클리프 블레스진스키의 게임 개발자 플래시카드

(Cliff Bleszinski's Game Developer Flashcards)

작성자: 클리프 블레스진스키(Cliff Bleszinski)

작성일: 2012년 7월 5일

이번 가마수트라 특집에서는 에픽 게임스(Epic Games)의 설계 디렉터 클리프 블레스진스키가 개발자들에게 공통적으로 나타나는 행동 양식을 정리한다.

내가 게임을 전문적으로 제작한지 올해로 20년이 되었다. 그 동안 캐릭터 마스코트 플랫폼 게임, 1인칭 슈팅 게임, 싱글 플레이어 게임, 멀티 플레이어 게임 등등의 기획을 맡아왔고 아주 뛰어난 프로그래머, 아티스트, 애니메이터, 작가, 프로듀서들과도 작업을 했다. 그러면서 창의력을 바탕으로 하는 우리 업계 사람들에게 특유의 커뮤니케이션 패턴이 있다는 걸 깨달았다.

개발자들은 놀라울 정도로 똑똑하지만, 함께 일하는 사람들에 비해 자신들이 얼마나 똑똑한지 모를 때가 가끔 있는 것 같다. 나는 온라인 개발자 게시판에서 사소한 문제를 지나치게 분석하며 물고 늘어지면서 십억 달러짜리 프랜차이즈부터 우수한 인디 업체에 이르기까지 모두를 완전히 찢어갈기는 걸 여러 차례 목격했다. 우리는 언제나 제일 먼저 아이디어를 떠올렸다는 걸 입증하거나 어떤 아이디어를 시도했거나, 성공했거나, 실패 했거나 어쨌든 써먹은 사례를 이야기하고 싶어한다.

간단히 말해 이 글은 게임 개발자들이 토의, 토론, 논쟁에서 '승리'하기 위해 종종 사용하는 커뮤니케이션 기법에 대한 글이다.

이런 관찰 결과나 내가 붙인 이름들에 악의는 전혀 없다. 실제로 여기에서 언급하는 일부 접근법은 단지 타당한 논리적 추론을 위해 사용한 것이다. 예를 들어 패턴 매칭(pattern matching)은 파생상품을 만들지 않기 위한 좋은 방법이다. 사실 어떤 극단적인 사례가 확고한 아이디어를 무너뜨리기도 한다. 그래서 게임 개발자의 커뮤니케이션 '플래시카드'를 소개한다.

"패턴 매칭으로 묵살(Pattern Match Dismissal)"

개발자가 어떤 아이디어를 묵살하기 위해서 머리 속 게임/대중문화 라이브러리에서 가장 비슷한 것(대개 나쁘거나 실패한 사례)을 찾아내는 것을 말한다.

예를 들어, <아바타(Avatar)>에 대해 "시퍼런 사람들이 숲 속에서 살면서 나쁜 군대, 기계와 싸우는 영화를 만들겠다고? 뭐야, <푸른 골짜기 에일리언 스머프(Smurf Ferngully Aliens)>야?"라고 하는 것이다. <기어스 오브 워(Gears of War)>도 삐딱하게 보면 80년대 싸구려 공포 영화 <C.H.U.D.>나 별반 다르지 않다.

"극단적 사례로 막기(Edge Blocking)"

극단적 사례로 기막히게 좋을 수 있는 아이디어를 묵살하는 걸 말한다. 일례로 왔던 길을 그대로 돌아가야 해서 지루할 수 있다는 이유로 <스카이림(Skyrim)>의 거인 세계(giant world) 아이디어를 거절한 사람이 있었다. '빠른 이동(fast travel)' 같은 확실하고 분명한 수정방법으로 문제를 쉽게 해결해 훨씬 큰 세계를 구현할 수 있는 데도 말이다.

극단적 사례로 막기의 변종으로 다음과 같은 것들이 있다.

"네트워커(The Networker)": 극단적 사례나 코옵(co-op) 상황에서 제대로 되지 않을 거라며 아이디어를 묵살하는 걸 말한다. 이것도 극단적 사례 막기다. "<맥스 페인(Max Payne)> 슬로 모드를 어떻게 코옵에 포함시킬 수 있겠어?

불가능해!"

