한국게임학회 논문지
제6권 제4호 2006년 12월

게임디자인에서 게임규칙 표현방법 조사연구

장희동
호서대학교 게임공학과
dooly@office.hoseo.ac.kr

A Survey of Representation Methods of Game Rules in Game Design

Chang, Hee Dong
Dept. of Game Engineering, Hoseo University

요 약

게임개발에서 설계내용은 디자인단계에서 뿐 아니라 구현단계와 테스트단계까지 자주 변경이 이루어진다. 게임의 설계내용은 게임규칙과 콘텐츠의 설계내용으로 이루어진다. 그 중에서 게임규칙의 설계 내용은, 모든 개발참여자들이 쉽고 정확하게 이해할 수 있어야 하고 자주 이루어지는 변경들이 효율적으로 관리되어야 하며 그리고 정확한 검증이 이루어져야 한다. 본 연구는 게임규칙의 설계내용에 대해, 게임디자인에서 적합하게 될 수 있는 표현방식을 찾기 위한 조사연구로서, 문서표현방식, UML 표현방식, 페트리네트 표현방식, 스크립트언어 표현방식에 대해 비교분석을 하였다. 비교분석은 게임규칙의 표현범위, 비주얼적 표현능력, 논리적 표현능력, 자동화된 검증 가능성, 그리고 효율적 형상관리 가능성에 대하여 이루어졌다. 비교분석결과 UML 표기방식이 가장 적합하였다. 그러나 UML 표기방식은 보다 편리한 자동화된 검증 방법의 연구개발이 필요한 것으로 판단되었다.

ABSTRACT

In game developments, the design results are often modified not only in the design phase but also in the implementation and test phases. The results of game design are consisted of the results of game rule design and the results of game contents design. The results of game rule design should be correctly understood to all the participants, be efficiently managed by the given configuration controls, and be accurately verified. In this study, we carry out a survey of representation methods of game rules in game design. We have the comparison analysis of the written representation, the UML representation, the Petri net representation, and script-language representation methods about the suitability of the representation method for game rule designs. The comparison analysis is about the representation scope, the visual representation, the automated verification, and the configuration management. The analysis results show that the UML representation is the best method but it needs more convenient automated verification method.

Keyword : Game Rules, UML, Petri Net, Script Languages

1. 서 론

게임개발은 실시간처리용 DB응용 소프트웨어 개발과 매우 비슷하면서 또한 엔터테인먼트 콘텐츠 개발이 합쳐진 경우이다[1]. 즉, 게임개발은 요구사항에 만족하는 기능, 동작, 인터페이스를 가진 시스템을 설계하고 구현해야 할 뿐 아니라 추상적인 게임 오락성을 플레이어가 실제 느낄 수 있도록 구현해야 한다. 이러한 게임개발 특성상, 게임디자인의 설계내용은 게임디자인단계에서 80%가 결정되고 그 후 구현단계와 테스트단계에서 나머지 20%가 결정된다[1]. 이는 게임개발에서 설계내용이, 게임디자인단계 뿐 아니라 구현단계와 테스트단계에서도, 자주변경 된다는 것을 의미한다.

게임디자인의 설계내용은 크게 게임 규칙과 게임 콘텐츠의 설계내용으로 나눌 수 있다. 설계내용의 검증에서 게임 콘텐츠의 설계내용은 문서검토나 또는 스토리보드 검토를 통하여 주관적인 관점에서 검증이 이루어지는 반면 게임규칙에 대한 설계내용은 문서검토 뿐 아니라 객관적인 테스트를 통해 정확한 검증이 이루어져야 한다.

따라서 효율적인 게임개발을 위해서는 게임규칙 설계내용의 변경에 대하여 신속한 검증이 이루어질 수 있는 게임규칙의 표현방식이 필요하다. 또한 이러한 표현방식은 모든 개발자들이 정확하고 쉽게 이해할 수 있어야 한다.

본 논문은, 게임디자인에 적합한 게임규칙의 표현방식에 대한 조사연구를 목적으로 한다.

