「STYX (The Boss)」

김지훈 / ㈜키프엔터테인먼트 개발팀장

〈Styx〉는 온라인 전용 게임이다. 유, 무료의 구분에 상관없이 유저의 요구를 받아들이며 끊임없이 발전하고 성장해나가야만 한다. 그렇기 때문에 〈Styx〉는 장르의 한계 내에서는 모든 가능성을 열어 두고 있다. 이는 우리 제작자들이 가장 염두에 두고있는 사항이다. 즉, 언제든 기꺼이 즐거운 마음으로 소화할 준비가 되어있다는 뜻이다.

1. 기획 과정

〈Styx〉는 기획적인 면에서는 처음부터 우여곡절을 많이 겪었다. 즉, 장르가 RPG나 전략이 아닌 단순 액션게임이라는 데에 기인하기도 하지만 더 큰 원인은 패키지게임이 아닌 온라인 전용이라는 데에 있었던 걸로 기억된다. 〈Styx〉의 제작목표가 각 단계마다 진행해야 할 미션 위주의 화려한 싱글 플레이가 아닌, 단순한 게임 설정과 안정화된 대단위 서버에 동시에 여럿이서 즐길 수 있도록 제작 초점을 맞추다 보니, 기본적인 FPS (First Person Shooting: 일인칭 슈팅)에서 구현하고 있는 것 중 가능하고 직관적인 사항들만 검토하게 됨으로써 기획적으로 어디까지 다가서야 할지 그 한계가 모호해서였다.

〈그림 1〉 일러 외주를 통한 초기 캐릭터의 모습

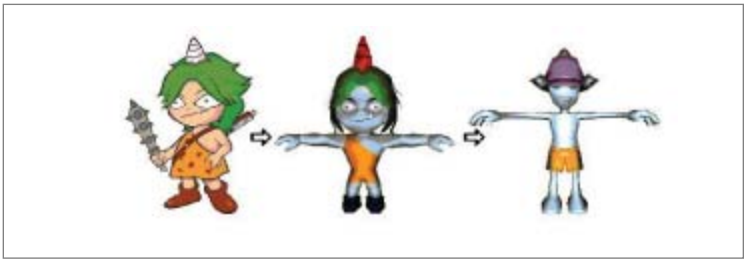
이 때문에 〈Styx〉의 기획과 제작은 공동기획, 공동제작이라는 형태를 띠고, 숱한 회의와 일러스트레이션 외주를 통해 캐릭터의 주된 컨셉을 정하게 되었고, 또 갖가지 테스트를 통해 여러가지 다양한 맵을 시험해보고 나서야 맵을 설정하게 되었다.

물론 어느 게임이나 대부분 그렇겠지만 일단 정해진 사항도 각종 기술적인 측면이나 주위의 우려(?) 혹은 좀더 잘 해 보려는 욕심으로 인해 그 또한 수정, 재고되기 일쑤이다. 특히 이번 프로젝트에서는 엔진 상의 구현문제와 분위기의 조화라는 측면에서 유독 말도 많고 탈도 많았던 부분이 바로 캐릭터의 설정이지 않나 싶다. 〈그림2〉에서 보듯이 조금씩 달라지는 캐릭터의 진화모습을 보면 이 점을 잘 설명해준다.

2. 그래픽 과정

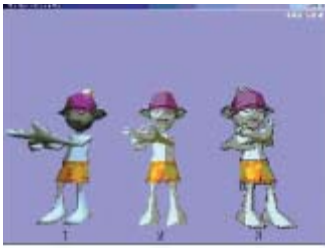
이제 닻을 올렸다. 정해진 수순이지만 일단 맵과 캐릭터의 설정을 정하고 난 후에는 바로 모델링에 착수를 하게 된다. 여기서 우리는 게임 상에서 캐릭터가 어떻게 보일런지에 대해서 심각하게 고민을 할 수 밖에 없었다. 일반적인 3D게임이라면 매핑(모델링이 끝난 후 그림을 덧씌우는 작업)이 끝난 연후에 단순히 빛만 고려해서 게임 상에 띄우지만, 뭔가 차별성을 구분하기에는 다소 부족해 보였다. 원래의 〈Styx〉 제작의도는 비교적 무거운 분위기 일색이던 FPS류(대표적인 예로 〈퀘이크〉, 〈언리얼토너먼트〉, 〈카운트 스트라이크〉 등)의 게임을 쉽고 가볍게 즐길 수 있도록 제작 해보자는 데 있었기 때문에 캐릭터만 귀엽게 제작하는 것으로 원래의 의도가 충분히 반영될 수 있다는 판단이 서지 않자 뭔가 다른 방법을 모색하게 되었다.

