공동 작업을 위한 게임 디자인 툴

(Game Design Tools for Collaboration)

작성자: 데미안 자우티(Damien Djaouti)

작성일: 2013년 3월 5일

게임 디자인의 모형 제작 툴이 디자이너가 엔지니어들과 일하는 것을 쉽게 만들 수 있을까? 다양한 도구와 프레임워크가 나와있는 가운데, 연구원 데미안 자우티(Damien Djaouti)가 그 실현 가능성과 적응성을 분석한다.

이 기사는 게임 디자인과 소프트웨어 엔지니어링을 가깝게 만들어줄 수 있는 "공식적인 게임 디자인 툴"을 선정해 소개한다. 요즘에는 게임 디자이너들이 비디오 게임을 만들 때 이론적인 도구 세트에 의존할 수 있다. 이 도구들은 처음엔 디자인 단계를 용이하게 하기 위한 것이었다. 하지만, 이 중 일부는 디자이너의 아이디어를 나머지 팀원들, 특히 엔지니어들과 소통할 수 있는 강한 잠재력을 가지고 있다. 우리는 어떻게 비디오 게임 스튜디오의 디자인과 엔지니어링 부서 사이의 소통을 쉽게 하는지 논의하기 전에, 이런 툴의 네 가지 사례를 살펴볼 것이다.

서론

전통적으로 게임 디자이너는 자신의 작업을 나머지 팀원들에게 발표하기 전에 게임의 모든 측면의 세부사항에 대한 게임 디자인 서류를 작성한다. 하지만, 2010년의 연구[1]에서 드러났듯, 산업 내 게임 개발자들은 팀원들이 읽을 시간이 없는 지나치게 포괄적인 문서는 필요 없다고 보고했다.

대신 커뮤니케이션 요구와 규율 사이에서 맞춘 새로운 방법을 시험하고 있다. 한편, 게임 디자인에 대한 연구는 2000년대 초부터 많은 연구가 이루어져왔다.

---

[1] 참조링크: http://www.digra.org:8080/Plone/dl/display_html?chid=10343.03567.pdf

이러한 책들과 연구 기사들을 통해, 몇몇 게임 디자이너들은 게임 디자인을 공식화하기 위한 도구를 제안했다. 과연 이 "공식적인 게임 디자인 툴"이 게임 디자이너들이 그들의 아이디어를 게임 개발자들과 공유하는 데 도움을 줄 수 있을까?

이 질문에 답변하기 위해서, 우리는 1999년에서 2012년 사이의, 게임 디자인에 연관된 다량의 책과 기사들을 모았다. 이 책들 속에서, 우리는 게임 디자인의 많은 측면과 연관된 몇 개의 공식 모델을 찾았다. 몇몇 모델은 게임 구조를 형식화하려고 했다. 다른 것들은 플레이어의 행동을 형식화하려고 했으며, 심지어 소수의 도구들은 모든 플레이어-게임 관계를 모델링하려고 했다.

이 기사를 통해, 우리는 디자이너가 게임 구조를 모델링할 수 있는 네 개의 모델을 강조할 것이다. 우리는 이런 도구들이 어떻게 개발 스튜디오의 내부 커뮤니케이션 도구로 활용될 수 있는지를 논하기 전에 우선 이 도구들을 소개할 것이다.

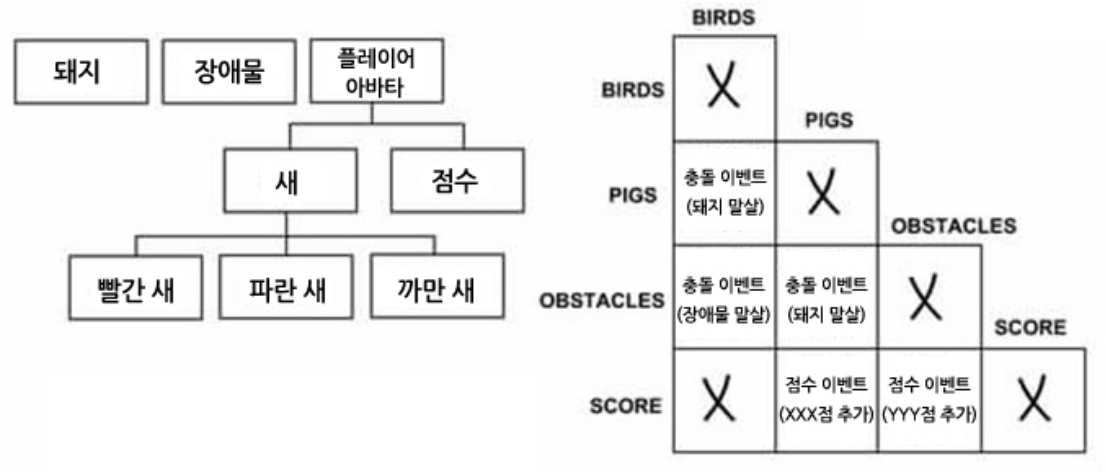
토큰 기반 모델(A Token-Based Model)

이 모델의 핵심 아이디어는 "토큰(tokens)"이라 불리는 기초 조각으로 게임 구조를 구축하는 것이다. 앤드류 롤링(Andrew Rollings)과 데이비드 모리스(David Morris)가 그들의 책『게임 구조와 디자인(Game Architecture and Design[2])』에서 설명했듯, 토큰은 게임을 구성하는 기본 요소이다. 따라서, 토큰은 모든 게임 요소의 계층적 구조를 구축하는 데 사용될 수 있다.

