제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

# 스크립트 언어를 이용한 게임 개발

최한용* 정진영**
경희대학교 컴퓨터공학과
골든벨엔터테인먼트(주)
e-mail:hychoi2@khu.ac.kr

# Game Development Using Script Language

Han-Yong Choi*, Jin-Young Jung**
*Dept of Computer Engineering, KyungHee University
**GoldenBell Entertainment Co.

요 약

엔진을 도입하여 개발할 경우 단지 하부 구조를 구현 하지 않았을 뿐, 엔진 내용을 이해할 정도의 수준이 되어야 한다는 것이다. 그러므로 게임 업계에서는 양질의 게임을 개발하기 위해서 여전히 고급 프로그래머를 필요로 하고 있으며, 이러한 부분은 여전히 개발비 상승에 영향을 미치고 있다. 그러므로 프로그램을 이해하는 기획자 수준에서도 게임을 개발할 수 있는 좀 더 상위 개념의 개발 방법의 필요성이 대두되고 있다. 본 논문에서는 이러한 문제를 해결하기 위해 스크립트를 이용하여 게임게임 전체 관리 부분, 모델의 AI 부분, 카메라 조작 부분으로 게임을 구성할 수 있도록 하였다. 그리고 스크립트 언어는 2 계층의 구조를 갖고 있으며, 하부 계층은 직접적으로 엔진을 제어하게 되어있다. 그러나 스크립트의 목표는 직접적인 엔진 제어를 줄이는 추상화작업을 목표로 하기 때문에 이를 추상화한 상부계층의 스크립트 언어를 설계하여 하부계층의 엔진제어 모듈로 변환하도록 하였다. 따라서 본 논문에서는 게임엔진을 직접적으로 모델링하지 않고 개발하기 위한 추상화된 상위 단계의 스크립트 언어를 이용하여 기획단계에서 게임을 구성해 볼 수 있으며,직접적으로 하부구조의 엔진 구현 및 엔진의 내용을 잘 이해할 정도의 수준이 아니더라도 스크립트언어를 이용하여 게임을 개발할 수 있다.

## 1. 서론

하드웨어 발달로 그래픽 표현력이 높아지고, 2D에서3D 게임이 주류가 이동하면서 게임의 개발 난이도와 비용 등이 계속해서 증가하고 있다. 이러한 문제를 해결하기 위해 반복적으로 사용되는 모듈의 재사용과 생산성을 높이기 위한 툴 도입을 위한 노력이 게임 개발 업계에서도 끊임없이 이루어지고 있다[1]. 가장 대표적인 예로 3D 게임엔진의 도입이라고 할 수 있는데, 제작사에서 자체 3D 엔진파트를 제작하는 경우도 있지만, 엔진 개발에 드는 비용 대비 효과가 떨어지기 때문에 전문 엔진 개발 업체의 상용 엔진을 쓰는 것이 일반적인 추세가 되고 있다[2]. 그러나 3D 게임 엔진을 도입한다고 해서, 3D 프로그래밍을 몰라도 게임 개발을 할 수 있는 것이 아니다. 어떤 엔진 업체의 경우에는 자사의 엔진을 사용할 수 있는 정도의 기술 수준이 되지 않으면 라이센스를 해주지 않는 다고 할 정도로, 엔진을 사용하여 개발하는 것 자체도 단지 하부 구조를 구현 하지 않았을 뿐, 엔진 내용을 이해할 정도의 수준이 되어야 한다는 것이다. 그러므로 게임 업계에서는 양질의 게임을 개발하기 위해서 여전히 고급 프로그래머를 필요로 하고 있으며, 이러한 부분은 여전히 개발비 상승에 영향을 미치고 있다. 그러므로 프로그램을 이해하는 기획자 수준에서도 게임을 개발할 수 있는 좀 더 상위 개념의 개발 툴의 필요성이 대두되고 있으며, 비슷한 장르의 게임을 지속적으로 만들어 낼 수 있는 툴을 제공하여 또한, 차후에는 엔진과 독립성을 유지하여, 개발 프로덕트의 최종 품질 및 개발비용 등에 관련하여 사용할 엔진을 교체한다고 하더라도 개발 환경은 동일하게 가져갈 수 있도록 한다[3][4].

