2

GameOVEN: 온라인 멀티 플레이어 게임을 지원하는 개방형 게임 개발 환경

NHN ▌박종목 · 성석현 · 김기범 · 이강인

1. 개 요

일반적으로 게임 개발에는 그래픽, 사운드, 인공지능, 네트워크 등의 요소들을 처리할 수 있도록 하는 미들웨어(middleware)들과 게임 엔진 그리고 통합된 환경에서 게임개발이 이루어질 수 있도록 하는 통합 개발환경(IDE)이 보편적으로 활용되고 있다[1]. 2D 그래픽을 기반으로 하는 통합개발환경으로는 Adobe의 Flash 기술[2]이 1996년에 소개되면서 쉽게 그래픽과 사운드를 조합하여 비디오 클립을 제작할 수 있게 되었고 여기에 ActionScript[3]라는 프로그래밍 언어를 지원함으로써 게임까지 개발할 수 있는 통합개발환경(Integrated Development Environment - IDE)으로 발전되었다. 3D 그래픽을 기반으로 하는 게임들의 경우, 1996년에 ID software에서 Quake[4]란 게임과 함께 게임 엔진을 함께 공개함으로써 지금까지의 많은 3D 게임 개발 환경들이 이로부터 영향을 받게 되었다. 초기에는 IDE 보다는 게임을 구동시키기 위한 게임 엔진의 발달이 이루어졌으나 최근에는 Unreal[5], CryEngine[6], Torque[7] 등에서 단순히 프로그래밍뿐 아니라 그래픽, 사운드 등의 요소들을 통합할 수 있고 인공지능을 쉽게 구현할 수 있는 IDE 형태로 발전되고 있다.

한편, 최근에는 고급 프로그래밍 능력이 없더라도 누구라도 쉽게 게임을 제작할 수 있는 개방형 게임 개발 환경들도 등장하고 있다. 몇 가지 예로서 Game Maker[8]와 Game Salad[9]가 있다. 이들은 주로 게임을 쉽게 제작할 수 있도록 하기 위해 2D 기반의 그래픽을 위주로 하고 있으며, 게임에 필요한 공통적인 요소들을 함께 제공함으로써 간단한 게임들은 프로그래밍 노력 없이도 개발할 수 있도록 해준다. 특히, Game Maker의 경우 게임의 개발뿐 아니라 yoyogames라는 게임 포털 사이트에 게임을 쉽게 퍼블리싱할 수 있도록 함으로써 만들어진 게임이 직접 게임 포털 사이트에서 큰 노력 없이 서비스할 수 있도록 한다.

하지만, 대부분의 게임 개발 환경들은 싱글 유저(single user) 게임 개발에 국한되어 있거나 제한된 멀티 유저(multi-user) 환경을 제공한다. Flash, Game Maker, Game Salad 등의 환경들은 기본적으로 싱글 유저 게임에 특화되어 있고 멀티 유저 게임을 개발하려면 별도의 게임 서버를 개발해서 연동시켜야 한다. Quake, Unreal, CryEngine, Torque 등은 멀티 유저 게임 개발이 가능하도록 되어 있으나 개발 환경 자체가 멀티 유저 게임에 특화되어 있지는 않으며 별도 게임 서버를 구축해야 한다. 또한, 게임 포털에서 온라인 상에서 멀티 유저 게임 형태로 서비스하려면 게임 포털의 시스템들과 연동을 하기 위한 추가 개발을 해야 하기 때문에 멀티 유저 게임을 온라인에서 서비스하는 데에는 많은 시간과 노력이 들며, 일반인의 자격으로 이러한 개발을 하는 것은 매우 어렵다.

이 논문에서는 개방형 온라인 멀티 유저 게임 개발환경인 GameOVEN(Game Online Virtual Environment)[10]을 소개한다. GameOVEN은 멀티 유저 게임 개발에 특화된 통합 게임 개발환경을 제공함과 동시에 누구나 개인의 자격으로도 멀티 유저 게임을 퍼블리싱할 수 있도록 하는 iDoGame[11] 이란 서비스 환경에 게임을 쉽게 퍼블리싱할 수 있도록 해준다. 멀티 유저 게임 개발에 특화된 기능들로는 게임 클라이언트와 게임 서버를 하나의 통합된 개발환경에서 프로그래밍, 디버깅과 실행을 할 수 있도록 한다. 또한, GameOVEN은 크게 IDE와 Runtime 엔진으로 구성되며, 동일한 Runtime이 IDE에서 구동되고 디버깅하는 데에도 활용되면서 iDoGame 서비스 환경에서도 최종 사용자들이 직접 게임을 구동시키는 데에도 활용되므로, GameOVEN의 IDE는 가상의 온라인 게임 실행 환경을 제공한다.

