry = {140, 400, 620, 55}  // 위치 및 크기
  .cornerRadius = 27.5              // 모서리 둥근 정도
  .fill = UIColor.White             // 색상 (흰색)
  .stroke = UIStroke():             // 검색 상자 테두리
    .width = 1                      // 테두리 두께
    .color = UIColor.Gray           // 테두리 색상 (회색)

// 이하 코드 1.1과 동일
...
```

마찬가지로 텍스트도 미리 준비된 이미지로 대체할 수 있지 않을까 생각할 수 있다. 불가능한 것은 아니지만 사용자 설정 또는 시스템 환경에 따라 언어와 폰트 등이 바뀔 수 있으므로 UI 앱이 직접 텍스트를 이미지로 대체하여 출력하는 방법[1]은 제약이 많고 비효율적이다.

정리하면, 현대의 UI 앱은 도형, 이미지 그리고 텍스트에 해당하는 그래픽 원소를 활용함으로써 원하는 화면 비주얼을 완성할 수 있다. 따라서 UI 앱을 구동하는 시스템은 최소 이 세 가지 그래픽 출력 기능을 제공함으로써 UI 앱의 필요 조건을 충족할 수 있다.

1.1.2 UI 컴포넌트

안드로이드, 맥OS, MS 윈도우 등 주요 플랫폼에서 동작하는 UI 앱을 개발해 본 적이 있다면, 앱 개발자가 앞선 예시와 같은 원시적 방법으로 UI를 구현하는 환경은 상상하기 어렵다. UI 시스템은 UI 컴포넌트(버튼, 스위치, 달력 등)[2]를

1 사실, 텍스트도 UI 렌더링을 거쳐 이미지(글리프)로 생성되어 출력된다.
2 UI 컨트롤이나 위젯도 같은 의미로 해석할 수 있다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

통해 화면을 구성하는 공통된 기능과 특성을 제공하므로 앱 개발자는 쉽고 빠르게 UI를 구성할 수 있다. 프로그래밍 관점에서 보면 UI 컴포넌트는 UI 객체로 통용되기도 한다.

코드 1.3 UI 컴포넌트를 이용한 앱 UI 구현

