POMMS, 플레이어가 게임을 테스트하게 하는 법

(POMMS: A Way to Get Your Players to Test Your Game!)

작성자: 로버트 호이셴(Robert Hoischen)

작성일: 2013년 1월 17일

거대한 인디 게임 <마인크래프트(Minecraft)>는 보통 게이머가 실제로 생각하는 "베타 버전"에 대한 인식을 바꾸어놓았다. "베타"라는 말은 더 이상 버그가 많고 실망스럽고 심하게 망가진 데다 전혀 재미없는 것을 의미하지 않는다. 지금은 상당히 플레이 가능하고 상당히 세련된, 그러나 불완전할 수 있는 초기의 개발 마일스톤[1]를 의미한다. "그렇다면 누가 실제로 이 마일스톤들의 베타 테스트를 하는가?"라고 물을 것이다. "<마인크래프트>를 베타 테스트" 했기 때문에 베타 테스터의 자격이 있다고 말하며 인디 게임 개발자 포럼에 오는 사람들은 아마 아닐 것이다.

먼저 명확하게 정의를 내려보자. 베타 버전[2]이란 기능은 완성되었지만 아직 버그가 고쳐지거나 세련화 작업을 거치지 않은, 개발 작업이 진행 중인 버전을 말한다. 게임의 작은 기능 하나일 수도 있고 전체 소프트웨어일 수도 있다. 이런 정의는 보편적인 것은 아니지만, 앞에서 이미 사례를 들었고, 이 글에서 보게 될 내용은 거부할 수 없는 명백한 사실이다. 노치(Notch)의 훌륭한 작품은 이 정의에 따라 매우 잘 다듬어진 중간결과물 이상도 이하도 아닌 상태로 출시되었는데, 스스로 "베타 테스터"라고 선언한 버전의 출시에 앞서 내부에서 알파와 베타 테스트를 한 것 같다.

---

[1] 참조링크: http://en.wikipedia.org/wiki/Minecraft#Development

[2] 참조링크: http://en.wikipedia.org/wiki/Software_release_life_cycle#Beta

최근 이런 변화들은 앞으로의 게임 제작에서 베타 테스트의 필요성에 심정적으로 어떤 영향을 끼쳤는가? 이 질문에 대한 답변은 타겟 이용자에 따라 크게 달라지지만, 일반적으로는 자유롭게 이용할 수 있는 일손들이 점점 "이것저것 해보는" 부류로 옮겨가고 있다.

블리자드 엔터테인먼트는 이런 사람들을 완벽하게 이용하는 방법[3]을 개발했지만, 이들은 소규모나 중간 규모의 테스트 팀에 잠재적인 위험 요소가 될 수 있었다. 유닛 시간과 테스터 당 (고급) 작업을 얼마나 해낼 수 있을지를 과대평가한 것은 즉각 실패로 이어졌고, 결국 신뢰를 잃고 궁극적으로는 판매를 저하시키는 결과를 가져오는 지연으로 이어졌다.

이 문제를 아무것도 계획하지 않거나 모든 것을 자신들이 직접 테스트함으로써 해결하는 인디 개발팀이 많은 것 같다. 실행 가능하지만 최선의 해결책과는 거리가 먼 방법이다.

이런 무지는 그리 놀랍지도 않다. 적절한 테스트 기반을 마련하고 테스트에 적절한 사람들을 찾고 끊임없이 변하는 테스트의 목표를 전달하는 데는 손이 많이 간다. 그게 바로 이 글에서 말하려는 POMMS 기반의 베타 테스트가 나온 이유다.

내가 개발한 POMMS 는 프로젝트 지향의 모듈 동기부여 시스템(Project-Oriented Modular Motivational System)의 약자로, 현재 <오토메이션: 자동차 회사 타이쿤(Automation: The Car Company Tycoon Game)>[4]을 개발하고 있는 소규모 인디 게임 개발 스튜디오 캠샤프트 소프트웨어(Camshaft Software)[5] 에서 베타 테스트라고 불리던 혼돈에 구조와 명확성을 부여하기 위한 도구다. 이 시스템은 테스트 과정 자체에 현대 게임 기획의 요소들, 즉 보통 "게임화 (gamification)"[6]라는 전문 용어와 관련된 개념을 구현한다. 테스트에 대한 POMMS 기반의 접근은 베타 테스트의 어려운 부분들, 즉 미리 계획하는 능력, 테스터 조정, 테스터 동기부여, 테스트의 초점, 테스터의 유연성과 개성 같은 것들을 공격한다.

