※ 본 아티클은 CMP MEDIA LLC와의 라이선스 계약에 의해 국문으로 제공됩니다

Gamasutra.com

# 상호작용성 디자인 패턴을 통해 유용성과 접근성이 뛰어난 게임 디자인하기

Eelke Folmer

2007년 5월 17일

http://www.gamasutra.com/view/feature/1408/designing_usable_and_accessible_.php


게임과 다른 엔터테인먼트 매체를 갈라놓는 것은 바로 상호작용성이다. 그렇지만 잘못된 방법으로 상호작용을 가능케 하는 것은 곧 사람들이 게임을 플레이하면서 실망을 하게 된다는 것을 뜻하며, 더 끔찍한 경우에는 게임을 제대로 플레이하지도 못 하는 경우가 있을지도 모른다.(예를 들어 플레이어가 생각하거나 요구하는 것과 다르게 작동한다던가)

이제는 “20 대 남성”이 아닌 사람들도 게임에 관심을 보이기 시작하고 있으며 이 말인 즉 이때까지 게임 산업에서 목표로 삼아온 대상이 바뀐다는 뜻이다. 콘솔에서 FPS 를 플레이 하기 위해서는 아날로그 스틱과 4 개의 버튼, 그리고 몇 개의 트리거를 다루는 방법을 알아야만 하고 때로는 이러한 것들을 조합하여 조작할 수도 있어야 한다. 모든 사람이 이러한 것을 쉽게 해내지는 못하며, 특히 노인이라던가 이때까지 게임을 해보지 못한 사람이라던가, 장애가 있는 사람 등은 점점 어려워지고 있는 게임들에 쉽게 접근하지 못하고 있는 것이다.

워드 프로세서가 누군가가 편지를 쓰는 것을 돕는 것처럼 대부분의 소프트웨어 시스템은 사용자의 일을 자동적으로 처리하기 위해서나

일을 하는데 도움을 주기 위하여 디자인되어있다.(ATM 소프트웨어 등) 이러한 점에서 게임들은 전통적인 소프트웨어 시스템과 다르다고 할 수 있으며 교육 또는 엔터테인먼트를 위해서만 개발된다는 점에서도 차이를 보인다.

상호작용성 디자인은 게임의 두 가지 요소에 영향을 끼친다.

- *유용성(Usability)*: 플레이어가 어떻게 게임을 플레이 해야 할 것인지를 파악하지만 못할 경우, 플레이어가 기다려야 하는 경우, 플레이 방법을 익히기 어렵거나 게임 오브젝트가 사용이 힘들 경우.
- *접근성(Accessibility)*: 플레이어가 청각 장애로 인해 컷신에서 무엇을 말하고 있는지 이해하지 못 하거나 누군가 자신의 뒤로 다가오고 있는 발소리를 듣지 못 하는 경우, 게임이 신체적 장애가 있는 플레이어를 위해 한 손 컨트롤러나 불기와 빨기를 통해 조작이 가능한 조이스틱을 지원하지 않는 경우.

유용성과 접근성은 서로 다르지만 매우 연관성이 깊은 요소이다. 접근성 문제는 특정 플레이어 그룹에 있어서는 유용성 문제가 될 수도 있다.(장애를 지닌 플레이어 등) 컷신에서 무슨 말을 하는지 이해할 수 없는 것은 소리를 들을 수 없는 사람에게는 접근성 문제이지만, 헤드폰 없이 시끄러운 환경에서 휴대용 게임을 즐기는 사람에게는 유용성 문제라고 할 수 있다.

이 문서에서는 유용성을 증대시키고 장애를 지닌 플레이어들에게 도움이 되는 해결책들에 대해 논의할 것이다. 게임의 접근성과 유용성은 게임플레이와 혼동되어서는 안 된다. 게임플레이는 재미있는 방향으로 상호작용성을 제공하는데 초점이 맞추어져야만 한다. 예를 들어 *퐁(Pong)*에서 막대를 움직여 공을 튕기는 것을 생각해보라. 유용성과 접근성과는 별개로 플레이어가 생각하는 그대로 상호작용이 이루어진다. 예를 들어 조이스틱을 위로 올리면 막대도 위로 올라가며 복잡하게 조작할 필요가 없다.

## 2. 게임에서의 유용성 & 접근성 문제

일반적인 유용성 & 접근성 문제에 대해 살펴보도록 하자. 우리는 이러한 문제들을 5 개로 분류하였다.

### 플레이어가 기다려야 한다

- 플레이어가 이미 본 적이 있는 컷신을 봐야만 하고 넘길 수가 없다.(재플레이 시에도 같은 문제가 발생)
- 뭔가를 완료하기 위하여 많은 시간을 기다려야 한다.(예: RTS 에서 건물을 건설하는 것)
- 게임을 시작하기 전에 플레이어가 많은 선택을 해야만 한다.(예: RPG 에서 캐릭터를 만들고 설정을 해줘야 한다)
- 플레이어가 실수를 하면 레벨 또는 세이브를 불러올 때까지 기다려야 한다.

### 플레이어가 실수를 한다

- 플레이어가 경험 부족이나 능력 부족으로 반복적으로 죽게 된다.(예: 결코 이길 수 없는 마지막 레벨 보스와 싸우는 것)
- 플레이어가 저장을 깜박해서 예전에 저장된 게임을 해야만 한다.
- 플레이어가 짧은 시간 내에 여러 행동을 해야 하는데 반복적으로 실패한다.(예: 버튼을 눌러 문을 열고 구멍을 뛰어넘은 후 적과 싸우고 문 밑으로 굴러들어가기)

### 게임이 플레이어에게 적합하지 않다

- 청각 장애가 있는 플레이어나 시끄러운 환경에서 휴대용 콘솔을 즐기는 플레이어가 자막이 없어서 컷신을 이해할 수 없다.
- 특정한 조작 방식에 익숙하거나 신체적 장애가 있는 플레이어는 그에 맞는 입력 장치가 필요하다.(한 손 컨트롤러라던가 불기와 빨기를 통해 조작이 가능한 장치 등) 이에 적합한 조작 방식이 필요하지만 플레이어들이 이것을 게임에서 바꿀 수가 없다.
- 부족한 경험이나 장애로 인해 FPS 에서 조준을 하는 것과 같은 특정 행동을 하기가 힘들다.

