코딩엔 무능, 그래도 게임은 만듦

(Suck at Coding, But Make Games Anyway)

작성자: 제이 웨스턴 (Jay Weston)

작성일: 2012 년 12 월 5 일

여러분이 기획자나 아티스트라면, 코딩은 자신이 가진 재능과는 아주 반대편에 있는 거라고 느끼고 있을 겁니다. 저도 마찬가지였거든요! 게임 기획에 열정을 갖고 있는 사람이라도, 코딩 없이는 게임을 만들 수 없습니다. 그렇죠? 하지만 유니티(Unity)와 플레이메이커(Playmaker, 유니티용 비주얼 스크립팅 시스템)와 같은 툴을 쓰면, 꼭 그렇지 않을 수도 있습니다.

저는 몇 년간 바이너리 스페이스(Binary Space)의 프로그래머 겸 CTO 인 잭슨 드루스(Saxon Druce)와 함께 <좀비 아웃브레이크 시뮬레이터(Zombie Outbreak Simulator)>와 <제 3 급 전염병(Class 3 Outbreak)> 두 개의 작품에 파트타임으로 참여한 적이 있었습니다. 그 후에 이력서 경력란에 채워 넣을 게임을 찾고 있었는데, 규모가 작고 완성 가능한 것이어야 했죠.

<좀비 아웃브레이크 시뮬레이터>를 iOS 로 포팅할 때 iOS 용 게임을 많이 해 봤는데, 그 중 <타이니 윙(Tiny Wings)>이 제 마음을 사로잡았습니다. 게이머의 입장에서 보면, 이어지는 점프를 정확히 해낼 때 무아지경에 드는 느낌이 정말 좋았습니다. 사업적 측면으로 보면, 기본적으로는 단 하나의 레벨에 극히 간단한 게임플레이만을 갖고 있기에 개발 기간이 짧다는 점이 마음에 들었습니다.

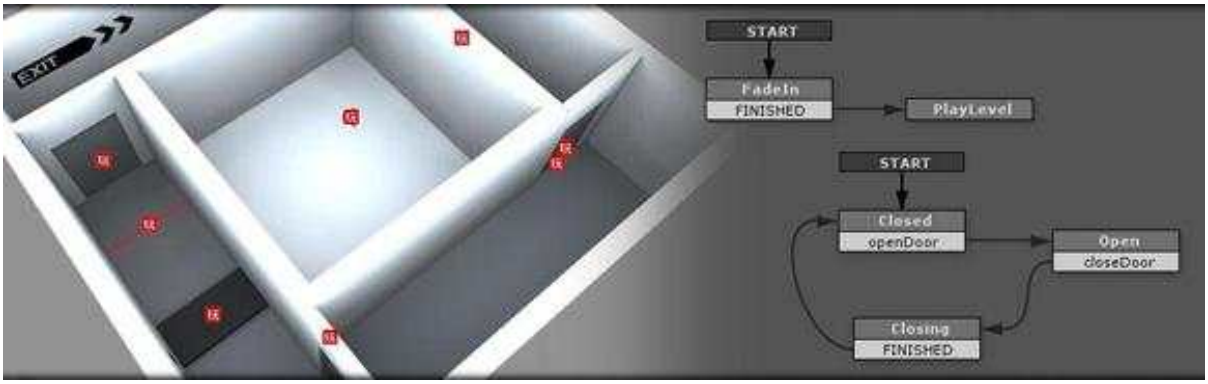
몇 가지 아이디어를 가지고 실험을 해보다가, <타이니윙>을 1 인칭 시점의 3 차원 게임으로 만들어 보기로 결정했습니다. 유니티만 사용해서 해봤는데, 지형 엔진에 공 하나를 굴러가게 하려고 기초적인 물리 코드와 씨름해야만 (웃지 마세요) 했고, 1 주 정도 지난 후에야 원하는 결과를 얻을 수 있었죠.

