HTML5 가 게임 제작 방식을 바꿀 것인가?

(Will HTML5 Change the Way Games are Made?)

작성자: 윌 이스트콧(Will Eastcott)

작성일: 2012 년 8 월 29 일

Will Eastcott 는 EA, Sony, Activision 에서 초거대 타이틀을 만들어 냈던 비디오 게임 테크놀로지스트이자, 클라우드 기반 HTML5 게임 개발과 퍼블리싱 플랫폼을 제공하는 PlayCanvas 의 공동 창립자다. 그는 이 글을 통해 급부상하고 있는 HTML5 기술이 게임 개발자들에게 실제로 어떤 영향을 미칠 지에 대한 의견을 공유하고, 게임 개발의 진일보가 이 언어를 통해 이루어지는 사례를 만들고자 한다.

아마 이런 기사 제목을 본 적이 있을 것이다. "콘솔 게임은 죽었다!" "HTML5 가 미래다!"

HTML5 를 둘러싼 논의는 너무 과장된 경우가 많고, 감히 말하건대 편견으로 가득 차 있다. 이런 것들은 일단 뒤로 제쳐놓고, 조금 객관적인 이야기를 하도록 하자. 내 의견을 조심스럽게 제시해 보자면, HTML5 는 비디오 게임 제작 방식을 근본적으로 바꿀 것이다. 그 이유를 설명하기에 앞서 현재 HTML5 의 상황을 요약해보자.

우리가 지난 30 년 이상 보아온 것은 하드웨어 플랫폼이 점점 더 강력해지는 일반적인 트렌드와 그 하드웨어를 전략적으로 더 잘 이용하기 위한 툴과 언어의 발전이었다. 현실을 뛰어넘는 게임 환경에 대한 욕구는 수그러들지 않았고, 콘솔 게임 제작사들은 개발자들의 손에 더욱 강력한 기술을 쥐어주었다.

그러나 모바일 게임이 부상하면서 우리는 새로운 현상을 목격하게 되었다. 게이머들이 어디에서나, 어느 기기에서나, 또 누구든지 함께 플레이할 수 있는 더 단순한 게임에 예외적으로 반응하기 시작한 것이다. 이것이 현재의 유저들이 요구하는 핵심이다. 기술은 여전히 중요하고, 또 잘 만든 그래픽도 여전히 염두에 두어야 한다. 하지만 이러한 게임에서 더욱 중요한 것은 접근성(accessible)과 연결성(connected)이다. 웹 브라우저보다 더 접근성과 연결성이 뛰어난 것이 무엇이란 말인가?

그러면, 이런 게임을 만드는 데 어떤 옵션이 있는가? 그냥 맨 처음부터 만들 수도 있다. 하지만 한 개 이상의 플랫폼을 목표로 설정하는 것은 작은 개발팀에게 무리일 수도 있다. 이럴 때 플래시를 이용할 수 있다. 조금씩 조금씩 매력도가 감소하고 있긴 하지만 말이다. 플래시의 모바일을 지원하지 않기로 한 일과 Stage3D 라이센싱 비용을 둘러싼 명쾌하지 않은 입장은 개발자들에게 좋지 않은 인상을 심어주었다.

Havok Vision Engine 과 Unity 와 같은 엔진을 예로 들어보겠다. 이러한 엔진들은 보통 데스크탑 브라우저는 자사 플러그인으로, 모바일 기기는 고유 실행파일로 지원하도록 되어 있다. 그리고 HTML5 가 있다. HTML5 를 선택하기에 앞서, 이것이 가진 장점과, 약점을 완화시킬 방법을 숙지하는 것이 매우 중요하다. 더 깊숙히 들어가보자.

진화하는 표준

HTML5 는 완성된 것이 아니다. 표준화 위원회인 W3C and WHATWG 에 의해 아직도 개발이 진행되고 있다. 즉 브라우저 판매사는 움직이는 타겟을 쫓고 있는 것이다. 그러므로 HTML5 를 지원하는 수준이 모든 브라우저마다 다를 수밖에 없다. 여기에 대해서는 caniuse.com 라는

사이트에서 자세히 설명하고 있다.