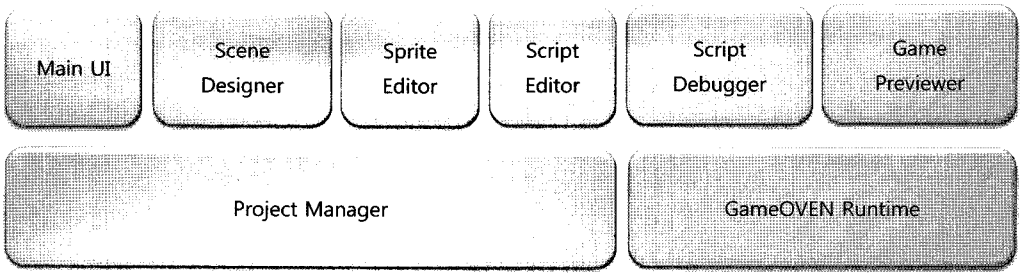

그림 1 GameOVEN IDE의 구조

논문의 구성은 다음과 같다. 2장에서는 GameOVEN의 전체 시스템 구성을 설명하며, iDoGame이란 서비스 환경과의 관계를 설명하여 게임이 개발되고 서비스 환경에서 구동되는 과정을 소개한다. 3장에서는 GameOVEN의 IDE를 소개한다. 4장은 게임 프로그래밍에 활용되는 언어와 멀티 유저 온라인 게임 프로그래밍에 활용될 수 있는 프로그래밍 모델들을 소개한다. 5장에서는 GameOVEN의 활용 사례들을 소개한다. 마지막으로, 6장에서 결론을 맺고 향후 발전 방향을 기술한다.

2. GameOVEN의 구조

이 장에서는 GameOVEN의 구조와 iDoGame 서비스 플랫폼과의 관계를 설명한다. 서론에서 언급되었 듯이 GameOVEN은 가상의 멀티 유저 온라인 게임 실행 환경을 제공하며, GameOVEN으로 개발된 게임은 iDoGame 서비스 플랫폼에 의해 온라인 상에서 사용자들에게 퍼블리싱하는 형태의 구조를 갖는다.

그림 1은 GameOVEN 툴의 통합개발환경 구성을 보여준다. 게임 화면을 디자인하기 위한 Scene Designer와 애니메이션을 위한 스프라이트 제작하는 Sprite Editor는 디자이너들을 위한 기능들을 제공한다. 한편, 게임 인공지능이나 사용자 인터페이스의 로직 프로그래밍을 위하여 LUA[12] 언어를 지원하며, 스크립트의 작성과 편집을 지원하는 Script Editor와 Debugger를 제공한다. 또한, Previewer는 게임 개발 도중 게임의 플레이과정을 미리 볼 수 있는 게임 preview 기능을 제공하여 게임 개발 결과를 직접 확인할 수 있도록 해준다. 이러한 preview 기능은 실제 게임이 수행되는 GameOVEN Runtime이란 실행 환경을 활용하며, 동일한 Runtime이 iDoGame 서비스 플랫폼에 수행됨으로써 실제 사용자들에게 개발한 게임이 동일하게 서비스될 수 있도록 한다.

그림 2는 GameOVEN과 iDoGame 서비스 플랫폼과의 관계를 도식화한다. 게임 개발자는 GameOVEN으로 개발한 게임을 iDoGame 웹 사이트에 업로드하면, 게임의 클라이언트 패키지와 서버 패키지가 구분되어 iDoGame Back-end server에 저장된다. iDoGame Back-end Server들에는 DB 서버, 다운로드 서버, 공지 서버 등의 게임 외적인 부분을 처리하는 서버들로 구성된다. 게임의 이용자들은 iDoGame 웹 사이트에서 iDoPlayer라는 게임 브라우저를 다운로드하고 PC에 설치 후