예를 들어, <앵그리 버드(Angry Birds)>(로비오 모바일(Rovio Mobile), 2009)는 네 개의 토큰으로 나뉠 수 있다. "새(Birds)", "돼지(Pigs)", "장애물(Obstacles)", 그리고 "점수 기록대(Score Counter)". "점수 기록대"와 "새"는 더 큰 "플레이어(Player)" 토큰의 일부분이다. "새", "돼지", 그리고 "장애물"은 몇 개의 서브-토큰(sub-tokens)으로 나뉠 수 있다. "빨간 새", "파란 새", "검은 새", "나무 장애물", "얼음 장애물"... 이런 토큰 세트를 이용하면, 게임에서 일어날 수 있는 모든 "토큰-토큰 상호작용"을 모델링하기 위한 상호작용 매트릭스(interaction matrix)를 쉽게 만들어낼 수 있다.

---

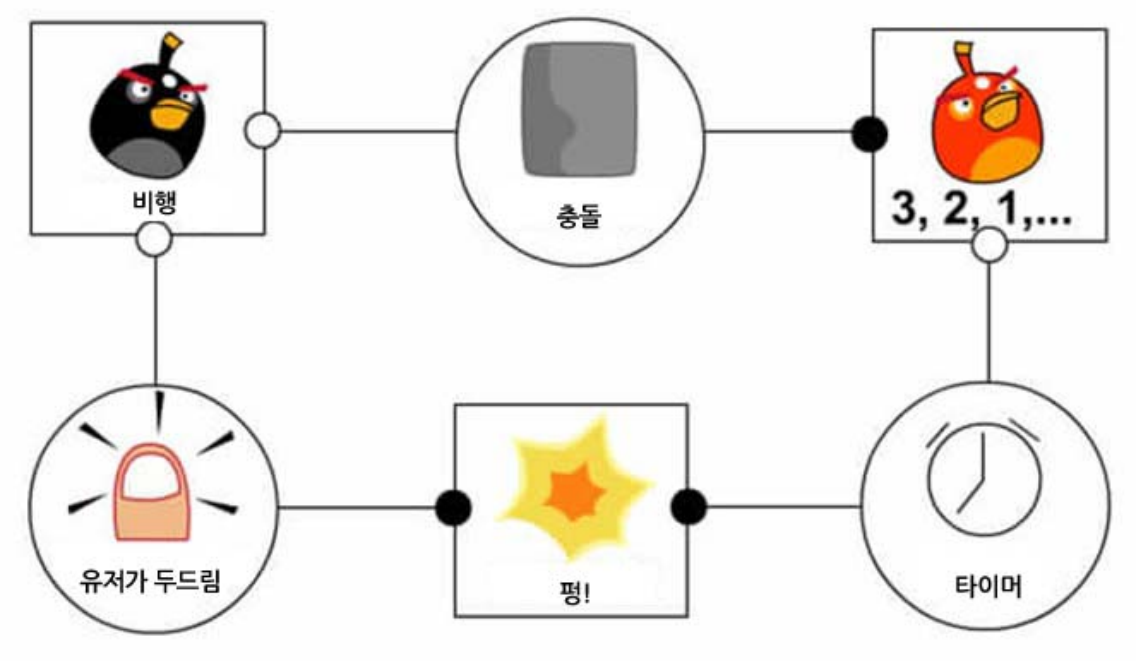
[2] 참조링크: http://www.amazon.com/Game-Architecture-Design-Andrew-Rollings/dp/0735713634



<앵그리 버드>의 상호작용 매트릭스와 토큰 체계 추출

두 저자가 언급했듯, 이 모델은 간단한 게임에는 잘 맞지만, 게임이 조금만 더 정교해지면 사용하기가 어렵다. 따라서, 그들은 더 복잡한 게임 구조를 다루기 위해 토큰 모델의 변형을 소개했다. 이 변형은 앞서 언급한 아이디어는 유지하지만, "가변적인 상태 토큰"을 소개한다.

각 "토큰"은 이제 "한정된 상태 기계(Finite State Machine)"이다. 이는 게임의 과정 동안 다른 상태를 가질 것이다. 그들은 각 토큰의 상태 변화를 대표하기 위해 도식화된 접근법을 제안한다. 하단의 <앵그리 버드> "검은 새" 토큰을 예로 제시했다. 상호작용 매트릭스와 상태 다이어그램 둘 다 이용함으로써, 게임 디자이너들은 완성된 비디오 게임의 구조를 형식화할 수 있어야 한다. 상호작용 매트릭스와 상태 다이어그램이 소프트웨어 엔지니어링 문화의 일부이기 때문에, 이 도구는 디자이너들이 그들의 창작품에 대해 프로그래머들과 소통하는 데 도움이 될 것이다.



"검은 새"토큰에 대한 상태 기계 다이어그램

다이어그램 기반 전략(Diagram-Based Machinations)

게임을 "기초 조각"으로 나누는 아이디어는 또한 라프 코스터(Raph Koster)의 "루뎀스(ludems[3])"와 같은 보다 복잡한 모델의 골조이기도 하다. 코스터에게 영감을 받아, 스테판 버라(Stéphane Bura)는 다이어그램으로 게임 구조를 설명하는 시스템을 소개[4]했다. 그는 자신의 모델을 위한 기반으로 페트리 넷(Petri net[5]) 모델링 언어를 사용했다.

