

# 스타크래프트는 어떻게 적절한 심성모형을 구축하는가? — 의미성 차원을 중심으로

---

아주대학교 소프트웨어융합대학 디지털미디어학과 박성범

# 목차

---

1. 서론
2. 이론적 배경
3. 첫 번째 단계: 학습성
4. 두 번째 단계: 변화제시성
5. 세 번째 단계: 이해가능성
6. 결론

참고문헌

# 1. 서론

---

# 1. 서론

---

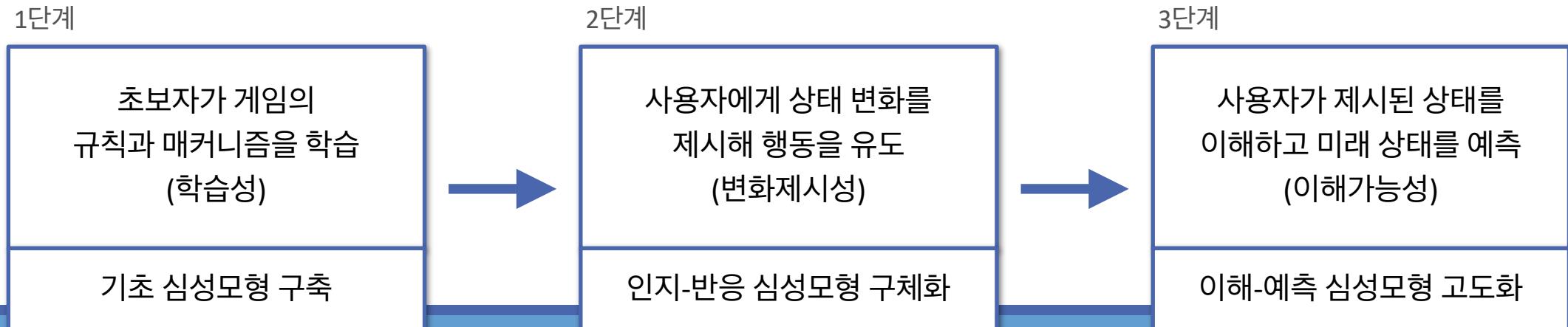
나는 스타크래프트의 사용성을 비판적으로 분석하는 큰 맥락에서 **스타크래프트(1998)**가 사용자의 심성 모형을 어떻게 구축하고 있는지 살펴보고자 한다. 적절하게 형성된 심성모형은 사용자로 하여금 복잡한 컴퓨터 시스템을 적절한 수준으로 추상화하여 효과적으로 이해할 수 있도록 돋는 동시에, 실제와 다르게 구성된 심성모형은 오히려 시스템을 오해하게 만들 수도 있다. 스타크래프트는 RTS(Real-Time Strategy, 실시간 전략) 게임의 황금기라 할 만한 90년대에 개발된 게임으로, 적절한 사용자 심성모형을 구축함으로써 방대한 세계관과 시스템을 쉽게 이해할 수 있게 만들고 유용성과 사용성을 제고하였다.

스타크래프트는 이전에 개발된 둔2(1992), 워크래프트(1994), 커맨드 앤 컨커(1995)와 같은 RTS 게임들의 설계를 적극적으로 계승했기에 온전히 독자적으로 사용성을 혁신했다고 평가할 수는 없지만, 스타크래프트가 앞서 출시된 RTS 게임들의 장점을 적절히 조합해 높은 사용성을 달성했다는 점, 스타크래프트 이후 출시된 RTS 게임들은 다른 게임이 아닌 스타크래프트를 교본 삼았다는 점, 그리고 좋은 사용자 경험을 설계했다는 평가를 넘어 상업적인 성공을 거뒀다는 점에서 스타크래프트의 사용성을 분석하는 것이 의미가 있다.

# 1. 서론

이 보고서에서는 스타크래프트의 심성모형 구축 방식을 (1) 초보 사용자가 게임의 규칙과 매커니즘을 익히며 기초적인 심성모형을 구축하는 단계, (2) 게임이 사용자에게 상태 변화를 제시해 적극적으로 사용자의 행동을 유도하며 사건에 대한 인지와 그에 따른 행동이 심성모형에 구체화되는 단계, (3) 사용자가 제시된 현재 상태를 이해하고 미래 상태를 예측하면서 심성모형을 고도화하는 단계로 나눠서 살펴본다. 이러한 단계적 접근의 맥락 하에 각 단계를 의미성 차원에서 (1) 학습성, (2) 변화제시성, (3) 이해가능성 측면을 중심으로 심성모형이 어떻게 형성되는지 분석하고자 한다.

RTS 게임이 매우 복잡한 시스템과 불확실성을 수반한 시뮬레이션 소프트웨어의 일종이라는 점에서 **스타크래프트의 심성모형 분석은 일반적인 소프트웨어 사용 경험을 설계하는 데에도 기여할 수 있을 것이라고 기대한다.**



## 2. 이론적 배경

---

## 2. 이론적 배경: 심성모형과 행위이론

---

도널드 노먼(2016)에 따르면, 심성모형은 “**사람들이 자기 자신, 다른 사람, 환경, 자신이 상호작용하는 사물들에 대해 갖는 모형**”이다. 즉, 심성모형은 어떤 시스템의 구조와 기능, 가치를 추상화한 정신적 모형이라고 할 수 있다. 심성모형은 주관적으로 추상화된 지식 구조이기 때문에 사회적 맥락과 개인적 배경에 따라 달라질 수 있다.

따라서 심성모형은 불완전하거나, 현실과 모순적일 수 있기 때문에 잘못된 심성모형이 구축되면 시스템을 오해할 수 있다. 또한 심성 모형은 동적으로 구성되며, 상황에 따라 변형될 수도 있다. 좋은 심성모형은 복잡한 시스템을 추상화해 시스템의 현재 상태와 현상을 묘사하고, 동작의 인과관계를 쉽게 이해할 수 있게 해준다. 또한 시스템의 상태가 어떻게 변화할지 예측할 수 있게 만듦으로써 사용자가 사용하는 시스템의 유용성과 사용성을 높일 수 있다.

시스템을 개발한 개발자는 자신이 시스템에 대해 가진 심성모형을 인터페이스로 구현된 시스템 이미지에 반영한다. 이때 개발자가 시스템 이미지에 갖는 심성모형은 개발자 모형이라고 한다. 한편, 시스템 인터페이스를 사용하는 사용자가 시스템 이미지에 갖는 심성모형은 사용자 모형이라고 한다. 사용자는 경험이나 교육을 통해 시스템에 대한 심성모형을 구축한다. 시스템 이미지를 적합하게 구성한다면 사용자 모형을 개발자 모형과 유사하게 형성하여 올바른 심성모형을 구축할 수 있다.

## 2. 이론적 배경: 심성모형과 행위이론

---

사용자가 자신의 심성모형을 바탕으로 시스템을 사용해 어떠한 목적을 달성하고자 할 때, 그 목적과 시스템이 제공하는 기능 및 정보 사이에 발생하는 차이를 실행차라고 한다. 사용자가 가진 실제 목적을 시스템에서 실행할 수 있는 목적으로 전환하면서 의도를 구축하고, 그 의도를 달성하기 위한 행위를 계획한 뒤, 계획된 행위를 실행하는 과정이 원활하다면 실행차를 줄일 수 있다.