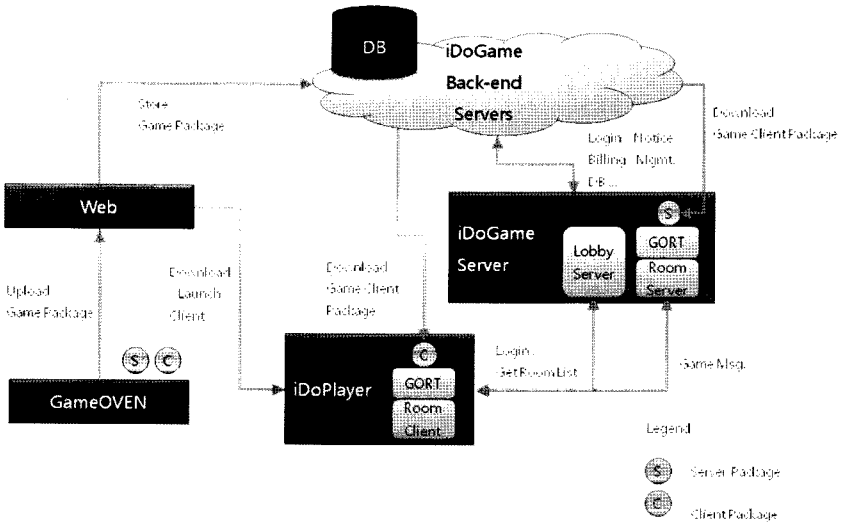

그림 2 GameOVEN과 iDoGame Service Platform

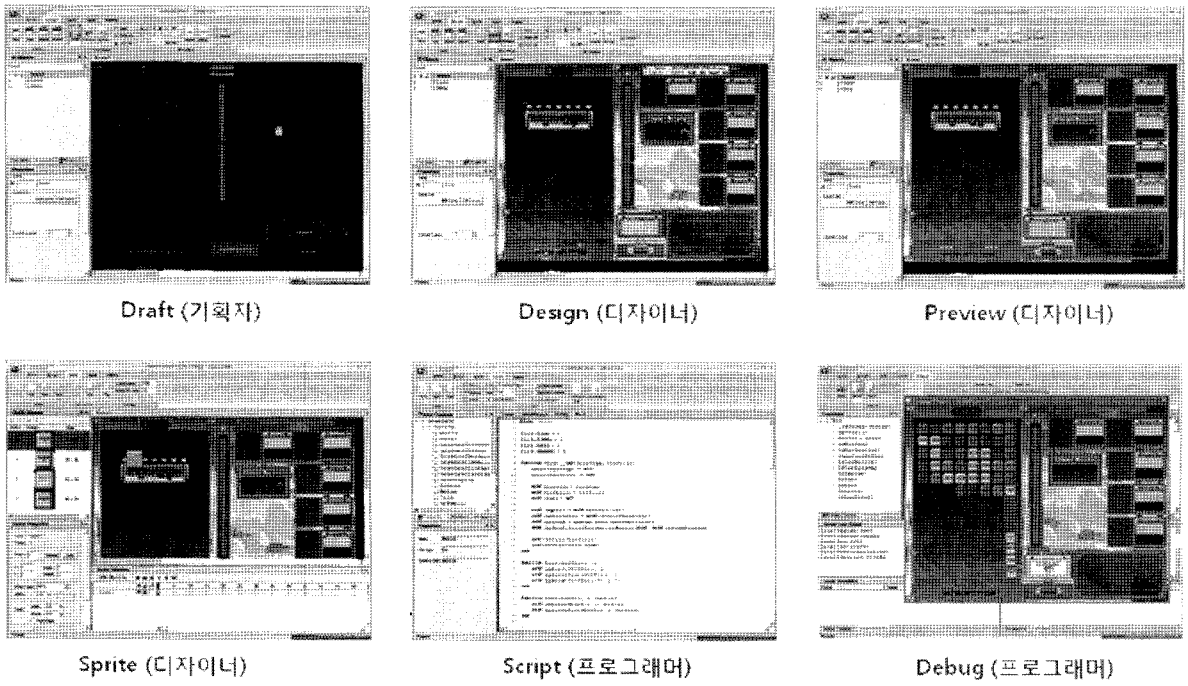

그림 3 디자이너와 프로그래머를 위한 기능들

게임들을 선택하여 수행시키게 된다. iDoPlayer는 개발자들이 업로드한 게임들의 목록을 보여주며, 멀티 유저 게임 방들의 목록을 보여주는 로비를 제공한다. 게임을 수행시킬 때에는 GameOVEN Runtime (GORT)의 클라이어트 부분을 활용하여 게임의 클라이언트 패키지를 다운로드 받아서 수행시킨다. GameOVEN Runtime의 서버 부분은 iDoGame Server에 로딩되어 게임의 서버 패키지를 수행시킨다.

3. GameOVEN의 IDE

이 장에서는 GameOVEN의 IDE가 제공하는 주요 기능들을 소개한다. GameOVEN은 크게 디자이너를 위한 기능들, 프로그래머(개발자)를 위한 기능들, 게임 패키징 기능을 제공한다. 그림 3은 디자이너를 위한 기능들과 프로그래머를 위한 기능들의 화면들을 보여준다.

이 장의 구성은 다음과 같다. 3.1절에서는 디자이너를 위한 기능들을 소개하고 3.2절에서는 프로그래머를 위한 기능들을 소개한다. 3.3절에서는 게임 패키징 기능을 소개한다.

3.1 디자인 기능

현재 iDoGame 웹 사이트에서 제공하는 GameOVEN 1.0 및 1.1 버전에서는 기본적으로 2D 그래픽 기반의 게임 개발 기능만 제공한다. GameOVEN IDE에서는 2D 게임의 화면이나 스프라이트을 편집하고, 게임에 필요한 이미지, 스프라이트, 텍스트나 버튼 등의 사용자 인터페이스 요소들 정의하고 이들을 조합할 수 있도록 해준다.

GameOVEN IDE는 게임 화면을 보여주는 다양한 뷰(View)를 제공한다(그림 3 참조).