하지만 세상에서 제일 무능한 코더가 하는 일이라 진행은 더뎠고 짜증만 치밀어 올랐습니다. 3 차원 물리학과 벡터는 물론이거니와, 유니티 자체와, 거기에 자바스크립트에도 골머리를 썩여야 했죠. 물론 세상에는 별다른 문제 없이 스크립트 코드를 다룰 수 있는 기획자도 많겠지만, 나와는 거리가 먼 이야기였고, 이쪽 분야를 배우려면 개발은 접어두고 연습하고 공부하며 몇 년을 보내야 할 텐데, 32 살 나이엔 쉽지 않은 이야기죠!

플레이메이커 입성

어쩌다 보니 - 유니티 애셋스토어를 살펴보던 중이었던 것 같은데 - 플레이메이커를 발견했습니다. 플레이메이커는 비주얼 스크립팅 시스템이자 스테이트 머신 관리자로써, 상태(행동 action 의 집합. 각 행동은 미리 작성된 짧은 코드)와 그 사이의 전이로 게임을 만들 수 있게 해줍니다. 간단한 예를 들자면, "전투중", "대기중", "이동중" 등의 상태를 가진 유한 스테이트 머신(finite state machine, FSM)을 캐릭터에 붙이는 식입니다.

각각의 상태에는 애니메이션 행동이나, 레이캐스팅/사격 행동, 이동 행동 등을 포함시킬 수 있습니다. 하나의 상태에서 다른 상태로의 전이는 이벤트를 통해 이루어집니다. 예를 들자면 "대기중" 상태에 왼쪽 마우스 클릭에 반응하는 액션을 추가하고, 왼쪽 버튼을 클릭하면 "사격중" 상태로 전이하는 이벤트가 발생하게 만들 수 있습니다. 상태에 들어가는 각 행동은 기본적으로 미리 작성된 짧은 코드이고, 여러분은 이를 조합하고 다듬어서 개별 스테이트 머신을 만들 수 있고, 궁극적으로 전체 게임까지도 만들 수 있는 것이죠.



플레이메이커의 실험실 예제. 단순한 문열기/문닫기를 제어한다.

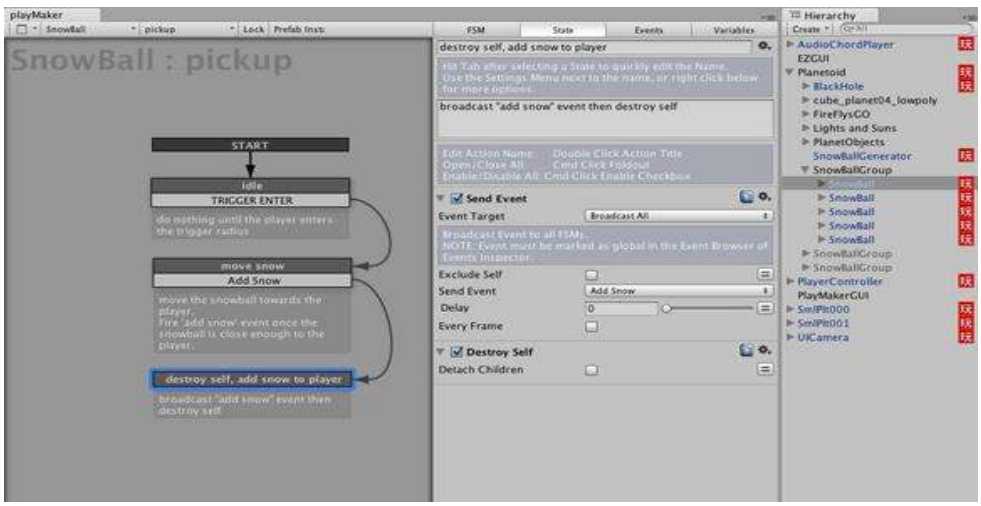
위 그림을 보면, 기술적 지식이 없는 사람에게는 이와 같은 비주얼 스테이트 머신이 얼마나 강력한 지 알 수 있을 겁니다. 제가 코드를 들여다보아도 그냥 검은 것은 글씨고 흰 것은 배경이다 정도만 알 수 있을 뿐이고, 언제 발동되는지, 어느 시점에 어떤 일을 하는지 이해해 보려고 해도 아무 소득이 없을 겁니다. 하지만 플레이메이커를 쓰면, 전체 FSM 을 한눈에 보고 무엇을 하는 건지 즉시 이해할 수 있죠.