본 논문에서는 C언어의 문법을 따르고 있는 스크립트를 이용하여 게임 내에서 사용되는 오브젝트별로 각각 코드를 작성하도록 하였으며, 게임 전체 관리 부분, 모델의 AI 부분, 카메라 조작 부분으로 구성하였다. 그리고 스크립트 언어는 2 계층의 구조를 갖도록 되어있다. 하부 계층은 직접적으로 엔진을 제어하게 되어있다. 그러나 본 논문의 목표는 직접적인 엔진 제어를 줄이는 추상화작업을 목표로 하기 때문에 이를 추상화한 상부계층의 스크립트 언어를 설계하게 되었다. 상부계층의 스크립트언어는 스캐닝과 파서, 코드 최적화 등의 작업을 거쳐 하부계층의 엔진 제어 모듈로 변환되도록 되어있다. 테스트에 사용된 VM은 애니

메이션, 컬링시스템, 라이트 맵, 빌보드, 파티클 시스템의 기능을 갖고 있다[5]. 또한 스크립트를 이용한 게임개발에 의해 엔진은 각 플랫폼별로 제작되고, 이 엔진을 구동할 수 있는 스크립트를 모든 플랫폼에 동일하게 제공하여 한번 프로그래밍을 해 놓으면 해당하는 타겟에 빌드하여 쉽게 플랫폼 별 이식을 할 수 있는 통합개발환경을 구축하였다. 통합개발환경은 엔진의 이해도가 낮아도 간단한 스크립트의 문법정도를 익힘으로써 사용자가 쉽게 게임을 개발할 수 있는 환경을 제공 하도록 하고 있다. 디버깅은 스크립트 언어를 디버그 모드로 컴파일 하면, 목적 코드에는 스크립트 정보가 함께 저장되며, 이 정보는 스크립트 언어 수준에서 실시간 디버그에 사용하도록 하였다. 그리고 디버깅 중 게임 데이터는 실시간으로 빠르게 변하는 속성이 있어서, 일반적인 화면 표시로는 눈으로 추적하기 어렵기 때문에 추적하고자 하는 변수의 범위를 설정하여, 범위를 벗어나는 순간에 경고(Warning)를 표시하도록 하였다

따라서 본 논문에서는 게임엔진을 직접적으로 모델링하지 않고 개발하기 위한 추상화된 상위 단계의 스크립트 언어를 이용하여 직접적으로 하부구조의 엔진 구현 및 엔진의 내용을 잘 이해할 정도의 수준이 아니더라도 스크립트언어를 이용하여 게임을 개발할 수 있도록 하였다.

## 2. 스크립트언어의 구성

### 2.1 구성요소

본 논문의 스크립트 언어는 그림 2.1과 같이 게임 전체 관리 부분, 모델의 AI 부분, 카메라 조작 부분의 3가지 스크립트 구성요소로 설계하였다.

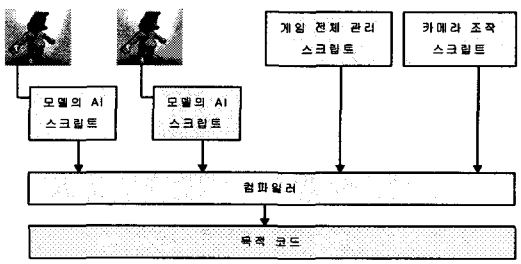


그림 2.1 스크립트 언어의 3가지 요소

가. 게임 전체 관리 부분

전체적인 게임의 흐름(flow)을 결정하고 게임 제작 전반에 관련한 변수들을 관리하는 부분이다. 게임의 내용을 적재(load)하고 저장(save)하는 기능을 갖는다.

나. 모델의 AI 부분

게임에 등장하는 모든 캐릭터의 AI 부분을 조작하는 스크립트 부분이다. 캐릭터의 AI부분은 각 캐릭터마다 따로 프로그래밍을 할 수 있도록 UI가 제공되는데, 게임 전체 관리 부분과 연동되어 게임 전체 상황 따라 캐릭터 역시 여러 가지 상황을 만들어 낼 수 있다.

다. 카메라 조작 부분

게임 전체와 모델들의 상황을 독자적으로 판단하여 적절한 카메라 뷰(view)를 만들어 내는 스크립트 부분이다

### 2.2 스크립트 언어의 작성

스크립트는 그림 2.2와 같이 두 가지 방법으로 작성할 수 있다. 스크립트는 각 개체(object)의 동작 스크립트를 편집기(editor)로 편집하여 작성되기도 하며, 간단한 마법사(wizard) 형식으로 자동 생성할 수 있다.