- 드래프트 뷰(Draft View): 게임 개발 초기 단계에서 실제 게임에서 사용되는 그래픽 리소스가 준비되지 않더라도 게임 화면의 구성 요소들을 화면에 배치할 수 있도록 한다.
- 디자인 뷰(Design View): 게임 화면의 구성 요소들에 그래픽 리소스가 적용된 상태의 화면을 보여준다.
- 프리뷰(Preview): 여기서는 디자인 뷰와 마찬가지로 그래픽 리소스가 적용된 상태를 보여주나, 스프라이트들의 경우 자동으로 동작하도록 되어 스프라이트의 애니메이션 형태들을 확인할 수 있도록 해준다.

게임 화면 디자인 시에 Drag & Drop으로 화면에 배치할 수 있는 기본 구성 요소들 다음과 같다.

- Image: 이미지로서 Image, StaticImage, ImageNumber 등이 있다.
- Text: 텍스트 상자로서 Text, TextList, Edit 등이 있다.
- Button: 눌렀을 때 반응하는 버튼들로서 CheckBox, RadioButton, Button 등이 있다.
- Bar: 어떤 값이나 상태를 조절할 때 사용하는 컨

트롤로서 ScrollBar, SlideBar, ProgressBar가 있다. 보통 독립적으로 사용되기 보다는 다른 구성 요소를 제어하기 위해 사용된다.

- Sprite: 애니메이션이 되는 이미지 객체를 의미한다. 스프라이트 파일의 경우 GameOVEN의 자체 포맷들로서 HSV가 있고, Flash로 제작된 SWF 파일로 만들어진 스프라이트도 지원한다.
- Particle: 효과를 나타내는 이미지 객체를 의미한다.

끝으로, 게임의 화면 디자인에 활용되지는 않지만 게임에서 사용되는 기본적인 리소스들인 사운드 객체와 스트링 객체를 정의할 수 있다. 사운드 파일의 경우 GameOVEN 자체 포맷인 HSO와 외부 포맷인 WAV, MP3, OGG를 지원한다. 스트링 객체는 문자열로 구성된 데이터를 객체로 정의하기 위해 사용되며, 게임의 다국어 지원 용도로 활용될 수 있다.

3.2 프로그래밍 기능

GameOVEN에서의 게임 프로그래밍은 주로 게임의 사용자 인터페이스나 인공지능의 구현을 쉽게 지원하는 데에 초점이 맞추어져 있다. 즉, 그래픽 처리나 네트워크 처리 등은 기본적으로 GameOVEN Runtime 엔진이 내부적으로 제공함으로써 스크립트 언어를 활용한 간단한 프로그래밍 만으로도 게임을 완성할 수 있도록 해준다. 또한, 게임 화면 디자인 기능을 활용할 때 게임 스크립팅을 도와줄 수 있도록 다음과 같은 프로그래밍 코드들을 자동으로 생성된다:

- 이벤트 핸들러: 게임 화면 구성 요소에서 대해서 개발자가 원하는 이벤트를 지정하여 해당 이벤트에 대한 핸들러를 자동 생성할 수 있다.
- UI/사운드 오브젝트: 게임 화면 디자인에서 사용한 객체들은 코드에서 사용할 수 있는 객체들로 정의되도록 코드를 생성해준다. 이러한 객체들은 게임 화면 디자인에서 저장한 ID를 통해 식별되며, 게임 화면 디자인을 저장할 때 마다 코드가 생성된다.

또한, GameOVEN의 IDE는 프로그래밍을 위한 스크립트 에디터와 디버거를 제공한다. 개발자는 IDE 환경에서 게임 코드를 작성하고 디버거를 통해 테스트 해 볼 수 있으며, GameOVEN Runtime을 이용해 게임을 실행해 볼 수 있다. 디버거는 스크립트의 수행 중에 발생하는 오류를 빠르게 찾아낼 수 있도록 필수적인 디버깅 기능들을 제공한다. 브레이크 포인트 지정, 호출 스택(call stack)과 변수 감시(variable watch), 에러 로깅(error logging) 등을 통해 오류를 쉽게 찾아낼 수 있도록 도와준다.

3.3 게임 패키징 및 퍼블리싱 기능

GameOVEN은 개발된 게임을 iDoGame 서비스에 업로드하기 위한 게임 패키지를 생성하는 기능을 제공한다. 게임 패키지는 크게 게임 클라이언트용 패키지와 게임 서버용 패키지로 구분되며, 다음과 같은 특징이 있다:

- 게임 클라이언트용 패키지: 게임 클라이언트에서 구동되는 콘텐트로서 게임 클라이언트의 코드를 저장하는 게임 클라이언트 패키지(gcp 형식)와 화면 데이터, 이미지, 사운드 등의 게임 리소스를 저장하는 게임 리소스 패키지(gsp 형식)가 생성된다.
- 게임 서버 콘텐트: 게임 서버에서 구동되는 코드가 포함된 게임 서버 패키지(gcp)가 된다.