사용자가 목적을 달성하기 위해 시스템의 상태를 변경한 뒤에, 변경된 상태와 사용자의 원래 목적 사이에 발생하는 차이를 평가차라고 한다. 사용자가 상태의 변경을 지각하고, 그 상태를 해석한 뒤, 원래 목적과 비교 평가하는 과정이 원활하다면 평가차를 줄일 수 있다.

## 2. 이론적 배경: 학습성

---

학습성은 시스템에 익숙치 않은 초보 사용자가 시스템을 유용하게 사용하기 위한 지식을 얼마나 쉽게 취득하고 학습할 수 있는지 의미한다. 학습성을 높이기 위해 선택할 수 있는 첫 번째 방법은 시스템의 작동 방법과 원리를 기술한 매뉴얼을 제공하는 것이다. 매뉴얼이 단순히 작동 절차만을 설명하는 것이 아니라, 절차의 배경 원리를 제공하면 적절한 사용자 모형을 구축하도록 유도하여 학습성을 높일 수 있다. 다만 방대한 매뉴얼은 오히려 학습 시간과 노력을 가중하는 요소가 될 수 있으며, 특히 유희를 목적으로 사용하는 상업 게임은 사용자로 하여금 매뉴얼을 읽도록 권장하기 어려울 수 있다.

두 번째 방법은 학습의 목표를 시스템의 목적에 맞는 수준으로 설정하고, 이에 적합한 시스템 디자인을 제공하는 것이다. 게임은 초보자부터 숙련자까지 넓은 범위의 사용자들을 위해 기획되므로, 과업을 달성하기 위한 다양한 방법을 설계해야 할 것이다. 세 번째 방법은 기억가능성을 높이는 것이다. 기억가능성이 낮은 시스템은 사용자가 시스템이 쉽게 익숙해지지 않아서 사용할 때마다 사용법을 다시 익혀야 한다. 게임은 실패의 파급 효과가 큰 시스템은 아니지만, 대중적인 성공을 목표로 하는 게임이라면 게임의 조작 방식이나 규칙을 설계할 때 기억가능성을 염두에 두어야 할 것이다.

## 2. 이론적 배경: 변화제시성

---

변화제시성은 시스템의 상태가 변경되었을 때 사용자가 변화된 상태를 인지할 수 있는지 의미한다. 여기에는 두 속성이 있는데, 먼저 즉시성은 변화 발생 즉시 사용자가 변화를 인지할 수 있도록 변화된 시스템의 상태를 제공하는 것을 의미한다. 이어서 부가성은 사용자가 변화된 상태를 보기 위한 부가적인 행위를 취해야 시스템의 상태를 제공하는 것을 의미한다.

스타크래프트와 같은 전통적인 RTS 게임의 경우 사용자가 월드 전체를 조망하기 어렵고, 사용자의 주의소재가 특정 지점에 집중되는 상황이 잦기 때문에 **주의소재의 전환을 목표로 상태 변화를 제시할 수 있어야 한다**. 게임의 진행 과정에서 적절한 상황에 적절한 수준으로 상태 변화를 제시한다면 역동적인 사용자 경험을 구축할 수 있을 것이다. 그러나 또 한편으로는, **의도적으로 사용자에게 상태의 변화를 숨김으로써 추가적인 게임성을 확보할 수도 있기 때문에** 시스템 설계에 다양한 가능성이 주어지게 된다.

## 2. 이론적 배경: 이해가능성

---

이해가능성은 사용자가 제시된 정보를 이해할 수 있는지 의미한다. 따라서 변화제시성의 원칙에 따라 제시된 시스템의 상태와 정보를 실제로 사용자가 이해할 수 있어야 한다는 점이 중요하다.

시스템의 이해가능성을 높이는 방법 중 하나는 **가독성을 향상시키는** 것이다. 정보를 시각적으로 눈에 잘 띠도록 노출하거나, 텍스트를 정확하게 해석할 수 있도록 구성함으로써 가독성을 높일 수 있다. 특히 RTS 게임은 복잡한 시스템을 갖추고 있어 많은 정보를 텍스트로 전달하곤 하는데, 이러한 텍스트를 빠르고 정확하게 읽을 수 있어야 하기 때문에 가독성이 더욱 중요하다.

이해가능성을 높이는 또 다른 방법은 **논리성을 확보하는** 것이다. 제공된 정보가 논리적인 순서나 구조를 지킬 때 논리성이 확보되며, 이를 통해 사용자는 일관된 규칙을 바탕으로 쉽게 정보를 해석할 수 있게 된다. 시뮬레이션 게임의 경우 비논리적인 정보 구조가 게임 시스템 전체에 대한 불신을 야기하고, 결국 게임성을 저하하는 결과로 이어질 수 있기 때문에 주의해야 한다.

# 3. 첫 번째 단계: 학습성

---

초보 사용자가 게임의 규칙과 매커니즘을 익히며 기초적인 심성모형을 구축하는 단계

### 3. 학습성: 스토리텔링의 역할

스타크래프트는 세계관과 스토리에 상당한 공을 들인 게임 중 하나다. 게임 실행 시 재생되는 시네마틱은 테란과 저그의 전투 장면을 보여준다. 이러한 시네마틱은 사용자의 흥미를 끌고 게임의 몰입도를 높일 뿐만 아니라, 소설 스타쉽 트루퍼스(1959) 또는 영화 에일리언(1979) 등 SF 장르에 배경지식을 가진 사용자에게 게임에 대한 전반적인 이해도를 높이는 역할을 한다.

이러한 시네마틱을 통해 사용자는 이미 알고 있는 SF 장르의 배경지식과 새롭게 접하는 스타크래프트의 세계관을 결합해 첫 심성모형을 구축하게 되며, 대략적인 게임의 주제와 구도를 이해하게 된다.



<그림 3.1> 스타크래프트: 브루드워 오프닝 시네마틱 장면.

### 3. 학습성: 스토리텔링의 역할

스타크래프트의 세계관은 테란, 저그, 프로토스 세 종족이 대립하는 구도이며, 사용자는 원하는 종족을 선택하여 게임을 진행할 수 있다. 스타크래프트가 크게 영향받은 기존 RTS 게임들도 종족내지는 세력 개념이 있었다. 둔2부터 이미 아트레이더스, 하코넨, 오르도스 3개 가문이 등장했다. 하지만 이들은 이름만 다를 뿐, 유닛이나 건물을 비롯한 기능에 실질적인 차이는 전혀 없었다. 한편 워크래프트에는 인간과 오크 두 종족이 등장하며, 커맨드 앤 컨커에도 GDI와 Nod 두 진영이 등장하는데, 이들은 둔2와 달리 각자의 특성이 뚜렷이 구분되어 고유한 유닛과 건물을 제공한다.

각자의 고유한 특성과 체계를 갖춘 세 종족이 등장한다는 기획은 두 세력만이 등장한 기존 RTS 게임에 비해 게임 플레이의 불확실성과 전략적 다양성을 높여 스타크래프트의 성공에도 큰 영향을 미쳤다. 하지만 이러한 독특한 종족 시스템은 어느정도 학습이 선행될 필요가 있다. 스타크래프트는 싱글플레이 캠페인을 통해 세 종족을 각각 플레이해볼 수 있는 3개의 에피소드를 제공하여 자연스럽게 게임 시스템과 종족별 특성을 익힐 수 있도록 유도한다.