사실, 그는 이 언어의 모든 요소를 게임 디자인과 관련 있게 만들기 위해 이를 "장악했다(hijacked)". 따라서 페트리 넷의 "토큰"은 이제 자원을 묘사하는 용어로 쓰이며, "장소(Places)"는 게임 요소를, 그리고 "이행(Transitions)"은 플레이어의 모든 행동을 정의한다.

그럼에도 불구하고 버라의 모델은 페트리 넷을 알고 있는 사람들에겐 흥미롭겠지만, 우리는 대부분의 소프트웨어 엔지니어들에게 조리스 도먼스(Joris Domans)의 모델이 이해하기 더 쉬울거라고 생각한다. 사실, 이 모델은 통합 모델링 언어(UML[6]/

---

[3] 참조링크: http://www.theoryoffun.com/grammar/gdc2005.htm

[4] 참조링크: http://www.stephanebura.com/diagrams

[5] 참조링크: http://en.wikipedia.org/wiki/Petri_nets

[6] 참조링크: http://en.wikipedia.org/wiki/Unified_Modeling_Language

Unified Modeling Language)에 기반한다.

이는 "협업 다이어그램(collaboration diagrams)"에 기대 각 게임 요소 사이의 상호작용을 설명한다. 도먼스의 첫 모델[pdf[7]]은 UML 표준을 엄격하게 따랐다. 이는 UML에서 영감을 받은 다이어그래밍 게임 구조의 관습적 방식으로 발전된다[8]. "전략(Machinations)"로 불리는, 이 모델은 예를 들어 게임의 자원이 플레이 하는 동아 발전되는 게임 같은, 국내 경제에 맞춘 디자인 게임에 잘 맞는다.

첫 단계는 게임 요소를 나타내는 "자원 풀(resource pools)"을 만들어내는 것이다. 그러고 나서, 풀은 네 가지 주요 방법에 따라 서로 상호작용할 수 있는데, 자원 생산, 자원 소모, 자원 전환, 혹은 자원 교환이 그것이다. 어니스트 아담스(Ernest Adams)의 도움으로, 도먼스는 최근 이 모델에 기반한 책 한 권[9]을 썼는데, 그 내용을 발췌한 것을 가마수트라에서 확인할 수 있다[10].

