XNA 프레임워크 기반의 게임 클라이언트 마이그레이션 연구

이승훈* · 변정원** · 류성열***

1. 서 론

문화산업의 주류로 자리를 잡아가고 있는 게임 산업은 비약적인 기술의 발전과 함께 더욱 빠른 성장을 하고 있다.

초창기의 컴퓨터 게임은 프로그래머가 어셈블리 언어를 통해 개발을 하기 시작했고, 요즘에는 C/C++ 언어를 통해서 개발을 하고 있다. 그러나 C/C++ 언어를 통해 게임을 개발하는 것은 결코 쉬운 일은 아니다.

목표하는 플랫폼에 대한 그래픽, 물리 계산, 사운드, 컨트롤러 등의 각종 엔진을 개발해야 하고, 메모리 관리 및 로직에 대한 부분도 신경을 써야 하기 때문에 개발 생산성이 낮기 때문이다.

또한, 최근의 게임 산업 트랜드 중 하나는 동일 운영체제 환경의 이기종 H/W에서 게임을 서비스 하는 멀티 플랫폼과 이기종 운영체제 환경의 이기종 H/W에서 게임을 서비스 하는 크로스 플랫폼이라고 할 수 있다[1,2]. 게임산업의 성장으로 인해 게임 플랫폼 다변화에 대한 요구가 증가하고 있으며, 빠른 개발과 신속한 대응에 대한 필요성이 높아지고 있는 가운데, 게임 사용자들의 다양한 플랫폼 환경을 지원할 수 있는 것이 바로 크로스 플랫폼과 멀티 플랫폼 지원 기술이기 때문이다.

이와 관련해서 마이크로소프트는 2004년 GDC (Game Developers Conference)에서 XNA라는 새로운 게임 개발 환경을 제안하였다. XNA는 사용하기 쉬운 개발환경이다. C/C++ 언어 대신 C# 언어를 사용하기 때문에 C# 언어가 가지는 빠른 생산 능력을 활용할 수 있다. 또한 플랫폼이 아닌 게임 개발 자체에 초점을 맞추고 있기 때문에 누구라도 간단한 게임을 쉽게 만들 수 있다는 장점이 있다[3,4].

다음으로 크로스 플랫폼과 멀티 플랫폼 환경을 지향한다. 하나의 게임 소스 코드를 여러 플랫폼에서 사용할 수 있다는 의미이다. 현재 PC와 XBOX 360, 아이팟과 같은 휴대용 기기인 ZUNE까지 지원을 한다.

소니·컴퓨터엔터테인먼트·유럽(SCEE)도 GDC2008을 통해 PHYRE ENGINE을 발표하였다. PHYRE ENGINE 풀 소스 서포트의 모듈 타입의 엔진으로, 아트 데이터 작성/처리의 파이프라인까지가 정비되고 있다. 샘플 게임은 아트 세

※ 교신저자(Corresponding Author) : 이승훈, 주소 : 서울시 마포구 상암동 DMC단지 1602번지 문화콘텐츠센터 8F (121-270), 전화 : 02)3153-2781, FAX : 02)3153-2780, E-mail : shlee@kgda.or.kr
* 숭실대학교 대학원 컴퓨터학과 박사수료, 한국게임개발자협회 회장
** 숭실대학교 대학원 컴퓨터학과 석사과정 (E-mail : Jimi01@ssu.ac.kr)
*** 숭실대학교 컴퓨터학과 교수 (E-mail : syrhew@ssu.ac.kr)

트 첨부의 풀 소스/풀 데이터 세트로 제공, 문서도 부속되어 있다.

PHYRE ENGINE을 통해 PS3 게임제작사들은 자신의 게임을 PS3 및 XBOX360, PC에 대응 가능한 타이틀로 개발할 수 있다.

마이크로소프트의 XNA 비해 늦게 출시되었지만, 유럽에서는 이미 PHYRE ENGINE을 사용하여 개발된 게임들이 Xbox LIVE Arcade, PS3, PC용으로 출시되고 있다.