편집기뿐 아니라 게임 화면에서도 각 오브젝트가 속해 있는 상태를 보여주기 때문에 실시간으로 어떤 일이 벌어지는 지 볼 수 있습니다. 이를 통해 FSM 은 물론 게임 전체적으로 무슨 일이 일어나고 있는 지를 이해하고 디버깅하는데 주는 이점은 말로 다 표현할 수 없을 지경입니다. 여기에 더하여 보조선, 브레이크 포인트 등과 같은 기능으로 무슨 일이 벌어지고 있는지 매우 명확히 이해할 수 있게 됩니다.

플레이메이커를 써서 제 첫 번째 창작물을 (거의 혼자서) 완성할 수 있었습니다. 제목은 <미지의 궤도(Unkown Orbit)>이고, 초현실적 3 차원 행성계에서 혜성이 되어 부유하고, 점프하고, 날아다니는 게임이죠. 파트타임으로 작업했기에 1 년 정도 걸렸는데, 풀타임으로 작업했다면 6 달이면 됐으리라는 생각이 듭니다. 게다가 지금은 유니티와 플레이메이커에 대한 이해도가 높아진 걸 감안하면, 이 게임을 다시 만든다면 몇 달 안 걸릴 것 같다는 생각이 듭니다.

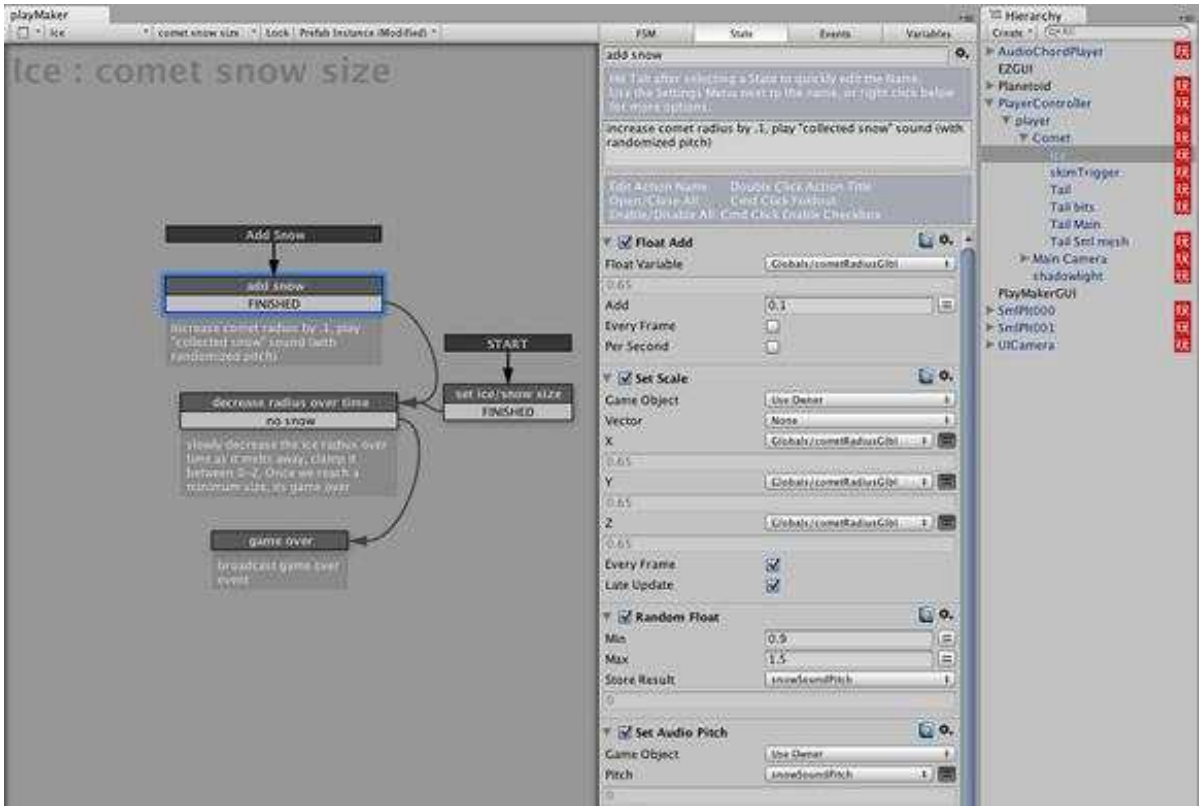
미지의 궤도

*미지의 궤도*에 쓴 FSM 중 플레이어가 눈덩이를 집을 때의 상황을 예로 들어 보겠습니다. 왼쪽에는 상태 사이의 이동을 제어하는 상태와 전이를 보여주고 있습니다. 가운데에서는 "자폭" 행동에 포함된 행동들을 보여주고 있습니다. 이 FSM 에서는 플레이어가 눈덩이를 향해 비행해서 이를 집어 들면, "눈 추가"라는 이벤트가 발송됩니다.



<미지의 궤도>에서 눈덩이 아이템에 쓰인 스테이트 머신