〈그림 2〉 캐릭터의 진화



그 결과 '카툰 렌더링' 이라는 방식을 게임에 도입하는 것은 어떨지 유사게임 분석을 통해 한번 시도 해보기로 했다. '카툰 렌더링' 이라 함은 보통의 3D 오브젝트들을 렌더링하는 데에 있어 실제에 가깝게 자연스러운 빛 처리를 하지만 이를 투톤, 혹은 그 이상의 색깔로 제한함으로서 카툰같은 느낌을 인위적으로 만드는 것이다. 실제 게임상에서 보이는 캐릭터를 여러가지 방식으로 띄워 놓은 〈그림 3〉을 보면 이 차이점을 좀더 명확이 느낄 수 있을 것이다.

〈그림 3〉 캐릭터의 렌더링



이와 같은 생각에서 우리는 프로그램 파트에서 지원만 가능하면 카툰 렌더링 방식을 도입하기로 정하고 작업에 착수 하였다. 단지 우려가 되는 것은 아무래도 일반 3D그

〈그림 4〉 스크린 샷

래픽 카드들이 하드웨어적으로 카툰 렌더링을 지원해 주지 않기 때문에 그러한 작업들을 몇 줄의 코딩으로서 소프트웨어적으로 해결하기에 무리가 따르지는 않을까 하는 것이었지만, 실제 테스트 결과 테이블 리스트를 통해 계산량을 조절하여 조금만 퍼포먼스를 높인다면 썩 무리가 가지 않을 것 같다는 판단을 내렸다.

실제 엔진상에서 띄운 캡쳐 화면인 〈그림 3〉을 보자. 1번 그림은 일반적인 3D게임처럼 빛이 자연스럽게 처리된 것을 볼 수 있지만, 2번 그림을 보면 그림자가 어둡고 밝은 부분이 명확히 드러남으로 인해서 조금은 채색된 듯한 느낌이 표현되었고, 3번 그림은 외곽라인을 한 번 더 처리해서 좀 더 '카툰' 같은 느낌을 살린 것을 볼 수 있다.

이는 맵도 마찬가지이다. 물론 여러 번의 시행착오를 거치기는 했지만 결국은 만족할 만한 분위기를 낼 수가 있었다. 일단 맵은 Vertex (3D 모델링의 기초가 되는 벡터를 의미한다)가 넓게 분포되어 있어서 캐릭터와 같은 자연스러운 그림자 처리가 불가능하기 때문에 Mapping만으로 그와 비슷한 분위기를 낸 다음 외곽라인만 처리해 줌으로써 캐릭터와의 이질감을 없애는 데 주력하였다. 〈그림 4〉는 실제 플레이 중인 게임화면이다. 맵을 자세히 보면, 매핑된 원화는 명암이 뚜렷한 그림으로 사용했고 그림자 처리는 다른 게임과 마찬가지로 라이트맵으로 처리한 후에 캐릭터와 마찬가지로 검정색의 라인을 둘렀다. 한가지 유일한 문제라면 라인을 그렸을 때와 그리지 않았을 때 그래픽카드에 따라 프레임 레이트가 차이가 난다는 것인데, 이는 옵션으로 선택, 처리하게 함으로써 자신의 컴퓨터 사양에 맞추어 할 수 있게 처리하도록 하였다.

〈그림 5〉 Interface 디자인 변천사

이 후로 게임성을 고려하여 다양한 맵을 테스트하고 제작하면

서 그래픽 작업은 박차를 가하게 된다. 캐릭터 역시 신선한 아이디어로 추가 제작을 하게 되는데, 이 때부터 내외부 게임시스템의 디자인에 대한 고민이 시작되었다. 이는 본 게임에 들어가기 전이나 플레이 중에 화면을 조작하기 위한 환경을 뜻하는데 이 역시 여러 번의 시행착오가 있었다.

〈그림 5〉는 어떻게 Interface 디자인들이 바뀌어 왔는지를 나타내준다. 처음엔 타 팩키지 게임처럼 3D환경에서 동적이면서 화려한 디자인으로 출발했으나 자체 회의를 통하여 온라인 게임의 특성을 고려한 간결하고 직관적인, 유저에게 익숙한 윈도우 위주의 그래픽으로 결론을 내게 되었다. 이는 네트워크 프로그램과의 효율적 연계 및 작업기간의 단축이라는 결과를 가져왔다. 즉, 디자이너는 윈도우의 프레임과 내부의 버튼 등을 유저의 취향에 맞게 디자인하고 배치함으로써 길지 않은 기간에 작업을 마무리 하게 되었고 네트워크 프로그래머는 익숙한 윈도우(API) 코드를 바탕으로 유기적으로 클라이언트 프로그램과 연계를 꾀할 수 있게 된 것이다. 이 외에도 무기디자인과 각종 효과들은 병행하게 되는데, 작업량은 생각보다 많지 않아서 전체 그래픽 일정에는 큰 무리가 가지 않았다.