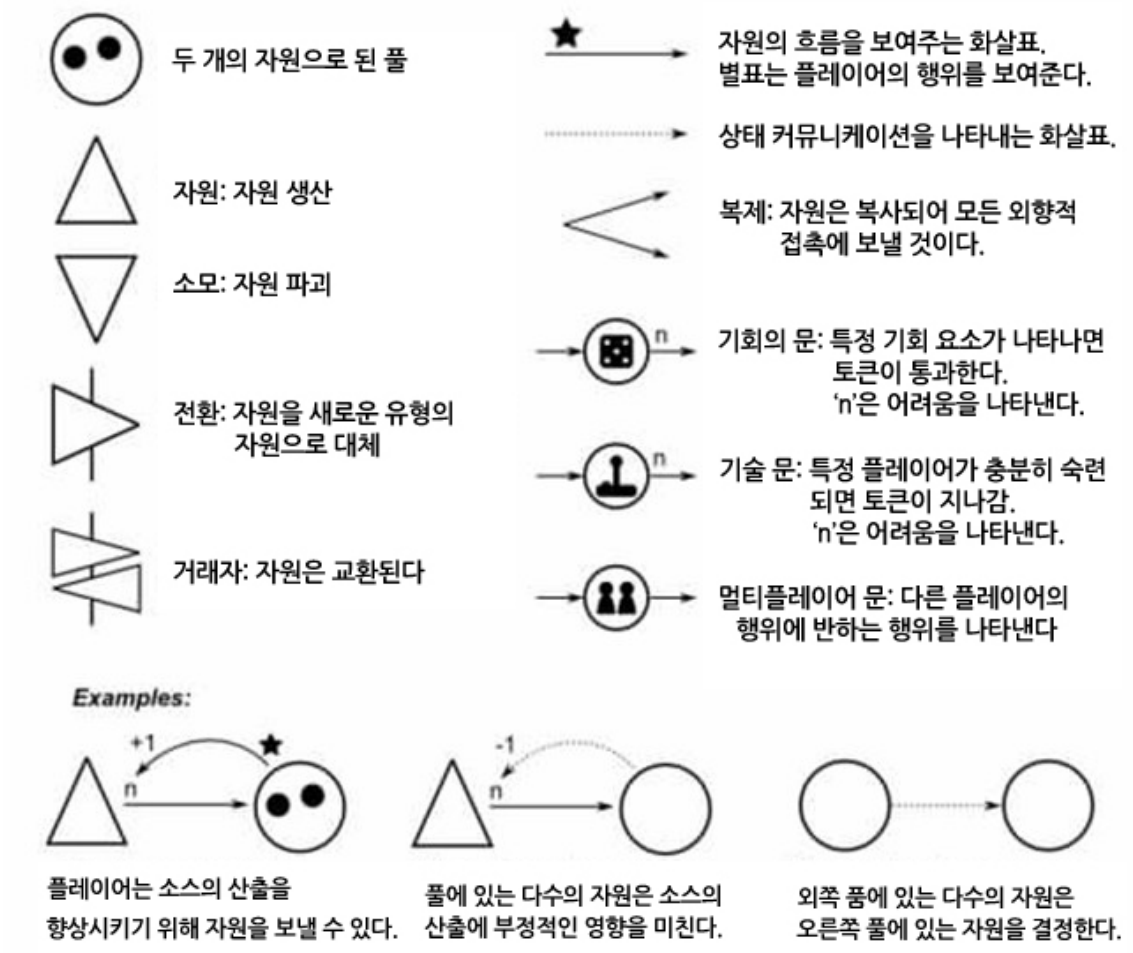
---

[7] 참조링크: http://meaningfulplay.msu.edu/proceedings2008/mp2008_paper_40.pdf

[8] 참조링크: http://www.jorisdormans.nl/article.php?ref=machinations

[9] 참조링크: http://www.amazon.com/Game-Mechanics-Advanced-Design-Voices/dp/0321820274/ref=sr_1_1?ie=UTF8&qid=1362430543&sr=8-1&keywords=Ernest+Adams+and+Joris+Dormans%2C+Game+Mechanics%3A+Advanced+Game+Design

[10] 참조링크: http://www.gamasutra.com/view/feature/176033/the_designers_notebook_.php



도먼스의 책략(Machinations)에서 사용되는 다이어그램 요소 목록

단순한 이론적인 모델을 넘어, 도먼스는 이러한 다이어그램을 그리는 소프트웨어 도구 역시 구축했다. 다이어그램이 그려지면, 게임 디자이너는 그를 테스트해보고 자원이 게임 플레이 시뮬레이션에서 어떻게 따라오는지 볼 수 있다. 이 도구가 어떤 코드도 만들어내진 않는다 해도, 이는 게임디자이너들이 그들의 디자인을 공식화하고 일종의 "자동 베타 테스팅"을 실행할 수 있게 해준다. 그러고 나서 결과로 나온 다이어그램은 개발팀에게 넘어가 디지털 프로토타입을 만들 수 있다.

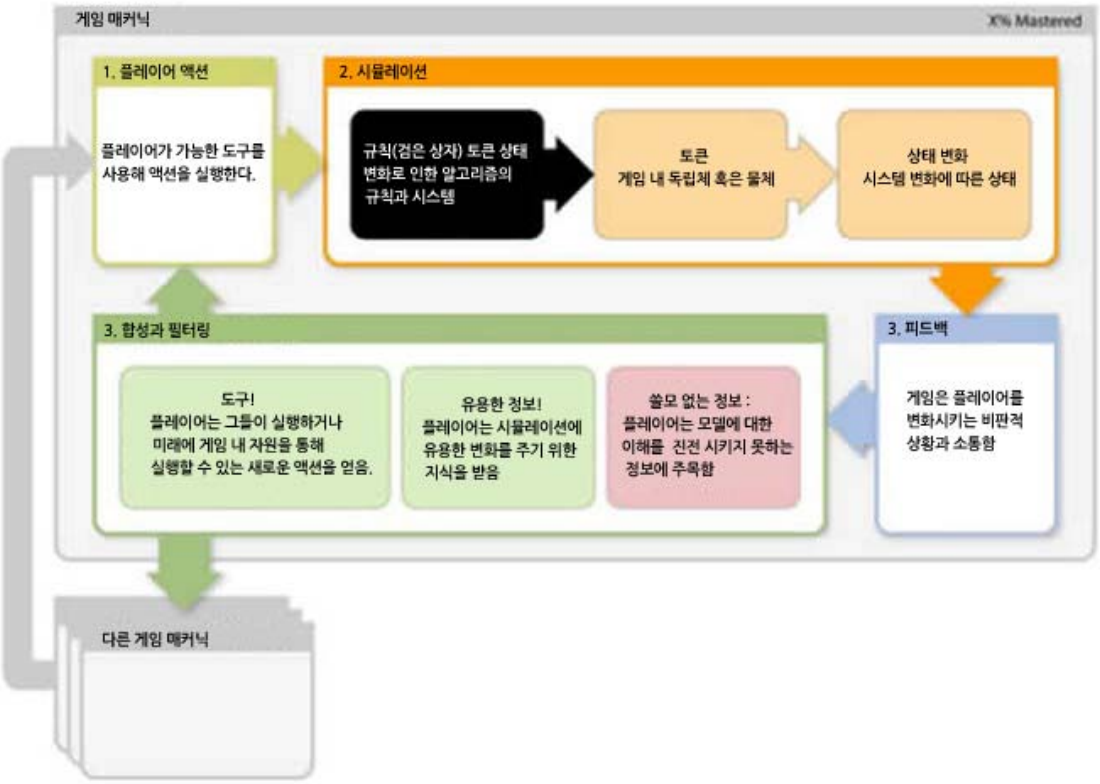
게임 연금술(A Game Alchemy)

다이어그램 아이디어에서 더 나아가, 다니엘 쿡(Daniel Cook)은 게임 구조뿐 아니라 플레이어-게임 상호작용까지 설명할 수 있는 도구를 제안[11]했다. 그의 모델을 구축하기 위해, 그는 플레이어를 "인지된 가치에서 높고 새로운 기술을 배우기 위해, 의

[11] 참조링크: http://www.gamasutra.com/view/feature/1524/the_chemistry_of_game_design.php

식적, 잠재의식적으로 운영되는 독립체. 그들은 성공적으로 기술을 획득하는 데서 즐거움을 얻는다."라고 정의한다. 이 정의는 코스터가 자신의 책 『재미 이론(A Theory of Fun[12])』에서 표현한 "재미(fun)"에 대한 관점과 유사하다. "내가 정의하는 재미는, 우리가 배우기 위해 새로운 패턴을 흡수할 때 뇌가 우리에게 주는 피드백이다."