### 게임이 도움말을 제공하지 않는다

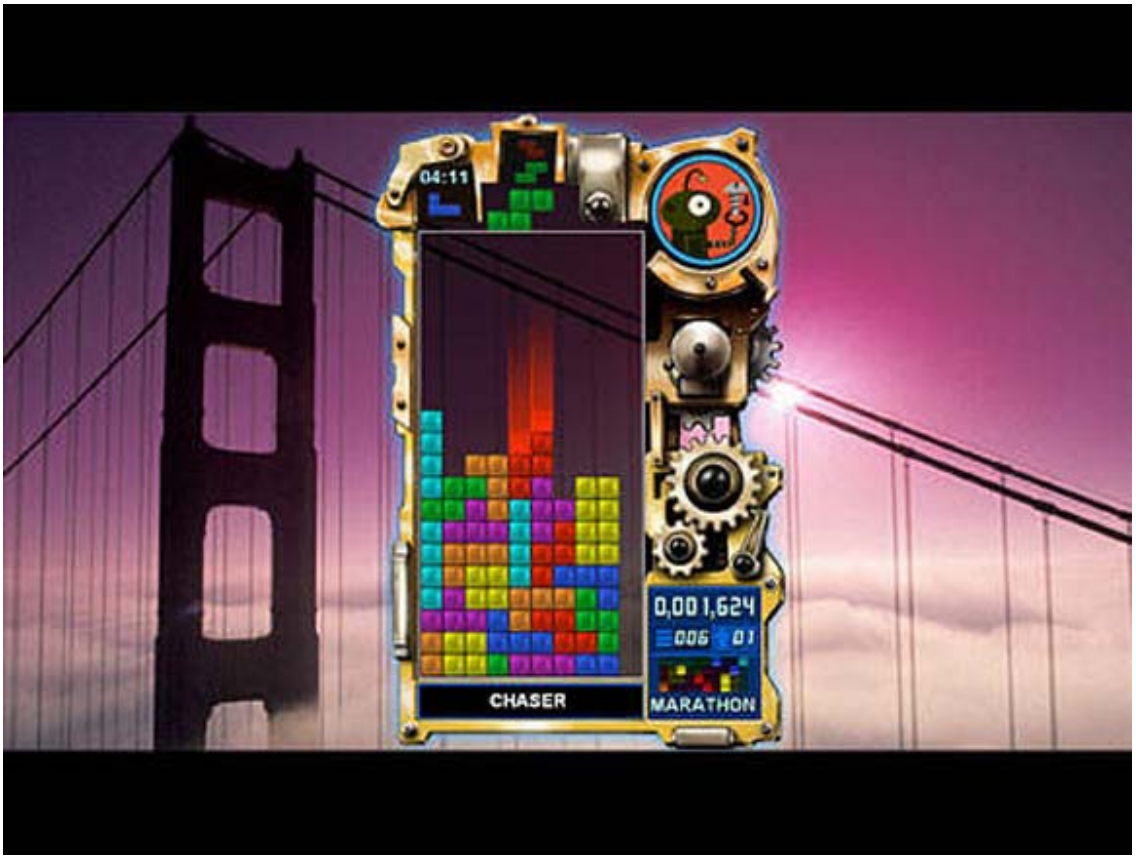
- 새로운 무기/탈 것/게임 오브젝트를 사용해보고 싶지만 그것이 현재 캐릭터/환경에 부정적인 영향을 끼칠 것을 걱정해 망설인다.
- 특정 게임 오브젝트를 어떻게 사용하는지에 대한 설명이 필요하지만 계속 매뉴얼을 보고 있기는 싫다.
- 게임을 플레이하기 위해서는 정보가 필요하다.(예: 퀘스트를 완료하기 위해서는 아티팩트의 위치를 알아야만 한다.) 이 정보는 게임 중에 제공되지만 언제든지 다시 확인하는 것이 불가능하다. 플레이어는 다시 정보를 얻으러 되돌아가야만 한다.

### 게임이 피드백을 지원하지 않는다

- 게임을 완료하기 위해서 필요한 플레이 시간이 어느 정도인지 알고 싶지만 찾을 수가 없다.
- 멋진 일을 해냈지만 게임이 계속 진행되어 그 순간을 만끽할 수가 없다.

### 문제에는 전후 관계적 이유가 있다

이러한 문제들을 살펴보면 문제들이 대개 전후 관계적인 이유가 있음을 알 수가 있다. 플레이어가 컷신을 계속 봐야만 하는 유용성 문제가 발생하는 경우는 컷신이 있는 게임에서이다. 세이브를 깜박하는 문제는 게임 진행을 저장할 수 있게 해주는 게임에서만 발생한다. 수많은 정보가 쏟아지는 것은 피드백이 거의 주어지지 않는 *테트리스(Tetris)*와 같은 게임에서는 문제가 되지 않는다.




접근성 문제는 플레이어가 겪고 있는 문제에 크게 영향을 받는다는 점에서 역시 매우 전후 관계적인 이유가 존재한다. 예를 들어 신체적 장애가 있는 플레이어에게는 자막이 없어도 문제가 되지 않으며 청각 장애가 있는 플레이어는 특수 컨트롤러 대신 일반 컨트롤러를 사용해도 문제가 되지 않는다.