<그림 3.2> 에피소드 선택 화면.

에피소드 1은 테란, 에피소드 2는 저그, 에피소드 3은 프로토스로 플레이하도록 구성되어 있다.



### 3. 학습성: 스토리텔링의 역할

싱글플레이 캠페인은 기본적으로 스토리텔링을 제공하는 역할을 하지만, 미션을 진행할수록 적절히 난이도가 높아지도록 설계된 매뉴얼로서 기능하기도 한다. 싱글플레이 캠페인의 첫 미션 이름은 “Boot camp(훈련소)”로, 외계 행성에서 근무를 시작한 행정관을 안내하는 내용으로 구성되어 있어 직접적으로 사용자에게 게임의 규칙과 방법을 교육하는 시나리오가 기획되어 있다.



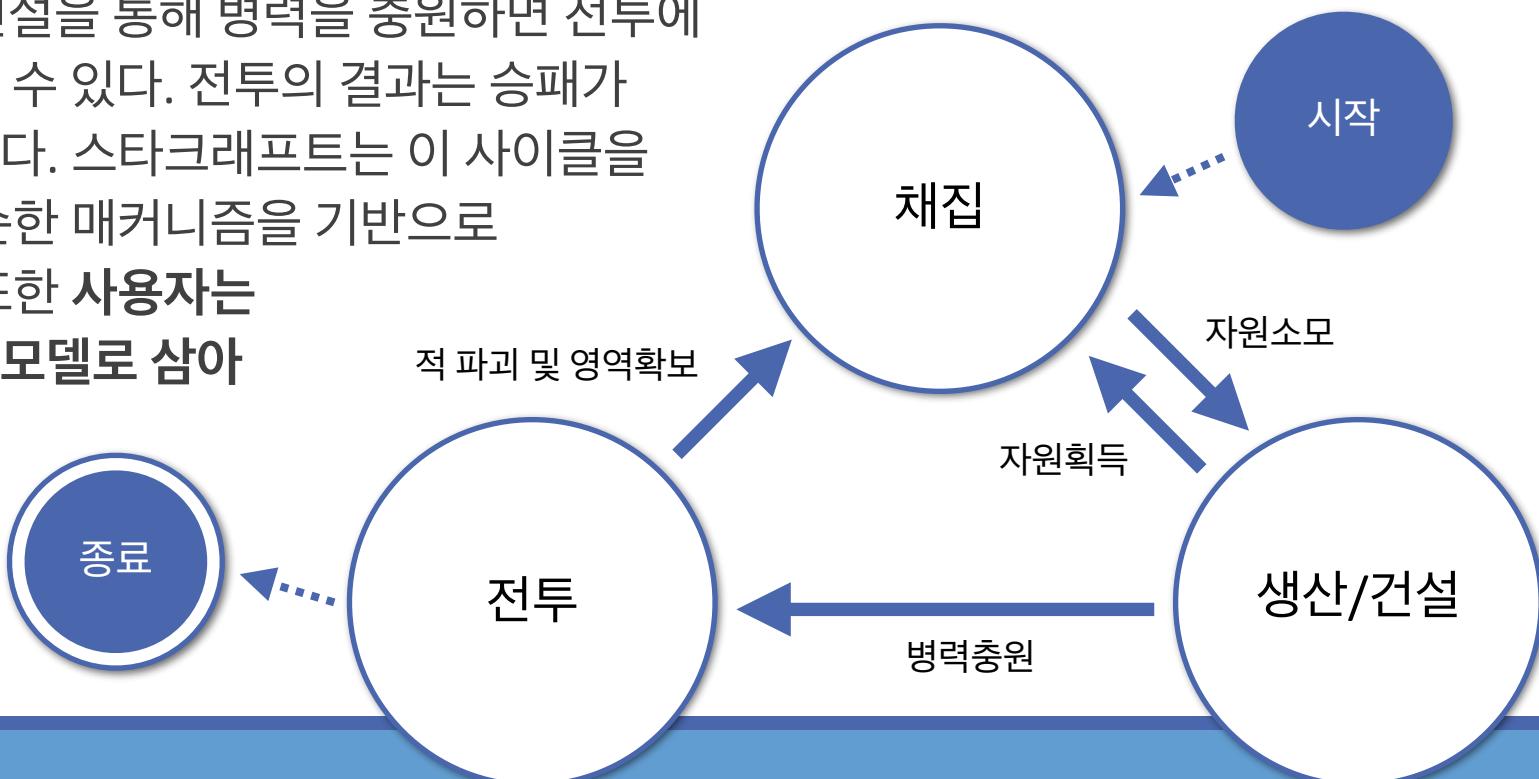
<그림 3.3> 오리지널 캠페인 에피소드 1(테란)의 첫 번째 미션 “Boot camp” 시작 화면.  
사용자가 자연스럽게 게임 시스템을 익힐 수 있도록 유도하고 있다.

### 3. 학습성: 게임 시스템 추상화

스타크래프트와 같은 대부분의 RTS 게임들은 사실성과 게임성을 트레이드 오프해야 한다. 사실성에 치중한 게임은 소수의 마니아층을 가진 시뮬레이션 게임이 되며, 게임성에 치중한 게임은 진입 장벽이 낮은 캐주얼 전략 게임이 된다. 스타크래프트는 사실성보다는 게임성에 집중한 게임으로 분류할 수 있다.

기획 의도에 맞춰 스타크래프트는 **전쟁을 채집-생산/건설-전투로 추상화했다**. 사용자는 자원(미네랄, 가스)을 채집하고, 자원을 소모해 병력을 생산하거나 건물을 건설한다. 이때 일부 유닛/건물은 자원(보급품)을 제공하기도 한다. 생산/건설을 통해 병력을 충원하면 전투에 투입해 적을 파괴하고 영역을 확보할 수 있다. 전투의 결과는 승패가 결정되는 게임의 종료 조건이기도 하다. 스타크래프트는 이 사이클을 반복하도록 설계되어 있는데, 이 단순한 매커니즘을 기반으로 무한한 전략적 가능성이 펼쳐진다. 또한 사용자는 이와 같이 추상화된 매커니즘을 심성모델로 삼아 **스타크래프트의 게임 시스템을 쉽게 이해할 수 있다**.

<그림 3.4> 상태 머신으로 추상화한 스타크래프트 매커니즘.