본 논문은 최근 이러한 게임산업 트랜드를 고려하여 기 개발된 DirectX 기반의 게임을 XNA 프레임워크 환경으로 변환하기 위한 과정과 절차, 그리고 변환 후에 DirectX 기반의 게임에 비해 어떠한 차이가 있는지를 비교 분석하여 프레임워크 기반의 게임 개발이 개발의 생산성 및 멀티 플랫폼 환경 개발에 어떠한 영향을 주는지를 살펴보고자 한다.

2. 관련연구

2.1 DirectX SDK

DirectX는 마이크로소프트 운영체제 환경에서 그래픽을 포함한 멀티미디어 관련 기능을 제공하는 하나의 라이브러리 세트이다. 대부분의 PC 기반 게임을 개발할 때는 DirectX SDK를 사용하여 게임을 개발한다.

DirectX SDK는 높은 성능을 요구하는 게임 및 멀티미디어 애플리케이션을 개발할 때 필요한 하위 수준의 API를 제공한다. 게임 개발자는 이러한 하위 수준의 API를 활용하여 입·출력, 화면처리, 오디오 등의 기능을 구현하여 게임을 제작한다.

그러나 DirectX SDK 기반의 게임 개발은 하위 수준의 API가 가지는 장점으로 인해 오히려 개발자들이 스스로 프레임워크를 구성해야 하는 문제

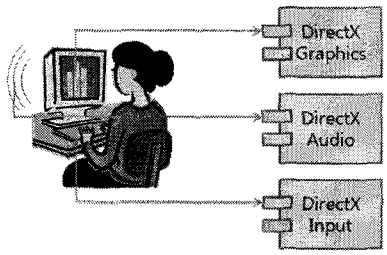


그림 1. DirectX SDK 기본 구성

점을 가지고 있다. 이는 게임 개발의 난이도 증가로 인해 개발 기간 및 비용 등에 있어 부담이 될 수밖에 없다.

또한, 개발에 참여하는 개발자마다 나름대로 프레임워크를 구성하기 때문에 설계의 수준이 달라지고 게임 로직과 DirectX 인터페이스가 코드에 같이 존재하게 되어 가독성 및 유지보수에 있어 많은 문제점이 발생하게 된다.

2.2 XNA 프레임워크

그림 2는 XNA 프레임워크 3.1을 기준으로 기본적인 구조를 보여주고 있다. XNA는 기본적으로 윈도우와 XBOX상에서 동작하며, 닷넷 프레임워크를 필요로 한다. 이러한 구조를 가지고 있기 때문에, 단일 개발 환경을 통해 거의 동일한 코딩을 통해 크로스 플랫폼 게임을 개발할 수 있다[4].

그림 3은 XNA 프레임워크 4 Layer 구조를 보여준다. 첫 번째는 Layer는 플랫폼 레이어로

그림 2. XNA Overview

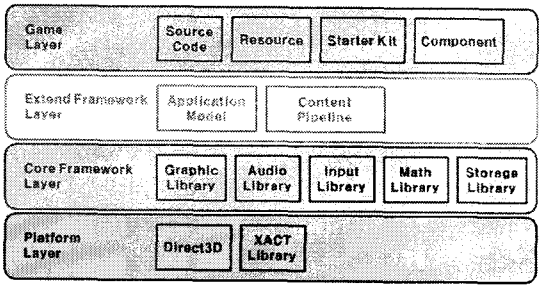


그림 3. XNA 프레임워크 4 Layer

Windows와 Xbox 360의 양쪽 플랫폼의 하드웨어를 효과적으로 동작시키기 위해 필요한 기능을 제공한다. 그래픽 조작을 위한 Direct3D와 사운드 조작을 위한 XACT 라이브러리 등이 속해 있습니다[5-7].

두 번째는 Layer는 코어 프레임워크 레이어로 Windows와 Xbox 360의 양쪽 플랫폼을 동일한 코드를 사용해 동작시키기 위해 필요한 레이어이다.