```
01 // 검색 상자 UI 컴포넌트(코드 1.1 대체 버전)
02 searchBox = UISearchBox():                  // 검색 상자 생성
03   .text = "Search Google or type URL"      // 가이드 텍스트 설정
04   .searchIcon = "./res/SearchIcon.png"     // 검색 아이콘 설정
05   .voiceIcon = "./res/VoiceIcon.png"       // 음성 아이콘 설정
06   .geometry = {140, 400, 620, 55}          // 검색 상자 위치 및 크기
```

기본적으로 UI 컴포넌트는 크게 비-컨테이너와 컨테이너 두 종류로 구분할 수 있다. 비-컨테이너는 외양을 기반으로 사용자와 상호 작용을 수행하는 UI 컴포넌트다. 대표적으로 앞서 등장한 검색 상자(SearchBox)가 이에 해당하며 버튼(Button), 토글(Toggle), 체크박스(Checkbox) 등이 있다. 반면 컨테이너는 비-

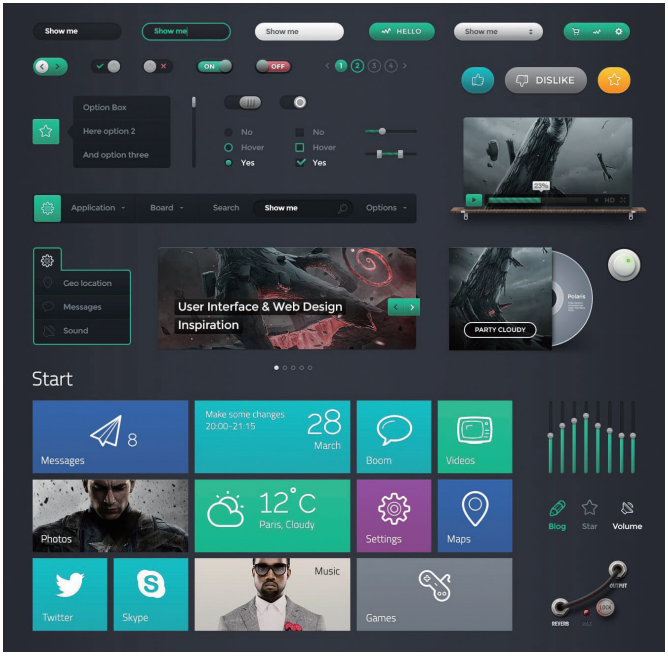


그림 1.2 다양한 종류의 UI 컴포넌트(Polaris UI)

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

컨테이너 컴포넌트를 화면에 배치하기 위한 레이아웃 기능을 제공한다. 대체로 컨테이너는 외양이 없거나 이를 부수적으로 제공하며, 스케일러블 UI(1.2절)를 보장하기 위한 원칙과 기능 동작을 구현한다. 여기서 '컨테이너'는 UI 시스템에서 쓰이는 표준 용어는 아니지만 통용되는 용어로 어떤 콘텐츠를 담는 기능을 말한다. 컨테이너 컴포넌트와 관련된 내용은 1.1.4 "레이아웃"에서 살펴본다.

다시 코드 1.3을 살펴보면, UI 컴포넌트를 이용하여 코드 1.1과 동일한 기능을 재구현하고 있음을 확인할 수 있다. 결과적으로 앱의 UI 구현이 더 간단해졌다. 또 UI 컴포넌트는 사용자와의 상호 작용을 위한 기능도 제공한다. 검색 상자가 사용자가 입력한 텍스트를 실시간 반영하여 출력하는 상황을 생각해 보면, UI 컴포넌트는 앞서 이미지와 텍스트를 이용한 원시적 구현 방식과는 비교할 수 없을 정도로 구현이 쉽고 간단하다. 실제로 버튼 컴포넌트는 사용자의 클릭 이벤트를 전달받고 앱에 그 상태를 전달한다. 이를 위해 버튼은 클릭 이벤트를 구현하며 클릭에 대한 상태 정의(Normal, Press)와 상태별 이미지를 다르게 출력하는 기능을 수행한다. 상태 전이 애니메이션 효과가 있다면 그러한 기능까지 모두 구현한다.

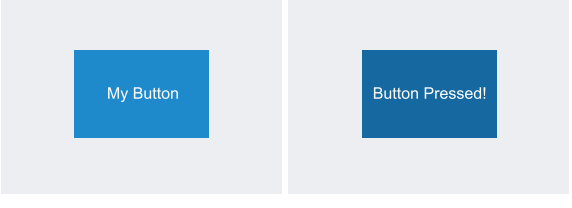


그림 1.3 버튼 클릭 상태 전이

이처럼 UI 컴포넌트는 앱 개발자 대신 색상, 이미지, 텍스트 등 UI 기본 요소를 내부적으로 조합, 배치함으로써 컴포넌트의 비주얼을 완성하고 동작 기능도 제공한다. UI 앱 개발자는 미리 완성된 UI 컴포넌트를 적재적소에 배치하고 각 컴포넌트가 제공하는 이벤트 동작을 구현함으로써 앱 UI를 빠르게 완성할 수 있다.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

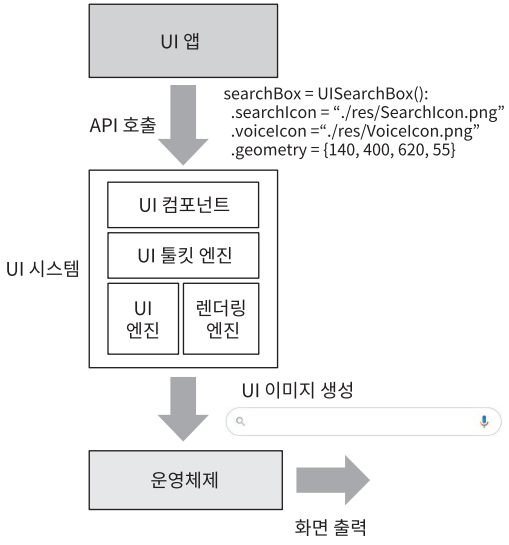


그림 1.4 UI 시스템 구성