게임규칙 표현방식에 대한 기존 연구들은 다음과 같다. 가장 전통적으로 게임규칙을 표현하는 방식은 글, 그림, 표를 사용한 문서표현(written representation) 방식이다[2, 7]. 이 방식은 게임규칙의 설계내용을 문서를 통해 모든 개발참여자들과 정보를 공유하기 위해 사용된다. 또한 게임규칙을 UML(Unified Modeling Language)[3]로 표현하는 방식이 연구되었다[4, 5]. 그 중 한 연구는 체스 보드게임에서 사용되는 말(예: 킹, 비숍, 나이트)들의 도달성(reachability) 규칙의 관계를 표기하기 위해 UML의 클래스다이어그램(class-diagram)을 사용하였다[4]. 또 한 연구에서는 게임소프트웨어의 설계내용에 대한 원활한 의사소통과 효율적인 구현을 위해 UML 다이어그램들로 표기하는 방식들이 소개되었다[2,5]. 그리고 게임규칙을 정보흐름의 분석모델인 페트리네트(Petri net)로 표현하는 방법이 연구되었다[6]. 이 연구는 설계내용을 시뮬레이션을 통한 검증을 위해, D&D(Dungeon and Dragon)[7]의 게임규칙을 페트리네트로 표현하였다. 그리고 또 다른 연구 활동으로, 게임의 규칙을 스크립트 언어(script language)로 표현방식이 연구되었다[8, 9]. 이 연구는 게임규칙의 설계내용을 즉시 실행할 있는 환경을 제공하기 위한 스크립트언어이다.

본 연구는 기존의 게임규칙 표현방식을 조사하여 그 특성들을 비교분석해서 게임디자인에서 사용 적합성에 대해 분석하는 것을 목표로 한다.

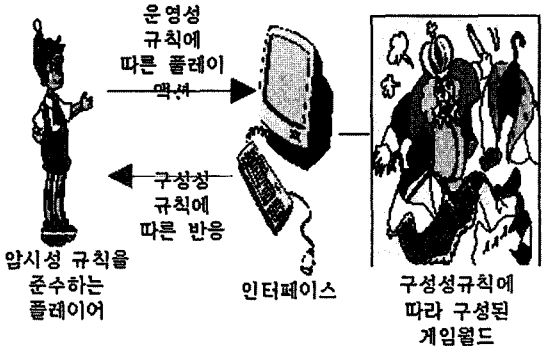
본 논문은 2장에서는 게임규칙의 개념과, 3장에서는 게임규칙 표현방법들을 소개하고, 4장에서는 게임규칙 표현방법들을 비교분석한 후, 5장에서 결론을 내린다.

2. 게임규칙의 개념

게임규칙(game rule)은 게임이 어떻게 동작하는지를 설명하는 추상적인 지침들의 고정된 집합으로 게임의 형식적인 구조를 나타낸다[10]. 게임규칙은 플레이어 액션(action)을 제한하고, 구체적이고 명백하며, 모든 플레이어들과 공유되며, 변하지 않고 구속력이 있으며, 반복가능하다. 게임규칙의 종류는, 〈그림 1〉과 같이, 운영성(運營性) 규칙(operational rules), 구성성(構成性)규칙(constitutive rule), 암시성(暗示性) 규칙(implicit rules)으로 나눌 수 있다[10]. 운영성 규칙은 플레이어가 게임을 어떻게 플레이해야 하는지에 대한 지침들이고, 구성성 규칙은 논리적이고 수학적인 게임의 형식적인 구조를 만들어 내는 내부적 메카닉스(mechanics)이다. 이 규칙은 플레이어의 액션에 대해 게임월드가 어떻게 구성되고 반응해야 하는지를 규정한다. 암시성 규칙은 플레이어가 서로 기분 상하지 않고 원활하게 플레이하기 위해 필요한 지침으로 묵시적으로 시행되는 에티켓을 위한 규칙이다.