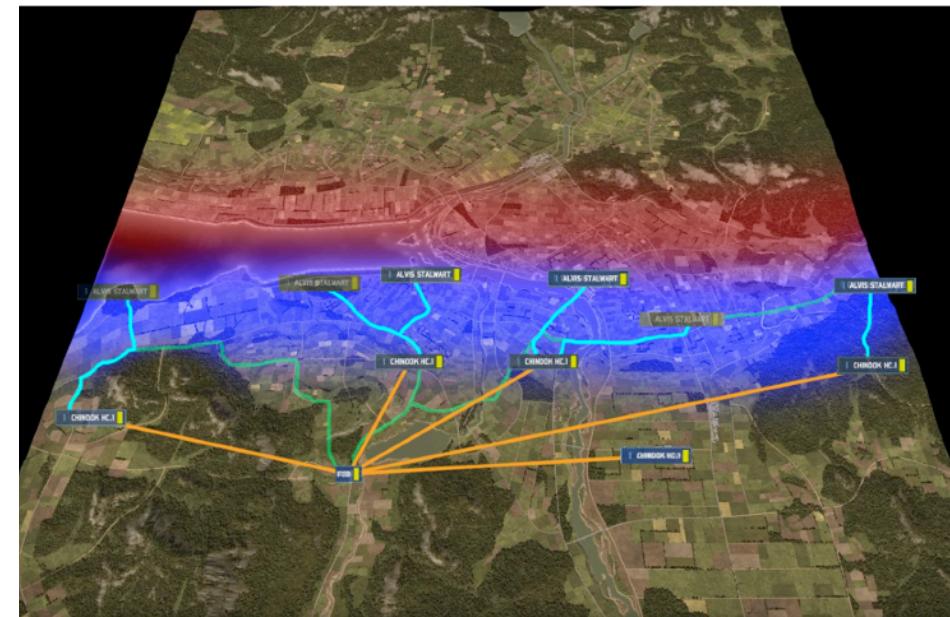


### 3. 학습성: 게임 시스템 추상화

**채집:** 병참이 추상화되어 있어 자원은 채집된 즉시 모든 생산 시설에서 사용 가능하다. 따라서 사용자는 자원의 수송과 물류를 신경쓰지 않아도 된다. 이는 전선까지 이어지는 보급 체인이 매우 중요한 실제 전쟁과 구분되는 지점으로, **사용자의 심성모형을 단순화함으로써 학습성과 게임성을 높인 사례다.**

반면 RTS 게임의 전신이라 할 만한 도상훈련의 전통적인 규칙을 계승한 워게임 장르는 높은 사실성을 추구한다. 서든 스트라이크 시리즈(2000~)나 워게임 시리즈(2012~)는 병참을 매우 중요하게 취급한다. 이 차이로 인해 스타크래프트와 같은 캐주얼 RTS 장르와 워게임 장르는 완전히 다른 정체성을 갖게 되며, 게임의 진행 방식과 난이도가 좌우된다.

채집은 게임의 방향을 새로운 방향으로 이끌기도 한다. 단순히 적에게 일시적인 피해를 입히고 파괴하는 것을 넘어서, **장기적 관점으로 자원을 확보해 게임을 유리한 방향으로 이끄는 전략이 중요해지기 때문이다.**



<그림 3.5> 워게임: 레드 드래곤(2014)의 보급 체인. ([reddit.com](https://www.reddit.com))

### 3. 학습성: 게임 시스템 추상화

**생산/건설:** 유닛의 생산은 특정 생산 건물에서만 가능하며, 건물의 건설은 특정 건설 유닛으로만 가능하다. 즉, 사용자는 어떤 건물에서 어떤 유닛을 생산할 수 있는지, 어떤 유닛으로 어떤 건물을 건설할 수 있는지 인지하고 있어야 한다.

이러한 생산/건설 시스템은 전작인 웍크래프트에서도 적용된 방식인데, 둑2나 커맨드 앤 컨커와는 다른 시스템이기 때문에 주목해볼 필요가 있다. 기존 RTS 게임은 병참을 추상화하듯 생산과 건설도 추상화했는데, 스타크래프트는 생산/건설 매커니즘을 각 건물과 유닛의 행동으로 노출하는 방식을 택한 것이다.

이와 같은 시스템은 사용자에게 추가적인 학습을 요구하지만, **인터페이스의 문법성과 심성모델의 일관성을 고려하면 매우 합리적인 방식**이라고 할 수 있다. 이에 대한 구체적인 내용은 후술할 “[4. 이해가능성: 명령 체계의 문법적 이해](#)”에서 다루도록 하겠다.

<그림 3.6> 스타크래프트 SCV의 건설 옵션.  
생산 건물이나 건설 유닛을 선택해 생산/건설 행동을 명령해야 한다.



<그림 3.7> 커맨드 앤 컨커: 레드얼럿(1996) OpenRA 버전.  
특정 건물이나 유닛을 선택하지 않아도 우측 패널에서 모든  
생산과 건설을 제어할 수 있다.



### 3. 학습성: 게임 시스템 추상화

**전투:** 전투는 게임의 승패를 결정하는 결정적인 요소로, RTS 게임의 핵심이라고 할 수 있다. 따라서 스타크래프트 역시 전투 시스템에 큰 노력을 들였는데, 전투는 채집, 생산/건설과 같은 다른 시스템에 비해 낮은 수준으로 추상화되어 있다. 여기서 낮은 수준으로 추상화되어 있다는 말은 조약하다는 의미가 아니라, 사용자에게 많은 제어권이 주어진다는 의미이다.

낮은 수준의 추상화로 사용자는 개별 유닛 하나하나의 행동을 제어할 수 있으며, 전략만큼 전술이 게임의 중요한 요소로 부각되어 APM(Actions Per Minute, 분당 행동수)과 같은 **사용자의 개인적인 역량이 스타크래프트에서는 매우 중요**해진다. 이러한 전술적 역량이 게임을 승리로 이끌 가능성을 높여 주기는 하지만, 게임의 불확실성으로 인해 항상 승리를 보장하지는 않는다는 점에서 스타크래프트의 공정성과 게임성이 확보된다.

또한 이렇게 마이크로 컨트롤을 필요로 하는 전투 시스템은 숙련을 어렵게 만들지만, 복잡한 전략 체계를 학습할 필요는 없기 때문에 초보자의 러닝 커브를 오히려 낮출 수 있다.

<그림 3.8> 스타크래프트 프로게이머의 입력 장치 조작 장면. ([youtube.com](https://www.youtube.com))  
빠른 조작 역량이 승패에 영향을 미치기 때문에 프로게이머들은 APM을 높이기 위한 연습을 한다.



<그림 3.7> 삼국지5(1995)의 전략 지도. ([saitamat.blog.fc2.com](http://saitamat.blog.fc2.com))  
사용자는 개별 병사가 아닌 추상화된 부대 단위를 제어한다.



### 3. 학습성: 다양한 과업수행 경로

스타크래프트는 같은 명령을 내리는 데 다양한 경로를 제공한다. 가령 한 유닛을 선택해서 적 유닛을 공격하도록 명령하는 경우 크게 3가지 경로 중 하나를 선택할 수 있다.

1번. (1) 유닛을 클릭한다. (2) 공격 아이콘을 클릭한다. (3) 적 유닛을 클릭한다.

2번. (1) 유닛을 클릭한다. (2) 키보드 A키를 누른다. (3) 적 유닛을 클릭한다.

3번. (1) 유닛을 클릭한다. (2) 적 유닛을 우클릭한다.

단축키 사용이 익숙치 않은 초보자는 2번 경로보다는 1번이나 3번 경로를 선호할 것이고, 보다 빠르게 명령을 입력하고자 하는 숙련자는 2번이나 3번 경로를 선호할 것이다. 더 나아가 3번 경로에서 적 유닛이 아닌 배경을 클릭할 경우 이동 중 조우하는 적과 전투한다는 것을 학습한 숙련자는 2번 경로와 3번 경로를 상황에 알맞게 선택할 것이다.

이렇게 하나의 과업을 수행하는 데 다양한 경로와 선택권을 제공함으로써 스타크래프트는 사용자의 제어주도성과 단축성을 확보하는 한편, 학습성도 훼손하지 않는다.

<그림 3.9> 공격 명령을 내리는 세 가지 경로.