혜성 오브젝트는 이 발송된 이벤트를 "전역 전이"를 통해 수신 받고, 혜성의 얼음핵 반지름을 증가시키는 상태로 진입합니다.



<미지의 궤도>에서 얼음/눈 오브젝트에 쓰인 스테이트 머신

교환비용

아마도 이런 이점에는 당연히 비용이 있다고 생각하고 있을 테죠? 물론, 이런 것들을 로딩하면서 메모리 오버헤드가 있을 수 있고, 또, 시스템의 능력을 다 활용하지 못하는 데에서 오는 성능상의 타격을 받는 부분도 있을 겁니다. 플레이메이커는 어떤 특정한 게임에 최적화된 물건이 아니고, 이건 유니티도 마찬가지입니다. 하지만 오늘날의 하드웨어는 충분히 강력해서, 여러분이 정신 나간 짓을 꽤 심하게 하지 않는 한 벽에 부딪치는 일이 다른 방식을 사용하는 것보다 눈에 띄게 빨리 찾아오지는 않을 겁니다.

비용이라고 할 만한 것 중 제가 겪은 또 하나는, 게임 제작 프로세스가 몇 단계에 걸쳐 추상화되기 때문에 문제를 수정할 방법이 없다고 느낄 수 있다는 점입니다. 버그를 발견하고 이를 수정하려는 노력에 며칠을 소모해도 문제의 원인은 오리무중인 거죠. 하지만 이런 문제는 어떤 식으로든 대부분의 게임에서 흔한 일이고, 특히 다른 누군가의 엔진/코드/API 를 써서 개발한다면 더 심하죠. 하지만 엔진을 쓰면서 그 엔진에 들어가는 플러그인과 함께 쓴다면, 문제는 더 심각해 질 수 있습니다. 다행스러운 점은 유니티와 플레이메이커 모두 훌륭한 지원체계를 갖고 있고, 거기에 더하여 내 스스로 풀 수 없는 문제를 만날 때면 유니티와 플레이메이커 전문가와 직접 작업함으로써 이를 해소할 수 있었습니다.

