Journal of Korea Multimedia Society Vol. 12, No. 2, February 2009(pp. 313-322)

이야기 생성을 위한 인터랙티브 스토리텔링 스크립트 언어에 관한 연구

김석규[†], 문성현[††], 박준[†††], 장준호[††††], 한상영[†††††]

요 약

사용자와의 상호작용을 통해 멀티 스토리를 생성하는 기법을 인터랙티브 스토리텔링(Interactive Storytelling)이라 한다. 본 논문에서는 인터랙티브 스토리텔링 시스템의 중요한 구성요소인 서술구조와 이를 표현하기 위한 스크립트 언어를 제시하고, 이 언어로 표현된 서술구조를 처리하여 스크립트 형태로 이야기를 생성하는 시스템을 구현했다. 본 연구는 인터랙티브 스토리텔링을 위한 저작 도구 생성의 기반이 되며 이는 다른 매체를 통해 이야기를 생성하기 위한 스토리 연구 도구를 만드는 데 사용될 수도 있으며, 사용자와 상호작용하며 만들어진 이야기를 텍스트, 이미지, 애니메이션 등의 형태로 표현하는 도구를 만드는 데 사용될 수 있을 것이다.

A Study on Interactive Storytelling Script Language for Generating the Stories

Seok-Kyoo Kim[†], Sung-Hyun Moon[††], Jun Park[†††], Juno Chang[††††], Sang-Yong Han[†††††]

ABSTRACT

A multi-story can be generated by the interactions of users in the interactive storytelling system. In this paper, I suggest narrative structure and corresponding Storytelling Markup Language and implement the system that processes a story presented using this language. This research will be basis of making an authoring tool for interactive storytelling. It can be used to make a story tool and story presentation tool using text, image, animation with user interaction.

Key words: Interactive Storytelling(인터랙티브 스토리텔링), Planning(플래닝), Game Programming (게임 프로그래밍), Script Language(스크립트 언어)

1. 서 론

오늘날 디지털 스토리텔링이라는 말은 사회 여러 분야에서 사용되고 있다. 문학과 같이 기존에 스토리텔링이라는 분야를 가지고 있던 학문뿐만 아니라 디지털과의 접목으로 인해 미디어학, 컴퓨터공학 등의 분야에서 연구가 이루어지고 있다. 이 용어는 명확하게 정의는 되어 있지 않지만 디지털 시대의 스토리텔링으로 인식되고 있으며, 몇몇 학자들이 정의를 내리고 있기는 하지만 공통된 정의는 없는 상태이다.

※ 교신저자(Corresponding Author) : 장준호, 주소 : 서울시 종로구 홍지동 7(110-743), 전화 : 02)2287-5393, FAX : 02)2287-0072, E-mail : jchang@smu.ac.kr
접수일 : 2008년 10월 6일, 완료일 : 2008년 12월 8일
[†] 준회원, 서울대학교 컴퓨터공학부 박사과정
(E-mail : anemonc@pplab.snu.ac.kr)
[††] 준회원, 서울대학교 컴퓨터공학부 박사과정
(E-mail : shmoon@pplab.snu.ac.kr)
[†††] 정회원, 홍익대학교 컴퓨터공학부 부교수
(E-mail : jpark@hongik.ac.kr)
[††††] 정회원, 상명대학교 디지털미디어학부 조교수
[†††††] 정회원, 서울대학교 컴퓨터공학부 교수
(E-mail : syhan@pplab.snu.ac.kr)

정의중의 하나로서 디지털 스토리텔링이란 디지털 기술을 매체 환경 또는 표현 수단으로 수용하여 이루어지는 스토리텔링이라는 것이다. 즉, 디지털 스토리텔링이란 넓게는 미디어영상물 제작공정에서의 매체환경 전체에 디지털 기술이 수용되거나, 최소한 이야기에서 담화까지의 창작에 표현 수단으로 디지털 기술이 수용된 경우를 말한다[1].

인터랙티브 스토리텔링은 디지털 스토리텔링의 한 분야인데, 이 개념은 이야기의 감정적이고 극적인 면과 컴퓨터의 상호작용성을 이용한 서사의 한 형태로서, 사용자가 스토리텔링에 영향을 주어서 이야기의 진행 방향을 변경시킬 수 있는 스토리텔링을 의미한다. 사용자의 선택에 따라 이야기의 진행과 결말이 다양하게 나타나는 에듀테인먼트나 비디오게임 등이 이에 근접하는 예라고 할 수 있다. 특히 게임에 있어서 RPG나 어드벤처 게임, 비주얼 소설 등에서 그런 특성을 찾아 볼 수 있다.

이러한 인터랙티브 스토리텔링에 대한 연구는 아직 활발히 진행되고 있지는 않지만, 온라인게임과 같은 온라인 콘텐트의 발전과 더불어 여러 가지 시도가 이루어지고 있다. 본 연구에서는 인터랙티브 스토리텔링의 중요한 구성요소로 다음과 같은 것들을 생각해 보고자 한다.

1) 이야기를 여러 방향으로 전개해나갈 수 있도록 해주는 서술 구조(Narrative Structure)
   - 이야기의 진행에 있어 필요한 요소들과 그것들을 표현할 구조가 필요하다.
2) 서술 구조를 구현하기 위한 스크립트 언어
   - 서술 구조를 저작자나 프로그래머가 이해할 수 있는 형태의 언어로 표현하여 컴퓨터가 처리할 수 있는 시스템을 마련해 주어야 한다.
3) 프로그래밍 언어에 대한 전문적 지식이 없어도 서술 구조 및 이야기의 생성을 도와주는 이야기 생성기 및 저작도구
   - 이야기 제작자가 스크립트 언어에 대한 지식이 없더라도 쉽게 이야기를 생성할 수 있는 개발환경이 지원되어야 한다. 이야기 생성기는 입력받은 정보를 서술 구조로 해석을 하여 스크립트 언어로의 변환 및 이를 이용하여 이야기를 생성하는 역할을 한다. 저작도구는 이야기생성기를 포함하는 그래픽 사용자 인터페이스 환경이 된다.