2차원 이미지 또는 3차원 모델 데이터를 간단히 실행시키기 위한 그래픽 라이브러리와 BGM · 효과음 등을 표현하기 위한 오디오 라이브러리, 키보드 · 마우스 · 게임 패드를 조작하기 위한 입력 라이브러리, 게임에 필요한 각도 · 좌표 · 행렬 등의 계산을 돕기 위한 수학 라이브러리, 데이터의 저장 · 읽기를 도와주는 저장 라이브러리 등이 속해 있다.

세 번째는 Layer는 확장 프레임워크 레이어로 게임 개발자가 게임의 개발에 집중할 수 있도록 준비된 레이어 이다.

윈도우 화면 · 타이머 설정 · 그래픽 장치 초기화와 같은 기본적인 어플리케이션 부품을 제공하는 어플리케이션 모델과 이미지 파일이나 3차원 모델을 프로젝트에 추가해 직관적인 콘텐츠 사용이 가능하도록 하는 콘텐츠 파이프라인 등이 속해 있다.

네 번째는 Layer는 게임 레이어로 게임 개발자가 작성한 코드 · 이미지 · 사운드 파일 등의 콘텐츠를 한데 모은 레이어 이다. XNA와 함께 설치되는 스타트 킷이나 커뮤니티로 공유 가능한 컴포넌트 등이 속해 있다.

표 1은 XNA 프레임워크를 구성하는 각 항목들에 대한 설명이다. 기존 DirectX로 게임을 개발할 때 사용한 많은 부분들이 모듈화되어 XNA 프레임워크에 구서되어 있는 것을 알 수 있다[8,9].

그림 4는 게임의 흐름에 따른 XNA 프레임워크

표 1. DirectX와 XNA 프레임워크 구성 비교

	DirectX 9	XNA Framework
화면 제어	IDirect3D9 - IDirect3DDevice9 외 다수	Microsoft.Xna.Framework.Graphics
음향 제어	IDirectSound8 - IDirectSoundBuffer8 외 다수	Microsoft.Xna.Framework.Audio
입력 제어 (마우스 등)	IDirectInput8 - IDirectInputDevice8 외 다수	Microsoft.Xna.Framework.Input
리소스 제어 (배경화면 등)	없음	Microsoft.Xna.Framework.Content
타입 제어 (Vector 등)	없음	Microsoft.Xna.Framework.Design
파일 관련 (저장, 로드 등)	없음	Microsoft.Xna.Framework.Storage

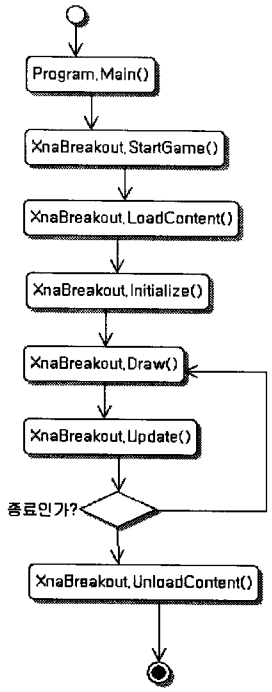


그림 4. XNA Game Main Flow

대응 함수에 대한 Flow이다.

XNA 프레임워크 기반의 게임은 main()에서 모든 것이 시작한다. 기본적인 환경에 대한 정보를 생성 및 초기화 한 후, StartGame(), LoadContent(), Initialize() 순으로 시간에 기반하여 이벤트를 감지하고 화면을 갱신하며 게임이 동작하게 된다. Initialize()는 게임의 초기화나 모든 시작 설정을 관장하고, LoadContent()는 게임에 사용되는 모든 컨텐츠(이미지, 사운드, 폰트 등)를 프로젝트 파일에 탑재한다.

Update()함수의 경우 1초에 60번 자동으로 호출된다. 처음 게임 프로젝트를 생성하면 그림 4와 같이 5개의 메소드가 나열된 소스가 자동 생성된다.

Update()는 각 프레임이 그려지기 전에 호출된다. 게임 로직을 위한 주요한 입력 처리 및 시간, 사운드 등을 처리한다. Draw()는 화면에 그려지는 부분을 담당한다.