브라우저간 불일치의 가장 대표적인 예가 바로 오디오다. 현재 세 가지 API 가 있는데, Mozilla 의 오디오 데이터 API, Google 의 웹 오디오 API, JavaScript API 의 Audio 엘리먼트가 그것이다. 최근에 Mozilla 는 자사의 API 를 곧 도태시키고, 웹 오디오 개발을 시작하겠다고 선언했다. 즉, 브라우저 판매사들이 개발자의 편의를 위해 일제히 혁신에 나서고 HTML5 플랫폼으로 수렴하게 될 것이라는 점은 자명하다.

특히 게임 개발자들에게 새로운 HTML5 API 를 제공해야 한다는 브라우저 제작사의 공통된 요구가 있다. 중요한 예로 GamePad API (플랫폼에서 지원하는 모든 종류의 게임패드에서 입력을 받을 수 있음), Pointer Lock API (마우스 커서를 숨기고 마우스 동작을 직접 읽음), Fullscreen API (모든 HTML 요소를 전체화면으로 표시)의 세 가지를 들 수 있다.

이들 API 는 짧은 기간 동안 일관성을 가지도록 사양이 정해졌고 많은 브라우저에 구현되었다. 이들을 도입함으로써 1 인칭 슈팅과 같은 류의 게임이 갑자기 HTML5 에서 훨씬 쉽게 돌아가기 시작했다.

자바 스크립트

개발자들은 프로그래밍 언어에 대해 상당히 강한 향수를 품고 있다. 콘솔 백그라운드가 있는 개발자들은 자바스크립트를 의심스럽게 보는 경향이 있다. 자바 스크립트는 새롭게 배워야 하는 언어이면서도 같은 내용의 C++ 코드에 비해서 느리고, 벡터화(vectorization) 같은 최적화 전략도 사용할 수 없다. 게다가 게임 코드는 브라우저의 개발자 툴에서 모두 볼 수 있다.

더글라스 크락포드(Douglas Crockford)의 자바스크립트가 <세상에서 가장 오해받는 언어(The World's Most Misunderstood Programming Language)>[1]라는 주장을 참고해볼 필요가 있다. C++ 프로그래머에게 자바스크립트 배우는 건 사실 식은 죽 먹기다. 전체적으로 C++과

[1] 참조링크: http://www.crockford.com/javascript/javascript.html

밀접하게 관련이 있다. 클로저(closure) 같은 컨셉이나 'this' 키워드에는 익숙해져야 할 테지만 말이다.

C++같은 언어의 경직성이나 장황함을 좋아하지 않는 사람들에게 자바스크립트는 환상적이고 다이내믹한 언어다. 1 급함수(first class function)같은 강력한 기능들이 작성해야 하는 코드와 작성시간을 크게 줄여준다. 자바스크립트를 일단 쓰기 시작한 사람들은 다시 돌아가지 않는 것 같다.

퍼포먼스도 생각하는 것보다 좋다. 우선 Web Workers API 로 자바스크립트에서 멀티스레드 프로그래밍이 가능하다. 더 중요한 사실은 자바스크립트 엔진이 지난 몇 년간 크게 발전했다는 것이다. 오픈소스 물리 엔진인 Bullet 에 이미 자바스크립트 포트가 있으며, 자바스크립트를 사용한 매우 CPU 집약적(intensive)인 알고리즘이 –비록 퍼포먼스가 눈부시지는 않아도- 제법 쓸만한 속도로 실행되는 걸 보면 알 수 있다.

톡 까놓고 얘기해보자 - 베끼기는 어느 플랫폼에서나 존재한다. 하지만 HTML5 게임에 관해서라면 몇 가지를 염두에 두어야 할 것이 있다. 구글의 Closure 컴파일러 같은 툴은 소스를 최적화하는데 뛰어난 성과를 보여준다. 애매모호한 소스코드를 분석하는 것은 원래의 코드에 접근하는 것과는 다르다. 코드 베이스의 사이즈가 확장되었기 때문에, 역설계(reverse engineer)하는 것이 점점 힘들어지고 있는 것이다. 두 번째로 어떤 게임에 온라인 콤포넌트가 있다면 많은 코드를 서버에서 실행시킬 수 있고, 클라이언트는 시각화만 담당하면 된다. 많은 "가치"를 서버 안에 숨길 수 있는 것이다.