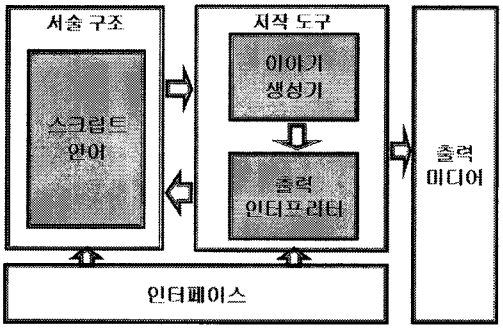


그림 1. 인터랙티브 스토리텔링 시스템

그림 1은 인터랙티브 스토리텔링 시스템의 구성요소를 보여준다.

이 논문에서는 인터랙티브 스토리텔링 시스템을 구성하고 있는 중요한 여러 가지 요소 중에서 서술구조 및 이를 이용한 인터랙티브 스토리텔링 스크립트 언어를 제안한다. 이를 위해 2장에서는 기존의 스크립트 언어 및 그 표현방식에 대해서 알아본다. 3장에서는 제약기반의 서술구조를 제시하고, 4장에서는 스토리텔링을 위한 SML(Storytelling Markup Language)을 제안하고, 이 스크립트 언어는 인터랙티브 스토리텔링 시스템의 기본이며 이를 이용하여 이야기 생성기 및 저작도구를 개발하고 이야기를 생성할 수 있게 될 것이다. 마지막 장에서는 본 연구를 정리하고, 본 연구의 활용 방안 및 향후 과제에 대해서 논의한다.

2. 관련 연구

지금까지 인터랙티브 스토리텔링 시스템에 사용되어진 언어들은 크게 3가지로 분류할 수 있는데, 자연언어, 논리 프로그래밍 언어, XML에서 파생되어진 다양한 마크업 언어가 있다.

자연언어는 인간의 언어를 그대로 표현한 것이기 때문에 행동, 인물, 사건 등을 서술하기에 가장 적합한 언어이다. 자연언어를 사용하게 되면 사용자는 편리해지고 접근성이나 가독성은 높아지는 반면, 기존 프로그래밍언어를 처리하는 것에 비해서 여러 가지로 어려운 점이 많이 있다. 기존의 연구에서는 이러한 자연언어시스템을 음성인식시스템과 통합하여 사용하였는데, 음성인식시스템을 통해 들어온 자연언어를 처리하여 최종적인 이야기를 보여주는 시스

템이다[2].

논리 프로그래밍 언어들은 인터랙티브 스토리텔링 시스템의 서술 구조에 인공지능분야의 플래닝기법(Planning Technique)을 사용하기 때문에 논리프로그래밍이 가능한 프로그래밍 언어를 사용하였다. 여기에는 STRIPS(STanford Research Institute Problem Solver)[3]및 STRIPS에서 파생된 언어들이 있다. STRIPS는 AI의 문제들을 해결하기 위해 제시된 방법으로 서술 구조의 플래닝 알고리즘을 표현하는데 가장 적절하기 때문에 인터랙티브 스토리텔링 시스템에 사용되었다[4,5]. STRIPS에서 파생된 언어로는 PDDL(Planning Domain Description Language)[6]가 있다.

마지막으로 XML (eXtensible Markup Language)을 들 수 있다. XML의 하위 개념인 HTML(Hyper-Text Markup Language)은 전 세계적으로 월드와이드웹의 표준 출력 형식으로 사용되고 있기 때문에 많은 사용자가 그 형식에 대해 잘 알고 있거나 배우기 쉬운 위치에 있다. 이런 환경요인에다, XML은 모든 정보를 사용자가 알 수 있는 문자로 표현하기 때문에 가독성이 매우 높다고 할 수 있다. 또한 XML은 W3C(World Wide Web Consortium)에서 제안된 이래 정보를 표현하는 표준으로 널리 쓰이고 있으며 HTML을 대체할 것이라는 전망도 있다. 많은 개발도구들이 이미 XML을 다루는 라이브러리를 포함시켰고 또한 끊임없이 발전시키고 있다. 이에 널리 쓰인다는 점에서 유리한 위치에 있다. 무엇보다도 XML은 여러 가지 형식을 자유롭게 추가하는 것이 가능하며 상상 가능한 거의 모든 형식을 나타낼 수 있는 일반성을 지닌 언어이다.

인터랙티브 스토리텔링 시스템에 사용되어지는 XML로부터 파생된 언어로는 MPML(Multi-modal Presentation Markup Language)[7], AIML (Artificial Intelligence Markup Language)[8] 등이 있다.

MPML은 현실세계와 비슷한 인물들의 행동을 제어하는데 적절한 마크업 언어이다. MPML은 웹상에서 2차원상의 인물들의 행동제어, 프레젠테이션 흐름, 외부객체의 통합을 할 수 있는 강력한 언어이다. 그러나 웹 에이전트를 제어하기 위한 언어이기 때문에 본 연구에서 하고자 하는 이야기 생성을 위한 스크립트 언어로는 부족한 점이 있다.

VISTA(Virtual Interactive Story Telling Agents) 프로젝트[9]에서는 AIML을 사용하여 프로그램을 작성하였다. AIML은 AI 응용프로그램을 지원하는 XML 스타일의 스크립트 언어인데, VISTA 시스템에서는 Prolog와 연동하여 AIML을 사용하였다. 이야기에 적용되는 질문응답 관계는 AIML을 이용하였고 Prolog는 여러 가지 행동규칙을 생성하는데 사용하였다. 그러나 AIML은 모든 질문 응답이 미리 정의되어 있어야 하고, 여러 가지 조건으로부터 추론된 다양한 결과를 얻기는 힘들다. 프로그래밍에 익숙하지 않은 일반 사용자가 Prolog 언어를 알아야 한다는 어려움도 있다.