"완벽주의자(Perfectionist)": 극단적 사례로 막기와 비슷하다. 개발자가 어떤 멋진 기능이 전적으로 완벽하게 보이지 않는 사례를 찾았을 때 나타난다. 예를 들어 아수라장 장면에서 일부 캐릭터가 서로 겹쳐 잘릴 거라고 하는 것이다.

"잘 될 리가 없어(Ne'er Do Well)"

혹은 "그런 기능을 본 적이 없어. 또는 그런 기능이 잘 되는 걸 본 적이 없어. 그러니 하면 안 돼."하는 것이다. 굳이 따로 설명이 필요가 없는 항목이다. 사실 이게 바로 어떤 아이디어를 반드시 실행에 옮겨야 하는 '이유'가 될 수도 있다. 이런 식의 논리를 따르자면 다른 사람의 성공을 모방할 수밖에 없다.

<기어스 오브 워>의 지하에서 온 로커스트(Locust) 족은 여러 역할을 하지만 꼭 필요한 것은 아니다. 하지만 결국 로커스트 덕분에 흔한 외계인을 내세운 다른 프랜차이즈와의 차별화에 도움이 되었다.

"일부러 반대하기(Devil's Advocate)"

개발자 대부분은 아이디어가 마음에 들어도 빠져나갈 구멍을 마련하려고 이렇게 행동한다. 변호사들이 대개 그렇듯.

"그냥 두 개 합친 거네(It's just X+Y)"

개발자가 어떻게 구성되어 있는지 쉽게 파악이 된다는 이유로 성공한 제품에 대해 별거 아니라고 말하는 것이다. 먹지 못하는 포도가 실 거라고 말하는 여우와 같다. 사실 간단하고 명백하기 때문에 성공하는 제품이 많다.

예를 들어 <워즈 위드 프렌즈(Words with Friends)>에 대해 이렇게 말하는 것이다. "그냥 비동기식 스크래블(asynchronous Scrabble)이잖아." 그렇다. 그래서 대단한 거다.

“나중에 넣자(Future Release)”

개발자가 좋든 나쁘든 아이디어를 듣고 후속 버전이나 속편에 잘 맞겠다고 하는 경우다. 사실 이 말의 속뜻은 “해볼만한 가치가 있는 아이디어 같지 않기 때문에 나중에 출시하겠다고 말해서 상대방 기분을 상하지 않게 하겠다”는 것이다.

“넘어뜨리기(Toppling)”[”바벨탑(Tower of Babel)”]

간단했던 기능에 개발자가 지나치게 많은 부가 기능을 넣는 바람에 결국 기능 자체가 제대로 발휘되지 못하는 경우다. 기능이 결국 스스로의 무게를 이기지 못하고 ‘넘어지는’ 것이다.

“애들을 생각 해야지(Think of the Children)!”

‘연속 의존(Cascading Dependencies)’라고도 한다. 애니메이션, UI, 미술 등 다른 부서의 작업량이 많아진다는 이유로 아이디어를 묵살하는 경우다. 하지만 대개 흥미롭고 시도해 볼만한 가치가 있는 기능은 전분야가 다 망라되어 있기 때문에 작업량이 많아질 수밖에 없는 경우가 많다.

“분석 마비(Analysis Paralysis)”

지나치게 생각을 많이 한 나머지 결국 아무것도 하지 못할 때 흔히 사용하는 용어다.

“뭐 하러 해(Why Even Try)?”

사실 “우리가 경쟁이나 되겠어?”란 뜻이다. 특정 분야의 격심한 경쟁에 겁을 먹은 개발자는 ‘최선의 노력(old college try)’을 다하기도 전에 이런 식으로 포기한다.

“거긴 개발자가 몇 명인지 알아(They Have N Developers)?”

개발자가 경쟁 팀의 게임 제작 인원이 얼마나 많은지 말할 때 흔히 쓰는 말로 대개 “뭐 하러 해?”로 이어진다. 우리 에픽(Epic)의 방식은 항상 업계 최고의 도구와 최고의 인력을 투입시켜 보다 스마트하게 작업을 진행하는 것이다.

“전통주의자(Traditionalist)”

“하지만 항상 이런 식으로 해왔다고!” 엔터테인먼트 산업, 특히 기술분야에서 혁신과 남다른 사고방식은 살아남기 위해 필수적이다. 현실에 안주하면서 계속 같은 식으로 일을 하는 것은 실패로 가는 지름길이다.