게임규칙에 의해 결정되는 형식적인 게임구조는 게임메카닉스를 결정하여 게임소프트웨어 설계를 위한 기술게임디자인(technical game design)의 핵심내용을 제공한다. 따라서 게임개발에서 게임규칙의 내용은 모든 개발참여자들이 공통적으로 이해해야 하며 게임디자인 단계에서 결정되고 게임구현단계와 테스트단계에서 수정된다. 일반적으로 게임규칙의 내용은 분기/조건/반복을 포함하는 진행절차,

게임플레이 상태를 판정하는 판정표 또는 수학공식, 게임 구성요소의 특성을 나타내는 매개변수들과 게임플레이 상황에 따른 매개변수 값들의 갱신내역들이다.



[그림1] 게임규칙에 의한 형식적인 게임구조

그러므로 게임디자인에서 적절한 게임규칙의 표현방식은 다음과 같은 요구조건들이 만족되어야 한다.

(1) 게임개발참여자들이 쉽게 이해할 수 있고 표현할 수 있는 비주얼적 표현(visual representation)

(2) 작성자의 성향에 영향을 받지 않고 게임규칙의 내용을 정확히 나타낼 수 있는 논리적 표현(logical representation)[1]

(3) 게임규칙의 설계내용의 무결성과 밸런스에 대한 자동화된 검증(automatic verification)

(4) 게임규칙의 설계내용의 효율적 형상 관리(efficient configuration management)

3. 게임규칙 표현방식들

본 연구를 통해 조사된 게임규칙의 표현방식들은 문서표현(written representation)방식, UML(unified modeling language) 표현방식, 페트리네트(Petri net) 표현방식, 스크립트 언어(script language) 표현방식들이 있다. 이들을 자세히 소개하면 다음과 같다.

3.1 문서(Written Representation) 표현방식

문서표현방식은 게임규칙을 표현하는 전통적인 방법으로, 〈표 1,2〉과 같이, 글, 표, 그림들을 통해 문서로 표현하는 것이다. 이 표현방식의 주 목적은 게임규칙의 설계내용을, 문서를 통해, 개발참여자들과 함께 공유하기 위함이다.

문서표현방식은 게임규칙을 작성할 때 그림과 표 중심으로 사용하면 비주얼하게 게임규칙을 표현할 수 있다. 게임규칙의, 객관적이고 정확하게 표현하는, 논리적인 표현품질은 작성자에 따라 결정된다. 그리고 문서로 표현된 게임규칙의 검증은 직접 컴퓨터로 자동화할 수 없다. 또한 게임규칙 설계내용의 변경 통제를 위한 형상관리는 주로 검토자의 문서검토를 통하여 이루어지기 때문에 정확하지 않고 비효율적이다.

전투 진행표
1. 행동선언: DM(Dungeon Master)이 모든 플레이어에게 각자의 캐릭터가 그 라운드 동안에 어떤 행동을 할 것인지 물어본다
2. 우선권(Initiative): 양쪽(PC일행과 몬스터)이 육면체 주사위를 1번 굴려 높은 쪽이 먼저 행동한다.
3.행동, 우선권을 갖는 쪽부터:
a)사기 체크: 필요한 경우에 사기 체크를 한다. (던전마스터 규칙책 25쪽 참조)
b)이동 (후퇴, 철수와 같은 특수이동도 포함. 플레이어 규칙책 69쪽 참조)
c)장거리 공격
d)마법 주문과 마법 아이템 사용
e) 근접전 공격
4. 행동, 우선권이 없는 쪽: 위의 3번과 같은 순서로 진행시킨다
5. 결과:
a)몬스터가 모두 죽으면 전투가 끝나고 한 턴이 지난다. '탐험의 진행표'로 돌아간다.
b)한쪽이 도망가면, 다른 쪽이 추격을 시도할 수도 있다. (던전마스터 21쪽 참조)
c) 전투가 계속되면, 1번으로 돌아가서 되풀이 한다