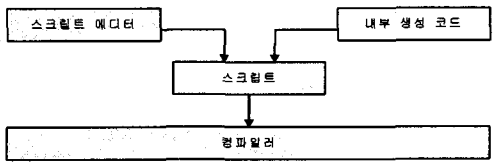


그림 2.2 스크립트 편집방법

### 2.3 스크립트의 추상화

IDE 환경을 통해 작성되는 스크립트는 컴파일 과정을 거쳐 목적 코드인 게임 전용 코드로 만들어진다. 목적코드에 해당하는 게임전용코드는 직접적으로 엔진을 조작하도록 되어있다. 따라서 그림 2.3와 같이 스크립트 언어는 사용자의 개발편의성을 고려하여 이 목적코드를 추상화한 것이다. 그러므로 게임엔진을 잘 이해하고 있다면 게임엔진 단계에서 직접적으로 제어할 수 있으며, 그렇지 못한 개발자의 경우는 엔진을 추상화한 개발환경에서 스크립트 언어를 이용하여 게임을 개발할 수 있다.

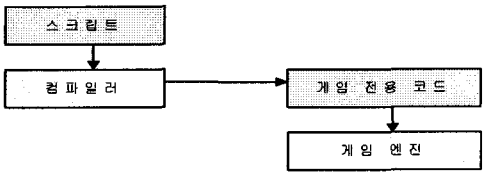


그림 2.3 2차 구조의 스크립트 추상화

게임 전용 코드는 게임 엔진을 구동하여 게임에 관련된 실제 작업을 엔진의 기능을 사용하여 수행하게 된다.

## 3. 스크립트의 어휘와 구문

가. 스크립트에서 사용하는 키워드와 특수 심볼은 다음과 같다.

int, string, else, if, return, void, while, for

+,-,*,/,++,--,&&,||,<,<=,>,>=,==,!=,;,,,(,),[,],{,},/*,*/

나. 이외의 토큰은 변수(var)와 숫자(number)가 있으며 다음과 같은 정규표현식으로 정의된다.

VAR = letter letter*
NUM = digit digit*
letter = a|..|z|A|..|Z
digit = 0|..|9

다. 공백 문자(white space)는 빈자리 문자(blank)와 줄바꿈 문자(new line)와 탭 문자(tab)가 있다. 공백 문자는 무시되며 키워드는 공백 문자로 구분된다.

라. 스크립트의 구문과 의미구조

본 논문에서 추상화한 스크립트의 구문과 의미 구조는 다음과 같이 BNF문법으로 정의할 수 있다.

```
1. program → declaration-list
2. declaration-list → declaration-list declaration | declaration
3. declaration → car-declaration | fun-declaration
4. var-declaration → type-specific VAR; | type-specific
VAR[NUM];
5. type-specifier → int | string | void
6. fun-declaration → type-specific VAR(params)
compound-stmt
7. params → param-list | void
8. param-list → param-list, param | param
9. param → type-specifier VAR | type-specifier VAR [ ]
10. compound-stmt → { local-declarations statement-list }
11. local-declaration → local-declarations var-declaration |
empty
12. statement-list → statement-list statement | empty
13. statement → expression-stmt | compound-stmt |
selection-stmt | iteration-stmt |
return-stmt
14. expression-stmt → expression ; | ;
15. selection-stmt → if ( expression ) statement |
if ( expression ) statement else
statement
16. iteration-stmt → while ( expression ) statement |
for (expression ; simple-expression ;
simple-expression) statement
17. return-stmt → return; | return expression ;
18. expression → var = expression | simple-expression
19. var → VAR | VAR [ expression ]
20. simple-expression → additive-expression relop additive
expression |
additive-expression
21. relop → <= | < | > | >= | == | !=
22. additive-expression → additive-expression addop term |
term
23. addop → + | -
24. term → term mulop factor | factor
25. mulop → * | /
26. factor → ( expression ) | var | call | NUM
27. call → VAR(args)
28. args → arg-list | empty
29. arg-list → arg-list, expression | expression
```