**해결책은 현존 게임에서 찾을 수 있다.**


다행스럽게도 일부 게임 개발사들은 이러한 문제를 개발 중 인식하였고 그 결과로 이러한 접근성 및 유용성 문제에 대한 해결책을 현존 게임에서 찾아볼 수 있게 되었다. 예를 들어 플레이어가 플래시 게임을 하려는데 파일 크기가 너무 크거나 플레이어의 회선 속도가 느려서 플래시 파일을 불러올 때까지 기다려야 한다면 그 시간 동안 간단한 미니 게임을 할 수 있게 만들 수 있을 것이다. 플레이어는 여전히 기다려야만 하지만 최소한 그 시간 동안 뭔가를 할 수가 있는 것이다.

*하프라이프(Half-Life)* 1 편은 자막을 제공하지 않는 문제로 비판을 받았었다. Valve 측에서 개발 중 이 문제를 고려하지 않은 것이다. 이때부터 Valve 는 청각 장애 게이머 커뮤니티와 가까운 거리를 유지하며 작업을 해나갔으며 *하프라이프 2(Half-Life 2)*는 자막을 지원하게 되어 청각 장애인들도 대사를 알 수 있게 되었다. 총소리나 뒤에서 다가오는 발소리 등을 자막에 표시한다면 더욱 좋은 해결책이 될 수 있을 것인데, *하프라이프 2*가 바로 이러한 해결책을 제시했다.

불운하게도 모든 게임이 이러한 해결책을 채택하고 있는 것은 아니며 이로써 게임들에서 유용성이나 (대개의 경우) 접근성 문제가 발생하게 된다. 유용성 문제는 모든 플레이어들에게 영향을 끼치므로 개발자는 유용성 문제를 최대한 줄일 필요가 있다. 그렇지만 게임 개발의 세계에서는 데드라인을 맞추고 제품을 출시하기 위해서 유용성 테스트를 거치지 않는 경우가 많다.

게임 개발자들이 게임의 접근성을 높이는데 있어서는 별다른 인센티브가 존재하지 않는다. 대상 인구는 적고, 게임 개발은 매우 위험 부담이 크고 비용이 많이 드는데다가 데드라인과 제품 출시의 압박은 어떤 것이 게임의 접근성을 더 높여줄 수 있을지에 대해 연구할 시간을 거의 남기지 않는다. 게임에는 W3C 웹 컨텐트 가이드라인 같은 접근성 가이드라인이 존재하지 않는 것이다.

게임 디자이너는 증명된 지식에 기반을 둔 효과적이고 유용한 툴이 필요하다. 만약 우리가 훌륭한 게임 상호작용 디자인(game interaction design)의 “예술”로 분류되는 지식을 알고서 그것을 디자이너 간에 공유할 수 있다면 미숙한 디자이너들도 더욱 높은 접근성을 지닌 게임 상호작용 디자인을 할 수 있게 될 것이다. 이것은 곧 유용성이나 접근성 문제를 지닌 게임을 만드는 위험을 최소화 할 수 있다. 만약 우리가 특정 유용성/접근성 해결책을 상세히 살펴본다면 그것을 도입하는데 필요한 것을 설명하고 파악하게 하는데 도움이 될 것이다.

이것은 게임의 유용성이나 접근성을 높이는데 있어 더 나은 디자인 선택을 할 수 있게 도와준다. 만약 자막을 넣는데 한 명의 프로그래머가 3 주의 노력을 투자했다면 이로써 ~1.2 백만 장의 게임을 더 팔 수 있다.(미국에는 3 백만 명의 청각 장애인이 있는 것으로 집계되고 있으며, 이 중 약 40%가 게임을 즐긴다.)

문제는 어떻게 우리가 이러한 솔루션(해결책)을 설명하고 이것이 게임 개발자들에게 유용한 디자인 툴이 될 것인가 하는 것이다.

## 3. 디자인 지식 습득

전통적으로 인터페이스/상호작용성 디자인과 관련하여 최고의 가이드라인 또는 휴리스틱(heuristic; 발견법)으로는 Nielsen 의 휴리스틱 또는 W3C 웹 컨텐트 접근성 가이드라인이 있었다. 가이드라인의 목적은 디자인 지식을 간결한 작은 규칙으로 만드는 것으로, 인터페이스와 상호작용성 디자인을 계발하는데 사용되었다.

디자인 지식을 습득하려는 시도는 게임 상호작용성 디자인과 연관하여 이루어졌다. Houser & Deloach 는 효율적인 게임 디자인을 위해 7 가지 원칙을 선보인다. Melissa Federoff 는 Nielsen 에 의해 발의된 현존 유용성 휴리스틱을 어떻게 게임에 적용시키는지에 대해 살펴보았으며 42 개의 게임 휴리스틱을 발의하였다. 이러한 가이드라인은 유용성 문제에 특히 초점을 맞추고 있으며 Noah Falsteins 400 프로젝트 같은 게임플레이를 설명하는데 다른 입장을 취하고 있다.

## 휴리스틱의 문제

가이드라인은 요구사항을 맞추는 것에 유용하지만 만약 우리가 이것의 유용성을 디자인 툴로서 살펴보았다면 정선(selection), 유효성(validity), 응용 가능성(applicability)과 관련하여 Welie 에 의해 정의된 일부 문제점을 발견할 수 있었을 것이다.

- 가이드라인은 일반적으로 절대적 유효성을 제공하지만 이것은 오로지 특정 상황에서만 적용이 가능하다. 예를 들어 Federoff 는 *"게임은 예기치 않은 결과가 나와야 한다."* 고 정의하고 있는데 이것은 어드벤쳐 게임에는 적용되더라도 *퐁* 같은 아케이드 게임에는 적용되지 않는 사항이다.
- 가이드라인이 해결하고자 하는 문제가 무엇인지에 대해서와 어째서 해결하려고 하는지에 불명확한 경우가 많다. Federoff 에서는 *"플레이어는 다른 상태에서 게임을 저장할 수 있어야 한다."* 고 정의하고 있지만 어떤 유용성 문제가 있는지에 대한 설명이 없으며 어째서 이 해결책이 유효하며, 이것을 어떻게 적용하는지에 대해서도 알 수가 없다.
- 간결한 작은 규칙으로 나뉘어진 디자인 지식은 결국 주변 사물을 설명하기 위해서는 수많은 규칙에 얽매이게 되는 문제로 연결된다. 수많은 가이드라인은 게임 디자이너로 하여금 올바른 규칙을 고르기 힘들게 만들며 정황(context)이 확실치 않으면 어떤 가이드라인은 다른 가이드라인과 충돌하기도 한다. 예를 들어 *"게임은 예기치 않은 결과가 나와야 한다."* 와 *"초반에 게임의 목적에 대해 분명하게 설명하여야 한다."* 는 것이 혼란을 일으킬 수 있다.