---

[3] 참조링크: http://www.tomshardware.com/news/Diablo-3-Concurrent-Action-RPG-Beta-Weekend-Blizzard,15440.html

[4] 참조링크: http://automationgame.com

[5] 참조링크: http://camshaftsoftware.com

[6] 참조링크: http://lithosphere.lithium.com/t5/science-of-social-blog/The-Magic-Potion-of-Game-Dynamics/ba-p/19260

POMMS 란 무엇이며 어떻게 작동하는가

이 시스템 자체는 금방 설명할 수 있다. 잘 정의된 하위 프로젝트들을 설명하고, 테스터가 하위 프로젝트의 목표를 실현한 각각의 업무(task)에 일정량의 "점수"가 부여된다. 이 점수가 이른바 테스터의 전투력(power level)[7]인데, 이것이 9,000 점 이상이면 하위 프로젝트들이 너무 길거나 점수당 작업량이 너무 적은 것이다.

하위 프로젝트의 내용에 따라 1 점당 대략 15 분 정도의 작업으로, 중요한 하위 프로젝트는 4 주에서 16 주가 걸릴 수 있다. 주어진 하위 프로젝트에서 각각의 테스터는 테스트에 계속 참여하기 위해 일정한 최고 점수에 도달해야 한다. 최고 점수를 획득한 테스터들은 하위 프로젝트가 진행되면 누적되는 별을 받는다. 별을 획득한 테스터에게 어떤 보상이 돌아갈지는 개발자에게 달려 있다. 예를 들어 개발자는 테스터에게 소프트웨어의 크레딧에 이름을 넣는 특권을 줄 수도 있고, 그 밖에도 특별한 포럼 등급을 부여할 수도 있다.

---

[7] 참조링크: http://knowyourmeme.com/memes/its-over-9000

테스터가 최고 점수는 획득했지만 별 순위 안에 들지 못한 경우 최종 점수는 다음 하위 프로젝트로 이월된다. 이것은 다음 하위 프로젝트에서 최고 점수에 도달하는 것이 아니라 별 순위 안에 드는 것을 유리하게 해준다. 그리하여 테스터가 결국 별을 받을 수 있게 되기 때문에, 하위 프로젝트가 진행되는 동안 작업을 꾸준하게 잘한 테스터에게 보상을 주며, 프로젝트가 끝난 후 테스터가 자신이 이름 없는 테스트 기계였다고 느끼지 않게 해준다.

각각의 하위 프로젝트는 소프트웨어의 특정한 측면에 초점을 맞추되, 활동을 너무 축소시키지 않도록 한다. 좋은 사례는 <오토메이션>에서 엔진 디자이너에 V8 엔진을 추가한 것이다. 이 하위 프로젝트는 8 주 안에 2D 와 3D 아트워크를 체크하고, 시나리오의 플레이를 테스트하고 균형을 맞추고, 실제 엔진의 자료 수집하고, 버그 추적 도구의 항목들을 다듬고, 게임의 모든 새로운 조건들을 체크하고, 마지막으로 가장 중요한 오래된 버그를 잘 보고하는 것을 아우른다.

각각의 하위 프로젝트마다 초반에 점수표가 정해진다. 모든 테스터는 이 점수표에서 베타 포럼의 "영웅의 전당" 스레드에 오르는데, 이 기록은 작업이 진행되는 동안 각 테스터가 꾸준히 업데이트한다. 다음은 위에서 언급한 V8 엔진 프로젝트의 점수표 사례다.

테스터 ID: PaLaDiN1337 (4*)
테스터 상태: 활동 중
현재 점수: 17 (+0)
시나리오 테스트: (V0.88) S1, S2, S4, S5, S7, S8, S11
트래커 항목: ID28804129, ID28796925, ID28786633
엔진 정보: 포드 모듈러 4.6L V8 3-밸브
테스트한 엔진: 포드 모듈러 5.4L "트리튼" V8 DOHC
포럼 장점: 교정 보조(2 점)
성과: 버그 트래커 정리(1 점) 3 회

점수표를 단순하게 유지하는 것이 가장 중요하다. 시나리오를 테스트하면 점수를 받는다. 엔진 정보를 조사하면 점수를 받는다. 버그 리포트를 기록하면 점수를 받는다. 이 시스템은 관리와 사용, 이해가 쉬워야 하고, 진행사항이 잘 보여야 한다.

각각의 하위 프로젝트가 끝날 때 영웅의 전당 표는 고정되고, 점수는 모든 테스터의 현재 상태, 파워 레벨, 이월 점수를 기록한 스프레드시트로 옮겨진다. 각각의 하위 프로젝트에 대해 이 모든 것이 타임라인으로 기록된다.