그림 1.4는 UI 컴포넌트 기능을 탑재한 UI 시스템의 구성과 동작 흐름을 개략적으로 보여준다. UI 컴포넌트와 UI 툴킷 엔진은 UI 엔진(1.3절 참고)과 렌더링 엔진(2.1절)에 의존하여 기능을 수행한다. 여기서 UI 컴포넌트는 여러 컴포넌트 기능을 구현하는 패키지 형태의 모듈을 가리킨다. 대표적으로 버튼(Button), 토글(Toggle), 아이콘(Icon), 레이블(Label), 캘린더(Calendar) 등의 기능이 여기에 속한다. UI 툴킷 엔진은 앞서 언급한 여러 컴포넌트의 공통 기능을 하나의 통합 요소로 분리함으로써 기능 구성을 최적화한다. 대표적으로 UI 툴킷 엔진은 클래스 확장 방식을 이용하는 경우 기반 컴포넌트 클래스(base class)를 정의하고 이를 툴킷 엔진의 여러 인프라 기능과 연동, 구현한다. 이후 버튼, 토글과 같은 실용 UI 컴포넌트는 기반 컴포넌트를 확장하는 방식을 통해 쉽고 안정적으로 기능을 추가할 수 있다. 툴킷 엔진은 1.4절에서 좀 더 자세히 설명한다.

한편, UI 엔진은 UI 컴포넌트 기능을 제외한 UI 핵심 기능을 정의한다. 시스템 통합, 이벤트, 입출력, 메인 루프(1.3.1절) 등이 이에 해당한다. 렌더링 엔진은 캔버스를 기반으로 UI를 드로잉하고 화면에 출력하는 기능을 담당한다. 여기서 캔버스는 UI 시스템의 구현 전략 중 하나로, 시스템 특성에 따라 다르게

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

설계/구현할 수 있다. UI 컴포넌트는 렌더링 엔진에서 제공하는 원시 기능(예: 도형, 이미지, 텍스트를 출력하는 기능)을 활용하여 컴포넌트 고유의 비주얼을 완성하고 컴포넌트 고유의 기능을 앱이 사용할 수 있도록 API를 제공한다.

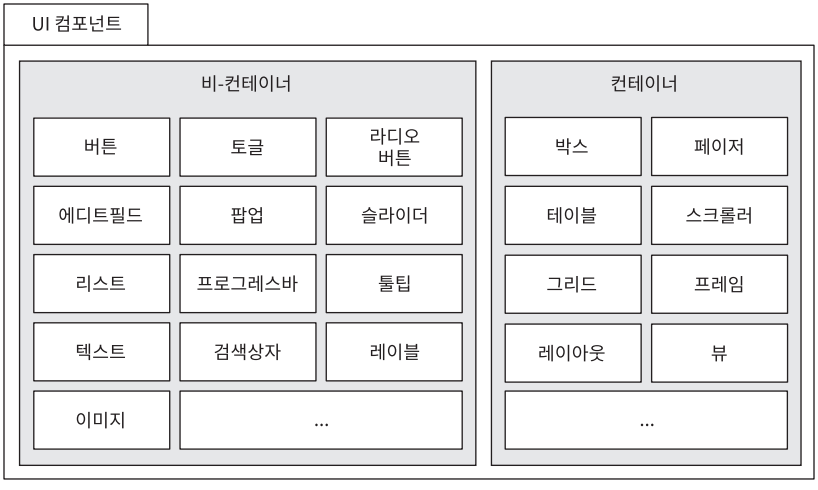


그림 1.5 UI 컴포넌트 패키지 기능 구성

그림 1.5는 UI 컴포넌트 패키지 내 세부 기능을 정리한 것이다. UI 컴포넌트는 앱에 필요한 다양한 종류의 기능 요소를 구성하여 탑재, 제공한다. 하나의 라이브러리 형태의 모듈[3]로 제공할 수도, 각 컴포넌트별로 독립 모듈로 제공할 수도 있다. 컴포넌트 종류는 UI 시스템이 제공하는 디자인 정책을 비롯하여 모바일, 데스크톱, 태블릿, TV 등의 디바이스 프로파일별로 차이가 있지만 오늘날 UI 앱에서 필요로 하는 UI 컴포넌트 범주는 상당 부분 정형화되어 있으므로 큰 맥락에서 살펴보면 컴포넌트 종류 및 기능은 서로 유사하다.

1.1.3 이벤트 처리

버튼은 UI 컴포넌트 중에서도 우리에게 가장 친숙한 기능이라 할 수 있다. 버튼의 핵심 기능은 사용자가 신호를 알릴 수 있도록 하는 것으로, 사용자가 버튼을 클릭하면 앱으로 신호를 전달한다. 앱이 버튼 신호를 전달받으면 그에 상

3 so, dylib 또는 dll과 같은 형태의 파일

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

응하는 어떤 동작을 취할 수 있다. 그림 1.6을 보면 버튼을 눌렀을 때 UI 컴포넌트 신호 처리를 설명하기 위해 버튼의 문구가 바뀐다.

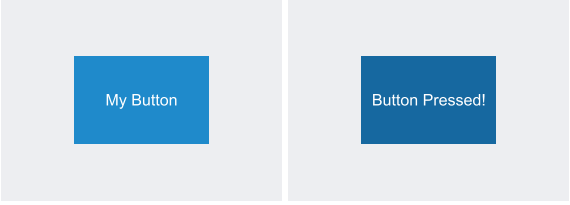


그림 1.6 버튼 눌림 전(왼쪽)/ 후(오른쪽)의 상태 변화

코드 1.4 버튼 생성 및 클릭 이벤트 등록