[표1] D&D[7] 던전마스터 규칙의 전투진행표

갑옷 방어도	
갑옷 종류	방어도
갑옷 없음	9
가죽 갑옷	7
체인 갑옷	5
판금 갑옷	3
방패	방어도 1점 보너스

[표2] D&D[7] 플레이어 규칙의 갑옷방어도

3.2 UML(Unified Modeling Language) 표현방식

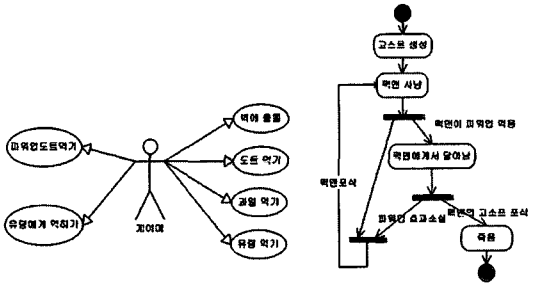
UML은 소프트웨어에 대한 내용들을 명시, 개발, 문서화를 위한 목적으로 개발된 객체지향적 모델링 언어이며 국제산업표준으로 사용된다[3]. UML은 총 8개 다이어그램,

[1] 논리적 신택스(syntax)들을 통해 표현하는 방법을 의미함

즉, 유스케이스 다이어그램(use case diagram), 클래스 다이어그램(class diagram), 시퀀스 다이어그램(sequence diagram), 협력 다이어그램(collaboration diagram), 상태 다이어그램(state diagram), 활동 다이어그램(activity diagram), 컴포넌트 다이어그램(component diagram), 배치 다이어그램(deployment diagram)들로 소프트웨어 시스템을 표현하는 비주얼적 모델링언어이다.

UML은 시스템에 대하여 다양한 이해관계자들의 관점의 모델링을 제공한다. 즉, 고객의 관점에서 모델링은 유스케이스 다이어그램이고, 분석 및 설계자의 관점에서 모델링은 클래스 다이어그램, 시퀀스 다이어그램, 협력다이어그램, 상태 다이어그램, 그리고 활동다이어그램이며 구현자 관점에서 모델링되는 것은 컴포넌트 다이어그램과 배치 다이어그램이다.

게임규칙의 UML 표현방식은, 〈그림 2〉와 같이, 운영성규칙(암시성규칙포함)은 유스케이스 다이어그램으로 표현되고 구성성규칙은 분석 및 설계 관점의 모델링 다이어그램들로 표현된다.

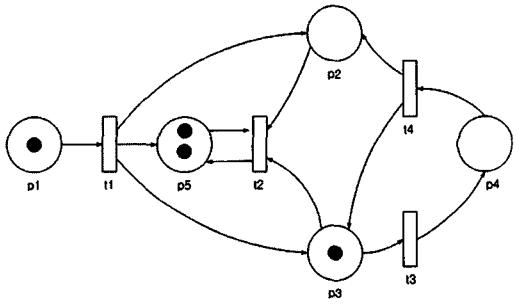


[그림2] 팩맨의 게임오브젝트 상호작용의 유스케이스 다이어그램(좌) 고스트 NPC의 상태에 대한 활동다이어그램(우)

UML 표현방식은 게임규칙을 다이어그램으로 나타내기 때문에 비주얼하게 표현할 수 있다. 그래서 모든 개발참여자들이 정확하게 그 내용을 이해 할 수 있다. 또한 UML의 다이어그램의 표현방식은 논리적인 표현을 요구한다. 따라서 UML 표현방식은 작성자의 성향에 따라 영향을 받지 않고 게임규칙 내용들이 정확하게 표현되기 때문에 개발과정에서 생길 수 있는 개발참여자들 간의 의사소통의 불일치를 해소할 수 있다. 게임규칙에 대한 UML 모델들의 자동화된 검증은 CASE(Computer Aided Systems Engineering) 툴을 사용하여 가능은 하지만 소프트웨어적인 상세한 명세가 이루어져야 하기 때문에 개념적이고 논리적으로 표현되는 게임규칙에는 적용하기 어렵다. 그러나 UML CASE 툴에 의한 형상관리는 효율적으로 이루어질 수 있다.