또한, 아직 제가 겪어보진 않았지만, 유니티용 플러그인을 더 많이 쓸수록 충돌할 가능성도 많아지고, 제작자가 연락 두절되거나 지원이 끊길 가능성도 높아집니다. 유니티용 플러그인을 고를 때에는 발표된 지 시간이 좀 지나서 업데이트가 자주 이루어지고 제작자가 포럼에 활발히 활동하고 있는지 확인할 수 있는 걸 선택하는 걸 추천합니다. 제 경우에는, 제가 선택한 인터페이스 플러그인 역시 상당 기간 업데이트가 지속되었는데, 플레이메이커가 이를 실제로 지원하고 있었기에 매우 편리 했습니다.

장점

게임 기획의 측면에서 유니티와 플레이메이커는 최대한 빨리 아이디어를 구현하고 테스트하는 데에만 온전히 집중할 수 있게 해준다는 점에서 장점을 가집니다. 개인적으로는 게임 기획에 중점을 두고 있고, 따라서 이런 툴을 통해 아이디어를 떠올리고 증명/반증하는 과정을 단축할 수 있고, 짧은 시간 동안 연습과 경험을 최대한 많이 쌓을 수 있었죠. 개발에서 사업으로 전환했다가 수년 전에 다시 개발로 돌아온 저로서는, 그 동안 잃어버린 시간을 당장 벌충하고 싶었으니까요!

여러분이 만약 전형적인 대형 개발사의 게임 기획자라면 필시 "9999 번째 후속작" 같은 것을 만들고 있을 테고, 그런 경우에는 창의력을 발휘할 여지가 조금 있긴 하겠지만 두 팔을 걷어 붙이고 매우 새롭거나 독특한 기획을 추진할 수는 없을 겁니다. 여러분이 이를테면, (예를 들어)

레이싱 게임에 (원하든 아니든) "고착되어" 있는 기획자라면, 상대적으로 짧은 기간 안에 쉽게 다른 장르의 게임을 연습해 볼 수 있습니다.

저는 또한 프로젝트의 인원이 적을 수록 기획은 더 독특하고 유일해지며, 개별 게임을 더 많이 만들게 된다고 믿는 쪽입니다. 이런 종류의 툴을 쓰면 여러분이 그래픽/물리/애니메이션/기타 등등의 엔진을 만들어낸 팀의 일원이 되는 효과가 있습니다. 원하는 어떤 기획이든 마음껏 만들면서도 제약은 적을 겁니다.

린 스타트업?

저는 에릭 리스의 유명한 개념인 린 스타트업 원칙의 신봉자입니다. 린 스타트업은 대체적으로 사업적 접근법이라고 할 수 있습니다. 이는 제품이나 서비스를 출시하기 전에 아이디어를 검증하고, 그 후에는 재빨리 최소 존속 제품(minimum viable product, MVP)를 개발하고, 개발 사이클을 반복한 후, 소비자 반응을 최대한 빨리 살펴보는 접근법입니다. 전통적인 게임 개발과는 반대라고 할 수 있습니다. 게임 개발은 대개 기획문서로 게임을 정의하고, 제작한 후, 잘 팔리기를 바라는 식이었습니다.

린 스타트업 전략을 고려하면, 유니티와 플레이메이커의 조합은 대형 개발사가 프로토타입이나 MVP 를 신속히 만들고자 할 때도 좋지만, 1 인이나 소규모 팀에도 적합한데, 특히 코더가 부족한 상황에서 스스로 타개하고자 할 때는 안성맞춤입니다! 킥스타터(Kickstarter, 소셜 펀딩의 일종)를 이용하고, 게임개발을 위한 적절한 린 스타트업 전략을 갖춰서 프로토타입을 재빨리 만들고, 킥스타터에서 개발에 피치를 올려서 사람들이 여러분의 게임을 구매할 것인지를 더 많은 것을 투자하기 전에 알아보십시오.

(더 자세한 것은, 타일러 요크(Tyler York)가 가마수트라에 게임 린 스타트업에 대해 쓴 글[1]을 참조하기 바랍니다.)