디자인 툴은 최우선적으로 유용성이 있어야 한다. 우리는 디자이너에게 정확히 언제 해결책이 필요하며, 이 해결책이 어떻게

작동하며 왜 그렇게 작동하는지에 대해 알려줄 수 있어야 한다. 요구사항은 “모든 비문자 요소에 대한 문자 표현의 제공”이라는 것보다 “제한 자막(closed caption)”이라고 표현하는 것이 더욱 쉽게 이해와 적용이 가능하다. 우리가 파악한 유용성과 접근성 문제는 전후 관계적 문제라고 할 수 있다. 필자는 디자인 경험을 습득하기 위해서 더욱 풍부한 설명을 제공하고 쓸모가 많으며 디자인 툴로도 사용할 수 있는 상호작용 디자인 패턴을 사용할 것을 제안하는 바이다.

## 4. 상호작용적 디자인 패턴


패턴과 이를 설명하기 위한 패턴 언어는 모범 사례, 훌륭한 디자인 그리고 다른 사람이 재사용할 수 있는 경험 습득을 위한 방법이라 할 수 있다. 각 패턴은 3 개의 파트(three-part) 규칙을 지니는데, 이는 특정 상황 사이의 관계(relation), 문제(problem), 해결책(solution)을 뜻하는 것이다. 일반적으로 이론적 근거로 함께 제공된다. 패턴은 Christopher Alexander 의 구조적 개념의 시발점이지만, 소프트웨어 내의 패턴은 디자인 패턴 책인 “갱 오브 포(Gang of four)”를 통해 유명해졌다. 이때부터 패턴 커뮤니티는 통합을 시작하여 상호작용적 디자인을 포함하여 모든 종류의 도메인을 위한 특정 패턴을 지니게 되었다.

패턴은 게임 디자인에 대한 지식을 설명하는데만 사용되었었다. 이 글에서는 상호작용적 디자인 패턴을 사용할 예정이다. 상호작용적 디자인 패턴은 일반적으로 발생하는 유용성 문제에 대해 반복이 가능한 해결책을 제안한다. 상호작용적 디자인 패턴을 묘사하는 가장 본질적인 예제는 undo 라고 할 수 있다.

**이름:** undo

**문제:** 사용자가 실수를 해서 이것을 되돌릴 수 없다.

**정황:** 워드프로세서나 그래픽 어플리케이션 등 사용자가 정보를 다루거나 새로운 내용을 만들 수 있는 어플리케이션.

**해결책:** 사용자의 행동 목록을 유지하고 선택된 행동을 사용자가

되돌릴 수 있게 해준다.

**왜:** 사용자는 자신의 실수를 되돌릴 수 있다는 것을 알고 있기 때문에 어플리케이션 학습이 더욱 촉진될 것이다.

---

패턴과 가이드라인은 서로를 배제하지 않는다. 패턴은 하나 혹은 그 이상의 규칙을 매우 상세한 상황 속에서 설명하여(심도) 디자인 툴로서의 효용성이 큰 것에 반해, 가이드라인은 더 높은 단계(범위)를 통하여 요구사항으로서의 효용성이 크다.

대여섯 개의 패턴 모음은 온라인에서 찾아볼 수 있으며 아마 Yahoo UI 패턴은 가장 잘 알려진 패턴일 것이다. 현존 패턴 모음은 웹과 범용 소프트웨어의 사용자 인터페이스(UI)의 문제에 초점이 맞춰져 있어서 게임 디자인을 위한 툴로 사용하기는 힘들다. 비록 마법사와 같은 일부 패턴은 게임에 사용할 수 있지만(예: 설치 마법사) 그 외의 컨테이너 네비게이션이나 달력 선택 등의 다른 것들은 게임 디자인에 적합하지 않다.

우리는 게임에 고유의 유용성/접근성 문제가 존재함을 살펴보았다. 예를 들어 제한 자막의 부재라던가, 컷신을 넘길 수 없는 것등은 웹 어플리케이션에서는 찾아볼 수 없는 문제이다. 그 결과로 우리는 게임의 유용성/접근성 문제를 해결할 수 있는 새로운 자체 제작 상호작용 디자인 패턴 모음을 개발하기로 결정하였다. 글의 다음 부분에서는 현존 게임에서 살펴본 패턴의 예를 3개 들어보았다.

## 5. 예제 패턴

디자인 지식을 설명하기 위해 패턴을 사용하는 것의 또 다른 이점은 바로 일관성 있는 형식이라 할 수 있다. 가독성을 높이기 위해 우리는 아래의 형식을 따르고 있다.

- **문제(Problem)**: 문제는 게임을 플레이하는 것에 연관되어 있으며 접근성이나 유용성 모두에 관련이 있을 수 있다.
- **정황(Context)**: 정황은 현재의 특수한 상황이나 문제가 일어날 수 있는 게임의 종류를 서술하여 일반적인 문제 해결 이분법을 확장시킨다.

- **영향(Force)**: 정황과 함께 다수의 영향력이 존재하며 이것 역시 해결해야 한다.
- **해결책(Solution)**: 증명된 해결책으로, 영향력을 해결한다.
- **왜(Why)**: 패턴이 적용되었을 때 유용성의 측면에 끼치게 되는 상세한 효과를 설명하기 위한 이론적 근거.
- **예제(Examples)**: 예제는 해당 패턴이 어떻게 게임에 성공적으로 도입되었는지를 보여준다.