3.3 페트리네트(Petri net) 표현방식

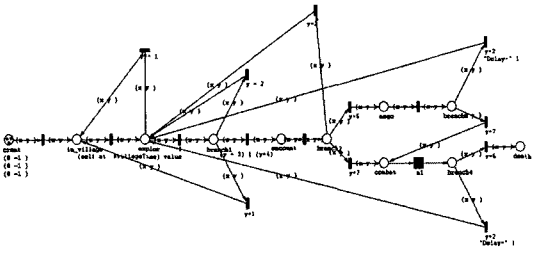
페트리네트는 1960년대 C.A. Petri에 의해 처음 개발 되었다. 페트리네트는 정보 흐름(information flow)을 표현하는 형식적 모델(formal model)로서 시스템의 정보 흐름과 제어를 분석하기 위해 사용된다[11, 12]. 페트리네트의 구성은, 〈그림 3〉와 같이, 시스템의 수동적인 요소들을 나타내는 place, 능동적인 요소들을 나타내는 transition, place와 transition 사이의 연결을 나타내는 arc, 그리고 정보 또는 제어 주도권을 의미하는 token으로 이루어져 있다. 즉 〈그림 3〉에서 원은 place를, 막대사각형은 transition을, 점은 token을, 그리고 화살표는 arc를 나타낸다.



[그림3] 토큰(token)들이 표시된 페트리네트

어떤 시점에서 각 place에 배치된 token들의 분포를 그 시점에서의 marking이라 부르며 이 marking은 그 시점에서의 시스템의 정보 흐름과 제어에 대한 상태를 나타낸다.

페트리네트는 다양한 메커니즘을 표현하는데 대표적으로 활용되는데 예를 들면, 상호배제문제(mutual exclusion problems), 일기-쓰기 문제(reader-writer problems), 생산-소비 문제(producers-consumers problems), 다중처리기 구조(multi-processor architecture)가 있다.



[그림4] 페트리네트로 모델링 된 D&D 게임규칙 모델[5]

페트리네트로 게임규칙을 표현할 때는 〈그림 4〉와 같이, 규칙내용을 페트리네트 표기방식으로 변환하여야 한다.

페트리네트의 표현방식은 페트리네트에 대한 기본 지식을 갖고 있으면 누구나 쉽게 정확하게 그 내용을 이해할 수 있는 비주얼적 표현방식을 제공한다. 또한 표현방식에서 논리적인 오류를 검증할 수 있기 때문에 논리적 표현이 가능하다. 페트리네트 시뮬레이터 개발도구(예:PACE5.0e)를 통해 표현된 게임규칙의 내용을 컴퓨터로 검증할 수 있다. 그리고 페트리네트 전용 CASE 툴(예: Net Case) 사용하면 모델의 형상관리를 비교적 쉽게 할 수 있다.

페트리네트의 표현방식은 정보흐름의 표현에 초점이 맞추어진 방식이기 때문에 게임의 운영성규칙과 암시성규칙의 표현은 쉽게 할 수 있으나 구성성규칙은, 예를 들면 게임월드의 구성요소들의 특성 파라미터들의 규칙은 오버헤드(overhead)가 많아서 적절하지 못하다.

3.4 스크립트 언어의 표현방식

스크립트 언어란 기계어로 컴파일(compile)되지 않고 별도의 번역기가 스크립트 내용을 분석하여 실행하는 방식의 프로그래밍 언어를 말한다. 스크립트(script) 언어의 예로는 자바스크립트, 유닉스 쉘 스크립트(Unix Shell Script)가 있다. 스크립트 언어는 기계어로 컴파일하는 프로그래밍 언어보다는 간단하고 유연한 신택스(syntax)를 사용하기 때문에 비교적 사용하기 쉽다는 장점이 있다.