3. 서술 구조 (Narrative Structure)

앞서 살펴본바와 같이 인터랙티브 스토리텔링 시스템에는 여러 가지 스크립트 언어가 사용되었는데 본 논문에서는 XML에 기초를 두는 SML을 제안한다. 이미 MPML이나 AIML같은 XML에서 파생된 스크립트 언어가 있지만 이 언어들은 서술구조를 정의할 수 없거나, 다양한 이야기의 생성이 어려운 점이 있다. 이에 비해 본 연구에서 제안하는 SML은 저작자가 직관적으로 쉽게 이야기를 프로그래밍할 수 있도록 서술구조를 정의하고, 이 서술구조에 따라 언어를 XML형식에 맞추어서 정의를 할 수 있는 장점이 있다.

3.1 제약 기반의 서술 구조 (Constraint based Narrative Structure)

기존의 연구들은 인공지능 분야의 플래닝 기법에 기반을 두어 문제를 해결하기 위해 주로 STRIPS 형태나 Lisp형태로 이야기를 나타냈다[2,5,10]. 서술 구조의 플래닝 알고리즘으로 HTN(Hierarchical Task Network), HSP(Heuristic Planning)[10]이 있는데, STRIPS는 HTN과 HSP를 사용한 시스템에서 사용되었다. HTN은 이야기의 결론을 루트로 하고 각각의 부결론들을 자식으로 하는 트리구조로 나타난다. 이는 탑다운 생성방식으로 만들었기 때문에 이야기의 통일성이 좋다. 반대로 HSP에서는 초기 상태에서 목표 상태로의 경로를 생성한다. 이는 이야기의 유연한 생성을 가능하게 한다[4]. 본문에서 소개하는 방식도 일종의 HSP에 해당한다고 볼 수 있다.

HTN의 경우에는 트리를 이용하여 이야기를 구성

한다. 루트 노드는 이야기의 결론을 의미하고 각각의 비단말 노드는 부결론을 의미하며, 단말 노드는 어떤 행동이 일어나는 것을 의미한다. 이것이 이야기를 이루는 방식을 보자면, 최종 결론을 여러 개의 부결론으로 나누고 그 각각을 또 부결론으로 나누어 각각의 부결론의 해결을 통해 이야기가 생성되는 방식이다. 여러 개의 행동이 모여서 가장 작은 단위의 부결론이 해결된다. 즉 이야기는 부결론의 결합이며 맨 하위 부결론은 특정 캐릭터의 행동들의 집합을 포함시키는 것으로 본다. 문제를 더 작은 단위로 쪼개어 그것을 해결한 뒤 결합하는 AI기반의 분할정복 플래닝기법으로 이야기를 생성한다. 이런 구조는 부결론이 만족하지 않으면 그것을 전제로 한 결론에 도달하지 못하여 이야기가 만들어지지 않기 때문에 흐름의 일관성이 보장되지만, 결론을 이루는 전제가 되는 행동들만을 중심으로 이야기가 진행되어 HTN의 경로 상에 나타나는 이야기들만 만들어지기 때문에 자유도가 떨어지게 된다는 단점을 가진다.

이상적인 형태의 인터랙티브 스토리텔링이라면 이야기의 흐름이 일관성을 유지하면서도 자유도가 높아야 하지만 보통 이 두 가지는 상충관계에 있기 때문에 양쪽을 모두 만족시키는 인터랙티브 스토리텔링 시스템의 구현은 쉽지 않다[10]. 본 연구에서 지향하는 이야기생성의 방향은 이야기의 자유도와 일관성의 정도를 작가에게 조절 가능하도록 하게 하는 것이다. 기존의 연구들은 인공지능기반의 분할정복 플래닝을 통해 이야기의 인과 관계를 유지하는 반면에 자유도의 한계에 부딪혔기 때문에, 본 연구에서는 계층 구조를 없앰으로써 그 틀을 벗어나는 형태를 제시한다. 즉, 기존의 구조들과 달리 부결론의 단계를 제거한 뒤 이야기를 캐릭터의 행동의 연속으로 본다. 부결론을 제거한 대신 이야기의 일관성을 유지하기 위해 각 행동간 인과 관계나 선후 관계 등의 제약 조건을 선언해 준다. 이 제약 조건의 정도를 통해 이야기의 일관성의 정도와 자유도를 작가가 원하는 수준에 맞게 적절히 조절할 수 있다.

본 연구에서 제시하는 서술 구조는 크게 요소(constituent) 선언부와 제약(constraint) 선언부로 나뉜다. 요소 선언부에서는 이야기에 필요한 등장인물(actor), 행동(action), 속성(property), 배경(stage), 소품(props) 등을 정의해준다. 제약 선언부에서는 이야기의 흐름을 조절하고 이야기의 비선형적인 진행을 지원할 수 있는 각종 조건 관계 구조를 제시한다.

3.2 요소 선언부(Constituent Declaration)

요소 선언부에서는 이야기에서 필요한 기본적인 요소들을 선언한다. 선언부에서 정의해주는 요소는 속성, 배경, 등장인물, 소품, 행동 등이 있다. 그림 2는 선언부에서 정의되는 요소들과 그들간의 의존성을 나타낸다.

- 속성(Property)

속성은 등장인물이나 소품 등이 지니는 특성을 수치로 나타낸 것이다. 등장인물이나 소품은 각각에 필요한 속성에 해당하는 값을 수치로 가질 수 있다.

- 배경(Stage)

배경은 이야기가 진행되는 공간을 나타낸다.

- 등장인물(Actor)

등장인물은 직접 행동을 행하거나 행하여지는, 이야기의 주체 혹은 객체가 되는 요소이다. 각각의 등장인물에 대해 이름이 있고, 그 등장인물이 가지는 속성의 종류와 각각의 값들, 그리고 이야기의 맨 처음에 어느 공간에 있는가를 표시해주는 배경(stage) 정보를 지닌다.

- 소품(Props)

소품은 이야기의 진행에서 행동의 주체는 되지 못하지만 객체가 되는 것을 지칭한다. 따라서 물건이라든지, 이야기 내에서 행동의 주체가 될 수 없는 등장인물을 소품으로 정의한다.