2D 대 3D

게임을 기획할 때에 가장 기본적인 선택 중 하나가 그래픽을 2D 로 할 것이냐 3D 로 할 것이냐 하는 것이다. 2D HTML5 게임은 일반적으로 페이지의 캔버스 요소에서 쿼리된 2D 컨텍스트를 사용한다. 유명 브라우저들은 이제 다들 2D 캔버스 컨텍스트를 잘 지원하고 있으므로 만일 스프라이트(sprite)를 쓰는 게임을 만들고 있다면, 걱정을 할 필요가 없게 되었다.

3D 게임을 개발하려고 한다면 고려할 것이 더 많다. HTML5 게임에서 하드웨어 3D 가속을 받는 고품질의 그래픽을 구현하는 유일한 방법은 OpenGL ES 2.0 과 거의 동일한 자바스크립트 인터페이스인 WebGL 이다. 데스크탑에서는 크롬(Chrome)과 파이어폭스(Firefox)에서 WebGL 이 강력히 지원되고 있으며 인터넷 익스플로러(Internet Explorer)에서만 지지부진하다. 다행스럽게도 구글 크롬의 Frame 과 같은 플러그인을 통해 WebGL 에 대한 마이크로소프트의 저항을 완화할 수 있다. 또한 애플이 사파리(Safari)에서 WebGL 을 기본적으로는 활성화시켜 놓지 않지만 개발자들에게는 열려 있다.

WebGL 은 2011 년 3 월에 1.0 규격이 발표된 후로 기나긴 길을 걸어왔다. OpenGL ES 3.0 이 SIGGRAPH 에서 공개되면서, 이를 반영한 WebGL 2.0 규격 또한 오래지 않아 발표될 것이다. 단언컨데, 3D 브라우저 게임 분야에 고무적인 시기이다.

모바일 지원

HTML5 게임이 데스크탑에선 매우 잘 실행되지만, 이것을 모바일로 이식하는 것은 쉬운 일이 아니다. 데스크탑 용 HTML5 게임을 손쉽게 모바일 플랫폼으로 이식할 수 있다는 주장은 대개는 과장이다. 모바일 이식을 위해선 (자바 스크립트 실행이 느릴 수 밖에 없는) 저성능 CPU 와 (3D 게임에서 버텍스/픽셀 셰이더의 처리량이 적을 수 밖에 없는) 저성능 GPU 에 대한 준비가 필요하다.

스크린 해상도도 매우 다양하고, HTML5 로 게임을 개발한다고 해서 데스크탑과 모바일 양 쪽에서 모두 잘 작동하는 마성의 조작법(control)이 생길 리도 없다. 6 개월 전과 많이 달라지기는 했지만 현재 모바일 플랫폼을 지원하는 WebGL 는 아직 발아 단계에 있다.

그 사이에 살펴볼 만한 옵션이 몇 가지 있다. PhoneGap, AppMobi 나 CocoonJS 와 같은 기술은 HTML 5 게임을 네이티브 코드로 실행하는 래퍼(wrapper)를 통해 더 나은 성능을 보여준다. 아직까지는 이러한 기술들이 작동 가능한 WebGL 을 지원하지 않는다는 한계가 있지만, 몇몇 미들웨어 개발자들은 이미 솔루션 개발에 착수한 것으로 보인다.

개발의 용이성

본질적으로, HTML5 게임 개발은 매우 쉽다. 브라우저는 놀랍도록 탄탄한 개발 플랫폼을 지향한다. 특히 몇몇 하드웨어 개발사들이 제공하는 툴과 비교해 보자면 그렇다.