쿡은 게임 구조를 "매커닉(mechanics)"으로 나눈다. 각각의 "매커닉"은 플레이어가 배운 "기술"과 관련된다(예를 들어 마리오(Mario)가 100코인을 모으면 추가 목숨을 얻는 것과 같은). 쿡에 의하면, "기술"은 단순히 게임에 이기는 법을 이해하기 위해 플레이어가 찾아내야 하는 정보 덩어리이다. 각각의 "매커닉"은 하나의 "기술"을 배우는 것과 관련이 있다. 이 "매커닉"들은 다음 도형에 언급되는 네 가지 단계로 구성되어 있다.



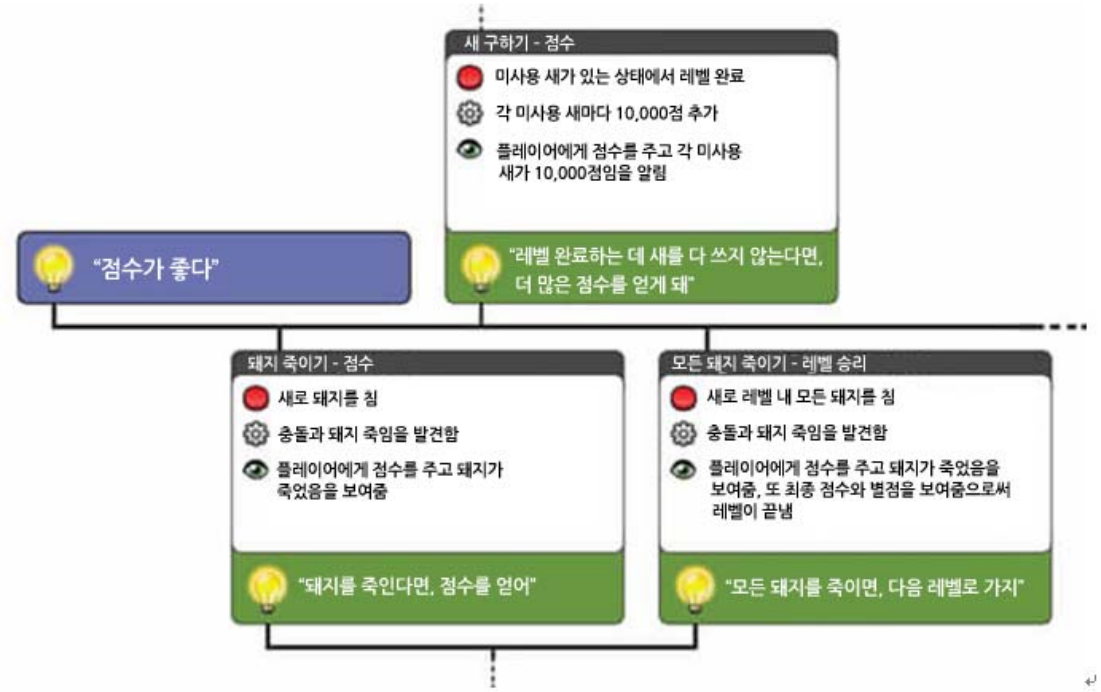
게임 연금술로부터의 매커닉에 대한 쿡의 상세한 견해

쿡은 "매커닉"을 다이어그램 유형에서 전체 플레이어/게임 구조를 설명하기 위한 기초 조각으로 사용했다. 각 매커닉은 "직선적이지 않은 기술 학습 체인"을 만들기 위

[12] 참조링크: http://www.amazon.com/Theory-Fun-Game-Design/dp/1932111972

한 다른 매커닉과 함께 연결될 수 있는 일종의 "원자(atom)"이다. 이와 같이, 이 도구는 게임 규칙 사이 그리고 게임과 플레이어 사이의 연결을 모두 보여주는 공식화된 구조를 만들어낼 수 있다.

이는 게임 디자인이 팀의 모든 부서와 소통하는 편리한 방법이다. 예를 들어, 개발자들은 이를 비디오 게임의 소프트웨어 아키텍처를 구축하기 위한 "상의하달식(top-down)" 접근법을 지원하는 데 사용할 수 있다. 그들은 모든 매커닉의 개요와 함께 일 층짜리 아키텍처를 구축하고 나서 각 메커닉의 내부 아키텍처를 따로따로 정제할 수 있다. 플레이어 연결 기술에 대한 정보를 제공함으로써, 이 도구는 또한 베타 테스트 동안, 게임의 최근 반복을 통해 플레이어가 성공적으로 배운 (혹은 배우지 못한) "기술"이 무엇인지 보기 위한 참고 가이드가 될 것이다.



쿡의 게임 연금술을 이용한 <앵그리 버드> 표현에서 발췌

게임 층(Game Layers)

게임 디자인을 공식화하는 다른 접근법은 디자이너가 게임을 만들 때 그의 주의를 집중시키기 위한 '슬라이스(slices)'로 나누는 것이다.

이 슬라이스들은 내부 게임 구조부터 플레이어 경험까지 비디오게임과 관련된 어떤 것과도 관련이 있다. 각 모델에는 다양한 예시가 존재한다.