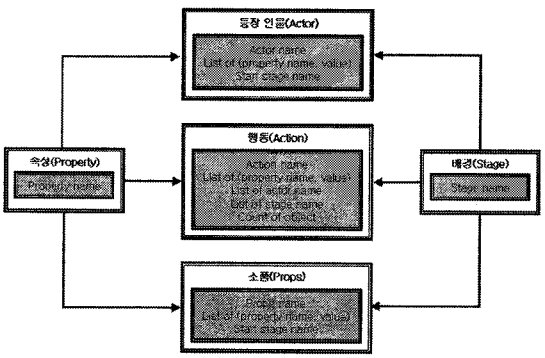


그림 2. 선언부 요소들과 그 관계

- 행동(Action)

행동은 등장인물이 취할 수 있는 동작을 나타낸다. 이 행동을 수행할 수 있는 등장인물들, 행동을 위해 필요한 속성들, 행동이 일어날 수 있는 배경들, 행동의 목적어가 몇 개 있느냐를 표시해준다.

본 제약 기반 서술 구조에서 속성은 등장인물과 행동 간에 공통적으로 존재하는데, 이는 등장인물의 동작 선택에 있어서 좀 더 합리적인 선택을 하게 만드는 이야기의 생성을 위한 장치이다. 이들의 쓰임을 이해하기 위해 한 가지 예를 들어보자. 등장인물 A가 talkative = 80, aggressive = 30 이라는 속성을 지니고, 행동 attack이 talkative = 30, aggressive = 80 이라는 속성을 지니며, 행동 talk가 talkative = 80, aggressive = 30 이라는 속성을 지닌다고 가정해보자. 이 때 등장인물 A가 attack이나 talk 가운데 한 가지 동작을 수행해야 하는 상황이라면, 속성의 값에 의해서 talk를 수행할 확률이 높아지는 것이다. 등장인물의 성격과 행동 간의 조화를 설정하기 위해서는 이야기 생성기에서 이 구조를 충분히 활용하여 이야기를 생성할 필요가 있다.

위에서 열거한 요소들이 조합되어 이벤트(Event)가 생성이 된다. 이벤트란 간단하게 말하면 특정 인물이 어떤 동작을 수행하는 것이며 행동의 정의된 형태에 따라 목적어의 유무가 정해진다.

그림 3에 이벤트에 필요한 요소들이 나타나 있다. 이를 풀어서 나타내면 "특정 등장인물이 어디에서 무슨 행동을 (누구에게) (무엇을) 하다." 와 같은 형태로 등장인물, 배경, 행동, 소품 등이 조합되어 표현된다. 본 구조에서 이야기의 생성이란 이런 이벤트의 생성을 의미한다. 그리고 이야기 생성의 복잡도를 줄이기 위해서 시점(viewpoint)라는 특별한 요소를 두는데, 이는 사용자가 현재 보고 있는 공간(stage)을 나타낸다. 모든 이야기 생성은 현재 시점에서만 이루어진다고 한정짓는다. 이는 연극에서 관객이 하나의 무대를 보는 것과 같은 이치이다. 연극에서 무대가 변경될 때 막이 내려오듯 본 구조에서 배경의 변화를 위해선 시점의 변화가 필요하다. 연극에서도 초기 무대가 지정되어 있어야 하듯이, 요소 선언에서도 이야기의 시작점이 되는 지점에 대한 시점의 지정을 해주어야 한다.

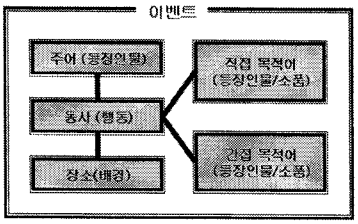


그림 3. 이벤트의 구성 요소

3.3 제약 선언부(Constraint Declaration)

제약 선언부에서는 이야기의 흐름의 방향을 제어할 수 있는 여러 가지 제약조건을 선언한다. 이 제약조건들은 이야기의 전개 방향을 잡아주는 뼈대(Framework) 역할을 하여, 이야기가 지나치게 엉뚱하게 흐르는 것을 방지해준다. 제약조건의 강약에 따라서 이야기의 자유도가 결정되는데, 제약조건의 종류는 아래에 나타나 있다. 여기서 설명하는 부분은 개념적인 것으로 이들을 실제 스크립트 언어로 나타낼 때에는 와일드 키('?')나 논리 연산을 지원해서 이야기의 제어를 더욱 정교하게 할 수 있도록 할 필요가 있다.

- 종결(Ending) : (event) → (end)

이야기를 종결시키는 조건이다. 주어진 이벤트가 발생하면 이야기가 종료된다. 이벤트의 종류를 여러 개 지정함으로써 이야기의 다중엔딩을 설정할 수 있다.

- 트랜잭션(Transaction) : (pre event) → (post event)

두 개의 이벤트가 트랜잭션처럼 함께 일어난다. 특정 이벤트가 발생했을 때 뒤이어 다른 이벤트가 바로 일어나주길 원하는 경우 지정해준다. 스크립트 언어에서는 post event가 pre event의 주어 및 목적어를 참조할 수 있도록 하는 기능이 제공되어야 풍부한 형태의 트랜잭션 조건을 만들어 낼 수 있다. 이를테면 임의의 인물을 지칭하는 와일드 키 '?'를 조건으로 주어서, pre event로 '?가 ?를 때리다.' 를 정의했다고 해 보자. 이에 관한 post event를 '[pre event의 목적어]가 [pre event의 주어]를 때리다' 와 같은 형태로 선언이 가능해야 한다. 이 경우 post event의 주어와 목적어는 pre event가 발생한 시점에서 결정된다.

- 변화(Transition) :
(event/action) → (change property)

지정된 이벤트 혹은 행동이 발생했을 시에 특정

속성값에 변화를 주도록 지정하는 것이다. 예를 들면 '때리다' 라는 동작에 대해서 '목적어의 health라는 속성을 30 감소시킨다.' 하는 형태로 조건을 지정하는 식이다. 이는 특정 상태의 변화를 수반하는 동작을 정의하는 데 유용하다.

- 발생(Induction) :
  (condition of property) → (event/action)

특정 속성이 어떤 조건을 만족하는 경우 이벤트나 동작을 수행하도록 선언할 수 있다. 예로서, "hungry가 30이하이면 밥을 먹는다." 같은 형태의 지정이다. 이 제어조건은 등장인물에게 상태의 변화에 따른 특정 동작을 수행하게 하고 싶을 때 지정해준다.