POMMS 베타 관리 예제					하위 프로젝트 1				하위 프로젝트 2				
타입	포럼ID	상태	제거	별	상태	경고	진행PL	End PL	Status	Warnings	Carry PL	End PL	Status
개발	Dev 1	Active	No										
	Dev 2	Active	No										
	Dev 3	Active	No										
베타 테스터	Tester 01	Active	Yes	1	Active		(+0)	72 (+0)	Active	1x shoddy	(+0)	41 (+0)	Active
	Tester 02	Passive	Yes	1	Active	1x shoddy	(+0)	41 (+0)	Active		(+20)	33 (+20)	Passive
	Tester 03	Retired	Removed	1	Active		(+0)	63 (+0)	Retired		(+0)	0 (+0)	Retired
	Tester 04	Active	Yes	1	Active		(+0)	38 (+0)	Active		(+19)	29 (+19)	Active
	Tester 05	Banned	Remove		Active	1x shoddy	(+0)	23 (+0)	Active	NDA violation	(+11)	0 (+9)	Banned
	Tester 06	Passive	Yes	1	Active		(+0)	47 (+0)	Active		(+0)	25 (+0)	Passive
	Tester 07	Passive	Remove		Active		(+0)	33 (+0)	Active	Failed Active	(+16)	10 (+16)	Passive
	Tester 08	Active	Yes		Active		(+0)	45 (+0)	Passive		(+22)	0 (+22)	Active
	Tester 09	Active	Yes	1					Active		(+0)	44 (+0)	Active
언론	Press 01	Passive	No										
	Press 02	Passive	No										

원본 이미지 링크 :

http://gamasutra.com/db_area/images/feature/184703/image2.jpg

모든 활동이 사전에 잘 계획되거나 준비되는 것은 아니고, 개발이 진행되는 동안 갑자기 발생하는 업무가 있을 수도 있다. 이 때문에 퀘스트 보드(Quest Board)가 있다. 퀘스트보드는 하위 프로젝트에 특화된 것으로 영웅의 전당 스레드에 꾸준히 업데이트되는 부분이다. 퀘스트 보드는 필요할 경우 테스트 초점을 빠르게 전환해주고 테스트 시스템을 역동적으로 만들어준다. 여기에는 테스터들이 완수했을 때 성과와 점수를 얻게 되는 비격식 업무들이 나와 있다. 이 업무는 "이 스프레드시트를 정리하라"나 "이 주제에 대한 정보 몇 가지를 찾아라" 또는 심지어 "괜찮은 테스터를 더 찾아라" 같은 것일 수도 있다.

대체로 테스터들에게는 사생활이 있으므로(가끔은 그런 척이라도 하는데) 테스트를 쉬거나 그만둬야 할 수도 있다. 이것을 수용하기 위해 테스터를 쉬게 해주되, 작업을 시작하지 않거나 개발 진행 알림을 더 이상 받지 않기로 한 테스터는 즉시 제적시키는 수동적인 규칙을 만들었다.

"삼등분 법칙(Rule of Thirds)"은 사진에서만이 아니라 이곳에서도 빛을 발한다. 팀에 새로 들어오거나 비활성 상태에서 시작하거나, 활성 상태에서 실패한 테스터는 하위 프로젝트 진행 초반 1/3 기간 내에 최소점수의 1/3 이상은 달성해야 제적당하지 않을 수 있다. 이 시스템은 관리도 간단하다. 스프레드시트에서 플래그

하나를 바꾸고, 논란이 되는 모든 후보자들의 해당하는 하위 프로젝트 1/3 점수표를 체크한다.

POMMS 가 개발자들에게 갖는 주요 이점은 다음과 같다.

가이드: 평균적인 테스트 완수분량을 측정하고 계획할 수 있다.
보상: 개발자와 테스터 모두에게 긍정적인 작업 환경을 만들어준다.
유동성: 새로운 업무의 전달과 생성이 간단하고 효율적이다.
관리: 관리하기 쉽고 사용자의 참여가 스스로 제어된다.

POMMS 가 테스터들에게 주는 주요 이점은 다음과 같다.

가이드: 다음에 무엇을 해야 하는지 추측할 필요가 없고 선택지만 있다.
보상: 작업의 진행과 팀에서 테스터의 가치에 대한 척도가 명확하다.
유동성: 테스터의 취향에 맞는 업무가 항상 있다.
관리: 이해하기 쉽고 점수표를 유지하기 쉽다.

잠재적인 문제점과 확장성

POMMS 가 가진 잠재적인 문제점은 앞에서 언급했듯이 테스터가 질이 아니라 양으로만 보상받는다는 점이다. 이 문제는 점수로 계산의 최소 요구치를 매우 높게 설정하여 해결할 수 있다. 예를 들어 버그 리포트는 버그 추적 도구에 정리하고, 적절한 제목, 의미있는 태그, 버그를 발생시키는 상황에 대한 상세한 묘사를 적어야 한다는 식이다.