### 3. 학습성: 테크 트리 시스템

스타크래프트는 테크 트리 개념을 채택해 게임의 진행 단계를 구조화하는 동시에, 생산/건설의 단계를 구분함으로써 합의된 심성모형을 구축한다. 테크 트리는 기본적으로 게임 경과 시간에 따라 플레이어간 화력의 균형을 맞춰주며, 사용자가 직접 트리의 경로를 선택하여 다양한 전략을 수립할 수 있게 만든다. (실제 구조는 DAG이다.)

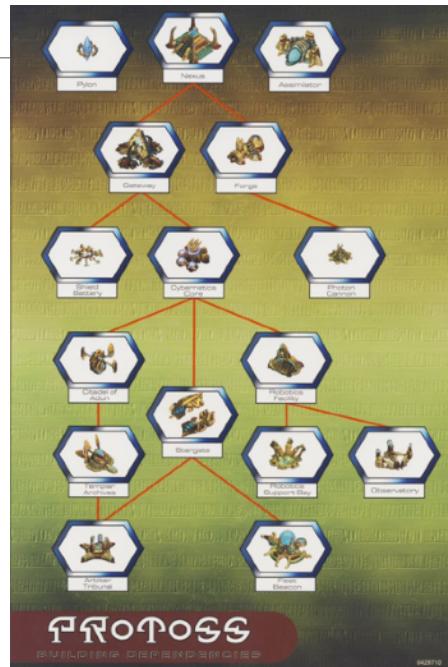
스타크래프트의 건설 테크 트리는 일반 건물과 고급 건물로 구분되어 있다. 고급 건물은 화력이 강한 유닛을 생산할 수 있으며, 더 많은 자원을 소모하기 때문에 자연스럽게 게임 후반부에 등장하게 된다. 일반 건물만으로도 채집과 전투가 가능하기 때문에 이러한 구분은 초보자로 하여금 학습의 부담감과 압도감을 낮출 수 있게 해준다.

그러나 사용자가 스타크래프트의 테크 트리를 명확히 트리 구조로 이해하려면 어느정도 학습이 필요하다. 건설 유닛의 건설 옵션을 선택하면 <그림 3.14>과 같이 현재 건설할 수 있는 건물과 선행 조건이 충족되지 않은 건물이 한번에 보이는데, 여기에서는 어떠한 선후 관계나 조건도 확인할 수 없기 때문이다. 건설하고자 하는 건물의 조건을 확인하려면 해당 건물에 커서를 올려 툴팁을 읽어야 하고, 선행 건물을 찾기 위해서는 다른 건물들에 하나씩 커서를 올려봐야 한다.

<그림 3.10> 프로토스의 건물 테크 트리. ([reddit.com](https://www.reddit.com/r/StarCraftII/comments/102qjwz/proto_stars_technology_tree/))

<그림 3.11> 일반 건물과 고급 건물의 구분.

<그림 3.12> 건설 조건이 충족되지 않은 건물에 커서를 올리면 선행 조건이 툴팁으로 나타난다.



### 3. 학습성: 테크 트리 시스템

테크 트리 개념은 둔2에서 처음 등장했다. 스타크래프트와 달리 둔2에서는 건물의 건설 조건을 충족해야 건설 리스트에 해당 건물이 노출되기 때문에 사용자가 적절한 심성모형을 구축하기 어렵고, 어떤 조건을 만족해야 다음 건물을 건설할 수 있는지 예측하기 힘들다. 다만 둔2의 테크 트리가 취하는 실제 구조는 연결 리스트이기 때문에 복잡하지는 않다.

방대한 테크 트리를 갖춘 문명 시리즈(1991~)는 트리 구조를 시각적으로 표현한다. 이러한 표현 방식은 사용자가 선후 관계를 직관적으로 이해할 수 있게 만들어주며, 전체 트리를 보며 미래 전략을 수립하는 데도 도움을 줄 수 있다.

단, 스타크래프트의 테크 트리는 문명과 달리 이미 건설한 건물도 다시 지을 수 있기 때문에 인게임 UI를 이와 같이 시각화하면 잘못된 심성모형을 구축하게 될 수 있다.

<그림 3.14> 문명5(2010)의 테크 트리. ([civilization.fandom.com](http://civilization.fandom.com))  
인게임에서 테크 트리가 시각화된 것은 문명2(1996)부터였다.



<그림 3.13> 둔2의 건설 선택 화면.  
건설 조건이 충족된 건물만 리스트에 노출된다.



## 4. 두 번째 단계: 변화제시성

---

게임이 사용자에게 상태 변화를 제시해 사용자의 행동을 유도하며 사건에 대한 인지와 그에 따른 행동이 심성모형에 구체화되는 단계

## 4. 변화제시성: 미니맵 알림

하단 정보 패널 UI의 가장 왼쪽에는 **게임 상의 유닛과 건물, 자원, 지형을 평면적으로 추상화해 제공하는 미니맵**이 위치해 있다. 미니맵은 실제 월드에 존재하는 유닛, 건물, 자원의 상태 변화를 실시간으로 반영한다. 스타크래프트에서는 사용자가 월드 전체를 조망할 수 없기 때문에 미니맵의 역할이 매우 중요해진다. 미니맵을 통해 사용자는 월드 전체의 대략적인 정보를 시각적으로 획득할 수 있고, 이를 통해 월드를 지리적으로 이해함으로써 적절한 심성모형을 구축할 수 있다.

미니맵의 또 다른 기능은 **상태 변화를 즉시 제공하는 것이다**. 유닛의 생산이나 건물의 건설이 완료되었을 때, 적의 공격을 받을 때, **미니맵 상에 하이라이트 애니메이션이 발생하며 사용자의 주의소재 전환을 유도한다**. 이 뿐만 아니라, 멀티플레이 게임에서는 한 사용자가 다른 사용자에게 특정 지점에 대한 알림을 보내는 기능을 제공하여 게임의 사회성을 제고하였다.

한편 커맨드 앤 컨커는 사용자가 자원(전력)을 확보하고 특정 건물(레이더)를 건설해야 미니맵의 기능이 활성화되도록 구현하여 **의도적으로 정보의 제공을 제한하고, 이를 통해 추가적인 게임성을 확보하기도 했다**.

<그림 4.1> 특정 사건(유닛 생산, 적 공격)이 발생했을 때  
미니맵에 사건이 발생한 위치가 하이라이트된다.