---

[1] 참조링크: http://gamasutra.com/view/feature/168647/making_lean_startup_tactics_work_.php

그러면 제가 린 스타트업 전략을 행한 방법이 궁금하시겠죠? 음, 했다고 하기도 안 했다고 하기도 묘합니다. 왜냐하면, 이는 제가 만든 첫 번째 유니티 게임이었고, 배울 것이 더 많았고, 린 방법론에서 다양한 개념을 차용하긴 했지만, 검증된 게임 컨셉트를 안전한 방식으로 적용하기로 결정했기 때문입니다. 그런 결정을 내리기 전에 <타이니 윙>, <다이노 런>, <돌핀 올림픽> 등의 유사한 다른 게임을 살펴보았는데, 이들 게임은 어느 정도 성공적이었고, 저는 충분히 성공 가능성이 있는 컨셉트를 목표로 하고 싶었거든요. 하지만 여러분이 원한다면, 정석대로 여러분의 아이디어가 얼마나 인기 있을 지 검증해 볼 수도 있습니다. 구글이나 페이스북 등에 배너나 텍스트 광고를 달고 클릭율이 얼마나 나오는지 보는 것이죠.

그 다음에는 저에게 익숙한 (3D 아트/텍스처를 공부한 적이 있음) 그래픽 스타일에, 애니메이션 없고(그쪽 기술은 전무), AI 없고, 느낌에 따라 튜닝 할 수 있는 지극히 단순한 게임 방식을 골랐습니다. 이 게임의 MVP 는 유니티의 자체 물리엔진을 이용한 덕분에 며칠 만에 완성되었고, 몇몇 친구들에게 보여주고 검증 받은 후, 제품 완성을 향해 매진했습니다. <좀비 아웃브레이크 시뮬레이터>를 작업하면서 잘 알게 된 freesound.org 에서 게임에서 쓸 거의 모든 사운드를 빠르고도 무료로(!) 구할 수 있었죠. 음악은 제 친구 리스 린지가 끝내주는 우주풍 음악을 녹음해 주었습니다. 이들 덕분에 저는 개발 공정 학습과 게임 튜닝에 온전히 집중할 수 있었습니다.

제 생각에는, MVP 와 최종 제품을 최소 애셋, 최소 비용으로 개발하고 개발 사이클을 빠르게 자주 했다는 점에서 크게는 린 스타트업 원칙을 따랐다고 말할 수 있을 것 같습니다. 하지만 겁먹은 나머지 친구들 외의 다른 사용자 피드백을 빨리 얻지를 못했습니다. 제작 방법을 익히느라 달팽이처럼 기어가는 동안 다른 누군가가 내 아이디어를 훔칠 까봐 염려했기 때문입니다. 또한 온전히 나만의 것을 만들고자 하는 마음이 있어서 외부의 반응을 원치 않았던 면도 있었습니다. 우스운 것은, 프로젝트 막바지에 이르러서는 테스터와 일하면서 그들의 피드백을 받는 것이 진짜 유용하고 매우 재미있었다는 점입니다. 더 많은 행성을 넣지 못한 점, 테스터들께는 사과 드립니다! 다음 업데이트를 기대하세요!

플레이메이커로 유니티 게임을 혼자서 완성해본 결과, 비슷한 상황에 놓인 개발자들에게 다음과 같은 조언을 해주고 싶습니다.

1 **스스로 할 수 있는 작고 단순한 게임을 목표로 하고, 자신의 현재 강점을 활용하라.**

<미지의 궤도>는 몇 달 만에 완성할 수 있을 정도로 단순하다고 생각이 들었습니다. 만약 제가 <타이니 윙>을 그대로 베끼고, 다소 복잡한 3D 행성/위성 설정을 뺀다고 하더라도, 게임을 잘 작동했을 겁니다. 하지만, 시간이 오래 걸리는 작업이 무엇인지를 생각해야 합니다. 여기서는 캐릭터와 애니메이션이 되겠습니다. 가장 단순한 3D 모델링이라고 해도 그래픽 쪽 경험이 없다면 많은 시간이 걸릴 겁니다. 저는 3D 모델러가 첫 직업이었지만, 블렌더에 익숙해져서 속도가 붙을 때까지는 시간이 다소 소요되었습니다.