게임규칙을 표현하기 위해 사용된 스크립트 언어는 게임규칙을 반영한 게임을 즉시 실행할 수 있는 환경을 지원하기 위함이다[8,9]. 관련 연구결과를 보면, 게임규칙을 표현하기 위해 사용된 스크립트 언어는 C언어[8], 그리고 Lisp언어[9] 신택스(syntax)를 기반으로 사용하고 있다. Lisp 신택스 기반의 스크립트 언어로 게임규칙을 표현한 예가 〈그림 5〉이다. 이들 스크립트 언어는 분기, 조건, 반복의 요소를 갖고 있는 처리절차, 여러 속성들 사이의 대응관계, 속성매개체들의 표현 등이 가능하기 때문에 모든 장르의 게임규칙을 표현할 수 있다.

스크립트 언어의 표현방식은 특정 문자기반 신택스에 맞추어 내용을 표현한다. 그래서 비주얼적으로 표현할 수 없다. 그러나 스크립트 언어의 경우는 컴퓨터 프로그래밍 언어의 공통적인 특징인 논리적인 표현을 요구하기 때문에 게임규칙 내용을 논리적으로 정확하게 표현할 수 있다. 그리고 스크립트 언어는 컴퓨터가 처리할 수 있는 방식이기 때문에 표현내용에 대해 컴퓨터의 검증이 가능하다. 스크립트 언어로 표현된 내용의 형상관리는 전용 편집기의 지원내용에 따라 가능할 수 있다.

```
(set feature-types (("peak") ("lake")))
(set feature-namers (("lake" "generic-lake-names")))
(namer generic-lake-names (grammar root 10
  (root (or 5 (foo "Lake " generic-names)
              1 (generic-names " Lake")
            ))
  (foo "")  ; works around a bug
  ))
(include "ng-weird")
```

[그림5] 게임월드의 호수에 대한 장면내역을 스크립트언어로 표현한 예제

4. 게임규칙 표현방법 비교분석

본 연구에서 조사된 표현방식들을 게임규칙의 표현 적합성에 대해 게임규칙 표현범위, 비주얼적 표현성, 논리적 표현성, 검증 자동화, 수정관리 편리성으로 나누어 〈표 3〉과 같이, 비교분석하였다.

	문서 표현방식	UML 표현방식	페트리네트 표현방식	스크립트 언어 표현방식
게임규칙 표현범위	구성성규칙, 운영성규칙, 암시성규칙	구성성규칙, 운영성규칙, 암시성규칙	운영성규칙, 암시성규칙	구성성규칙, 운영성규칙, 암시성규칙
비주얼적 표현	가능, 품질 가변적	우수	우수, 내용 간접적표현	불능
논리적 표현	가능, 품질 가변적	우수	우수, 내용 간접적표현	우수
자동화된 검증	불능	가능	우수	우수
효율적 형상관리	불능	우수	우수, 이중 형상관리	가능
표현력 한계	작성자의 표현능력의 존	표형식의 대규모 자료 표현	구성성규칙 표현능력	비주얼 표현능력
장점	자유로운 표현	비주얼표현 논리적 표현	정보흐름 표현	논리적 표현

[표3] 게임규칙 표현방식의 적합성에 대한 표현방식들의 비교분석표

게임규칙 표현범위에 대해 비교하면, 문서 표현방식은 가장 범용적 표현능력을 갖고 있기 때문에, 구성성규칙, 운영성규칙, 암시성규칙 모두 표현할 수 있다. UML 표현방식도 클래스다이어그램, 패키지다이어그램, 배치다이어그램, 활동다이어그램으로 구성성규칙을 표현할 수 있고, 유스케이스다이어그램과 활동다이어그램을 통해 운영성규칙과 암시성규칙을 표현할 수 있다. 그러나 페트리네트 표현방식은 정보흐름 표현에 맞추어져 있기 때문에 운영성규칙 표현은 우수하지만 구성성규칙은 표현은 표현하기 어렵다. 스크립트 언어 표현방식은 프로그래밍 언어의 특성상 구성성규칙, 운영성규칙, 암시성규칙 모두 표현할 수 있다.