- 필수(Must) : (pre event) → (post event)

여러 가지 기능을 수행하는 제약조건이다. 의미상으론 'post event가 일어나지 않은 상태에서 pre event가 수행된 경우, post event는 이야기가 종결되기 전까지 반드시 일어나야 한다.'는 것을 나타낸다. 즉, 인과관계에 있어 원인에 해당하는 이벤트가 발생했을 때, 그에 따른 결과로 일어나는 이벤트가 반드시 필요하다면 이 제약조건을 사용한다. post event에 negation옵션을 둬서 pre event가 발생한 경우 post event는 절대로 일어나지 않도록 지정할 수도 있다. 이는 특정 사건이 발생했을 때는 그에 따라 또 다른 어떤 사건은 절대로 일어나지 못하게 하는 제약을 둘 수 있다. 그리고 pre event를 두지 않고 post event만 지정함으로써 이야기에서 어떤 경우라도 일어나야 하는 사건을 나타낼 수 있다.

- 순서(Ordering) : (pre event) → (post event)

이야기의 논리적 흐름을 맞춰주기 위한 선언이다. 여기서 지정되는 이벤트들은 일어나지 않아도 상관이 없다. 단 'pre event는 post 이벤트보다 항상 먼저 일어나 줘야 한다.' 는 것을 나타낸다. 즉 pre event가 발생하지 않았다면 post event 또한 발생하지 않는다.

- 배경 변경(Stage Change) :
  (event) → (change stage)

특정 이벤트가 발생하는 경우 이벤트의 주어 혹은 목적어의 현재 배경을 변경해주도록 하는 제약 선언이다. 이는 연극에서 무대의 등장인물을 퇴장시키거나, 등장인물을 무대로 나타나게 하는 효과와 비슷하다.

정리하자면, 요소 선언부에서는 이야기에서 요구하는 기본 환경과 구성 요소들을 정의하고, 제어 선언부에서는 전체적인 이야기 흐름의 윤곽을 잡아주는 제어사항들을 지정해준다.

4. SML (Storytelling Markup Language)

4.1 SML

앞에서 인터랙티브 스토리텔링을 목적으로 한 서술 구조를 정의하였다. 그러나 앞에서 정의한 구조는 일종의 추상 데이터 타입에 불과하다. 이야기를 생성하기 위해선 위의 구성 요소들을 이야기 생성 엔진이 처리할 수 있도록 명시적인 언어로 표현해 줄 필요가 있다. 서술 구조를 표현하기 위한 언어에는 다음과 같은 특성들이 필요하다.

첫째로, 가독성이 좋아서 의미파악이 쉽고 사용자가 원하는 것을 추가하기가 쉬워야 한다. 기본적으로 저작도구를 이용하기 때문에 코드를 작성할일은 없겠지만 상황에 따라서 직접 코드를 작성해야 하는 경우라도 어려움이 없도록 가독성이 좋아야 한다.

둘째로, 표현성이 좋아야 한다. 앞에서 제시한 구조를 보면 인물이나 동작의 경우 복잡한 형태로 하위 요소들을 가진다. 특히 동작이 정의되는 것을 보면 동작에 필요한 속성들과 동작을 수행할 수 있는 등장인물들, 동작이 일어날 수 있는 배경들에 대한 정보를 요구한다. 하지만 각각의 동작마다 그 요소들의 숫자는 다르다. 제어 선언부에서 정의되는 구조들은 AND, OR 관계로 묶이고, 와일드키와 참조 구조 등을 요구하기 때문에 더욱 복잡하다. 이런 복잡하면서 다변적으로 정의되는 구조들을 표현 가능한 언어여야 한다.

셋째로, 다루기가 쉬운 구조여야 한다. 저작 도구나 이야기 생성기에서 주어진 언어로 된 코드를 생성하거나 혹은 파싱하는 작업이 필요하다. 언어의 구조가 다루기 쉬워야 이 언어를 지원하는 저작 도구라든지 이야기 생성기 등의 개발이 용이하다. 저작 도구와 이야기 생성기, 스크립트 언어가 각각 독립적인 경우에 언어의 이런 특성은 매우 중요하다고 볼 수 있다. 이는 나중에 이 언어를 지원할 다양한 저작 도구나 이야기 생성기의 개발까지도 가능할 수 있도록 해 준다.

위의 조건들을 염두에 두고 언어를 정의하여야 만

족할만한 결과를 얻을 수가 있다. 이미 존재하는 언어의 구조들 가운데서 생각해보자면 우선 손쉽게 XML(eXtensible Markup Language)형식을 고려할 수 있다. XML은 HTML(Hyper-Text Markup Language)의 상위개념으로 볼 수 있는데, HTML은 월드와이드웹의 표준 출력 형식으로 사용되기 때문에 많은 사용자들에게 잘 알려져 있다. 이로 인해 많은 사용자들이 HTML을 읽고 작성할 수 있다. 따라서 XML형식은 가독성에 있어선 상당히 좋다고 할 수 있다.

그리고 XML자체가 여러 가지 형식을 자유롭게 추가하는 것이 가능한 언어이기 때문에 표현성도 상당히 좋다. XML은 또한 널리 알려지고 쓰이는 까닭에 관련 라이브러리도 많이 존재한다. 라이브러리의 도움을 받으면 언어를 다루기가 용이하기 때문에 XML 형식은 '다루기가 쉬운 구조' 라고도 말할 수 있다.

이렇듯 XML은 위에서 나열한 언어적으로 요구되는 조건들을 모두 만족한다. 따라서 본 연구에서는 앞에서 정의한 서술 구조를 XML 형식으로 정의하였고 그 언어의 이름을 SML(Storytelling Markup Language)이라고 명명하였다. 앞서 제시한 서술 구조의 구성 요소들과 제어 조건들의 양이 많고 복잡하기 때문에 SML의 DTD(Document Type Definition)는 본문에 첨부하지는 않았다.