가장 간단한 것은 MDA 모델[pdf[13]]이나 케이티 살렌(Katie Salen)과 에릭 지머만(Eric Zimmerman)의 주된 개요(Primary Schemas[14])처럼 세 개의 슬라이스를 이용한 것이다. 아키 자브리넨(Aki Jävrinen)의 "게임 요소(game elements[15])"는 9개의 서로 다른 게임 슬라이스를 이용함으로써 더 자세한 사항들을 제공하기 위한 진화된 MDA 모델이다. 이 모델들은 심하게 세세해질 수도 있는데, 제시 셸(Jesse Schell)이 그의 책 『게임 디자인의 기술(The Art of Game Design)』에서 보여준 "렌즈(lenses)"는, 백 개의 슬라이스[16]를 제공한다.

이 "슬라이스 기반" 도구들 가운데서, 아키텍처에 가장 잘 맞는 것 중 하나는 파올로 타제(Paolo Tajè)가 제안[17] 했다. 그의 모델은 게임-플레이어 관계를 6개의 "층(layers)"으로 나눈다. 그의 주의를 각 층에 따로따로 집중하면서, 게임 디자이너는 보다 쉽게 게임 디자인을 만들거나 수정할 수 있게 될 것이다. 이 층들은 정확한 방식으로 "게임"에서 "플레이어"까지 정렬되어 있다.

- 맨 아래의 **토큰**은 모든 게임 요소에 참조되는 층이다(즉 롤링과 모리스가 정의한 "토큰"을 의미한다)
- **받침대(Prop)** 층은 토큰의 다른 "속성(properties)"을 포함한다. 우리는 여기서 객체 지향 패러다임(Object-Oriented paradigm)과 높은 유사성을 발견했다. "토큰"은 "객체", "받침대"는 그들의 내장된 "수단(methods)"이 될 것이다.
- **동력(Dyn)**은 플레이어가 실행할 수 있는 모든 액션을 열거한다.
- **목표(Goal)**은 플레이어가 게임에서 승리하기 위해 성취해야 하는 모든 목표를 열거한다.
- **메타(Meta)**는 이전 층에서 정의되었지만, 여전히 플레이어 경험을 형성하는데 활발한 역할을 하는 다른 모든 요소를 참조한다.
- 마지막으로 하지만 중요한 **사이코(Psycho)**는 플레이어가 플레이 하는 동안 느낄 수 있는 어떤 감정이든 요약한다.

---

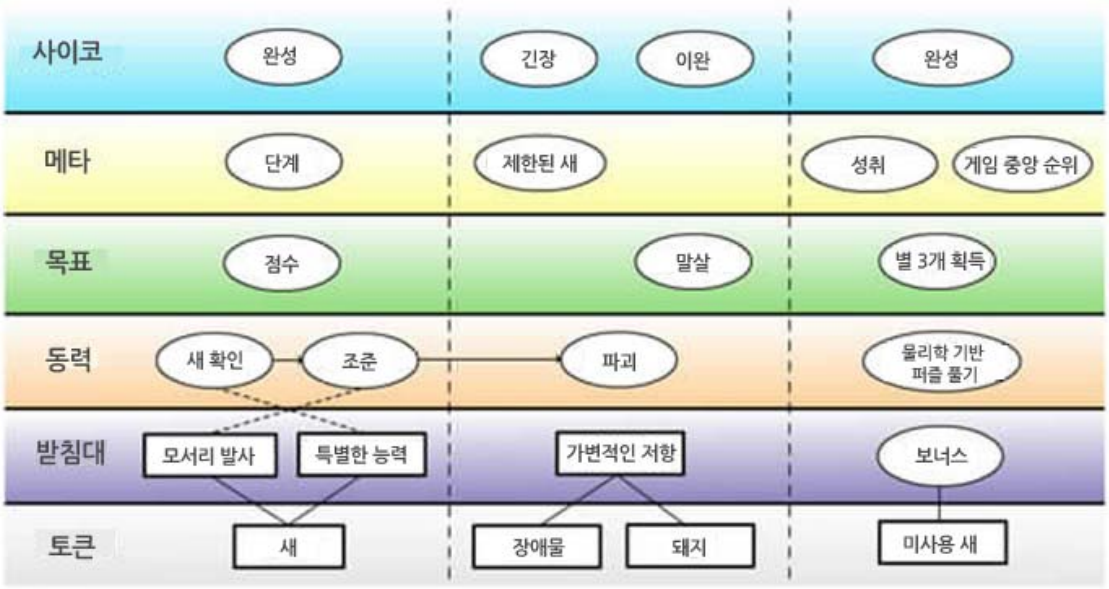
[13] 참조링크: http://www.zubek.net/robert//publications/MDA.pdf

[14] 참조링크: http://www.amazon.com/Rules-Play-Game-Design-Fundamentals/dp/0262240459/ref=sr_1_1?ie=UTF8&qid=1362431895&sr=8-1&keywords=rules+of+play

[15] 참조링크: http://urn.fi/urn:isbn:978-951-44-7252-7

[16] 참조링크: http://www.amazon.com/The-Art-Game-Design-lenses/dp/0123694965/ref=sr_1_1?ie=UTF8&qid=1362432237&sr=8-1&keywords=schell+lenses

[17] 참조링크: http://www.gamecareerguide.com/features/355/gameplay_deconstruction_elements_.php



타제의 게임 층을 이용한 <앵그리 버드> 표현

이 모델은 게임에서 디자이너들이 잠재된 디자인 결점을 발견하도록 도와주는 서로 다른 요소들을 개략적으로 제공한다. 모든 게임 특징을 하나의 문서에 요약하고 카테고리화함으로써, 이 도구는 다른 부서와 게임 디자인에 대해 빠르게 소통하기 쉽게 해준다. 우선, 토큰, 받침대, 동력, 목표, 그리고 메타 층은 소프트웨어 엔지니어가 디지털 프로토타입을 구축하는 데 참고자료로 활용될 수 있다. 그리고 사이코 층은 베타 테스트 동안 플레이어들이 기대된 감정을 느끼는지 확인할 수 있게 해준다.