다른 분야에서는 20년을 몸담아 온 베테랑이라는 게 이점이 될 수 있지만 기술 분야에서는 한계가 될 수도 있다. 개발자는 나이가 들수록 항상 열린 마음으로 배우는 자세를 유지하는 것이 아주 중요하다.

“하지만 우린 (스튜디오명)잖아 (But We’re …)

일정 수준의 성공을 거뒀으니 다른 스튜디오가 감히 범접하지 못할 거라 생각하며 성공에 도취돼 하는 말이다. 사람들이 이렇게 말하기 시작하면 스튜디오가 파멸의 길을 걸을 날이 얼마 멀지 않은 것이다. 세상에는 그렇게 성공하는 꿈을 꾸고 그 꿈을 간절히 이루고 싶어 하는 젊은이들이 많이 있기 때문이다.

“전에 해 봤어(We Tried That Before)”

어떤 아이디어를 전에 시도해봤다가 실패했다고 말하며 새로운(제대로 될 수도 있는) 아이디어를 묵살하는 경우다.

“지나치게 멋지네(Too Cool)”

아이디어는 훌륭하다! 사실 너무 멋지고 혁신적이다. 그래서 할 수가 없다. 일이 많을 것 같으니까.

“전문용어로 압박하기(Jargonating)”

개발자가 다른 분야 개발자와의 말싸움에서 이기기 위해 자기 분야에서만 사용되는 전문용어를 사용하는 것이다. 코드 개발자가 미술담당자에게 코드 용어를 사용하거나 기획자가 애니메이터에게 기획 용어를 사용하는 경우가 이에 해당한다.

“우리가 왕이다(The Tribal Leader)”

자기 분야(미술, 코드, 설계 등)가 스튜디오의 다른 분야에 비해 중요하다고 생각해 개발자가 다른 분야 사람들을 무시하는 경우다.

“범위를 벗어난 거네(Noscope)”

“아이디어는 훌륭하지만 프로젝트 범위를 벗어났네.” 불행히도 최고의 기능은 레이더 망에 포착되지 않는 주변부에 존재하거나 원래의 일정에 포함되어 있지 않은 경우가 적지 않다.

“플레이테스트 수정(Playtest Grandstanding)”

새로운 기능이나 무기 개발에 실패한 개발자가 소리 높여 플레이테스트 때 수정해서 ‘균형(balancing)’을 맞추자며 자기 맘대로 하는 상황을 말한다. 스나이퍼 건을 들고 꼼짝 못하다가 다른 사람들한테 죽는 경우가 생겨도 괜찮다고 넘어갈 태세다.

“아이디어 도둑(The Repitcher)”

남의 아이디어를 듣고 처음에는 못 들은 척 하다가 원래 어디서 들었는지도 잊고 자기 아이디어인양 펼치는 사람을 말한다. 멋진 아이디어가 어딘가에서 나와서 잘 실행되기만 한다면 궁극적으로 별 문제는 없다!

“필리버스터(Filibuster)”, “너무 길어서 안 읽게 되는 사람(TL;DR [1] Guy)”

---

[1] TL;DR =Too long;Didn’t read

기획 제안이나 논의에 3 페이지짜리 이메일로 답을 하는 사람이다.

그것도 매번.

한 번도 빠짐 없이.

결국 이 사람 메일은 필터링된다.

“온라인 바보(The E-Douche)”

그럴 의도가 없는데도 불구하고 이메일 쓰는 능력이 없어 한결 같이 어처구니 없는 이메일을 보내는 사람이다.

“고질라(Godzilla)”

진전된 사항을 어떻게 해서든 모두 뒤엎어 버리고 자기 혼자만 더 낫다고 생각하거나 때로는 더 나쁘기까지 한 완전히 새로운 아이디어를 내놓는다. 결국 모든 사람이 처음부터 다시 시작하게 만드는 사람이다.

“우유부단의 극치(The Doubter)”

뚜렷한 이유도 없이 “잘 모르겠는데……”라고 하며 어떤 아이디어를 좋지 않다고 하는 사람이다. 유용하게 쓰이는 경우도 있다.