현재 우리는 이때까지 23 개의 상호작용적 디자인 패턴을 파악했으며, 이것들은 모두 게임의 유용성을 증대시킬 수 있으며 8 개는 접근성을 증대시킬 수 있다. 아래에는 3 개의 예제 패턴을 나열할 것인데, 적응적 난이도와 심리스 게임세계, 그리고 슬로우가 그것이다. 이 글에는 오직 3 개의 패턴만이 포함되어 있지만 이 글의 독자들은 우리의 웹사이트를 방문하여 나머지 패턴들도 읽어볼 것을 권하는 바이다. http://www.helpyouplay.com

## 적응적 난이도

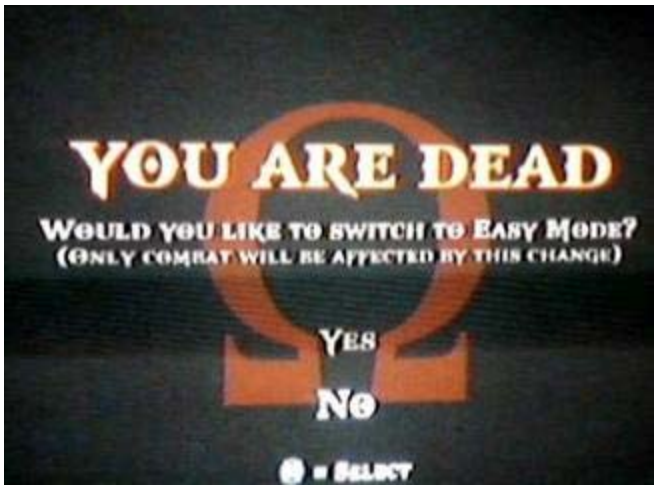



그림 1: *갓 오브 워(God of War)*에서의 적응적 난이도

**문제**

**문제** - 플레이어가 다음의 문제로 인해 지속적으로 죽거나 피해를 입었다

**접근성** - 현재 설정으로는 능력이 부족한 사람에게 게임이 너무 어렵다.(예: 빠르게 대응을 못하거나 조준점을 정확한 위치로 조절을 못 한다.(신체적) 혹은

많은 사건을 동시에 다루지를 못 한다.(신체적/지각력)

**유용성** - 현재 난이도 설정으로는 게임이 너무 어렵다.

**언제 사용** 여러 난이도 설정을 제공하는 게임은 다양한 종류의 플레이어를 받아들일 수 있다. 난이도를 구분하는 일반적인 단어는 쉬움(easy), 보통(normal), 어려움(hard)이다. 난이도에 따라서 다음의 사항이 변한다.

- 적이 강해지거나 약해진다.
- 퍼즐이 어려워지거나 쉬워진다.
- 더 적거나 많은 안내가 제공된다.
- 더 적거나 많은 조작 도움이 이루어진다.(예: 자동 조준이나 자동 스티어링)

플레이어는 게임을 시작하며 난이도를 선택하게 되며 대개의 경우 게임을 다시 시작하지 않고 난이도를 바꿀 수 없다.

**영향**

- 플레이어는 게임을 포기하는 것을 싫어하지만 도전적이 되길 원한다. 예를 들어 게임을 플레이하기 위해서는 힘들어야 하지만 같은 지점을 지나가기 위해 반복적인 시도를 하는 것은 원하지 않는 것이다.
- 일부 플레이어는 비슷한 게임을 해본 경험이 있을 수 있다. 다른 플레이어는 경험이 거의 없다.
- 일부 플레이어는 다른 플레이어보다 특정 게임(슈팅이나 퍼즐 풀이)등에서 더 낫거나 나쁜 실력을 지니고 있을 수 있다.
- 특정 요소와 관련된 게임의 난이도는 게임 중에 달라질 수 있다. 게임의 한 부분은 슈팅에 크게 중점을 두지만 또 다른 부분은 퍼즐 풀이에 중점을 두고 있을 수 있다.

**해결책** **플레이어가 난이도를 조절할 수 있게 하라.**

게임은 여러 파트를 지나며 여러 플레이어를 수용해야 한다. 이러한 패턴을 도입하는 것에는 두 가지 선택사항이 있다.

- 플레이어의 실력에 따라 다른 난이도를 제공한다. 예: 플레이어가 실패하거나 특정 지역을 지나갈 때 일정 횟수 이상의 죽음을 맞이하게 되면 더 낮은 난이도를 선택할 수 있게 제안한다.
- 플레이어의 실력에 따라 자동적으로 난이도를 조절한다. 플레이어가 죽거나 실패할 때마다 다음으로 쉬운 난이도로 설정이 바뀔 확률이 존재한다. 만약 플레이어가 한동안 잘 해나간다면 난이도가 상승할 확률이 존재한다. 이 해결책은 넓은 범위의 난이도를 제공할 수 있다는 점에서 훌륭하다.(일반적인 쉬움 / 보통 / 어려움 보다 많다.) 난이도의 변경을 플레이어에게 알릴 수도 있지만 보이지 않게 할 수도 있다. 플레이어는 게임을 시작하며 난이도를 선택할 필요가 없다.

**왜** **접근성** - 난이도를 조절하는 것은 실력이 부족한 플레이어가 게임을 더욱 쉽게 즐길 수 있으며 더 적은 실수를 하게 만들어준다. 게임은 플레이어에게 적절한 난이도를 자동적으로 설정하게 될 것이다.

**유용성** - 플레이어가 난이도를 설정하는 것은 만족감과 효율성을 증대시킬 수 있다. 플레이어는 실망하지 않을 것이다.

**예제** ***갓 오브 워*** - 이 3 인칭 액션 게임에서 만약 플레이어가 짧은 시간 내에 자주 죽게 되면 난이도 조절이 가능하다. 예: 연속으로 7 번 죽을 경우 화면에 쉬운 난이도로

변경할 것인지를 물어오는 화면이 뜬다.