3. 클라이언트 프로그램의 검토

이렇게 제작된 캐릭터를 설계된 맵(배경)과 같이 엔진 상에 띄우고 게임성을 접목시키는 일이 프로그래머의 역할이다. 일반적으로 알고 있는 바와 같이 맵과 캐릭터를 엔진 상에서 보이도록 하는 것은 크게 문제가 되지않는다. 단지 모델링된 데이터를 불러내서 엔진 상에서 렌더링만 시킨다면 이는 그래픽카드 테스트용이지 게임이 아니기 때문이다.

충돌체크 (collision detection)

스프라이트라는 2D좌표(x, y)를 가지는 점을 그대로 뿌려주는 2D게임과 3D좌표(x, y, z)들로 된 점, 면으로 구성된 오브젝트를 화면에 뿌려주는 것에서는 서로 공통점을 갖지만, 이를 운용하는 방식에서는 근본적인 차이가 난다. 그러한 것들이 충분히 답보되어 살아있는 유기체처럼 아주 조직적으로 움직여야 게임이라 말할 수 있는데 3D게임에선 2D게임에 비해 훨씬 복잡하고 까다로운 과정을 거치게 된다. 그 중의 하나가 충돌체크라는 것으로, 쉽게 말해서 총알과 주인공, 맵과 주인공의 충돌여부를 검사하는 것을 뜻한다. 좀더 과장되게 말하면 한마디로 충돌체크라는 게 어느 3D엔진에서건 가장 중요한 부분이자 전부라고 할 수 있겠다. 충돌체크가 없다면 게임은 매우 현실적이지 못할 것이다. 어떤 등장 인물, 무기 혹은 어떤 물체가 배경들을 뚫고 돌

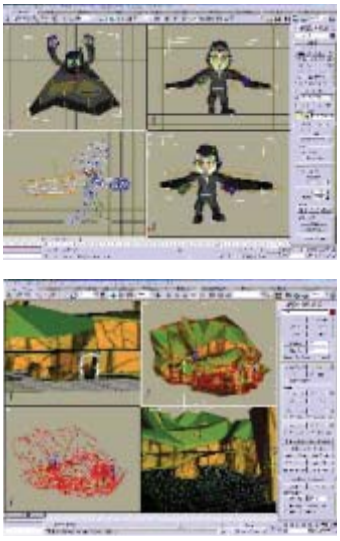
아다닌다면 어떻게 게임이라 할 수 있겠는가.

따라서 충돌체크란 3D 환경에선 가장 기본적인 요소이기 때문에, 무기 - 캐릭터, 캐릭터 - 캐릭터, 무기 - 맵, 캐릭터 - 맵 등 상호간의 충돌을 포괄적으로 다루기 위한 작업을 해야만 했었다. 그러면서도 가능한 한 충돌 로직을 단순화 시키고 각자의 특성에 맞게 수정이 용이하도록 함수를 일원화한 다음, 검출해야 할 충돌범위를 최소화시키는 작업을 통해 전반적인 퍼포먼스를 높이는 데 주력을 했다.

배경 (map)

〈Styx〉에서 맵을 구현하는데 있어서는 일반적인 FPS에서 구현되고 있는 이원 스페이스 분할(Binary Space Partition)이라는 재귀 알고리즘을 통해 구현되었다. 물론 이외에도 다른 공간 분할 기법들이 다양하게 존재하고 있지만 자체 테스트 결과, 몇가지 문제점을 제외하고는 큰 무리가 없는 것으로 판단이 되어 기존의 BSP 작성 함수들을 〈Styx〉의 제작환경에 맞게 수정하는 선에서 일단락 지었다.

〈그림 6〉 캐릭터와 배경 제작 화면[1]



캐릭터 (character)

전작이었던 〈XERO〉의 방식과는 전혀 다른 캐릭터의 구현이 관건이었다. 일단 상체, 하체를 구분하는 작업과 마우스의 상하 변동 폭에 따라 상체만 허리를 중심으로 회전 시키는 일, 좌우변동에 따라 어느 일정 범위 내에서 좌우로 회전 시키는 일 등이 그것이다. 이 작업들은 순조롭게 진행이 된 편이었다. 단, 문제가 되었던 부분이 각 캐릭터의 동작을 결정짓는 애니메이션에 있었는데, 이 역시 여러 가지 방법들이 시도되었다.