“예언자(Prophet)”

기획자가 어떤 아이디어에 급히 흥분한 나머지 설계나 영향에 대해서는 곰곰이 생각해보지 않은 경우를 말한다. 이런 기획자는 그 아이디어가 왜 좋은지 궁리하지 않고 모든 사람이 그저 아이디어가 잘 될 거라는 믿음을 갖기 바란다. 젊고 경험이 부족한 기획자에게서 흔하게 나타난다.

“포기를 모르는 인간(Captain Ahab)”[2]

어떤 아이디어가 제대로 진행되지 않는다는 걸 인정하지 않고 끊임 없이 소중한 코드와 미술 자원을 사용하며 언젠가는 게임이 재미있어질 거라고 생각하는 기획자를 말한다.

“데이터 폭발(Data Bombing)”

이런 이야기는 대개 이렇게 흘러간다. “여기 별 관계 없고 완전히 편견에 가득한 데이터에 따르면 너의 아이디어는 절대 성공할 리 없어. 많은 사람들이 불쾌해 할 거야. 우린 이렇게 할 수가 없어.”

“초자연적 기대(Psychic Expectations)”

코딩을 하는 사람들이 쓰는 방법으로 정확하게 원하는 방식대로 요청이 들어오지 않으면 주문 받은 기능을 이해하지 않으려고 하는 상황을 말한다.

“완전히 못 본 척 하기(Ignorannihilation)”

개발자가 고의적으로 (혹은 모르고) 뻔한 것을 보지 못하고, 그게 짤리기 전까지 왠지 좋을 것 같은 기능에 매달려 있는 상태를 말한다.

“내 아이디어 아니니까 난 안 해(Not My Idea, Not Going to Do It)”

---

[2]소설『백경(Moby Dick)』에 등장하는 선장 이름. 3일 동안 사투를 벌이지만 결국 고래에 목숨을 잃고 배는 침몰하게 됨

기획자가 다른 사람에게 아이디어를 원한다고 해놓고서는 자기 생각이 아닌 건 다 무시하는 걸 말한다.

“정원사(The Gardener)”

정원사는 일찌감치 아이디어의 씨를 뿌리고 사무실 사람들과의 미팅이나 일상 대화를 통해 계속해서 그 아이디어를 키운다. 결국 아이디어는 사람들 사이에서 뿌리를 내리고 자라나 게임의 실제 기능이 되지만 아무도 애초에 그 아이디어가 어디에서 나왔는지 기억하지 못한다. *이건 진짜로 매우 유용한 기법이다.*

“확실한 버그 인간(Obvious Bug Guy)”

진행 중인 기능 작업을 보여주었을 때 그 기능이 어디로 향해 가야 할지, 무엇을 할 수 있을지 생각하지 않고 Z 파이팅(Z-fighting)처럼 나중에 분명히 수정될

오류만을 지적하고 싶어하는 개발자를 말한다.

“내 상사는 누구?(Multiboss)”

상사나 명령 체계가 확실하지 않아 여러 사람으로부터 집중적으로 해야 할 일에 대한 지시를 받는 경우이다. 설계 디렉터, 책임 프로듀서, 사장이 작업에 대해 서로 다른 견해를 갖고 있으면 이런 직원은 혼란에 빠질 수밖에 없다.

“약속 남발자(The Promiser)”

언론에 어떤 기능을 구현하겠다고 홍보 또는 약속을 하는 바람에 팀 전체를 곤란에 빠뜨리고 결국 언론에 공개한 기능의 코딩 작업을 하게 만드는 사람을 가리킨다.

“시류에 편승하는 자(The Bandwagoner)”

혁신적인 다른 방법을 찾지 않고 최근 인기 있는 게임에서 본 기능이라면 무엇이든 추가해서 게임을 개선하려는 사람을 가리키는 용어다.

위의 내용은 필자가 일을 하면서 그 동안 만나왔던 사람들의 특성과 기법을 정리한 것이다. 본 목록을 정리하는 데 도움을 준 에픽 게임스, 체어(ChAir), 피플캔플라이(People Can Fly)의 동료들에게 감사의 말을 전한다. 앞으로 게임 개발자가 되고 싶은 사람이 뭔가 배울 것이 있었으면 좋겠고, 기존 개발자들은 킬킬거리며 위의 글을 읽었으면 한다.

그림: 후안 라미레즈(Juan Ramirez)[3]

[3] 참조링크: http://www.buttermonster.com/