```
myBtn = UIButton():                 // 버튼 생성
  .text = "My Button"               // 버튼 텍스트
  .geometry = {110, 80, 200, 130}   // 버튼 위치 및 크기

/*
 * 버튼 클릭 이벤트를 람다 함수로 구현
 * Clicked 이벤트 발생 시 func() 수행
 * obj 인자로 myBtn 인스턴스 전달
*/
func(UIButton obj, ...):
  obj.text = "Button Pressed!"  // 버튼 레이블 변경

myBtn.EventClicked += func      // 클릭 이벤트 수행 함수 func() 등록
```

코드 1.4의 구현 코드를 통해 사용자가 버튼을 클릭하면 func()를 수행한다는 사실을 짐작할 수 있다. 사실 이러한 이벤트 처리는 UI 시스템, 즉 UI 컴포넌트와 UI 엔진 내부의 복잡한 과정을 거쳐 발생하지만(6.3절 참고) 앱 개발자는 그러한 지식을 알지 못하더라도 버튼의 이벤트 기능을 쉽게 구현할 수 있다.

UI 컴포넌트를 잘 활용하기 위해서는 앱 개발자가 해당 컴포넌트가 제공하는 기능과 이벤트 종류를 알 수 있어야 한다. 버튼의 경우 기본적으로 클릭 이벤트를 제공하지만 눌림(pressed), 눌림 해제(unpressed), 길게 누름(long-pressed)[4]과 같은 다른 이벤트도 제공할 수 있다. 앱 개발자는 버튼 이벤트에

4 일정 시간(약 0.25초) 버튼을 누르고 있을 때 발생하는 이벤트.

미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/

대한 명세서를 이해하고 이벤트를 적재적소에 활용함으로써 사용자 시나리오에 맞는 비즈니스 로직을 구현할 수 있다.

코드 1.5 버튼 복수 이벤트 등록

```
// 눌림 이벤트 발생 시 pressedCb() 수행
pressedCb(UIButton obj, ...):
  ...

// 눌림 해제 이벤트 발생 시 unpressedCb() 수행
unpressedCb(UIButton obj, ...):
  ...

// 롱프레스 이벤트 발생 시 longpressedCb() 수행
longpressedCb(UIButton obj, ...):
  ...

myBtn.EventPressed += pressedCb
myBtn.EventUnpressed += unpressedCb
myBtn.EventLongpressed += longpressedCb
```

물론, UI 시스템은 다양한 컴포넌트 기능과 구현 방법을 이해할 수 있도록 앱 개발자가 참고할 명확한 개발 가이드 문서를 제공해야 한다.

코드 1.6 Doxygen 형식에 맞춰 작성한 버튼 개발 가이드 문서 예시

```
/**
 * @defgroup UIButton Button
 * @ingroup UIComponent
 *
 * This is a push-button. Press it and run some function. It can contain
 * a simple text and icon object and it also has an autorepeat feature.
 *
 * This widget inherits from the @ref Layout one, so that all the
 * functions acting on it also work for button objects.
 * ...
 * This control emits the following signals,
 * besides the ones sent from Layout.
 * @li EventClicked: the user clicked the button (press/release).
 * @li EventPressed: button was pressed.
 * @li EventUnpressed: button was released after being pressed.
 * @li EventLongpressed: the user pressed the button without releasing it.
 * ...
```


미리보기용 PDF 입니다. 구매: https://ebook.insightbook.co.kr/