GameOVEN으로 생성한 게임 패키지들은 iDoGame 웹 사이트를 통해서 업로드할 수 있다. iDoGame 웹 사이트에 업로드된 게임 패키지들은 우선 게임을 온라인 환경에서 테스트해볼 수 있는 테스트 서버에 저장되며, 개발자는 iDoGame Test Player를 다운로드 받아서 자신이 업로드한 게임을 다른 사람들과 함께 테스트할 수 있다. 테스트가 완료된 게임은 게임 심의 요청 후 최종 심의 완료된 게임은 실제 사용자들에게 제공되는 서버에 업로드되며, 사용자들은 iDoPlayer를 다운로드 받아서 게임을 play하게 된다.

iDoGame 서비스에서는 각 게임이 unique한 ID가 부여되고, 게임 패키지들에 버전 번호가 부여되어 서버에 저장된다. 따라서, 사용자들은 특정 ID를 갖는 게임의 특정 버전의 게임 패키지를 내려 받아 사용하게 된다. 이외에도 iDoGame 서비스 플랫폼에서는 게임 패키지에 대한 무결성(integrity) 검사나 코드의 사전 컴파일/압축 등의 부가적인 기능을 제공한다.

4. GameOVEN의 프로그래밍 모델

GameOVEN에서 제공하는 스크립트 언어는 LUA 이다. LUA는 다른 언어에 비해 가볍고 유연한 장점을 지니고 있어 게임에서의 스크립트 프로그래밍에 많이 활용되고 있다. 또한, C++ 언어와의 바인딩이 가능하여 외부 모듈과의 연동이 매우 쉬워서 확장성도 제공할 수 있다. 다만, 현재의 GameOVEN에서는 보안을 고려하여 아직까지 C++ 언어로 작성된 모듈과의 연동은 허용하고 있지 않다. 이러한 제약 사항은 향후 개선될 예정이다.

이 장에서는 GameOVEN에서 온라인 게임 개발을 위해 필요한 프로그래밍 모델들을 소개한다. 게임 프

로그래밍을 위해 제공하는 모델로는 이벤트 기반 모델, 객체지향 모델, 게임 개발 프레임워크, 클라이언트/서버 모델, 그리고 데이터 액세스 모델이 있다. 다음 각 절에서 각각의 모델들을 설명하기로 한다.

4.1 이벤트 기반 모델

GameOVEN의 Runtime 엔진은 이벤트 기반 프로그래밍을 지원한다. Runtime 엔진은 각종 키보드, 마우스, 네트워크 등의 이벤트들을 개발자가 프로그래밍한 이벤트 핸들러에게 중계하는 역할을 한다. 개발자는 이벤트들에 대해 이벤트 핸들러 함수를 등록하는 형태로 프로그래밍하게 된다.

또한, 개발자가 직접 이벤트를 정의하고 이에 대한 이벤트 핸들러를 작성할 수 있다. 이러한 사용자 정의 이벤트는 메시지 형태로 정의되며, 어떤 객체가 다른 객체에 메시지를 전송하면 메시지를 받는 객체의 해당 메시지에 대한 이벤트 핸들러가 호출된다. 아래의 예제 코드는 클라이언트와 서버 간의 메시지 이벤트 예제를 보여주고 있다.

```
Evt_Hello = Event(1)

function tutorial1_1ClientApp:__init()
        self:SetMsgHandler(Evt_Hello, self.OnHello)
end

function tutorial1_1ClientApp:OnHello(server, msg)
        print("Hello")
end
```

이 예제에서는 Evt_Hello라는 Event를 미리 정의해 놓은 후 SetMsgHandler 함수를 사용하여 이벤트 핸들러 함수를 등록하고 있다. Event의 생성시에는 이벤트 아이디를 함께 입력하게 되는데 이 이벤트 아이디는 이벤트 식별자이다. 만약 Evt_Hello라는 이벤트가 발생하면 OnHello라는 함수가 호출되어 print("Hello") 라인이 실행되게 된다.

예제에서 확인할 수 있듯이 Evt_Hello와 같은 이벤트는 이 이벤트가 네트워크 이벤트인지 사용자 입력에 대한 이벤트인지 구분이 되지 않는다. GameOVEN에서는 키보드, 마우스 등의 입력이나 네트워크 등으로 인해 발생하는 이벤트에 대해 동일한 이벤트 기반 프로그래밍 모델을 제공하여 어떤 이벤트도 모두 같은 방식으로 사용할 수 있다. 따라서, 자세한 지식이 없이도 기본적인 개념만을 파악한다면 다양한 이벤트에 대한 구현을 손쉽게 할 수 있다.