비주얼적 표현성에 대해 비교하면, 문서 표현방식은, 작성자가 그림과 도표를 많이 사용할 경우는 비주얼성이 높아지는 반면 문자중심으로 표현할 경우는 비주얼성이 낮아지기 때문에, 비주얼적 표현성이 가변적이다. UML 표현방식은 게임규칙의 내용을 UML 다이어그램으로 표현하기 때문에 비주얼적 표현성이 우수하다. 그리고 페트리네트 표현방식도 비주얼 기호들을 통해 내용을 나타내기 때문에 비주얼적 표현성이 우수하다. 그러나 페트리네트 표현방식은 원, 막대형 사각형, 화살표, 점들을 사용하여 페트리네트 표기규칙에 따라 표현하기 때문에, 게임의 규칙내용을 페트리네트 표현방식으로 변환하여 표현하는 간접적인 표현방법을 사용한다. 그리고 스크립트 언어 표현방식은 게임규칙을 스크립트 언어로 표현해야 하기 때문에 그림이나 도표를 사용하지 못하고 오직 텍스트로 표현해야 하기 때문에, 비주얼적 표현이 불가능하다.

논리적 표현성에 대해 비교하면, 문서표현방식은 작성자에 따라 논리적으로 표현할 수도 그렇지 않을 수도 있다. 그러나 문서 작성에 대한 기본원칙을 준수하면 논리적 표현은 가능하다. UML 표현방식의 경우는 다이어그램 표현과 동시에 논리적 표현이 이루어지도록 특정 표기지침의 준수를 요구하기 때문에 논리적 표현성이 우수하다. 페트리네트 표현방식의 경우에도 표기지침의 준수를 요구하기 때문에 논리적 표현성이 우수하다.

검증 자동화에 대해 비교하면, 문서 표현방식은 게임규칙 내용을 인간이 사용하는 사회적인 언어로 표현하기 때문에 컴퓨터가 그대로 처리할 수 없어 내용 검증을 자동화할 수 없다. UML 표현방식은 UML을 지원하는 CASE툴이 존재하기 때문에 게임규칙의 검증 자동화가 가능하다. 그러나 CASE툴을 사용한 검증의 자동화는 게임규칙을 프로그램 코드 수준까지의 상세한 명세를 입력해야 만 가능하다. 이러한 작업은 UML 표기법외에 객체지향적 시스템 모델링과 프로그래밍에 대한 지식을 필요로 한다. 페트리네트 표현방식은 페트리네트 시뮬레이터를 이용해서 자동적으로 게임규칙을 검증할 수 있다. 페트리네트가 본래 검증과 분석을 목적으로 하는 범용적 모델링도구이기 때문에 검증능력이 우수하다. 또한 스크립트언어 표현방식도 개발하고자 하는 게임규칙의 검증과 분석을 목적으로 개발되었기 때문에 검증능력이 우수하다.

게임규칙의 효율적인 형상관리에 대해 비교하면, 문서표현방식은 문서내용 검토에 의존하기 때문에 효율적인 형상관리가 불가능하다. 스크립트언어 표현방식에서도 전용편집기를 사용하기 때문에 전용편집기의 지원기능에 따라 형상관리의 가능여부가 결정된다. UML 표현방식은 CASE툴을 사용하면 형상관리의 많은 부분을 자동적으로 해준다. 마찬가지로 페트리네트의 경우에는 CASE툴을 사용하면 형상관리의 많은 부분을 자동적으로 해주어 편리하지만, 게임규칙 내용에 대해 간접적으로 표현되기 때문에 수정관리에 이중적인 작업이 이루어지는 불편함이 있다.

본 연구에서 조사된 4가지의 표현방식을 비교분석한 결과 UML 표현방식이 게임디자인에서 가장 적합한 방식으로 판단된다.