4.2 SML의 예

그림 4는 이야기의 특정 부분을 그림으로 표현한 것이고, 그림 5는 그림 4에 나타난 이야기를 필요한 부분만 간략하게 SML로 표현한 것이다. pre event가 없는 필수(must)를 통해 '홍부가 마당에서 제비를 발견' 혹은 '놀부가 마당에서 제비를 발견' 하는 이벤트가 꼭 일어나도록 설정해 주었다. 그리고 트랜잭션(transaction)을 통해 '누군가가 마당에서 제비를 발견' 하는 이벤트가 생성되는 경우 '그 발견한 사람은 제비를 돌봐주거나 괴롭히는 행동을 수행' 한다. 이 때 제비를 괴롭힐지 돌봐 줄지는 홍부나 놀부의 속성에 달렸다. 홍부의 경우 속성을 보면 '돌봐준다' 가 '괴롭힌다' 보다 속성값이 더 높고, 놀부의 경우 '괴롭힌다' 와의 속성값이 더 높다. 이 경우 홍부는 '돌봐준다' 는 동작이 일어나는 이벤트가 생성될 확률이 더 높고, 놀부는 '괴롭힌다' 는 동작이 일어나는 이벤트의 생성 확률이 더 높아지게 된다.

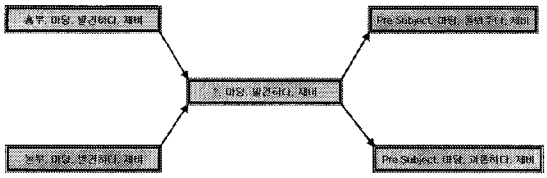


그림 4. 이야기 구조의 예

```
<actor actorId = "홍부">
    <property propertyId = "배려심">80</property>
    <property propertyId = "심술">20</property>
</actor>
<actor actorId = "놀부">
    <property propertyId = "배려심">20</property>
    <property propertyId = "심술">80</property>
</actor>

<prop> propId = "제비">
    <feature featureId = "호감도">50</feature>
</prop>

<action actionId = "발견하다" type = 1>
    <property propertyId = "배려심">50</property>
    <property propertyId = "심술">50</property>
    <actor actorId = "홍부"></actor>
    <actor actorId = "놀부"></actor>
    <stage stageId = "마당"></stage>
</action>
<action actionId = "괴롭히다" type = 1>
    <property propertyId = "배려심">20</property>
    <property propertyId = "심술">80</property>
    <actor actorId = "홍부"></actor>
    <actor actorId = "놀부"></actor>
    <stage stageId = "마당"></stage>
</action>
<action actionId = "돌봐주다" type = 1>
    <property propertyId = "배려심">80</property>
    <property propertyId = "심술">20</property>
    <actor actorId = "홍부"></actor>
    <actor actorId = "놀부"></actor>
    <stage stageId = "마당"></stage>
</action>

<must mustId = "must1">
    <preevent></preevent>
    <postevent conj = "or">
        <event>
            <subject subjectId = "홍부"></subject>
            <action actionId = "발견하다"></action>
            <object objectId = "제비"></object>
            <stage stageId = "마당"></stage>
        </event>
        <event>
            <subject subjectId = "놀부"></subject>
            <action actionId = "발견하다"></action>
            <object objectId = "제비"></object>
            <stage stageId = "마당"></stage>
        </event>
    </postevent>
</must>

<transaction transactionId = "transaction1">
    <preevent>
        <event>
            <subject subjectId = "?"></subject>
            <action actionId = "발견하다"></action>
            <object objectId = "제비"></object>
            <stage stageId = "마당"></stage>
        </event>
    </preevent>
    <postevent conj = "or">
        <event>
            <subject subjectId = Rsubect></subject>
            <action actionId = "돌봐주다"></action>
            <object objectId = "제비"></object>
            <stage stageId = "마당"></stage>
        </event>
        <event>
            <subject subjectId = Rsubect></subject>
            <action actionId = "괴롭히다"></action>
            <object objectId = "제비"></object>
            <stage stageId = "마당"></stage>
        </event>
    </postevent>
</must>
```

그림 5. SML을 이용한 이야기 구조 표현의 예

4.3 이야기 생성기

앞에서 서술 구조를 정의하였고, 그것을 표현해 줄 언어도 정의하였다. 주어진 언어로 작성된 코드를 해석하고 작업을 수행할 컴파일러 혹은 인터프리터가 필요한 시점이다. 즉 주어진 SML 코드를 입력받아 이야기를 생성해주는 이야기 생성기가 필요하다.

이야기를 연속된 이벤트의 연결로 봤을 때 품질이 좋은 이야기의 생성은 이 이벤트들을 얼마나 적절한 순서로 나열하느냐에 달려있다. Sitem을 item을 도메인으로 하는 집합이라 할 때 Sevent는 다음과 같이 나타낼 수 있다.

Ssubject = Sactor
SDO = Sactor ∪ Sprops
SIO = Sactor ∪ Sprops
SEvent = {(xsubject, xstage, xaction, xDO, xIO)
| xsubject∈Ssubject, xstage∈Sstage,
xaction∈Saction, xDO∈SDO, xIO∈SIO}

즉 event의 집합은 subject, stage, action, DO, IO 도메인들에 대한 카테시안 프로덕트로 볼 수 있고, 하나의 event는 (subject, stage, action, DO, IO)의 튜플로 나타난다. 좀 더 확장해서 이 튜플을 그래프의 노드로 본다면 이야기를 노드들의 연결로 볼 수 있다. 따라서 이야기의 생성이란 이벤트를 노드로 하는 그래프의 적절한 경로를 설정하는 것으로 볼 수 있다.

이런 관점에서 본 연구는 기존의 연구들과 달리 인공지능기법을 배제하고, 이벤트를 노드로 하는 그래프를 이용하는 이야기 생성기를 제시한다.

일단 위에서도 설명했다시피 이벤트를 그래프의 노드로 본다면, 이야기를 그래프에서의 일련의 노드의 연결하는 하나의 경로로 볼 수 있다. 이 경우 이야기 생성은 곧 제약조건에 위배되지 않는 순서의 그래프 경로를 설정하는 문제로 귀결된다. 따라서 고려해야 할 문제들은 대부분 그래프 문제로 변환해서 생각할 수 있다.