툴 자체는 폭넓고도 제약 없이 브라우저 자체에 통합되어 있으며 코딩 후 컴파일 단계가 없다. 자바스크립트의 동적인 특성은 강력한 수정후-바로실행(edit-and-continue) 형식의 개발을 지원하며 기존의 콘솔도 매우 유용하다. 웹 개발에 관한 모든 것은 빠른 반복시간과 단순성에 초점이 맞춰져 있다.

API 는 군더더기 없이 깔끔하며, 거대한 웹 개발자 커뮤니티의 덕에 방대한 양의 샘플 코드와 동영상 튜토리얼 또한 쉽게 접할 수 있다.

접근성

컴퓨팅에 있어 보편적으로 받아들여지는 하나의 패러다임이 있다면, 브라우저 안에서 웹 주소로 이동하여 그 자리에 존재하는 컨텐츠를 소비하는 방식일 것이다. 어쨌든 게이머들은 인내심이 부족한 종족들이다 (성급한 일반화에 양해를..). 다운로드 받고 설치를 하는, 첫 번째 문턱에서 많은 수의 고객이 떨어져나간다. 또 모든 사람이 게임을 실행하기 위해 특정한 브라우저의 플러그인을 기꺼이 설치하려고 하지도 않고, 할 수 있는 것도 아니다.

HTML5 게임은 게임을 실행하기 위해 어떤 것도 설치하거나 허가를 받도록 요구하지 않는다. 몇몇 사람이 믿고 있는 것과는 반대로, 한 번 리소스를 다운 받은 다음 장치에 영구적으로 캐시로 저장할 수도 있다. 게임에 인터넷 연결이 필요하지 않다면 오프라인 플레이도 분명히 가능하다.

즉 게임이 무척 빠르게 시작되고 페이지 로딩이 매끄럽다면 그만큼 레벨 1 로 기꺼이 진입할 유저를 많이 확보할 수 있다는 뜻이다.

고객 확보

Google I/O 2012 에 따르면, 구글 크롬은 3 억 1 천만 명의 실제 사용자를 확보하였는데 이 숫자는 현재 전 세계에 출시된 현 세대 게임 콘솔의 숫자를 훨씬 웃도는 것이다. HTML5 를 지원하는 브라우저를 고려한다는 것은 잠재적으로 광대한 고객을 바라보고 있다는 뜻이다.

게임이 URL 주소 안에 들어있고, 쉽게 찾을 수 있다고 해서 사람들이 이 게임을 찾아낸다는 뜻은 아니다. 발견해내는 것 또한 중요한 역할을 한다. 앱 스토어는 좁디 좁은 "진열장"으로 악명이 높고, 또 게임은 수 많은 유사 게임들 사이에 쉽게 파묻히곤 한다. HTML5 게임 또한 크게 다르지 않다. 하지만 이러한 어플리케이션보다 유리한 고지를 차지할 수 있는 열쇠가 있다.

HTML5 게임은 기본적으로 웹페이지이기 때문에, 검색 엔진에서 구조적으로 인덱싱할 수 있다. 또, 게임을 널리 알릴 수 있는 크롬 웹 스토어를 포함해 다수의 게임 포탈과 스토어를 이용할 수도 있다. 배포 방법이 웹으로만 제약되는 것도 아니다. 한 예로, Mac 앱스토어에서 HTML5 게임을 판매할 수도 있다. 임베디드 WebView 요소를 이용한 네이티브 어플리케이션의 형태로 패키징해서 말이다.

가능한 많은 유저들에게 도달하는 것이 중요한데, HTML5 는 어떤 하나의 경로가 아닌 다양한 유통망을 조합할 수 있게 해 준다.

주목 받기

좋든 싫든, 게임을 만든다는 것은 최첨단의 벼랑 끝에 서 있다는 말이다. "핫"한 테크놀로지는 커버리지가 더 나을 수밖에 없다. 만약 당신이 똑같은 두 게임 코드를 C++과 HTML5 로 짠다고 할 때 어느 것이 미디어의 더 큰 주목을 받게 될지는 자명하다. 반드시 공평하다고는 할 수 없지만, 어쨌든 이것이 현재 HTML5 의 위상이라고 할 수 있다 - 사람들은 HTML5 로 무엇을 할 수 있는지 보고 싶어 하고, 개척자가 될 기회는 얼마든지 많이 있다.