논의(Discussion)

이 아키텍처 도구들은 모두 공식화된 방법으로 게임 구조를 제시하는데 약간의 도움을 준다. 각 모델이 다양한 단계의 세부 사향과 복잡성을 특징으로 한다 해도, 이 기사에서 제시한 네 모델들은 게임 디자이너가 그의 창작물을 소프트웨어 엔지니어에게 설명해야 할 때 매우 유용할 수 있다.

이 기사의 서론에서 이야기 했듯, 업계의 몇몇 게임 개발자들은 전통적인 게임 디자인 서류를 대체하거나 보완한 새로운 도구를 찾고 있다. 나는 이 기사에서 제시한 모델들이 게임 스튜디오의 디자인 부서가 개발 부서와 소통하는 편리한 방법이 될 수 있다고 생각한다. 이러한 공식적인 모델들을 사용함으로써, 게임 디자이너는 어느 정도 자기 작업을 소프트웨어 엔지니어가 사용하는 형태로 만들 수 있다.

몇몇 모델들이 (UML이나 페트리 넷과 같은) 이전에 존재했던 표준에 기반하지 않았다 해도, 그들은 모두 프로그래밍 언어와 관련된 관념의 과정에, 특히 "객체 지향 프로그래밍 패러다임"에 영향을 받았다. 정말, 이 모든 도구들이 게임을 분명하고 자율적인 요소로 나누었으며, 이 요소들이 서로 어떻게 상호작용했는지 명확하게 이야기한다. 아래의 표는 이 모델들의 잠재적 사용에 관한 나의 관찰을 요약한 것이다.

네 개의 "공식적인 게임 디자인 툴" 비교.

	"토큰"	"전략"	"연금술"	"층"
게임 요소를 설명하는 것	"토큰"	"자원 풀"	"토큰"	"토큰"
규칙을 설명하는 것	상호작용 매트릭스	"자원 풀"사이의 연결	"매커닉"	"받침대", "동력", "목표" 층
플레이어의 내적인 생각을 설명하는 것	-	-	획득할 "기술"	"사이코" 층의 감정
플레이어-게임 상호작용을 설명하는 것	-	링크의 "별점"과 "관문"	"기술"과 "매커닉" 사이의 연결	"토큰"과 관련 있는 "속성"
추천 게임 장르	액션, 아케이드, 퍼즐	경영, 전략, RPG	무관	무관
추천 복잡성 (즉 규칙의 양)	단순한 게임	복잡한 게임	복잡한 게임	단순한 게임
추천 소통 사용	*게임 디자인에서 엔지니어링	*게임 디자인에서 엔지니어링	*게임 디자인에서 엔지니어링 *게임 디자인에서 테스팅	*게임 디자인에서 엔지니어링 *게임 디자인에서 테스팅
강점	사용 편의성	경제적인 모델에 특화됨	철저함	간결한 문서 생성
약점	단순한 게임에 제한됨	UML에 친숙하지 않으면 배우기 어려움	매우 많은 문서 생성	규칙에 대한 깊은 설명에 약함

이 표에서 제안했듯, 네 모델은 개개 게임 프로젝트에 유용할 수 있다. 예를 들어, <팜빌(FarmVill)>과 같은 게임을 만들기 위해, 게임 디자이너는 모든 게임 요소를 "토큰" 으로 요약(작물, 동물, 토지 단위...)하는 데서 시작할 수 있다. 그러고 나서 상호작용 매트릭스를 통해 토큰 사이의 기초 상호작용을 정의할 수 있다.(작물은 토

지 단위 위에 심어진다 등...)

하지만, 각 게임의 꽤 복잡한 경제 모델에는 전략 모델이 더 쉬울 것이다(예를 들어 자원의 비율은 작물의 각 유형마다 서로 다른 시간을 제공하는 식으로 한다 등...). 이 두 모델 덕분에 소프트웨어 엔지니어는 좀 더 쉽게 게임의 실행 프로토타입을 구축할 수 있다. 연금술과 층 모델은 디자이너가 베타 테스트를 위한 가이드 라인을 형식화할 수 있게 해준다. 연금술이나 층 모델 중 어떤 것을 사용할지는 게임 내부의 복잡성에 달려 있다.

또 다른 접근법은 각 모델에 의존한다. 예를 들어, 연금술은 엔지니어링과 테스팅 부서 모두와 소통하는데 사용할 완전한 문서를 만들 수 있다. 사실, 이 도구는 "매커닉" 시스템을 통해 각 규칙과 게임의 요소를 세분화할 수 있다. 이는 엔지니어를 안내할 것이다. 이 "매커닉"은 "기술" 습득으로 구성된 "기대 플레이어 경험"과 연관될 수 있다. 이 정보는 베타 테스터에게 참고 자료로 활용될 수 있다.