우선은 이벤트의 표현 문제를 생각해보자. 이벤트를 노드로 볼 수 있다고 하였지만 실제로 다른 그래프 문제처럼 생성가능한 모든 종류의 이벤트를 생성하고 각 노드간 경로를 설정해준다면 데이터의 양에 따라 노드의 수가 너무 많아지는 문제가 있다. 특히 와일드키가 들어가는 이벤트의 경우 이것을 모두 구체화시켜준다면 그 양은 기하급수적으로 늘어난다. 따라서 개념적으로 그래프 문제로 인식한다 해도 실질적으로 현실적인 수준에서 메모리를 사용하도록 하는 구현을 고려해야 한다. 전체 이벤트를 노드로 생성하지 않고 생성된 이벤트를 노드로 관리하는 형태로 접근할 필요가 있다. 그리고 제약조건에 대한 리스트를 관리하며, 이를 생성할 이벤트와 비교해서 경로화할 것인지를 고려하는 식의 구현이 현실적이다.

이벤트를 생성하는 문제는 현재 노드에서 고를 수 있는 많은 경로 가운데 하나를 선택하는 문제로 볼 수 있는데, 이 경우 경로선택의 기준이 되는 값이나 함수가 필요하다. 예를 들어 prim의 MST 생성 알고리즘을 보자면 경로선택의 기준으로 '현재까지 만들어진 MST와 가장 가까운 거리에 존재하는 노드를 둔다. 본 연구에서는 선택할 수 있는 행동들 가운데 등장인물의 속성값이 가장 높은 조합을 가지는 노드에 비중을 두는 방법을 생각한다. 이는 앞에서 그림 4를 설명하면서도 언급된 내용으로 특정 인물이 두 가지 이상의 행동 가운데 하나를 선택해야 한다면 그 인물의 성향에 맞는 행동을 하는 것이 합당하다고 보기 때문이다. 이야기의 획일성을 피하기 위해 이 선택의 조건은 확률개념으로 처리하도록 한다.

제약 조건을 지키는 문제는 그래프의 노드간 경로를 동적으로 관리하는 문제 및 위상적 순서를 관리하는 문제, 경로의 무한반복을 방지하는 문제로 볼 수 있다. 개념적으로 A라는 이벤트 이후에 B라는 이벤트가 바로 일어날 수 없다면 노드 A에서 B로 향하는 경로가 존재하지 않는다고 볼 수 있다. 그리고 '트랜잭션'이나 '필수', '순서' 등과 같이 두 이벤트 사이에 순서의 선후관계가 존재하는 경우 위상적 순서를 설정함으로써 제약조건을 지킬 수 있다. 그리고 무한반복의 생성을 방지함으로써 제약조건 간의 충돌을 막을 수 있다.

4.4 이야기 생성의 예

흥부전 이야기의 구성을 간략화 한 뒤 위에서 정의한 SML로 표현한 후 위에서 제시한 알고리즘을 토대로 한 이야기 생성기를 통해 이야기를 생성했다.

그림 6은 현재 구현된 저작도구이고 그림 7은 만들어진 이야기의 한 예이다.

앞의 예에선 흥부와 놀부가 둘 다 제비를 돌봐주어 그 결과 모두 부자가 되는 이야기가 생성되었다. 흥부와 놀부의 속성값에 따라서 우리가 기존에 알고 있는 것과는 전혀 다른 이야기가 생성이 될 수 있는 것이다. 이외에도 여러 가지 다양한 결말이 생성될 수 있다. 결과를 보면 단순한 텍스트 형태의 출력물로 되어 있지만 본 연구를 확장하여 이를 스크립트화

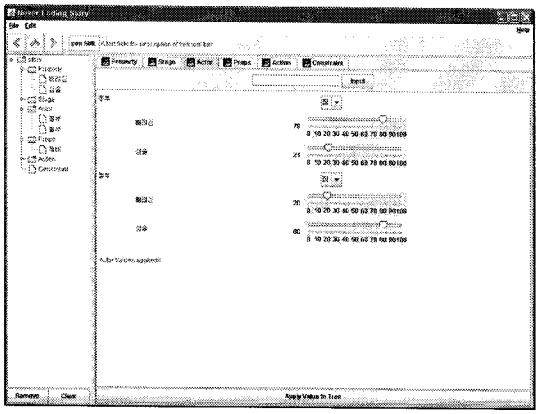


그림 6. 인터랙티브 스토리텔링 저작도구

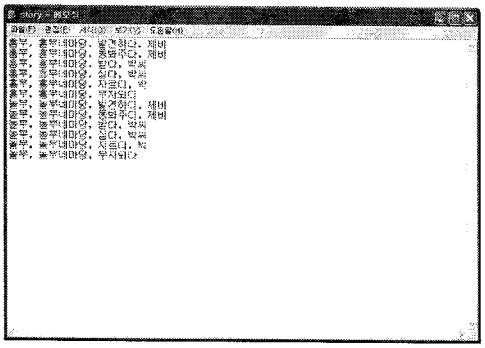


그림 7. 이야기 생성의 예

시킨 후 그것을 해석하는 출력 인터프리터와 결합한다면 그림, 애니메이션 등 다른 매체물로의 표현이 가능할 것이다. 현재 생성된 텍스트를 연속된 그림의 형태로 출력하는 출력인터프리터가 개발 중에 있다.

5. 결 론

본문에서는 인터랙티브 스토리텔링 시스템의 개념과 구조에 대해 알아보고, 서술 구조와 이를 표현할 언어를 제시하였다. 본 연구에서 제시한 서술 구조는 이야기를 생성하는 데 있어 이상적이라고 할 수는 없다. 다만 이런 식으로 구조를 만들고 이를 언어적으로 어떻게 표현할 수 있으며, 그렇게 표현된 언어를 이용하여 이야기를 생성하기 위해서는 또 무엇이 필요한가에 대해서 하나의 가능성을 제시한 것이다.

기존의 인터랙티브 스토리텔링 시스템에 비해 본 논문에서 제안한 서술 구조 및 SML은 다음과 같은 장점을 가지고 있다.