테스터가 완수한 작업의 퀄리티는 간접적으로만 통제한다. 조잡한 작업은 언제가 됐든 겉으로 티가 날 것이다. 이 시스템은 또 테스터와 개발자 사이의 신뢰를 강화하는데, 직접적인 통제로 테스터들이 자신이 발견한 것을 기록하는 대신 실수를 두려워하기 시작하면 시스템만 손상될 뿐이다. 우리는 모두 실수를 한다. 중요한 것은 실수의 밀도지 절대 수치가 아니다.또 다른 문제는 시스템의 확장성이다. 테스트 시스템의 전환은 어렵고 지루한 일이다. 일단 POMMS 같은 시스템을 마련하면 이 시스템이 곧 쓸모 없어지지 않으리라는 확신이 있어야 한다.

지금까지 우리가 구현한 POMMS 는 소수에서 50 명에 이르는 테스터에게서 매우 양호하고 거의 대수적인(logarithmic) 확장성을 보여주었다. 테스터 100 명을 관리하는 것은 10 명을 관리할 때보다 일이 2 배가 된다. 단, 100 명 이상은 자동화된 도구 없으면 쉽게 관리하기는 어렵긴 하다.

시스템 공정성과 작동

많은 테스터들이 같은 점수를 두고 경쟁하다 보면 경쟁이 매우 치열해질 수도 있다. 특히 항상 선착순으로 하게 되는 관심이 집중된 버그 리포트의 경우 그렇다. 얼핏 보면 작업 진행이 빨라져서 개발자들에게 좋을 것 같지만, 결국은 테스트 환경의 전반적인 질을 해칠 수도 있다. 이 문제는 경쟁심이 약한 테스터들에게 퀘스트 보드를 통해 다른 테스트 활동을 보상해줌으로써 대개 막을 수 있다.

한 가지 놓치기 쉬운 점은 시스템 공정성이 매우 중요하다는 것이다. 새롭게 구현한 POMMS 베타 테스트는 기본적으로 MMO 게임처럼 움직이며 여기서 테스터는 플레이어가 된다. 아무리 사소한 불균형과 불공정함이라도 만 배 규모로 커져 마치 테스터의 인생을 파괴하기라도 한 듯이 개발자들에게 돌아온다. 가끔은 평균적인 MMO 게임 포럼과 무섭도록 비슷하다.

가장 중요한 것은 점수 시스템과 보고 시스템을 110 퍼센트 공정하게 만드는 것이다. 규칙에 예외를 만들지 말라. 예외를 두면 모두가 예외를 요구할 것이기 때문이다. 테스터들은 처음에는 개발자 편에 있을 것이다. 그러나 시스템의 준비와 관리에 크게 주의를 기울여 처리하지 않는다면 상황이 빠르게 변할 수 있다. 규칙이나 구조의 변화에 항상 투명성을 기하고 이유와 때를 설명하라. 테스터들이 해낸 작업에 감사를 표하고 양쪽 모두에게서 일이 잘못될 수도 있음을 받아들여라. 테스터 각각에 대한 개발자들의 직접적이고 개인적인 인정과 감사는 아무리 사소하더라도 동기 부여와 테스트 분위기에 놀라운 작용을 한다.

우리가 처음 POMMS 를 구현했던 6 개월 동안은 매우 긍정적인 결론이 도출되었다. 약 90 퍼센트의 테스터가 POMMS 구조 안에서 작업하는 것을 좋아했고, 10 퍼센트는 주로 이 시스템의 경쟁적인 성격에 불만을 표하고 떠났다. 이것은 아마도 경쟁심이 강한 테스터들과 그렇지 않은 테스터들 모두에게 공정하게 규칙이

적용되기 전이었던 준비 단계에서 저질렀던 실수에서 비롯한 불행한 결과인 것 같다.

현재 테스트 팀이 완수한 작업의 양을 우리가 체계적이지 않은 베타 버전을 실행하는 동안 완수했던 작업의 양에 비교해보면 생산성이 10 배 증가했다. 그닥 목소리 높이는 일 없이 작업을 컨트롤할 수 있고 가끔 경쟁적이긴 하지만 전반적인 분위기는 긍정적이다.

요약과 결론

요약하자면, 이 글에서 설명한 테스트 시스템은 문제점이 거의 없고, 소규모와 중간 규모의 소프트웨어 제작에 엄청난 이익을 준다. POMMS 는 한번 설정되면 인간의 힘이 별로 들어가지 않으며, 처음 구현했을 때 이미 매우 효율적임이 증명되었다.