2 **단순한 그래픽 스타일로 애셋을 재사용하고 게임플레이를 반복하라.** (반복 vs 연결을 다룬 로스트 가든의 글[2] 참조)

막대한 양의 콘텐츠와 20 개 이상의 독특한 맵이 있는 스토리 기반 게임은 애셋을 제작하는 데에만 많은 시간이 걸리지만, 수퍼 헥사곤과 같은 게임은 그래픽 애셋을 조금만 쓴 완벽한 예가 될 것입니다.

3 **공개(Creative Commons) 애셋을 활용하라.** 특히 사운드와 음악에 있어서 freesound.org 는 최고입니다! CC 로고만 확인하지 말고, 라이센스를 읽고 상업적 사용을 허용하는지 확인해야 합니다. 저 같은 경우는 파일이름을 그대로 유지하여 모두의 권리가 올바르게 보장되도록 했습니다.

4 **어떤 식으로든 조언을 구하라.**

조언의 형태는 유니티/플레이메이커 포럼이나 유니티 채팅 채널의 사용자가 될 수도 있고, 꽤 많은 수의 플레이메이커/유니티 3D 튜토리얼이 될 수도 있습니다. 개인적으로는 "잘돌아가는게임(WellPlayedGame)" 튜토리얼을 추천합니다. 도움의 손길이 없었다면, 아마도 저는 벽에 머리를 찧으며 보내는 시간이 더 많았을 겁니다.

5 **코딩을 한 두줄 정도는 해야만 합니다. (충격과 공포!)**

저 같은 경우는 바이너리 스페이스의 잭슨에게서 득점 순위 시스템에 도움을 받았습니다. 플레이메이커가 발전하면 코딩을 할 필요성이 점점 줄어들겠지만, 현재로서는 상당수의 게임은 조금이라도 코딩이 필요하리란 생각이 듭니다. 최근에

[2] 참조링크: http://www.lostgarden.com/2012/04/loops-and-arcs.html

추가된 기능 중 하나는 쟝 파브르가 만든 배열 생성기인데, 그게 있었더라면 득점 순위 시스템에 사용할 수 있었을 지도 모릅니다.

정리하자면, 몇몇 게임에서는 코딩이 여전히 필요합니다만, 전체 게임을 혼자서 제작/출시 할 수 있음이 증명되면 자신을 포함, 잠재적 파트너, 투자가, 킥스타터 참여자 등에게도 작업을 성실하게 하며 일을 마무리 지을 수 있다는 점을 인정받을 수 있을 겁니다.

첨언하자면, 묘하게도 플레이메이커를 쓰면서 객체지향 설계 같은 프로그래밍/설계 개념에 대한 이해가 더 확고해 졌습니다. 이런 개념을 배우는 데에도 유용하다고 할 수 있겠죠!

그 최종 결과물[3]이 바로 1 년간 (파트타입으로) 작업한 <미지의 궤도>로써, 프로그래밍 경험은 전혀 없고, 3D 기술 약간, 그리고 약간의 게임 기획 경험만을 가진 사람이 iOS 용 유니티와 플레이메이커로 만들어낸 것입니다.

미지의 궤도는 익스블리티브의 제이 웨스턴이 개발하였고, 아이폰과 아이패드용으로 앱스토어에서 판매 중입니다. 미지의 궤도는 플레이어가 초현실적 3 차원 행성계에서 혜상이 되어 떠다니고, 점프하고, 비행하는 게임입니다. 여기에서 확인할 수 있습니다.[4]

[3] 참조링크: http://youtu.be/W1CRM5EMMYo

[4] 참조링크: http://www.exbleative.com/unknown_orbit/