5. 결론

게임개발은 기획자, 그래픽디자이너, 프로그래머 등 다양한 분야의 전문가들이 함께 참여하여 진행된다. 게임개발의 성공을 위해, 게임설계내용은 모든 참여자들이 반드시 이해해야 한다. 게임개발 특성상, 게임 설계내용은 게임디자인단계부터 구현단계와 테스트단계까지 자주 변경이 이루어진다. 게임의 설계내용은 게임규칙과 콘텐츠의 설계내용으로 구성된다. 그 중에서 게임규칙의 설계 내용은, 모든 참여자들이 쉽고 정확하게 이해할 수 있어야 하고 자주 발생하는 변경관리를 효율적으로 할 수 있어야 하며 그리고 정확한 검증이 이루어져야 한다.

본 연구는 게임규칙의 설계내용에 대해, 게임개발에 적합한 표현방식을 찾기 위한 조사연구이다. 조사된 표현방식은 문서표현방식, UML 표현방식, 페트리네트 표현방식, 스

크립트 언어 표현방식으로 4가지이다. 이 표현방식들을, 게임규칙의 표현범위, 비주얼적 표현, 논리적 표현, 자동화된 검증, 그리고 효율적 형상관리에 대하여 비교분석하였다.

비교분석결과 UML 표기방식이 가장 적합하였고 특히 "자동화된 검증" 항목을 제외하고는 모든 비교분석 항목에서 우수하다는 것을 알게 되었다. UML 표기방식은 CASE툴을 사용하여 자동화된 검증이 가능은 하지만 객체지향적 프로그래밍 기법의 지식과 프로그래밍 코드 수준의 명세화 작업이 필요하다.

따라서 앞으로 필요한 연구는 UML 표기방식으로 이루어진 게임규칙의 설계내용의 검증을 쉽고 편리하게 이루어질 수 있는 컴퓨터 자동화 방법의 개발연구가 필요하다.

참고문헌

[1] Andrew Rollings, Dave Morris, "Game Architecture and Design", Coriolis, p. 2000.

[2] Erik Bethke, "Game Development and Production," Wordware Publishing, Inc. 2003.

[3] UML 공식사이트: http://www.uml.org

[4] Vladimir Yakhnis, "A UML Based Notation for Specifying Abstract Board Games," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 5, pp. 4925-4929, 1998.

[5] Thor Alexander, Editor, "Massively Multiplayer Game Development," Charles River Media, Inc., 2003.

[6] 장희동, "게임메카닉스 시뮬레이터 구현," 정보처리학회논문지B, Vol. 12, No. 5, pp.595-606, 2005.

[7] 유수현 외 4인 번역, "Dungeons & Dragons 한글판 룰북," , (주)커뮤니케이션, 1995.

[8] John R. Rankin, Xiaohang Ma, "Game Design Language," Proceedings of ICITA 2002, pp. 283-285, 2002.

[9] Xconq Homepage: http://sourceware.org./xconq/

[10]Katie Salen, Eric Zimmerman, "Rules of Play Game Design Fundamentals," The MIT Press, 2004.

[11] James L. Peterson, "Petri Nets", Computing Surveys, Vol 9, No. 3, September 1977.

[12] Claude Girault and Rudiger Valk, "Petri Nets for System Engineering: a Guide to Modeling, Verification, and Application", Springer, 2002.

장 희 동 (Hee Dong Chang)

E-mail: dooly@office.hoseo.ac.kr
1984년 계명대학교 수학과 졸업(학사)
1987년 한국과학기술원 응용수학과 졸업(석사)
1995년 포항공과대학 수학과 졸업(박사)
1987년 - 1997년 한국전자통신연구소 영상통신연구실 선임연구원
1998년 - 2002년 숭의여자대학 컴퓨터게임과 조교수
2003년 - 현 재 호서대학교 컴퓨터공학부 게임공학과 조교수
관심분야: 게임메카닉스 설계, 게임밸런싱, 게임알고리즘