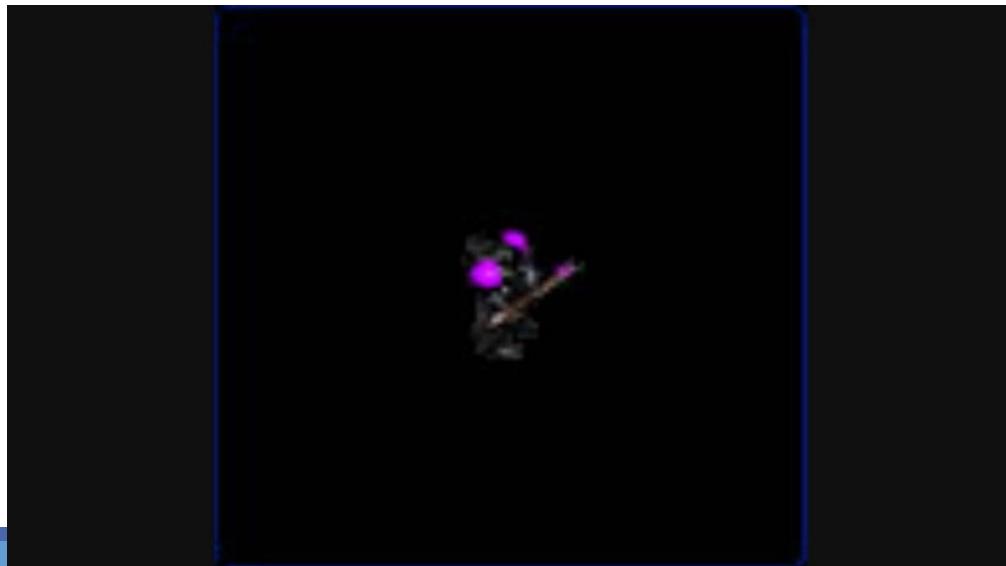
## 4. 변화제시성: 오디오 인터페이스

스타크래프트는 각종 사운드 효과를 통해 적극적으로 시스템의 현재 상태와 변화를 표현한다. 오디오 인터페이스를 이용하면 시각적인 효과 이상으로 사용자의 주의를 환기할 수 있기 때문에 특정 지점에 집중하고 있는 사용자의 주의소재 전환을 효과적으로 유도할 수 있다. 이러한 목적은 앞서 살펴본 미니맵 알림과도 유사하며, 스타크래프트는 사운드 효과와 미니맵 알림을 병용하여 청각적인 장치와 시각적인 장치를 동시에 사용한다.

이러한 사운드 효과의 특성을 응용해 게임의 균형을 조율하기도 했다. 게임 상에서 핵 미사일이 발사되면 미사일이 탄착해 폭발하기 전에 게임에 참여하고 있는 모든 사용자에게 “Nuclear launch detected”라는 경고 음성이 출력되는데, 이는 핵 미사일의 파괴력을 감안하여 경고를 제공하는 것이다.

사운드 효과는 변화제시성 뿐만 아니라, 게임 요소를 풍부하게 만들어 몰입도와 흥미도를 높이기도 한다. 실제로 유닛을 클릭했을 때 출력되는 유닛의 음성이 선풍적인 인기를 끌어 릴(Meme)이 되기도 했다.

<그림 4.2> 마린의 음성 대사. ([youtube.com](https://youtube.com))



## 4. 변화제시성: 건물의 상태 변화

스타크래프트에서 건물의 내구도가 일정 수치 이하로 내려가면 시간이 경과함에 따라 내구도가 서서히 떨어지며 파괴된다. 건물의 완전한 파괴를 방지하려면 사용자가 직접 수리 명령을 내려야 하기 때문에 건물의 내구도가 어떤 상태인지 인지하는 것이 중요하다.

스타크래프트는 건물의 상태 변화를 내구도에 따라 외형을 변경함으로써 제시한다. 가령 테란과 프로토스의 건물은 내구도가 낮을수록 건물에서 더 많은 불이 나는 방식으로 표현된다.

만약 구체적인 내구도 수치를 알고 싶다면, 사용자가 해당 건물을 클릭해 하단 상태 패널에서 확인해야 한다. 이는 변화제시성의 속성인 즉시성과 부가성을 섞은 것으로, 스타크래프트2(2010)에서 클릭한 유닛이나 건물의 내구도를 항상 유닛 위에 표시해준 것과 상반되는 디자인이다.



<그림 4.3> 내구도가 일정 수치 이하로 떨어져 불이 나는 건물.

<그림 4.4> 스타크래프트2는 선택한 유닛/건물의 내구도를 항상 표시한다.

## 4. 변화제시성: 유닛의 목표 표현

스타크래프트는 대부분의 경우 적절한 방식으로 사용자에게 정보를 제공하지만, 명령을 받은 유닛이 어떤 명령을 수행 중인지, 어떤 목표를 위해 행동 중인지 제시하는 데에는 부족함이 있다.

스타크래프트에서 유닛에게 명령을 내리면 유닛은 즉시 명령의 목표를 달성하기 위한 동작을 실행하지만, 이동 시간과 수행 시간 등의 요인으로 인해 그 **목표를 달성하는 데는 지연이 발생**한다. 이러한 지연 사이에 사용자가 다른 곳에 주의를 집중한 뒤 다시 돌아오면 과거에 내린 명령이 무엇인지, 유닛이 어떤 행동을 하고 있는지 기억하지 못할 수 있다. 특히 건설이 예정된 지점에 유닛을 배치하거나 다른 건물을 건설하도록 명령을 내려 기존 건설 명령이 취소되는 상황은 비일비재하다.

스타크래프트2는 건설을 명령하면 **건설 예정 지점에 완성될 건물의 실루엣을 보여주는** 식으로 문제를 해결한다. 또한 커맨드 앤 컨커: 레드얼럿2(2000)의 경우 유닛을 선택하면 해당 유닛이 어디를 목적지로 이동하고 있는지, 어떤 대상을 공격할 예정인지를 시각적으로 보여준다.



<그림 4.5> 스타크래프트2는 건설 예정지에 완성될 건물 실루엣을 표시한다. ([youtube.com](https://www.youtube.com))

<그림 4.6> 커맨드 앤 컨커: 레드얼럿2(2000)는 유닛과 공격 대상 사이에 선을 표시한다. ([youtube.com](https://www.youtube.com))

## 4. 변화제시성: 유닛의 목표 표현

최근 RTS 게임들은 사용자의 명령을 받은 유닛이 이동할 목적지 뿐만 아니라, 그 경로를 시각적으로 표현하기도 한다. WARNO(2022)는 유닛에게 특정 목적지로 이동하도록 명령하면 유닛이 이동할 경로를 보여준다. 다만 이 같은 방식은 **이동 명령과 동시에 목적지까지의 경로를 계산할 수 있을 때만** 사용할 수 있다. 스타크래프트는 우선 맵 데이터만으로 A\* 탐색 알고리즘을 수행해 유닛의 대략적인 경로를 계산하고, 유닛이 이동함에 따라 주변 장애물을 바탕으로 최적 경로를 매번 다시 계산하도록 구현(Patrick Wyatt, 2013)되어 있기 때문에 완성된 경로 정보를 미리 제공할 수는 없었을 것이다.

또한 스타크래프트에서 유닛의 공격 범위 밖에 공격을 명령하면 유닛은 공격 지점이 공격 가능 범위에 도달될 때까지 이동하기 시작한다. 화면 상으로는 사용자가 유닛의 공격 가능 범위를 쉽게 판단하기 어렵기 때문에 **공격 명령에 의도치 않은 이동 명령이 동반되곤 한다.**

WARNO의 경우 유닛의 시야와 공격 가능 범위를 시각화해 보여줌으로써 이러한 문제를 예방한다.