HTML5 로 게임 개발을 다시 생각하다

이제 여러분은 만들고자 하는 게임을 HTML5 로 만들 수 있을지 더 잘 판단 할 수 있게 되었기를 바란다. 하지만 이 지점이 바로 논의가 충분치 못한 부분인 것 같다. HTML5 는 단순히 비디오 게임의 구동을 위한 표준들의 집합이 아니며, 그 한가지 기준에 의해서 평가되어서도 안 된다.

HTML5 는 우리의 디지털 라이프를 클라우드로 이행하는 데에 연료가 될 차세대 생산성 앱의 동력이 된다. 이 트렌드는 우리가 비디오 게임을 만들 때 쓰는 툴과 워크플로우를 흔들어놓을지도 모르는 놀라운 잠재력이 있으며, 아직 우리는 수박 겉핥기조차 하지 못하고 있을 뿐이다.

클라우드로 가는 창

오늘날 우리는 클라우드 앱을 빼놓고 아무 것도 생각할 수 없다. 웹 메일은 가장 대중적인 예인데, 한때는 원격 서버에 개인적인 메일을 저장하는 것에 일말에 걱정이 있었을지 모르지만 지금은 훨씬 마음을 놓을 수 있게 되었다. 현재는 생산성에 도움을 주는 앱이 클라우드 앱의 형태로 존재한다. 워드 프로세서, 이미지 편집기, 코드 편집기 등 말이다.

클라우드는 게임이라는 맥락에 있어 다양한 이점을 가져다 주었다. 웹 앱이 서버에서 브라우저(혹은 브라우저의 캐시)로 전송되기 때문에, 아무 것도 설치를 할 필요가 없다. 툴은 언제나 깔끔하게 업데이트되어 유저가 언제나 서로 싱크되어있게 해주고, 늘 최신 릴리즈를 접할 수 있게 해 준다. 유저 데이터는 늘 안전하게 백업되고 또 쉽게 복구할 수 있다. 유저의 OS 나 브라우저가 충돌할 수도 있지만, 데이터는 이미 서버에 안전하게 저장되어 있을 것이다.

이상의 모두가 개발을 할 때 진정 유용한 특성이다. 유저에게 클라우드 기반 시스템의 창구를 제공하고자 한다면 브라우저가 필요할 것이다. 그렇다면 브라우저의 인터페이스를 만드는데 가장 좋은 방법은? 내 생각엔 HTML5 이다. 플래시나 자바, 게임을 실행하기 위한 사용자 지정 플러그인을 사용하는 것도 물론 가능하지만, 복잡한 웹 앱을 만들 때에는 HTML5 이 가장 합리적인 솔루션이다. ExtJS, Google Closure 나 SproutCore 를 비롯해 사용이 간편한 많은 웹 앱 프레임워크가 지속적으로 진화하고 있기 때문에 HTML 을 사용하는 사례가 점점 강력해지고 있다.

협업의 새로운 방식

몇 년 간 많은 게임의 에디터로 일해오면서, 나는 대부분이 몇몇 버전 컨트롤을 통합하여 협업을 지원해 왔음을 알 수 있었다. 하나의 게임은 우리가 "레벨"이라고도 부르는 다수의 문서로 만들어진다. 하나의 레벨은 XML 의 형태로 지속될 수도 있고, 다른 형식을 띨 수도 있다. 만일 두 명 이상이 동시에 이 데이터 파일을 수정한다면 몇 가지 사항을 염두에 두어야한다. 첫째, A 는 B 가 바꾸는 것을 알아차리지 못할 수도 있고, 혹은 그 반대일 수도 있다. 둘째, A 가 변경사항을 반영할 때, B 는 A 가 만든 것을 자신의 것과 통합해야 한다.