4.2 객체지향 모델

LUA 스크립트 언어는 본래는 객체지향 모델을 지원하지 않지만 객체지향 모델을 지원할 수 있도록 확장이 가능하다. GameOVEN은 Luabind[13]라는 LUA의 확장 기능을 통하여 LUA 스크립트 언어에서 객체지향 모델을 활용할 수 있게 하였다. 다음 예제는 Luabind를 사용하여 객체 클래스를 표현한 예제 코드이다.

```
class 'TestClass'

function  TestClass:__init()
          self.memberVar1 = 1
          self.memberVar2 = "var2"

          print("__init")
end

function  TestClass:MemberFunc()
          print("called MemberFunc()")
end

function  TestClass:__finalize()
          print("__finalize")
end
```

이 예제는 첫번째 줄에서 "TestClass"라는 이름의 클래스를 선언하고 있다. 그 후 3개의 함수가 선언하는 내용을 살펴볼 수 있는데 __init, MemberFunc, __finalize라는 함수명 앞에 "TestClass:"라는 문자열을 함께 적어줌으로써 3개의 함수가 모두 TestClass 클래스의 멤버 함수라는 것을 명시하고 있다. 여기서 __init, __finalize 함수는 특수 함수로서 각각 생성자와 소멸자를 의미하고 있다. 그리고 TestClass의 인스턴스가 생성되었을 때 그 자신을 가리키는 self라는 키워드를 통해서 memberVar1, memberVar2라는 멤버 변수들도 함께 선언하고 있다.

이외에도 클래스 간의 상속관계를 정의할 수 있으며, 연산자 재정의도 가능하다. 단, 클래스 간의 상속관계는 단일 상속만 지원한다.

4.3 게임 개발 프레임워크

GameOVEN은 개발자들이 효율적으로 게임을 제작할 수 있도록 GO framework라는 게임 개발 프레임워크를 제공한다. GO framework은 게임을 완성하기 위해 필요한 기능들 중에서 공통적으로 활용되는 기능들을 클래스 라이브러리 형태로 제공한다. 여기서의 클래스 라이브러리는 GameOVEN의 객체지향 프로그

래밍 모델을 활용하여 구성된다.

또한, GameOVEN IDE는 게임 애플리이케이션에서 필요한 기본적인 코드들은 자동으로 생성해준다. 새로운 프로젝트를 생성하면 네트웍 통신을 하기 위한 기본적인 클라이언트/서버 구조를 구현한 클래스와 클라이언트의 화면 제어 클래스, 네트워크 게임에 필요한 기본적인 이벤트 핸들러 함수 등이 생성된다. 이외에도 3.2절에서 설명하였듯이 화면 디자인 과정에서 게임 프로그램과 연동하기 위한 코드들도 자동 생성된다.

더불어 GO framework은 소스코드 형태로 배포되고 있는데 이를 통해서 사용자가 자신만의 framework을 만들어 기존의 GO framework과 연동하여 사용할 수도 있고 혹은 GO framework 자체를 수정하여 자신만의 framework으로 개조하여 사용할 수도 있게끔 하고 있다. 이러한 요소들을 통해서 사용자들은 간단하게는 자신만의 독특한 동작을 하는 버튼을 만들어 GO framework과 연동하여 사용하는 일에서부터 크게는 액션, RPG, 시뮬레이션 등의 특정 게임 장르에 적합한 프레임워크를 개발하여 사용하는 것이 가능하다.

4.4 클라이언트/서버 통신 모델

GameOVEN은 클라이언트/서버 통신 모델을 지원한다. 클라이언트는 각 게임 사용자의 PC 환경에서 구동되며 주로 사용자 인터페이스나 그래픽, 사운드 등의 사용자에게 직접적으로 출력하는 내용들을 제어하는 것이 목적이다. 서버는 클라이언트들과 메시지 교환을 통하여 전체 게임의 상태를 저장하고 게임의 수행을 제어하는 것이 목적이다.

GameOVEN에서의 서버는 논리적인 서버를 의미하며, 물리적으로는 서버가 구동되는 환경이 서비스 플랫폼에 의해 결정된다. 현재의 iDoGame 서비스 플랫폼에서는 GameOVEN의 서버를 물리적인 서버에서 구동되도록 하고 있다. 향후에는 사용자의 PC에서도 서버를 직접 구동할 수 있는 형태를 지원할 예정이다.

iDoGame 서비스에서 사용자가 게임 방을 개설하면 해당 방을 서비스하는 GameOVEN의 서버가 구동된다. 게임 방에 접속하는 사용자들에게는 각 사용자에 해당되는 GameOVEN의 클라이언트가 구동되게 된다. 게임 방을 개설하거나 게임 방에 접속하는 등에 대한 모든 처리는 iDoGame 서비스 플랫폼에 의해 진행되기 때문에 GameOVEN을 사용한 게임 개발 과정에서는 이러한 처리에 대해서 신경 쓸 필요가 없고, 오로지 게임 내에서의 클라이언트와 서버 간의 메시지 처리에 대해서만 구현하면 된다.

GameOVEN에서의 클라이언트/서버 간의 메시지 교환 방식에는 2가지 방식이 있다. 한가지는 Send() 혹은 SendUnreliable() 함수를 통해서 메시지를 네트워크로 직접 전송하는 방식이고 또 한가지는 SyncObject를 사용하는 방식이다. Send 함수는 TCP 기반의 메시징을 활용하며, SendUnreliable은 UDP 기반의 메시징을 활용한다. SyncObject는 클래스의 형태로 제공되는데 이 클래스는 변수의 값이 변경되면 변경된 내용을 네트워크를 통해서 서버나 클라이언트로 자동으로 전송해 주는 역할을 한다. 이를 이용하면 서버와 개별 클라이언트간의 동기화를 보다 쉽게 구현할 수 있다.