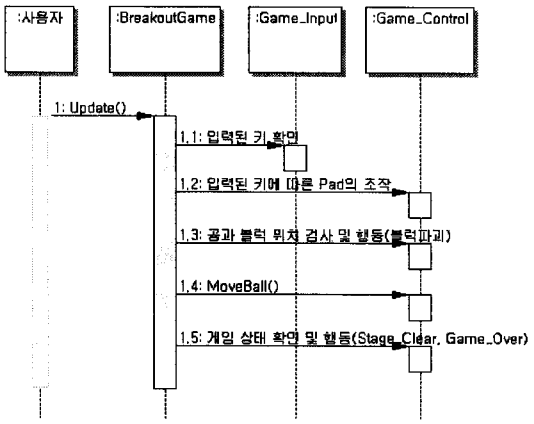


그림 5. Update Sequence Diagram

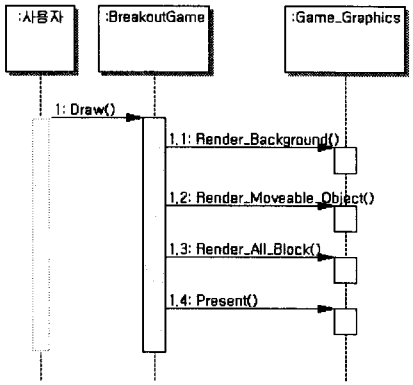


그림 6. Draw Sequence Diagram

3. 마이그레이션 접근 방법

3.1 Source Code Reverse

DirectX 기반으로 개발된 게임을 XNA 프레임워크 기반의 게임으로 마이그레이션 하는 첫 번째 단계는 Source Code Reverse이다. Source Code Reverse는 앞에서도 설명했듯이, DirectX 기반의 게임 소스에는 DirectX API와 게임 로직이 함께 존재하고 있는 형태이기 때문에, 게임 로직만을 추출하기 위해서 진행하는 작업이다.

그림 7과 같이 DirectX 기반으로 개발되는 게임 Source code는 일반적으로 3가지로 영역으로

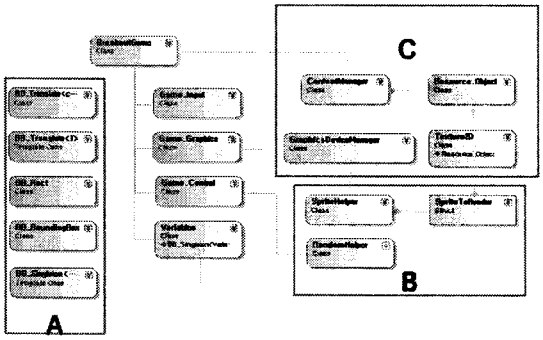


그림 7. DirectX Source Code 구조

구분해서 볼 수 있다. A 영역은 Utility, B 영역은 Helpe,r C 영역은 Framework로 이야기 하는데, 전부다 클래스로 구성된다. Source Code Reverse 에서 주 타겟이 되는 영역이 바로 C로 대부분의 게임 로직이 들어 있는 부분이다. 결국 C 영역의 C++로 개발된 Source Code를 두 번째 작업에서 C#으로 변환 후 XNA 프레임워크에 탑재하면 XNA 환경에서 동작하는 게임을 개발할 수 있게 되는 것이다.

3.2 Language Conversion

DirectX SDK와 XNA 프레임워크 환경의 차이 중 가장 중요한 포인트가 바로 언어가 다르다는 것이다. 대부분의 게임 개발자는 C++에 익숙하며, 그래서 DirectX SDK 기반의 게임 개발에 익숙하다.

그러나 XNA 프레임워크는 .Net 프레임워크 기반이기 때문에 C# 언어를 사용한다. 따라서 표2와 같이 언어의 차이를 숙지하여 게임 로직을 C++에서 C#으로 바꾸는 작업을 진행하여야 한다.