툴의 작동을 배타적 수정권한 방식으로 제한하는 것도 가능하지만, 그렇게 되면 그저 한 유저가 다른 유저를 가로막고 개발을 지연시키는 것에 불과하다. 자동화가 가능하다면 코드를 병합하는 것도 좋겠지만, 이게 언제나 가능하지는 않다. 시각적으로 코드를 합치는 것은 코더가 해오던 일이지만, 기획자에게 어떻게 직관을 이용해 두 개의 레벨 데이터를 일일이 합치라 할 수 있단 말인가? 불가능하지는 않더도, 잘 해 내기란 무척 어려운 일일 것이다.

그러므로 다른 접근방법을 택해보자. 우리의 개발 툴이 클라우드에서 실행되고, 모든 클라이언트가 서버에 연결되어 있다면, 유저에게 편집을 허용하지 않을 이유가 뭐가 있겠는가? 다시 한번, HTML5 는 여기에 필요한 것을 모두 제공한다.

WebSockets (TCP 상의 양방향 통신)을 사용하며, 우리는 클라이언트 간에 편집 내용을 전송할 수 있게 되었고, 이를 통해 실시간 협업이 가능해졌다. 일단 한 번 이 방법으로 작업을 해 보고

이 방식의 효용을 체험해 본다면, 홀로 작업하는 환경의 "배타적 수정권한"으로는 절대로 돌아가고 싶지 않을 것이다.

소셜 게임 개발

개발자라면 아마 GitHub 이나 BitBucket 의 계정을 가지고 있을 것이다. 최근 몇 년 사이, 우리들 대부분은 당연히 Subversion 나 Perforce 와 같은 유명한 버전 컨트롤 솔루션의 로컬 인스턴스를 설치했다. 그리고 당연히 이것들은 데이터를 백업하고 코드를 관리하는데 참으로 좋은, 환상적인 소프트웨어들이다.

그렇다면 왜 서비스로서 버전 컨트롤이 부상하는 것일까? 많은 이유가 있겠지만, 핵심은 커뮤니티일 것이다. 개발자들은 현재 예전 어느 때와도 다른 네트워크를 형성하고 있으며, 다른 개발자나 흥미로운 프로젝트를 팔로잉하고 업데이트가 되자마자 즉각적인 알림을 받는다. 열정적인 팔로어가 밑바탕이 돼 버그가 고쳐지고, 특성이 추가되곤 한다. 프로젝트 또한 누군가의 하드 드라이브에 들어있거나, ZIP 파일로 누군가의 웹사이트에 저장되어 있을 때보다 훨씬 더 많이 노출될 기회를 얻는다.

코딩하는 사람들만 이러한 재미를 보는 것일까? 아마 이 원칙은 웹 서비스로 제공되는 총체적 게임 개발 과정과 더 밀접한 연관이 있을지도 모른다. 이것은 코드 개발도 포함하지만, 자산 관리, 레벨 편집, 현지화, Q&A, 그리고 퍼블리싱 등과 같은 다른 기능을 포괄하는 말이다.

솔직히 말하자면 게임을 만드는 것은 어려운 일이고, (좋은) 게임을 혼자서 만드는 것이 개발자들에게 주어진 유일한 선택지는 아니다. 요구되는 기술의 범위는 종종 너무나 커다랗다. 개발자에게는 서로를 찾아내고, 각자에게 가장 흥미를 불러 일으키는 게임 프로젝트를 찾아 그 안에서 서로 연합해야 한다. 커뮤니티에서 게임에 추가 레벨을 만들어주길 바라는가? 문제 없다. 게임을 스페인어로 번역하고 싶은가? 아마 팔로워들이 도와줄 것이다.

다중 플랫폼이라는 성배

다중 플랫폼(cross platform)으로 게임을 만드는 것이 관건이다. 이것은 잠재적인 고객 범위를 넓혀주므로 결과적으로 상업적으로 더 성공할 수 있도록 도와준다. 썬 마이크로시스템즈가(Sun Microsystems)가 "한번 작성으로, 어디서나 실행(Write once, run anywhere)"이라는 문구와 함께 자바 프로그래밍 언어를 만든지 여러 해가 지났고, 많은 현대적 게임 엔진이 비슷한 기조 아래서 만들어지고 있다. 비슷한 견지에서, HTML5 는 확실히 게임 컨텐츠를 다양한 장비에 적용시킬 수 있게 해 준다.