*레지던트 이블 4(Resident Evil 4)* - 이 3 인칭 슈터 게임은 5 종류의 난이도를 제공하며 플레이어의 실력에 따라 자동으로 변화한다.

*신 에피소드(Sin Episodes)* - 이 FPS 는 매우 진보적인 동적 난이도 시스템을 지니고 있다. 지속적으로 플레이어의 실력을 확인하고서 적의 체력이나 총탄, 방어력, 공격력 등을 특정 플레이어 스타일에 맞춘다.

---

## 심리스 게임세계

그림 2: *던전 시즈(Dungeon Siege)*에서의 심리스 게임세계

**문제** **유용성** - 플레이어는 게임의 (새로운) 파트로 진입하기 전에 기다려야만 한다.

**정황** "자유 행동(free roaming)" 게임에서 일반적. 예: 3 인칭 슈터, 롤플레잉, 거대한 세계에서 자유롭게 돌아다닐 수 있는 액션 또는 시뮬레이션 게임. 이러한 게임들은 대개

비선형적인 게임 플롯을 지니고 있다. 전체 세계를 메모리에 불러올 수 없기 때문에 게임 세계는 대개 여러 개의 구역으로 나누어진다. 플레이어가 하나의 구역에서 다른 구역으로 이동하게 되면(예: 집 안으로 들어가기) 그 구역이 메모리에 올라갈 때까지 기다려야만 한다.

**영향**

- 플레이어는 기다리는 것을 싫어한다.
- 플레이어는 게임이 방해 받는 것을 원하지 않는다.

**해결책** **심리스 게임 세계를 제공한다.**

플레이어가 새로운 구역으로 들어갈 때 기다리게 하지 말고 그 구역에 들어가기 전에 미리 로딩(pre-load)을 해놓는다. 거대한 심리스 세계를 렌더링 하는 것은 게임 엔진 디자인에 큰 영향을 미칠 것이다. 예제는 아래와 같다.

- 전체 세계가 조각으로 나뉘어진다.(각 조각의 크기는 메모리에 따라 달라진다.)
- 한 번에 오직 9 개의 조각만을 불러올 수 있다. 현재 플레이어가 위치한 조각 주위에 8 개의 조각이 감싸고 있는 형식이다.
- 플레이어가 중앙 조각에서 벗어나서 경계면에 존재하는 조각으로 이동하게 되면 플레이어로부터 가장 먼 곳에 위치한 3 개의 조각은 소멸되고 플레이어가 들어선 구역이 중심 조각이 된다. 그러면 새로운 3 개의 조각이 불러져서 3x3 의 조각이 만들어진다. 좀 더 상세한 적용 방법(조각의 부분을 로딩하는 것이나 하드웨어적 제한을 해결하는 방법)은 Bilas 에서 확인할 수 있다.

**왜** 거대한 퍼시스턴트 월드(persistent world)를 지니는 것은 플레이어에게 지속적인 몰입감을 제공해주며 게임이 결코 멈추거나 불러오기를 하게 되는 경우가 없다. 이 해결책은 효율성과 만족도를 높여준다.

**예제** *던전 시즈* - 이 롤플레잉 게임은 “컨텐트 스트리밍”을

이용하여 “레벨 불러오기”를 방지하고 있다.

*월드 오브 워크래프트(World of Warcraft)* - 이 MMORPG 는 수많은 장소가 연결되어 하나의 부분으로 보이는 심리스 세계를 제공한다. 불러오기를 하는 경우는 드물며 매우 짧다. 게임의 거대한 대륙을 건너거나 각 플레이어 그룹을 위한 특별한 인스턴스 고레벨 지역에 들어가는 경우에만 불러오기를 한다.

---

## 슬로우(Slow)

그림 3: *맥스 페인(Max Payne)*에서의 슬로우

**문제** 플레이어는 짧은 시간 내에 여러 행동을 해야 하는데 이것이 어렵다.

**접근성 문제**- 만약 플레이어가 신체적, 인지적, 시각적 문제를 겪고 있다면 동시에 여러 일을 처리하는데 더 많은 시간이 필요할 것이다.

**유용성 문제** - 플레이어가 숙련되지 않았을 경우.

**정황** 이것은 FPS 나 플랫폼 게임과 같은 액션 게임에 흔한 일이다.

특정한 목표(레벨 완수 등)를 수행하기 위해서는 여러 행동을 성공적으로 수행해야만 한다. 가끔은 정해진 시간 내에 행동을 해내야 할 때도 있다. 예를 들어 플레이어는 버튼을 눌러 문을 열었다면 이 문이 어느 정도의 시간이 지나면 닫히기 때문에 그 너머로 가기 위해서는 구덩이를 뛰어넘거나, 적을 처치해야만 한다.

게임 디자이너는 이러한 '도전'을 게임에 삽입하여 재미를 돋군다. 뛰어난 플레이어에게는 이러한 도전이 장애물이 되지 못하지만 (초보) 플레이어는 도전을 완수하는데 매우 큰 어려움을 겪을 것이며 성공하기까지 대여섯 번의 시도를 하게 될 것이다.(만약 성공한다고 쳤을 경우)

**영향**

- 시간 조정은 멀티플레이 게임에서 사용할 수 없다.
- 플레이어는 죽었을 때 같은 부분을 계속해서 하고 싶어하지 않는다.
- 게임을 너무 쉽게 만들면 게임플레이를 망치게 된다.

**해결책** **해결책 - 플레이어에게 시간을 느리게 만들 수 있게 해준다.**

자신은 원래 속도로 이동하면서 주변 세계를 슬로우 모션으로 만드는 것은 몇 가지 이득을 가져다 준다.