MSDN의 Language Equivalents(http://msdn

표 2. C++과 C# 언어 주요 특징 비교

특 징	관련 토픽
상속 : 오직 1개의 상속만을 허용합니다. 하지만, 다수의 인터페이스를 상속받을 수는 있습니다.	class interface
long 타입: C#에서 long 타입은 64비트입니다. C++에서는 32비트입니다.	long
struct 타입: C#에서 class와 struct는 문법적으로 다릅니다. struct는 1개의 값 타입이며, class는 참조 타입입니다.	struct class
switch 문장: C++과 다르게, C#은 1개의 케이스에서 다른 케이스로 이동하지 않습니다.	switch
delegate 타입: Delegate는 C++의 포인터와 같습니다. 그러나 타입에 대한 안정성과 보안성을 가지고 있습니다.	delegate
new를 사용하여 상속된 맴버를 명시적으로 숨길 수 있습니다.	new
전처리기 지시자는 conditional compilation로 사용합니다. 해더파일을 C#에서는 사용할 수 없습니다.	C# Preprocessor Directives
메소스 파라메터 : C#은 **ref**와 **out** 파라메터를 지원합니다. 이것은 파라메터 전달에 대해 포인터를 대신하여 사용합니다.	ref out
Strings: C# string은 C++ string과 다릅니다.	string
C#에서의 지역변수는 초기화되기 전에 사용할 수 없습니다.	5. Variables
소멸자 : C#에서 소멸자를 명시적으로 불러 사용하지 않으며, 가비지 컬렉션을 통하여 자동적으로 호출됩니다.	Destructors
생성자 : C++과 비슷하게, 만약 명시적으로 만들지 않는다면, 자동적으로 만들어 줍니다. 기본 생성자는 모든 필드(어트리뷰트)를 기본 값으로 초기화 시킵니다.	Instance Constructors Default Values Table
bit 필드를 지원하지 않습니다.	C++ Bit Fields

.microsoft.com/en-us/library/ aa293030(VS.71).aspx) 문서를 참고하면 보다 자세한 내용을 알 수 있다.

3.3 Source Code Mapping

Source Code Mapping은 C#으로 변화된 게임 로직 Source Code를 XNA 프레임워크 상에 배치하는 작업이다.

Game Studio 3.0에서 템플릿을 사용해 신규 프로젝트를 생성하면 그림 8과 같이 자동 코드가 생성이 되는데, 만들어진 소스 코드는 그리 복잡하지 않다.

하나는 Program.cs 파일이고, 나머지 하나는 Game1.cs 파일이다. Program.cs 파일은 단지 Game1.cs 파일을 호출하는데 사용한다. 언어의 엔트리 포인트가 정의된 곳이다.

Game1.cs 파일 내부가 XNA 프레임워크에서 사용자가 직접 코드를 작성해야 할 부분으로, C#

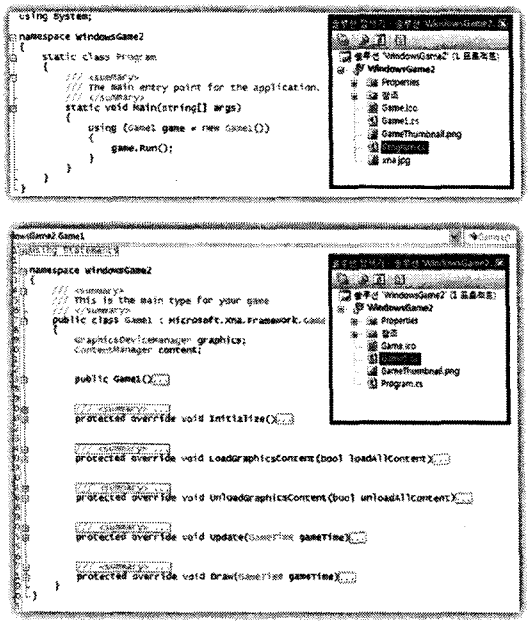


그림 8. 개발 템플릿을 선택하고 나온 자동 코드

으로 변환한 게임 로직을 이 곳에 작성해서 게임을 만들면 된다.

3.4 Debugging

마지막 단계는 게임 로직이 제대로 변환되고 XNA 프레임워크에서 동작하는지를 확인하는 단계이다.

Visual Studio Express C# 2008의 Debug 기능을 사용하여 마이그레이션 이전의 기능과 동일하게 동작하는지, 컴파일 중에 발생하는 오류는 없는지 등을 확인한다.