게임 개발 툴에 있어 플랫폼 다중성을 갖추는 것이 매우 중요하다는 점 또한 이야기하고 싶다. 15 년 전쯤, 한 툴 프로그래머가 MFC 와 같은 기술을 이용해 앱을 만들어 냈다. 시간이 지나자 더 나은 기술이 개발되었고, .NET 어플리케이션 개발이 누워서 떡먹기라 알려질 정도가 되었다-- 하지만 그러면서도, 우리는 아직 Windows 라는 제약에 갇혀있다. 다양한 기기를 사용하는 우리의 삶 속에서, 프로젝트에 접근하게 위해 데스크탑 PC 앞에 앉아야 한다는 것은 너무나 커다란 제약이다. 일관된 인터페이스와 적당한 컨트롤을 지닌 툴을 이용해 어느 기기에서나 접속하고, 편집하고, 퍼블리시 할 수 있어야 한다.

HTML5 를 이용한 인터페이스 구축은 빠르고 군더더기가 없으며 다중의 브라우저를 통해 웹 어플리케이션을 설계하는 방법도 잘 다져져 있다.

그저 링크 한 단계

툴로서 HTML5 의 핵심 강점 중 하나는 모든 컨텐츠가 URL 에 담겨있다는 점이다. 게임을 만들어내는 각각의 리소스는 특정한 웹 주소를 통해 접근할 수 있다. 스크립트 파일, 텍스쳐, 사운드, 레벨, 심지어는 게임 그 자체까지도 말이다. 새로운 특성 하나가 프로젝트에 추가된다면, 개발자는 즉시 트윗을 날릴 수도 있고 메신저나 메일로 링크를 보낼 수도 있다. 다른 사람들은 싱크를 위해 아무 것도 할 필요가 없다. 그냥 즉시 보기만 하면 되는 것이다.

이것은 또한 게임 퍼블리싱을 간소하게 해준다. 게임이 이미 서버에 있고 플레이가 가능하다면, 더 넓은 웹 상에 퍼블리싱하는 것은 그저 올바로 접속하느냐의 문제일 뿐이다. 개발자는 게임에

빠르고 쉽게 숨을 불어넣을 수 있어야 한다 - 업로드할 것도 없고, 작성할 HTML 도 없다. 퍼블리싱은 단지 웹의 프론트 엔드에 달려있는 손잡이를 잡아당기는 것일 뿐이다.

요약

비디오 게임에 있어서 완벽한 기술이란 없다. 만들고자 하는 게임에 따라, 상업적인 목적에 따라 다른 기술을 버리고 이 기술을 선택한 데에 완벽한 이해가 있어야 한다. HTML5 는 단지 게임을 작동시키는 데에 뿐만 아니라 이것들을 구축하는데 사용되는 툴을 작동시키는 데에도 진정 놀라운 플랫폼으로 여겨지고 있다. 만약 당신이 HTML5 한계를 끌어안고 진화하는 표준을 이용해 작업을 해 나간다면, 이것은 당신의 생산성을 더욱 향상시켜주고, 더욱 많은 사람이 당신의 게임을 접할 수 있도록 잠재능력을 발휘할 것이다.

앞으로 게임을 만드는 과정은 계속해서 접근성이 향상되어 갈 것이다. 개발 툴은 더 많은 플랫폼에서 실행될 것이며, 사용하기가 더 쉬워지고, 협업의 과정은 더 간소해 질 것이다. 게임 개발자들의 글로벌 커뮤니티는 강화되고, 지식은 더욱 손쉽게 이동할 것이다. 그리고 궁극적으로 더 많은 사람들이 더 나은 게임을 만들고 우리 모두는 소비자로서 어떤 게임을 플레이 할만한가를 훨씬 더 잘 알게 될 것이다. 이 모든 것들이 웹과 HTML5, 그리고 이것이 속한 오픈 웹 플랫폼 없이는 불가능한 일이다.