- 빠른 이동 - 달리기, 도약, 구르기, 싸우기, 쏘기 등을 할 수 있다는 것은 적들과 장애물에 대해 플레이어에게 이점을 가져다 준다. 이것은 시간과 연관된 목표를 이루는데 특히 도움이 된다.
- 향상된 공격력 - 적과 싸우며 더 많은 공격을 할 수 있기 때문에 더 많은 피해를 입힐 수 있으며 적들은 공격을 방어하기가 힘들어진다.

슬로우 모션은 짧은 시간 동안만 사용되어야 한다는 것을 명심해야 한다. 만약 계속해서 시간을 느리게 할 수 있다면 게임은 너무 쉬워질 것이다. 이러한 구성을 위해서는 플레이어가 슬로우를 작동시킬 때 뭔가를 희생하게 만들어야 한다. 액션 게임인 *페르시아의 왕자(Prince of Persia)*에서는

게임 중에 모을 수 있는 토큰(token)을 이용하여 슬로우를 작동시킬 수 있다. 동시에 나오는 토큰의 수는 제한되어 있기 때문에 플레이어는 슬로우를 남발할 수 없게 된다.

능력이 부족한 게이머에게는 슬로우의 남발이 문제가 되지 않으며 원할 때 슬로우를 사용할 수 있어야 한다. 슬로우는 다수의 적을 공격하면 작동하기도 한다. 플레이어는 슬로우를 작동할 수 있을 경우 혼란을 최소화하기 위해 그것을 인지할 수 있어야 한다. 슬로우는 FPS 에서 불렛타임(bullet-time)이라고 불리기도 한다.

왜 **접근성** - 주변을 모두 느리게 만드는 것은 능력이 부족한 플레이어가 실수를 적게 하도록 만들어준다.

- 빠르게 대응하거나 동시다발적인 상황에 대응하는데 어려움을 겪는 플레이어는 게임을 느리게 만들어 문제를 해결할 수 있다.
- 조준점을 정확한 위치로 움직이는데 어려움을 겪는 플레이어는 게임을 느리게 만들었을 때 이것을 완수하는데 더 많은 시간을 이용할 수 있다.

**유용성** - 게임을 느리게 만드는 것은 플레이어에게 반응할 시간을 더 제공하고 실수를 적게 하도록 만들어주므로 신뢰도와 만족도를 높일 수 있다.

예제 *맥스 페인* - FPS 에서 매트릭스 스타일의 불렛타임을 최초로 선보인 게임이다. *맥스 페인*의 게임플레이는 불렛타임에 크게 연관되어 있다. 불렛타임을 사용하면 시간의 흐름이 느려지며 총알의 움직임을 볼 수 있게 된다. 플레이어의 움직임 역시 느려지지만 반응이나 조준은 원래의 속도대로 할 수 있기에 적들보다 이점을 지닐 수 있다.

*페르시아의 왕자: 전사의 길(Prince of Persia: Warrior Within)* - 이 플랫포머/액션/퍼즐 게임에서는 메인 캐릭터가

단검을 이용하여 시간을 느리게 하는 것이 가능하다. 단검에는 모래시계에서 나온 시간의 모래가 “충전”되어 있으며 이것을 이용하면 왕자가 잠시 동안 시간을 “느리게” 만들 수 있다. 단검의 사용은 제한되어 있지만 적을 처치하고 나면 시간의 모래가 그 자리에 남게 되며 이것을 단검으로 흡수해 충전을 하는 것이 가능하다. 이것은 시간을 조정하는 힘을 충전하기 위해 적을 피하는 것 대신에 직접 상대하고 처치하도록 플레이어를 장려하고 있다.

*블링스: 타임 스위퍼(Blinx: The Time Sweeper)* - 이 3 인칭 플랫폼 게임은 시간의 흐름을 조절할 수 있게 해준다. 예: 느리게 만들기, 빠르게 만들기, 되돌리기, 멈추기

---

## 6. 패턴으로 디자인하기


우리가 수집한 패턴 모음을 사용하는 방법은 여러 가지가 있다. 현재 우리는 패턴들을 두 개의 모음으로 나누어 두었는데, 하나는 유용성에 중점을 둔 것이고 다른 하나는 접근성에 중점을 둔 것이며, 앞으로는 게임의 장르와 패턴의 적용에 필요한 노력의 정도를 기준으로 정리할 예정이다.

### 유용성을 위해 디자인하기

글의 2 번 부분에서 유용성 문제를 파악하며 우리는 패턴을 플레이어의 문제에 맞게 정리했다. 일부 분류는 Nielsen 의 휴리스틱과 동일하기도 하다. 슬로우와 같은 특정 패턴은 두 개의 분류에 속하기도 한다.

- **기다림 막기**
  - 심리스 게임세계
  - 넘길 수 있는 컷신
  - 빠른 전진
  - 빠른 저장/불러오기
- **상태 전달**
  - 즉석 리플레이
  - 게임 진행도
- **플레이어에게 적응**
  - 제한 자막

---

    - 빠른 재시작
    - 빠른 시작
    - 프리로더(Pre Loader) 게임
    - 아케이드 모드
- **실수 방지**
    - 슬로우
    - 되감기
    - 자동 저장
    - 화면 저장
    - 멈추기
    - 자유롭게 둘러보기
    - 상호작용 지원
    - 적응적 난이도 설정
    - 재설정 가능한 버튼
    - 슬로우
    - 아케이드 모드
- **도움 제공**
    - 튜토리얼 요원
    - 상호작용 지원
    - 플레이그라운드
    - 일지

---

게임 디자이너는 우리의 5 개 분류를 요구사항으로 사용하여 휴리스틱적으로 분류 내의 각 패턴을 평가한 후 어떤 것을 적용해야 할 것인지 말아야 할 것인지를 결정할 수 있다. 이것은 디자인의 초기에 이루어질 수도 있고, 개발의 후기에 이루어질 수도 있다.