## 4. 개발프로세스

### 4.1 통합개발 환경

통합 개발 환경은 그림 4.14과 같이 게임 엔진을 이용하여 게임을 개발하는 프로세스 상에서, 기획에서 나온 스토리보드 상의 내용을 게임 로직에 적절히 적용할 수 있도록 통합적인 개발 시스템을 제공하여 복잡한 게임 개발 프로젝트의 제작 관리가 쉬워진다. 스크립트 에디터와 리소스 관리 및 파티클을 비롯한 게임 정보를 구성할 수 있도록 하였다. 그리고 스크립트를 이용하여 스켈레톤 게임을 구성할 수 있으므로, 게임개발 작업에 착수하기 전에 기획단계에서 구성된 시나리오에 의해 게임이 어떻게 보여질 수 있을지 데모게임을 작성하여 시나리오를 검토할 수 있으며 기획자와 개발자간의 의사교환 방법으로 이용할 수 있다.

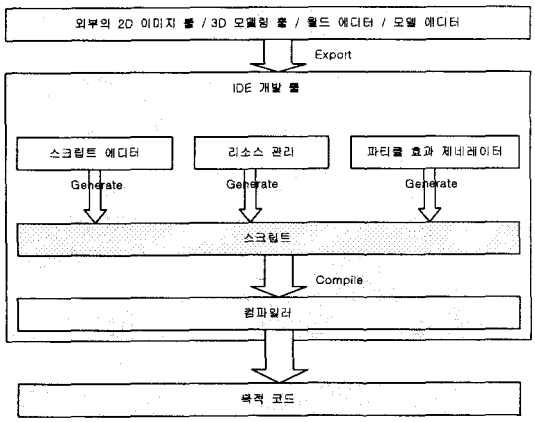


그림 4.1 개발 프로세스

### 4.2 리소스 관리

IDE 툴의 좌측 창은 리소스 관리 창으로서 게임용 리소스 편집을 위한 외부 에디터와 연결되어 있다. 따라서 리소스와 IDE의 개발과정은 동적링크에 의해 리소스를 관리할 수 있다. 그림 4.3는 리소스에서 제공하고 있는 정보를 스크립트로 조합하여 모델을 생성한 예를 보이고 있다.

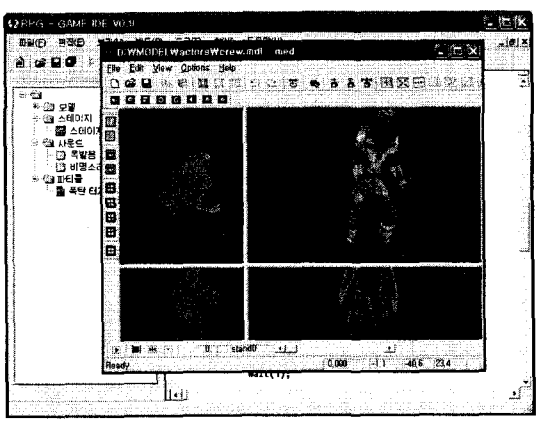

그림 4.2 모델 리소스 에디터

### 4.3 게임 구성

IDE 툴은 각 모델에 대해 고유의 스크립트를 독립적으로 제공한다. 각 모델에 대해 게임 루프를 가질 수 있으며, 모델은 독자적으로 편집이 가능하다.

스크립트에 의해 관리되는 모델은 컴파일 되는 목적 코드 상에서 게임 루프를 그림 4.3과 같이 구성하고 있으며, 이 때 모델의 AI 파트는 모델의 숫자가 늘어나는 만큼 리스트로 추가되어 관리된다. 이때 AI는 그림 4.4와 같이 모델에 사용할 AI를 스크립트로 제어할 수 있으므로 FSM(Finite State Machine), FuSM(Fuzzy State Machine), Decision Tree 로 구성 되어있는 엔진레벨의 AI 루틴을 간단하게 제어할 수 있다. 따라서 모델의 AI 파트 정보는 카메라 조작파트의 속성을 이어받도록 하였다.