결론

현재 시점에서, 이 기사에 제시된 어떤 도구도 게임 개발 스튜디오에서 광범위하게 사용되고 있진 않아 보인다. 몇몇 전문가들은 이를 터무니 없다고 생각하기도 하지만, 우리는 이 상황이 대체로 두 가지 요인 때문이라고 생각한다. 이러한 모델의 존재에 대한 무지, 그리고 게임 창작 과정에 맞추기 어려움.

게임 디자인과 관련된 네 개의 공식 모델을 선택해 제시함으로써, 나는 이런 이론적 도구의 존재를 사람들에게 알리는 데 도움이 되었으면 한다. 이를 게임 창작 과정에 맞추는 방법에 대해서는, 두 가지 경우에 유용할 수 있다고 생각한다. 우선, 이는 "정의된 골조"로 디자이너를 안내함으로써 게임 디자인을 만드는 것을 도와줄 것이다.

덧붙여, 이 도구들은 비디오 게임의 소프트웨어 아키텍처를 구축하는 데 도움이 된다. 그것은 게임 디자이너가 객체 지향 페러다임과 유사한 형식으로 그의 작업을 표현하게 해준다. 이는 고체 아키텍처를 구축하는 데 있어 게임 디자이너가 소프트웨어 엔지니어와 소통하는 것을 도와줄 것이다. 이 도구 중 몇몇은 심지어 플레이어의 기대 반응을 모델링함으로써, 디자이너들이 테스팅 부서에 분명한 가이드라인을 제시할 수 있도록 도와준다. 따라서, 공식적인 게임 디자인 도구는 전통적 디자인 서류를 대체할 새로운 소통 도구를 실험하고자 하는 게임 스튜디오에게 해결책

이 될 것이다.

하지만, 지금 이 도구들은 이론에 머물고 있으며 직접적인 코드 생성을 하지는 못한다. 게임 디자인 공식화의 다음 단계는 이 모델에 기반한 프로토타이핑 소프트웨어를 만들기가 될 것이다. 사실, IBM의 래셔널 로즈(Rational Rose)와 같은 프로그램을 통해, 소프트웨어 엔지니어는 UML 다이어그램만 그리면 자동적으로 코드를 생성할 수 있다. 따라서, 게임 디자이너들이 비디오 게임을 위한 비슷한 도구에 접근해 보는 것은 어떨까?

우리는 이미 몇몇 아마추어 게임 창작 도구를 관찰해보았는데, 게임 메이커(GameMaker[18])나 게임 팩토리2(The Games Factory 2[19])와 같은 것들은 종종 업계에서 프로토타입을 구축하는데 사용되기도 한다. 그들은 이 기사에 소개된 도구들과 비슷한 컨셉을 가지고 있다.

예를 들어, 게임 규칙 창작을 쉽게 하기 위해, 게임 팩토리2는 거대 표를 특징으로 하는데, 이는 롤링과 모리스의 "토큰 상호작용 매트릭스"와 유사하다. 하지만, 현재 시점에서, 어떤 프로토타이핑 소프트웨어도 사실상 기존의 이론적 모델에 기반하지 않는다. 이런 이유로, 몇몇 연구자들은 새로운 확고한 이론적 모델에 기반한 소프트웨어 도구를 구축하기 위한 실험을 하고 있다.

나는 이미 전략[20]에 기반한 소프트웨어 도구를 언급했다. 이는 잠재적 디자인 결함을 확인하기 위한 몇몇 자동 시험을 실행할 수 있지만, 아직 플레이 가능한 프로토타입을 만들 수는 없다. 또 다른 예시는 루도코어(Ludocore)[pdf[21]]인데, 이는 이벤트 미적분 논리 언어(event calculus logical language)를 사용함으로써 플레이 가능한 게임 프로토타입을 구축할 수 있는 도구이다. 강력한 이론적 토대 적분에, 이는 유용한 "게임플레이 추적(gameplay traces)"를 생성한다. 이 도구는 플레이어의 행위를 기록해 이 도구를 사용하는 디자이너가 게임의 규칙을 향상시킬 수 있도록 도와준다.

나 역시, 연구원 팀과 함께 이 분야의 몇 가지 실험을 하고 있다. 우리의

---

18 참조링크: http://www.yoyogames.com/gamemaker/studio

19 참조링크: http://www.clickteam.com/website/world/the-games-factory-2

20 참조링크: http://www.jorisdormans.nl/machinations/

21 참조링크: http://games.soe.ucsc.edu/sites/default/files/ieeecig10_ludocore.pdf

"Gam.B.A.S.[22]" 도구는 게임 구조를 "게임플레이 벽돌(GamePlay bricks)"라고 불리는 기초 조각으로 나누는 이론적인 모델을 기반으로 한다. 우리는 최근 더 나은 소프트웨어 도구를 구축하기 위해 이론 모델 개선 버전을 작업하고 있다.

이런 도구의 디자인은 연구 개발에 많은 투자를 필요로 하는 실험적인 일이다. 그래픽이나 물리학 관련 연구와는 다르게, 새로운 전문 디자인 도구 창작은 상업적 비디오 게임을 위한 고유의 장점으로 전환하기 힘들다. 그래서, 이 분야는 비디오 게임 관련 연구에 관여하는 대학 연구소에게 기회로 떠오르고 있다. 소수의 학업적 프로젝트가 이미 존재하지만, 우리는 많은 다른 연구자들과 산업 전문가들이 "공식적인 게임 디자인 툴" 분야를 탐험하는 데 관심을 가지게 되길 바란다.

[22] 참조링크: http://www.ludoscience.com/EN/diffusion/257-Towards-a-classification-of-Video-Games.html