하지만 개발 후기에 특정 패턴을 적용하는 것은 너무 많은 비용이 필요하게 될 수도 있다. 비록 게임 디자이너가 각 패턴의 적용에 필요한 비용이 어느 정도인지 알 수 있는 좋은 위치에 있다고 하더라도 우리는 이러한 것이 중요하다고 생각하며, 이는 미숙한 게임 디자이너에게 전체적인 필요 비용을 예측할 수 있게 해주는 것이다.

예를 들어 넘길 수 있는 컷신은 쉽게 적용이 가능하다. 심리스 게임세계는 하부 구조에 제약이 존재하므로 개발 후기에 이것을 적용하는 것은 많은 비용을 필요로 하게 된다. 요구사항 분석 중 패턴이 어떻게 유용성을 증대시킬 수 있는지에 대해서와 무엇이 시스템에 필요한지를 논의해보면 소프트웨어 엔지니어링과 유용성 사이에 존재하는 제약의 상호 인식이 증가할 수 있을 것이다.

## 접근성을 위해 디자인하기

접근성을 위해서 우리는 패턴을 각각 다른 장애에 대응시켜 정리하였다. 4 종류의 장애가 분류되어있다.

- 시각 장애 - 실명, 저시력(low vision), 색맹.
- 청각 장애 - 귀머거리 또는 작은 소리를 못 듣는 것.

- 신체 장애 - 마비, 신경 장애, 반복운동손상(RSI), 나이 문제.
- 인지 장애 - 난독증, 행동 부전, 자폐증, 주의력 결여 장애 등의 학습 장애

- **청각 장애**
  - 제한 자막
  - 상호작용 지원
- **신체 장애**
  - 슬로우
  - 상호작용 지원
  - 적응적 난이도 설정
  - 재설정 가능한 버튼
- **시각 장애**
  - 상호작용 지원
  - 일지
- **인지 장애**
  - 슬로우
  - 상호작용 지원
  - 튜토리얼 요원

---

청각 장애를 지닌 사람들에게 접근성을 높이는 것이 가장 쉬운 것이 될 것이다. 왜냐하면 적은 노력을 통해서 제한 자막을 넣을 수 있기 때문이다. 실명한 사람들에게 게임의 접근성을 높이는 것은 가장 어려운 일이 될 것이지만, 저시력에 대해서는 이미 FPS 에서 자동으로 적을 마주보게 하는 것 등의 기계가 존재하고 있다.(상호작용 지원) 게임 디자이너는 다시 한 번 패턴을 살펴보며 어떤 것을 적용해야 할 것인지에 대해 고려해볼 수 있을 것이다.

## 7. 논의


게임에 필요한 ID 패턴이 얼마나 상세해야 하는지에 대한 의문은 아직 남아있다. 숙련된 게임 디자이너는 대부분의 패턴이 평범한 것으로 생각할지도 모른다. 그렇지만 초급자에게는 유용할 수도 있다. 패턴의 상세함(detailedness)는 디자인 패턴 연구에서 일반적인 '문제'로 받아들여진다. 그렇지만 우리의 패턴은 UI 디자이너와 소프트웨어 엔지니어에게도 유용하면서도 너무 세세한 부분까지 다루고 있지는 않다.

가끔 패턴을 설명할 때 아래에 가려진 구조에 대해 기본적인 가정을 하지 않고서는 설명이 힘든 경우도 있다. 자동 저장 패턴을 설명할 때

일반적인 의미에서 “저장”이라는 것을 자세히 설명하는 것이 매우 어려운 것이다. 우리의 패턴에서 플레이어가 게임을 저장할 수 있다는 것은 추측할 수 있지만 어째서 이것을 써야만 하는가? 저장이 유용성과 관련된 기능일까, 아니면 그저 순수하게 분리된 하나의 개별 기능일까? 워드프로세서의 경우 저장 기능이 없는 것은 상상할 수도 없지만 일부 게임의 경우에는 저장 기능이 없을 수도 있다.(적을 전부 때려잡는 게임 등.)

우리가 패턴을 모으기 시작했을 때 우리는 우선 게임 영역에서 “고유한(unique)” 패턴들을 파악하기 시작했다. 자동 저장이나 화면 저장 등의 기능은 데스크탑 소프트웨어에서도 찾아볼 수 있는 것이기에 이것은 매우 어려운 일이었다. 프리뷰 패턴은 우리의 화면 저장 패턴에 연관되어 있다.

이러한 일반적인 패턴(generic pattern)에서 화면 저장이 게임만의 특징적인 기능이라고 주장할 수도 있다. 패턴 모음의 완성도를 위해 우리는 이 패턴이 더욱 일반적인 패턴의 파생 패턴이라고 할지라도 이것을 패턴 모음에 포함시키기로 마음먹었다. 일반 패턴을 게임 전용 패턴으로 옮기는 것은 우리의 목적이 아니고, 패턴은 소속 영역이 최대한 자유로운 것으로 정의되어야 하기 때문이다.

## 8. 결론

게임을 위한 상호작용적 디자인은 매우 어려우며 수년간의 경험이 필요할 때가 많다. 상호작용적 디자인 패턴은 게임 휴리스틱보다 더욱 풍부한 포맷으로 게임 상호작용 디자인의 모범사례가 될 수 있으며 디자이너에게 더 유용한 툴이라고 할 수 있다. 현존 상호작용 디자인 패턴 모음은 웹과 범용 소프트웨어를 위한 UI 에 초점을 맞추고 있기에 게임 디자인에 적용하기가 힘들다. 우리는 게임의 일반적인 유용성 및 접근성 문제를 해결할 수 있는 상호작용적 디자인 패턴을 개발하였고, 이 패턴들은 현존 게임으로부터 수집한 것들이다.

우리 패턴의 포맷과 양은 더 많은 패턴의 개발에 도움을 줄 수 있다.

이러한 지식은 게임에 상호작용적 디자인을 알리는데 효율적으로 사용될 수 있으며 초기 디자인 시 의견 소통을 돕고 게임의 유용성 및 접근성 문제를 줄이고, 잠재적인 판매량을 증가시킬 수 있다.