2D 게임의 경우 C++에서 C#으로 변환하는 과정에서의 오류가 가장 많이 발생하며, 3D 게임의 경우 DirectX와 XNA 프레임워크의 3D 좌표계가 바뀌는 부분도 Debugging 중에 반드시 확인해야 하는 부분이다.

4. 사례 연구 및 평가

4.1 사례 연구

그림 9와 같이 DirectX로 개발된 Alcanoid 게임에서 Source Code Reverse 과정을 통해 게임 로직을 추출한 후, C++로 구성 되어 있는 게임 로직 Source Code를 C#으로 변환하여, XNA 프레임워크 기반의 템플릿 프로젝트에 적용한다.

이 후에 컴파일 및 디버깅 과정을 통해 마이그레이션을 완료한 후 게임 로직이 정상적으로 작동하는지 확인한다.

그 후에 코드 절감, 수정/보완 효율성, 멀티 플랫폼 지원 등에 대한 부분에서 마이그레이션의 효과에 대하여 평가한다.

4.2 평가

(1) 코드 절감 : XNA 프레임워크에서 제공하는

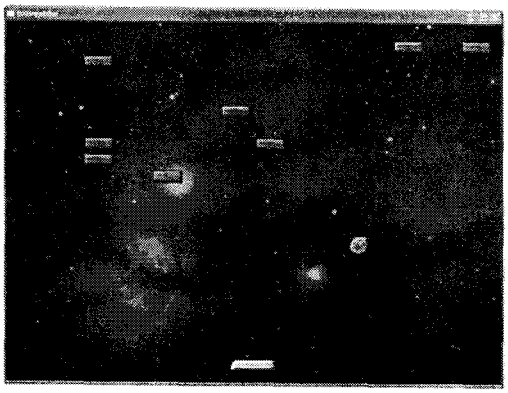


그림 9. Alcanoid 게임

만큼 DirectX 기반보다 Source 코드를 절감할 수 있다. Alcanoid 사례 연구의 경우 30% 정도 Source 코드의 절감이 있는 것을 확인할 수 있었다.

또한, 기본적인 자료구조에서 특별한 변경이 없기 때문에 혼돈이나 케스팅에 대한 위험도 없다고 할 수 있다. (예, XNA : Rectangle => DX : RECT, D3DXRECT, D3DRect)

(2) 수정 및 보완 효율성 : DirectX의 경우 개발자별로 프레임워크를 구성하기 때문에, 동일 게임 프로젝트의 경우에도 여러 형태의 프레임워크가 존재할 수 있고 개발자 변경 시 개발기간 지연 및 재작업의 문제점들이 항시 존재한다.

그러나 XNA 프레임워크를 사용할 경우 기본적으로 공통적인 프레임워크 구조를 사용하기 때문에 수정 및 보완 효율성이 높아진다고 할 수 있다.

단지 C#에 익숙하지 않은 C 또는 C++ 개발자의 경우 포인터를 마음대로 사용하지 못하고 소멸자를 명시적으로 부르지 못하는 것이 오히려 수정 및 보완 등의 작업에 있어 큰 문제점이 될 수 있다.

(3) 멀티 플랫폼 지원 : DirectX로 제작된 게임은 PC환경에 최적화 되어 있기 때문에, 멀티 플랫폼 환경에서 사용이 안 된다. 따라서 멀티 플랫폼 환경을 고려하는 경우, 최악의 상황에서는 언어 및 개발 환경을 변경하여 신규로 개발을 해야 하는 경우도 발생할 수 있다.

표 3. 평가 결과 비교

	DirectX SDK	XNA 프레임워크
코드 절감	△	○
수정 및 보완 효율성	△	○
멀티 플랫폼 지원	×	○

(높음 : ○, 보통 : △, 낮음 : ×)

그러나 XNA 프레임워크 기반으로 게임을 제작할 경우 기본적인 호환성을 보장받을 수 있기 때문에, 게임 개발사 입장에서는 개발 기간 및 비용적인 측면에서 장점으로 작용할 수 있다.