4.5 데이터 저장 모델

일반적으로 온라인 게임을 만들기 위해서는 사용자의 전적 정보 혹은 게임 내에서 진행한 점수 등을 저장하는 것이 필요하다. 이와 같은 데이터를 저장하는 것을 지원하기 위해서 GameOVEN에서는 데이터 저장 모델을 제공한다. GameOVEN에서의 데이터 저장 모델은 Serialization에 의한 저장 모델이다. 즉, 어떤 사용자 정의 클래스도 Serializable 클래스로부터 상속을 받으면 데이터베이스에 저장될 수 있다. Serializable 클래스는 객체의 멤버 변수들을 모두 하나의 바이트 스트림으로 만드는 기능을 제공한다.

데이터의 저장소를 제공하는 클래스로는 GameData와 PlayerGameData가 있다. GameData는 모든 사용자들이 게임에서 공통으로 활용하는 데이터를 저장하기 위한 목적으로 제공되며, PlayerGameData는 특정 사용자에 해당하는 데이터를 저장하기 위한 목적으로 제공된다. 또한, 이러한 저장소에 해당되는 클래스들은 서버에서만 액세스가 가능하다.

5. GameOVEN의 활용 사례

이 장에서는 GameOVEN을 직접 활용한 몇 가지 사례들을 소개하고자 한다. 초기에 GameOVEN 1.0 베타가 개발되었을 때 어느 정도 활용하기 쉬운 지를 확인하기 위한 몇 가지 검증 작업을 수행하였고, 그 결과에 대해 공유하고자 한다. 크게 초보자와 경험자의 두 가지 부류의 개발자들에 대한 검증을 수행하였다. 초보자의 경우 게임 개발 경험이 전혀 없고 LUA란 언어와 GameOVEN을 처음 접한 개발자에 해당되며 3명에 대해서 동시에 검증 작업을 진행하였다. 경험자의 경우 이미 3년 이상 DirectX, C++ 기반의 게임을 개발한 경험이 있고 LUA란 언어도 어느 정도 익숙한 상태에서 GameOVEN을 처음 접한 경우이다.

표 1 GameOVEN을 활용한 간단한 게임 개발 사례

게임명	개요	소요기간
세균전	상대편을 감염시켜 영역을 넓혀가는 보드 게임	총 6일(게임오븐 학습/게임 기획: 3일, 게임 개발: 3일)
알까기	장기판에서 장기 알을 튕겨서 상대편 알을 밖으로 나가게 하는 보드 게임	총 7일(물리엔진 학습/게임 기획: 3일, 게임 개발: 3일)
당구	당구공을 맞추면서 점수를 획득하는 게임	총 7일(게임 기획: 1일, 게임 개발: 6일)

초보자의 경우 2009년에 대학교를 졸업한 신입사원에 해당되며, GameOVEN에 대한 교육과 간단한 실습을 수행 후 약 한달 간 총 3가지 게임을 개발하였으며, 하나의 게임당 약 일주일의 기간이 소요되었다. 이는 GameOVEN, LUA 문법 등에 대한 학습과 게임 선정 및 기획을 포함한 기간이다. 표 1은 개발된 게임과 소요된 기간을 보여준다.

표 1에서 실험한 게임들은 모두 턴(turn) 기반의 보드 게임들로서 GameOVEN 1.0 베타가 처음 오픈되었을 당시 UDP 기반 네트워킹을 지원하지 않아서 실시간 액션 게임은 포함되지 못했었다. 또한, 디자인 요소들은 기존의 리소스들을 최대환 활용하였으므로 디자인에 들어가는 시간은 제외되었다. 초보자들에게 주어진 개발들은 주로 간단한 게임 개발이었지만, 게임 개발 경험이 전무한 신입사원들이라는 점을 고려하였을 때 빠른 시간 안에 LUA 언어와 GameOVEN을 학습하고 멀티 플레이어 게임을 개발할 수 있었다는 점에서 GameOVEN의 유용성을 입증하였다.

한편, GameOVEN이 상용 게임의 개발 가능성을 확인하기 위해 게임 개발 경험이 많은 전문 개발자로 하여금 실제로 서비스 중인 테트리스 게임을 거의 동일한 spec으로 혼자서 멀티 유저 게임을 GameOVEN으로 개발하도록 하였다. 단, 이미지, 사운드, 스프라이트 등의 디자인 요소들은 기존의 서비스 중인 게임의 리소스들을 활용하도록 하였다. 결과적으로 QA 기간을 포함해서 총 18일 만에 게임을 완성하였다. 여기서 주목할 만한 것은 개발자 한 명이 클라이언트와 서버를 모두 개발할 수 있었다는 것이다. 통상적으로는 멀티 유저 게임 개발에 클라이언트 개발자와 서버 개발자가 별도로 게임을 개발하고 연동해야 하는데, GameOVEN에서 지원하는 멀티 게임 디버깅, 클라이언트간 공유 데이터에 대한 동기화 지원 등의 기능으로 인해 빠른 시간에 상용 수준의 멀티 유저 게임의 개발이 가능하였다.