첫째, 기존의 연구는 응용프로그램에 치중한 나머지 언어에 대한 내용이 미흡한 점이 많았다. 그러나 본 논문은 언어에 대해서 연구함으로써 통합, 연결될 수 있는 시스템에 대한 확장가능성을 높였다.

둘째, 본 연구의 서술 구조는 필요에 따라서 추가 및 삭제가 가능하고, 새로운 서술 구조를 적용할 수도 있다. 하나의 예로서 토큰을 이용하는 데이타플로우 시스템을 응용하는 서술 구조를 만들 수도 있다.

셋째, SML을 이용하는 다양한 인터랙티브 스토리텔링 시스템의 개발이 가능해서, 게임 및 교육용 멀티미디어 시스템에 적용될 수 있다.

현재 본 연구는 세 가지 방향으로 진행이 될 예정이다. 첫 번째는 서술 구조와 언어의 기능을 향상시키는 것이고, 두 번째는 저작도구와 출력 인터프리터를 이용해서 결과물을 텍스트가 아닌 그림, 애니메이션과 같은 다양한 멀티미디어 매체로 생성할 수 있는 인터랙티브 스토리텔링 시스템을 구현하는 연구이다. 마지막으로는 SML을 응용하여 MMORPG (Massively Multiplayer Online Role Playing Game)에서 NPC (Non Playable Character)의 행동을 제어할 수 있는 스크립트 언어 및 제어시스템을 국내 모 게임사와 협력하여 진행하려고 하고 있다. 이와 같이 SML은 멀티미디어 및 게임 분야에 폭넓게 적용될 수 있는 언어로서의 가능성을 보여준다고 할 수 있다.

참 고 문 헌

[1] 이인화외 7명, "디지털 스토리텔링," 황금가지, 서울, 2003.

[2] F. Charles, S. J. Mead, and M Cavazza, "User Intervention in Virtual Interactive Storytelling," *Proceedings of VRIC 2001*, Laval, France, 2001.

[3] R. Fykes, and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence 2*, pp. 189-208, 1971.

[4] M. Cavazza, F Charles, and S. J. Mead, "Interacting with Virtual Characters in Interactive Storytelling." *ACM Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 318-325, 2002.

[5] L. M. Barros and S. R. Musse, "Introducing narrative principles into planning-based interactive storytelling," In *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pp. 35-42, 2005.

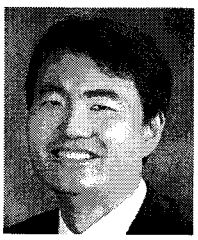
[6] S. Edelkamp and J. Hoffmann, "PPDL2.2: The language for the classical part of the 4th Internal Planning Competition," *Technical Report 195,* University of Dortmund, German, 2004.

[7] H. Prendinger, S. Descamps, and M. Ishizuka. "MPML: A markup language for controlling the behavior of life-like characters." *Journal of Visual Languages and Computing*, pp. 183-203, 2004.

[8] A.L.I.C.E. AI foundation, "Artificial Intelligence Markup Language (AIML)," *Technical Report*, URL: http://alice.sunlitsurf.com/TR/2001/WD-aiml/, 2001.

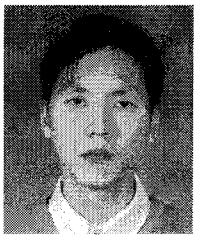
[9] E. Figa and P. Tarau, "The VISTAProject: An Agent Architecture for Virtual Interactive Storytelling," In *Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment*, p. 106, 2003.

[10] F. Charles, M. Lozano, S. J. Mead, A. F. Bisquerra, and M. Cavazza. "Planning formalisms and authoring in interactive storytelling," In *Proceedings of TIDSE'03: Technologies for Interactive Digital Storytelling and Entertainment*, pp. 216-225, 2003.



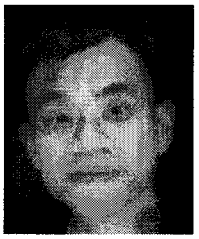
김 석 규

1988년 3월~1992년 2월 서울대학교 계산통계학과 이학사
1992년 3월~1994년 2월 서울대학교 계산통계학과 전산학 석사
1994년 1월~1996년 7월 현대전자 소프트웨어연구소 연구원
1996년 8월~1999년 6월 현대정보기술 연구원
1999년 7월~2003년 6월 엔씨소프트 팀장
2003년 9월~현재 서울대학교 컴퓨터공학부 박사과정
관심분야 : 디지털 스토리텔링, 컴퓨터 게임, HCI, 증강현실



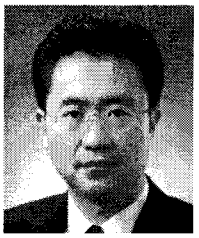
문 성 현

2002년 University of Maryland, Information System Management 학사
2007년 서울대학교 컴퓨터공학부 석사
2007년~현재 서울대학교 컴퓨터공학부 박사과정
관심분야 : 디지털 스토리텔링, 컴퓨터 게임, HCI, 증강현실



박 준

1993년 서울대학교 계산통계학과 학사
2001년 University of Southern California 전산학 박사
2001년~2002년 Rockwell International, Science Center Post-Doc Researcher
2002년~현재 홍익대학교 컴퓨터공학과 부교수
관심분야 : 증강현실, HCI, 의료영상, 컴퓨터비전



장 준 호

1990년 서울대학교 계산통계학과 학사
1992년 서울대학교 전산학 석사
1998년 서울대학교 전산학 박사
1998년~2003년 i2 Technologies 이사
2003년~현재 상명대학교 디지털미디어학부 조교수
관심분야 : 비즈니스SW, 미들웨어, 디지털 스토리텔링

한 상 영

1972년 서울대학교 응용수학과 학사
1977년 서울대학교 전산학 석사
1983년 University of Texas at Austin 전산학 박사
1977년~1978년 울산대학교 공과대학전임강사
1984년~2000년 서울대학교 계산통계학과 조교수, 부교수, 교수
2000년~현재 서울대학교 컴퓨터공학부 교수
관심분야 : 병렬처리, 보안, 디지털 스토리텔링