5. 결 론

최근 게임 산업 트랜드는 빠른 개발을 통한 비용 절감 및 멀티 플랫폼 환경을 지원하는 게임 개발을 목표로 한다. 그러나 DirectX 기반으로 개발된 게임의 경우 PC 환경에 최적화 되어 있기 때문에 멀티 플랫폼 환경을 지원하기 위해서는 수정 및 변경, 최악의 경우는 재개발이라는 위험(Risk)을 감수해야 하는 상황이다.

XNA 프레임워크 환경의 게임 개발은 이러한 최근 게임 산업 트랜드에 맞는 게임 개발 환경이다. 따라서 본 논문에서 제시한 DirectX 기반의 게임을 XNA 프레임워크 환경으로 마이그레이션 하는 과정을 통해 DirectX 기반의 게임을 안정적으로 멀티 플랫폼 환경을 지원하는 게임으로 변환할 수 있다. 또한 XNA 프레임워크를 통해 개발의 표준화 및 생산성, 효율성 등의 향상을 확인 할 수 있었다.

향후 연구에서는 마이그레이션 후 게임 클라이언트가 어느 정도의 성능을 유지하고 있는지에 대한 평가 및 기존 Code의 재사용성에 대한 방안, 자동화 할 수 있는 Tool의 설계에 대하여 보다 구

체적으로 목표를 세분화 하여 연구를 할 계획이다.

참 고 문 헌

[1] 한용희, "MS의 게임 개발 플랫폼 XNA 실전분석," 월간 마이크로소프트웨어 2007년 8월호.

[2] 이승훈, "XNA 게임개발 플랫폼," 한국게임학회, pp. 30-35, 2008년.

[3] Chad Carter, "Microsoft XNA usleashed: graphics and game programming for Xbox 360 and windows," SAMS, 2008.

[4] Benjamin Nitschke, "Professional XNA game programming : for Xbox 360 and Windows," Wiley Technology Pub., 2007.

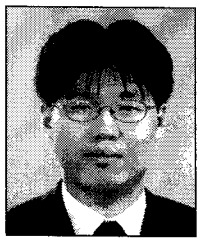
[5] 한용희, "MS의 게임 개발 플랫폼 XNA 실전분석," 월간 마이크로소프트웨어 2007년 8월호.

[6] Riemer Grootjans, "XNA 2.0 Game Programming Recipes," Apress, 2008.

[7] http://creators.xna.com/

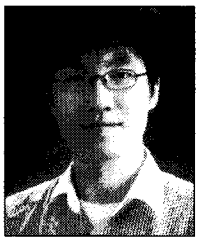
[8] http://www.xnatutoral.com/

[9] http://en.wikipedia.org/wiki/Microsoft_XNA



이 승 훈

- 2002년 숭실대학교 컴퓨터학부 (공학사)
- 2005년 숭실대학교 대학원 컴퓨터학과 (공학석사)
- 2007년 숭실대학교 대학원 컴퓨터학과 (박사수료)
- 2006년~현재 한국게임개발자협회 이사
- 2008년~현재 마이크로소프트 XNA/DirectX MVP
- 2009년~현재 동국대학교 게임멀티미디어공학과 겸임교수
- 관심분야 : 게임개발 프로세스, 소프트웨어 품질평가, 게임 프레임워크, 소프트웨어 유지보수, 소프트웨어 재사용 등



변 정 원

- 2001년~2007년 숭실대학교 미디어학부 학사
- 2008년~현재 숭실대학교 대학원 컴퓨터학과 석사과정
- 관심분야 : 소프트웨어 아키텍처/프레임워크, 소프트웨어 요구공학, 게임 개발 프로세스

류 성 열

- 1981년~현재 숭실대학교 교수
- 1997년~1998년 George Mason University 객원 교수
- 1998년~2001년 숭실대학교 정보과학대학원 원장
- 2004년~현재 한국품질재단 운영위원회 위원장
- 2006년~현재 공정거래위원회 성과관리위원회위원
- 2008년~현재 정보통신연구진흥원 비상임 이사
- 관심분야 : 소프트웨어 유지보수, 게임 개발 프로세스, 오픈소프 소프트웨어 등