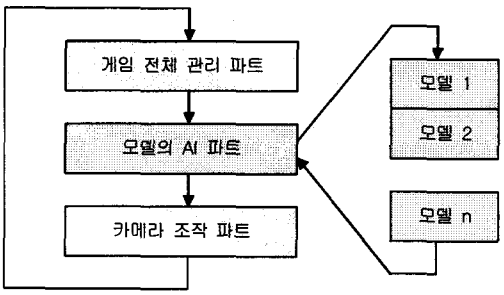


그림 4.3 게임 구성

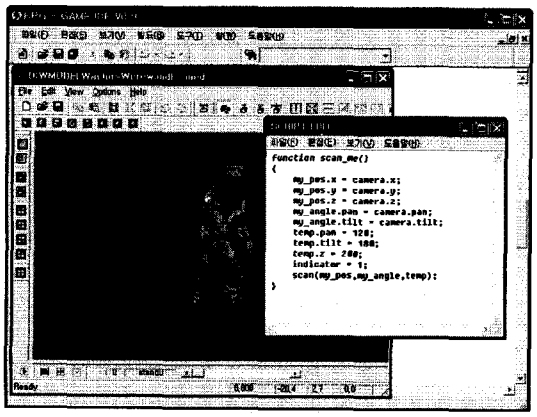

그림 4.4 모델의 AI 루틴

## 5. 결론

일반적으로 고가의 게임 엔진(수천만원 ~ 수억원대)을 도입한다면, 생산성과 개발 편의성은 크게 향상시킬 수 있으나, 엔진 자체의 라이센스 비용이 수억원에 달하는 등 개발비용에 큰 부담을 준다는 단점이 있다. 그리고 저가의 게임 엔진의 경우(수십만원대 ~ 수백만원대)에는 통합적으로 개발을 할 수 있는 IDE 환경이 없으며, 자체 스크립트 편집도구 조차 없는 경우가 대부분이라 일반 개발 환경에 비해 편의성이 열악해진다는 단점이 있다. 따라서 기존의 개발방법 중 엔진을 도입한 개발방법에서 나타나는 개발자의 숙련도 문제 그리고 기획자와 개발자간의 의사교환문제가 있었다. 본 논문에서는 게임개발에서 소요되는 노력 중 엔진개발이후 엔진을 활용한 게임개발 교육에 소요되는 노력을 줄이고, 엔진도입 시점에서 유사게임을 빠르게 개발하려는데 목적이 있으며 또한 개발자와 기획자간의 의사교환을 위해 개발이전에 엔진을 이용하여 게임을 정의해 볼 수 있었다. 스크립트를 이용하여 엔진을 제어할 경우 표 5.1과 같이 생산성을 향상시키기 위한 요소로써, 본 과제에서 개발한 스크립트를 활용한 IDE는 엔진을 정확히 이해하지 못한 초급 개발자들도 쉽게 적응할 수 있는 수준의 쉬운 스크립트를 제공하여 게임을 개발할 수 있으며, 기획자가 의도한 바를 스크립트 조합으로 개발단계의 스켈레톤 게임을 구성하여 개발자와 검토하기 위한 의사교환 방법으로 이용할 수 있다. 따라서 스크립트를 이용한 게임 개발 방법은 게임개발 라이프사이클을 빠르게 할 수 있으며 생산성을 향상시킬 수 있다.

표 5.1 IDE 생산성

| | 엔진제어 | 스크립팅 |
|---|---|---|
| 엔진용 코드개발 | 엔진의 이해도가 높아야 엔진용 코드를 작성할 수 있다. | 상위 스크립트만 이해하면 엔진용 코드는 컴파일러를 통해서 생성한다. |
| 개발자 요구수준 | 중/고급 프로그래머 | 초/중급 프로그래머 |
| 개발기간 | 엔진을 이해하는데 걸리는 시간과 게임 전체를 설계해야하는 부담이 있다. | 장르에 따른 기본적인 프레임 구조를 가지고 있어서 빠르게 개발할 수 있다. |
| 개발환경 | 고가의 엔진이 아닌 경우에는 개발 환경의 수준이 많이 떨어지고, 개발자들이 적응하기가 어렵다. | 통합 IDE 환경에서 제공되는 쉬운 스크립트를 사용하여 쉽게 게임을 개발할 수 있다. |

### 참고문헌

[1] 가상현실연구부, 온라인 3D 게임엔진 표준화 연구, 최종연구보고서, 2001.11, 한국전자통신연구원.

[2] Michael Lewis, Jeffrey Jacobson, "Game Engines in Scientific Research," Communications of the ACM, Jan. 2002, Vol 45. No, 2002.

[3] Steve Rabin, AI Game Programming Wisdom, Charles Rivers Media, 2002.

[4] John E.Laird, "Using a Computer Game to Develop Advanced AI," IEEE Computer, July 2001, pp.70-75, 2001.

[5] Steven Woodcock, "Game AI: the state of the Industry," Game Developer, Aug. 2000, pp24-28, 2000.