6. Conclusion and Future Directions

이 논문에서는 멀티 유저 게임 개발에 특화된 통합 게임 개발환경을 제공함과 동시에 누구나 개인의 자격으로도 멀티 유저 게임을 퍼블리싱할 수 있도록 하는 개방형 온라인 멀티 유저 게임 개발환경인 GameOVEN을 소개하였다. GameOVEN은 IDE와 Runtime으로 구성되며, GameOVEN IDE의 기능으로 디자인 및 프로그래밍 기능들을 소개하였고, GameOVEN Runtime이 제공하는 프로그래밍 모델을 소개하였다.

아울러 GameOVEN을 활용한 게임 제작 사례들을 살펴봄으로써 GameOVEN의 유용성을 확인하였다. 먼저 초보자로서 게임을 개발한 경험이 없고 LUA언어와 GameOVEN을 처음 접하는 경우에도 일주일 정도면 간단한 멀티 유저 온라인 게임의 개발이 가능함을 확인하였다. 또한, 숙련된 개발자도 쉽게 상용 수준의 게임을 개발할 수 있음을 확인하였고, 특히 개발자 한 명이서 게임 클라이언트와 서버를 모두 개발할 수 있음을 입증하였다.

GameOVEN이 멀티 유저 온라인 게임 개발에 필요한 요소들을 제공한 다는 측면에서 다른 상용 IDE나 게임 엔진들과 차별화된 부분이 있으나 IDE 환경이나 Runtime의 완성도, 다양한 편리 기능을 제공하는 데에 있어서는 보완할 부분이 많이 있다. IDE 측면에서는 보다 편리한 기능들을 제공하면서 향후에는 프로그래밍 능력이 없더라도 쉽게 게임을 제작할 수 있는 환경으로 발전시킬 계획이다. Runtime 측면에서는 3D 그래픽과 P2P 네트워킹, C++ 기반의 모듈 연동 기능 등을 보완하여 보다 다양하고 전문적인 게임 개발이 가능한 형태로 발전시킬 계획이다.

감사의 글

iDoGame 서비스 플랫폼에 대한 기술적인 내용을 검토하고 지도해주신 성운재 랩장님과 논문에 필요한 자료들을 모으고 취합해주신 장필봉 차장님께 감사의 말씀을 전합니다.

참고문헌

[1] 편집부, 대한민국 게임백서 2006, 한국게임산업개발원, 2006.
[2] Adobe Flash, http://en.wikipedia.org/wiki/Adobe_Flash
[3] ActionScript, http://en.wikipedia.org/wiki/ActionScript
[4] Quake, http://en.wikipedia.org/wiki/Quake
[5] Unreal Engine, http://en.wikipedia.org/wiki/Unreal_Engine

[6] CryEngine, http://en.wikipedia.org/wiki/CryEngine
[7] Edward F. Maurina III, The Game Programmer's Guide to Torque, GarageGames, 2006.
[8] Game Maker, http://en.wikipedia.org/wiki/Game_Maker
[9] Game Salad, http://gamesalad.com/
[10] GameOVEN, http://idogame.hangame.com/guide/gameoveninfo.nhn
[11] iDoGame Lab, http://idogamelab.hangame.com
[12] Paul Schuytema and Mark Manyen, Game Development with LUA, Charles River Media, 2005.
[13] luabind, http://sourceforge.net/projects/luabind/

박 종 목

1990 연세대학교 전산과학과 학사
1992 한국과학기술원 전산학 석사
1997 한국과학기술원 전산학 박사
1998 IBM Almaden Research Center Visiting Scientist
1999 삼성전자 소프트웨어 센터 선임 연구원
2000 ㈜와이즈엔진 개발본부장
2002 ㈜신텔정보통신 기술연구소장
2003 삼성전자 소프트웨어 연구소 수석 연구원
2007~현재 NHN 주식회사 게임개발센터장/이사
관심분야 : 온라인 게임 퍼블리싱 플랫폼, 가상 머신, 게임 엔진 및 툴
E-mail : chongmok.park@nhn.com

성 석 현

1998 서울산업대학교 산업공학과 학사
2001 L&H Korea, 새롬기술연구소
현재 NHN 게임개발센터 게임빌더팀 팀장
관심분야 : 2D/3D 게임 개발, 게임 엔진 및 툴 개발
E-mail : oyar@nhn.com

김 기 범

2006 서울대 컴퓨터공학과 학사
현재 NHN 네트웍게임엔진팀 팀장
관심분야 : 분산 시스템, 네트워크 엔진, 가상 머신
E-mail : giveme98@nhn.com

이 강 인

2006 서울대 컴퓨터공학부 학사
현재 NHN 네트웍게임엔진팀 대리
관심분야 : 분산 시스템, 네트워크 엔진, 가상 머신
E-mail : ashira11@nhn.com