<그림 4.8> WARNO가 유닛의 이동 예정 경로를 시각화하는 방식. ([steamcommunity.com](https://steamcommunity.com))

<그림 4.7> WARNO는 유닛의 시야 범위와 현재 위치에서의 공격 가능 범위를 직관적으로 시각화해 표현한다. ([steamcommunity.com](https://steamcommunity.com))



# 5. 세 번째 단계: 이해가능성

---

사용자가 현재 상태를 이해하고 미래 상태를 예측하면서 심성모형이 고도화되는 단계

## 5. 이해가능성: 명령 체계의 문법적 이해

스타크래프트의 명령 체계는 주어-동사-목적어 순서로 구성된 문법으로 이해할 수 있다. 사용자는 명령을 내릴 대상을 선택함으로써 주어를 설정한 뒤, 이어서 행동을 선택함으로써 동사를 설정할 수 있다. 마지막으로 필요하다면 행동의 대상을 선택해 목적어를 설정할 수 있다. 예를 들어, 마린을 클릭(Marine)한 뒤 공격 아이콘을 클릭(Attacks)하고, 공격 대상인 적을 클릭(the Enemy)하면 “마린이 적을 공격한다 (Marine attacks the enemy)”라는 완결된 문장이 만들어지는 셈이다.

이러한 문법적 명령 체계는 사용자에게 익숙한 논리적 구조와 순서를 제공하여 인터랙션과 이를 통해 변경된 상태를 직관적으로 이해할 수 있도록 만들어준다. 또한 게임 전체에서 통용되는 문법 체계를 대원칙으로, 사용자는 무수히 많은 상태를 일관적인 규칙으로 이해하고 변경하며 응용할 수 있게 된다.

유닛 뿐만 아니라 건물에도 같은 문법이 적용되기 때문에 어떤 유닛을 생산하려면 생산 건물을 주어로 선택한 뒤 어떤 유닛을 생산할지 선택(동사+목적어)해야 한다. “[3. 학습성: 게임 시스템 추상화](#)”에서 언급했듯이, 만약 커맨드 앤 컨커와 같이 생산/건설을 통합된 패널에서 수행할 수 있게 한다면 주어를 생략해 문법 체계의 일관성을 위배하게 된다.

<그림 5.1> 일련의 문법적 명령 인터랙션.

- (1) 특정 마린을 클릭하고 (2) 공격 아이콘을 클릭한 뒤 (3) 적 유닛을 클릭해 사용자가 마린이 적을 공격하도록 명령을 내릴 수 있다.



## 5. 이해가능성: 명령 체계의 문법적 이해

---

제프 래스킨(2003)은 주어를 먼저 선택한 다음, 여기에 동사를 적용하는 **주어-동사(Object-Action) 구조가 올바른 인터랙션 설계**라고 강조했다. 동사를 먼저 선택 한 뒤 주어를 선택하는 동사-주어(Action-Object) 구조의 경우 동사를 선택하면 다른 동사를 선택하기 전까지 그 상태가 지속되어 필연적으로 모드(Mode) 방식이 되며, 잘못된 동사를 선택한 경우 모드 상태에 진입하기 때문에 선택을 취소할 인터페이스가 추가적으로 필요하다는 문제가 생긴다.

스타크래프트는 모든 명령 체계를 주어-동사 인터랙션 기반 위에 구현했다. 이렇게 **통일된 체계를 기반으로 사용자는 상태 변화에 따라 제시되는 정보를 쉽게 이해할 수 있고, 쉽게 변경할 수 있게 된다.** 예외가 있다면 앞서 언급한 것과 같이 동사와 목적어를 통합한 단축 경로를 제공한다는 점이다. 가령 건물 선택 후 생산할 유닛을 선택하는 경우, 유닛 선택 후 공격할 적 유닛을 우클릭하는 경우가 있다. 이러한 단축 경로의 제공은 사용성을 위한 최소한의 타협으로 봄야할 것이다.

## 5. 이해가능성: 상태 패널

상태가 실시간으로 바뀌는 RTS 게임은 그 특성상 사용자에게 게임의 현재 상태를 제시하는 것이 상당히 중요하다. 스타크래프트는 3분할된 하단 패널 중 가운데 위치한 상태 패널을 통해 선택한 대상의 상태를 보여준다. 상태 패널은 변화제시성을 위한 인터페이스이기도 하지만, **상태 패널의 핵심은 문법 체계 상의 주어를 표현하는 것이다.** 즉, 상태 패널 역시 문법적 명령 체계의 일부이며, 이러한 장치를 통해 사용자는 자연스럽게 문법 체계를 중심으로 게임과 상호작용하는 심성모형을 갖게 된다. 이와 같이 일관된 체계로 정보를 표현함으로써 궁극적으로 이해가능성이 높아진다.

하나의 유닛을 선택하면 해당 유닛의 상세 정보가 표시된다. 한편, 여러 유닛을 선택하면 선택된 유닛들의 아이콘만이 표시된다. 이때 상태 패널에서 특정 유닛을 클릭하면 해당 유닛을 단일 선택할 수 있고, 컨트롤 키를 누른 채로 유닛을 클릭하면 선택된 유닛 그룹에서 해당 유닛을 제외할 수 있다. 즉, **상태 패널은 단순히 현재 상태를 제시하는 역할을 넘어, 사용자가 직접 명령을 내릴 수 있는 제어 패널로서 기능한다.**



<그림 5.2> 하나의 유닛을 선택한 경우.



<그림 5.3> 여러 유닛을 선택한 경우.

## 5. 이해가능성: 커서 인터페이스

스타크래프트는 커서의 호버링 대상에 따라 다양한 커서 UI를 제공하여 사용자가 마우스를 클릭하면 상태가 어떻게 변경될지 예측할 수 있게 디자인 되었다. 커서의 종류는 일반(Normal) 커서, 검사(Inspection) 커서, 대상(Target) 커서로 분류할 수 있다.

- **일반 커서:** 화살표 형태의 일반적인 커서로, 일반적인 UI를 선택할 때 사용한다. 마우스를 이용해 동사를 설정하는 경우에도 패널의 버튼 UI를 거치므로 일반 커서를 사용하게 된다.
- **검사 커서:** 유닛이나 건물을 선택하는 커서로, 주어를 설정할 때 사용한다. 클릭하면 상태 패널에서 클릭한 대상의 상태를 확인할 수 있다. 주어가 설정되어 있을 때 우클릭하면 클릭한 대상을 목적지로 이동하거나, 대상을 공격하게 된다.
- **대상 커서:** 어떤 동작을 취할 대상을 선택하는 커서로, 동사가 설정된 상태에서 목적어를 설정할 때 사용한다. 대상 커서를 클릭하면 해당 대상을 상대로 즉시 어떤 동작이 실행된다.

커서의 색상은 피아식별 기능을 한다. 예를 들어 사용자의 유닛/건물 위에 커서를 올리면 커서 색상이 녹색으로 표시되고, 적 유닛/건물 위에 커서를 올리면 커서가 빨간색으로 표시되는 식이다.

이처럼 상황에 따라 커서의 형태와 색상을 바꾸는 것은 평가차의 저하를 선제적으로 방지하기 위한 매우 일반적인 전략이다.

(위에서 아래로) 일반 커서,  
검사 커서: 아군, 중립, 적군.  
대상 커서: 아군, 중립, 적군.

<그림 5.4> 스타크래프트의 커서 UI. ([sprites-resource.com](http://sprites-resource.com))



## 5. 이해가능성: 피해 공식과 유닛 상성

스타크래프트에서 각 유닛은 기본적으로 크기를 기준으로 분류(Small, Medium, Large)되며, 적에게 피해를 주는 방식에 따라 피해 형식으로도 분류(Concussive, Normal, Explosive)된다. 이러한 분류에 따라 적에게 얼마나 피해를 입힐 수 있는지 계산하는 피해 공식이 세워지는데, 가령 피해 형식이 Explosive인 유닛은 크기가 Small인 적 유닛을 공격할 때 공격력의 50%만이 적용되는 식이다. 유닛의 공격력은 유닛을 선택한 뒤 상태 패널의 공격력 아이콘에 커서를 올려야 확인할 수 있고, 인게임에서 유닛의 크기와 피해 형식을 확인할 방법은 없다. 이 정도의 정보만으로 사용자가 정확한 피해 공식을 바탕으로 어떤 유닛이 어떤 유닛을 상대하는데 효과적인지 구체화하기는 어렵지만, 게임을 경험하면서 사용자의 심성모형에는 유닛간 상성이 어렵듯이 형성되고, 이것이 개인간 역량 차이로 이어진다.

얼마 지나지 않아 숙련된 사용자들은 맵 에디터를 통해 유닛의 정확한 크기와 피해 형식을 찾았고, 이를 통해 구체적인 유닛간 상성 관계를 밝혀냈다. 심지어 유닛의 피해 판정 영역까지 분석하는 등 사용자 모형과 개발자 모형의 경계가 흐려지는 상황도 펼쳐졌다. 스타크래프트2는 아예 인게임에서 이러한 정보를 제공한다.

<그림 5.5> 유닛을 선택하면 공격력 아이콘에 커서를 올리면 공격력 정보가 툴팁으로 표시된다.

<그림 5.6> 스타크래프트2는 인게임에서 유닛 크기와 피해 형식 등 정보를 제공한다. ([sc2-coop.fandom.com](http://sc2-coop.fandom.com))



## 5. 이해가능성: 피해 공식과 유닛 상성

전략 시뮬레이션 게임에서 유닛간 상성 시스템은 매우 광범위하게 채택되곤 하지만, 이것을 인게임에서 구체적인 수치로 밝히는 경우는 많지 않다. 이는 사용자의 마이크로 컨트롤이 중요한 RTS 게임의 특성에 따라, 상성 관계를 파악하고 이를 바탕으로 전술적 판단과 선택을 내리는 것 역시 사용자의 역량이라고 본 관례라고 할 수 있다. 또한 RTS 게임은 전투의 호흡이 빠르고, 상황에 따라 다양한 전략과 전술을 선택할 수도 있다. 이러한 배경에서 제한된 정보를 바탕으로 현재 상태를 이해하고, 미래 상태를 예측하며 적절한 판단을 내리도록 하는 것이 게임성을 높이는 방법이 되기도 한다.

마이크로 컨트롤이 필요한 전술보다는 오랜 시간 숙고하는 전략이 중요한 문명 시리즈의 경우 전투 명령 전에 유닛간 상성 관계와 유닛의 상태, 지형 등 다양한 맥락을 바탕으로 전투 결과를 예측하여 사용자에게 제시한다. 이와 같은 디자인은 유닛간 상성을 직관적으로 이해할 수 있게 만든다.

<그림 5.7> 문명 시리즈는 유닛간 상성 관계를 바탕으로 전투의 예측 결과를 사용자에게 명시적으로 제시한다. ([steamcommunity.com](http://steamcommunity.com))



# 6. 결론

---

# 6. 결론

---

여기까지 스타크래프트가 어떻게 적절한 사용자 심성모형을 구축하는지 단계적으로 살펴보았다.

첫 번째 단계는 **사용자가 게임의 기본적인 규칙과 구도를 익히면서 기초적인 심성모형을 구축하는 단계로**, 시스템의 학습성을 중점에 두고 분석했다. 스타크래프트는 스토리텔링을 제공함으로써 사용자가 자연스럽게 게임 시스템과 세 종족의 특성을 학습할 수 있도록 유도하고, 복잡한 게임 시스템을 채집-생산/건설-전투라는 단순한 사이클로 추상화해 사용자가 쉽게 심성모델을 구축할 수 있도록 돋는다. 또한 다양한 과업수행 경로를 제공해 사용자의 제어주도성과 단축성을 확보하는 동시에 학습성도 잃지 않았다. 다만 테크 트리 시스템은 선후 관계를 파악하기 어려워 학습하기 어려운 측면이 있었다.

두 번째 단계는 **게임이 사용자에게 상태 변화를 제시하여 사용자가 사건을 인지하고, 이에 따른 행동이 심성모형에 구체화되는 단계로**, 변화제시성의 측면에서 세부적인 요소를 살펴보았다. 사용자의 주의소재가 특정 지점에 국한되는 RTS 게임의 특성을 가진 스타크래프트는 미니맵 알림과 오디오 인터페이스를 통해 사용자의 주의소재 전환을 유도한다. 또한 건물의 상태 변화를 시각적으로 제시해 건물의 내구도에 대한 적절한 정보를 제공한다. 한편, 유닛의 현재 목표 상태를 제시하는 데에는 부족한 부분이 있었다.

## 6. 결론

---

세 번째 단계는 사용자가 제시된 정보를 바탕으로 현재 상태를 이해하고, 미래 상태를 예측하면서 심성모형을 고도화하는 단계로, 이해가능성을 중심으로 분석하였다. 스타크래프트는 주어-동사-목적어로 구성되는 문법적 명령 체계를 통해 사용자에게 익숙한 논리 구조와 순서를 제공하고, 일관된 규칙을 정립한다. 상태 패널은 주어를 표현하는 인터페이스로, 자연스럽게 사용자의 심성모형에 문법적 명령 체계를 구축한다. 또한 맥락에 따라 커서의 외형을 달리 함으로써 평가차를 줄인다. 마지막으로 유닛의 피해 공식과 유닛간 상성관계의 경우 제공되는 정보가 제한되어 있어 쉽게 이해하기 어렵지만, 이러한 제약이 오히려 게임성을 높이는 요소가 될 수 있음을 짚었다.

스타크래프트는 둔2, 웍크래프트, 커맨드 앤 컨커와 같이 앞서 출시된 RTS 게임들의 영향을 크게 받은 게임으로, 이들의 핵심적인 시스템과 장점을 적절히 채택, 개선하여 높은 수준의 사용자 경험을 이끌어 냈다. 이러한 성취는 복잡한 규칙을 직관적으로 이해할 수 있도록 돋는 장치와 접근법을 취해 적절한 사용자 심성모형을 구축했기에 가능했다. 스타크래프트가 실현한 체계적 인터랙션 디자인은 오늘날에도 많은 게임과 범용 소프트웨어에서 찾아볼 수 있다.

# 참고문헌

---

- 김진우, “Human Computer Interaction 개론”, 안그라픽스, 2012.
- 도널드 노먼, “디자인과 인간 심리”, 박창호 역, 학지사, 2016.
- 도널드 노먼, “도널드 노먼의 디자인 심리학”, 범어디자인연구소 역, 유엑스리뷰, 2018.
- 제프 래스킨, “인간 중심 인터페이스”, 이건표 역, 안그라픽스, 2003.
- Bart Farkas, “StarCraft: Prima’s Official Strategy Guide”, 1999.
- Patrick Wyatt, “Tough times on the road to Starcraft”, [codeofhonor.com](http://codeofhonor.com), 2012.
- Patrick Wyatt, “The StarCraft path-finding hack”, [codeofhonor.com](http://codeofhonor.com), 2013.