처음 시도할 때는 키 프레임 정보를 읽어와서 부분 애니메이션을 한 후에 관절을 이

1) 실제 작업은 3ds-Max를 통해서 하고 프로그램과의 연계는 자체 제작한 플러그 인(Plug-In)을 통해 Export해서 사용한다.

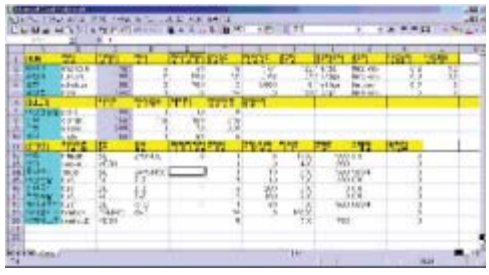
어 붙이는 스키닝작업과 함께 캐릭터를 구현했으나 다소 딱딱해 보인다는 중론에 따라 자연스러운 애니메이션이 가능하도록 모핑을 지원하는 샘플링 방식을 취하게 되었다. 이의 단점이라면 각 점의 이동정보를 모두 기억하고 있어야 하기 때문에 정보량이 생각보다는 많다는 것이었는데 이는 최소의 프레임을 운용하면서 끊김이 느껴지지 않도록 각 프레임 간에 진행시간에 따라 중간점을 찾아가는 방식을 통해 극복하게 되었다.

효과 (effect)

이후 배경과 캐릭터를 제외한 그래픽 엔진을 전반적으로 손보게 되는데, 대표적으로 게임 플레이 도중 발생되는 효과가 있겠다. 효과의 종류로는 폭발 장면, 잔상, 무기 효과, 아이템 소멸, 배경 효과 등이 있는데 주로 파티클 엔진과 단순한 텍스쳐 애니메이션 또는 직접 모델링한 오브젝트로 구현이 된다. 이 효과들을 구현하는 데 있어서는 이미 전작 〈XERO〉를 통해 충분히 숙지하고 있었으므로 별다른 어려움을 느끼지는 못했지만, 카툰 방식을 고집하다 보니 텍스쳐 소스를 배경과 조화시키기 위해 신경을 많이 써야만 했다.

〈그림 7〉 각종 EFFECT[2]

〈그림 8〉 각 무기에 따른 밸런스 조율을 위한 엑셀파일



게임 시스템 (system)

게임 자체가 슈팅인 까닭에 게임 내부 시스템 역시 무기와 연관이 많았다. 각 무기마다 기본 제공되는 총알과

2) 파티클 효과가 다른 모듈에서 제작되는 과정과 실제 게임에서 효과들이 적용되는 모습

그리고 연사 횟수, 상대에게 입히는 데미지, 자동화기 여부 등 모든 것들이 슈팅감을 살리자는 의도 아래 밸런스 조율이 들어갔다. 계속하여 나중에 추가한 것이지만, 슈팅을 했을 때 화면이 흔들리면서 조준점도 덩달아 흔들려 정확한 사격이 방해 받도록 한 점, 데미지를 입었을 때 화면에 빨간색 필터링과 함께 일시적으로 캐릭터의 움직임을 제한한 점 등이 유사게임(일인칭 슈팅게임) 분석을 통해 지속적으로 이루어졌다. 아울러 맵에서 습득할 수 있는 아이템 역시 기본적인 체력회복 아이템을 제외하면 거의 무기와 관련되어 있었기 때문에 각각의 결정요소를 조율하는 데 있어서 세심한 관심을 기울여야만 했다.

4. 서버 프로그램의 문제

아쉽게도 〈Styx〉에는 별다른 인공지능이 가미되지 않았다. 실제 주인공인 플레이어 외에 등장하는 적(Enemy)이 바로 타 계정을 통해 접속해 있는 또 다른 유저이기 때문이다(연습모드에서 날아다니는 박쥐 정도가 유일하다).

장르가 RPG나 전략이 아니면서도 단순 액션게임도 아닌 온라인 전용이라는 점에서 네트워크(Network)의 역할은 상당히 중요한 역할을 차지했다. 네트워크 게임을 만듦에 있어서 가장 먼저 대두되는 문제가 다음 두 가지일 것이다. 첫째는 상대편과의 대전이 주목적이기 때문에 처음에는 한 방(room)에 들어가서 게임을 하는 유저의 수를 결정하는 문제이다. 여러 명이서 전략과 전술을 겸비할 수 있도록 16 내지 32명 또는 그 이상으로 해야 할 것인지, 아니면 동시 시작, 동시 종료라는 온라인 특성상 4명이나 8명 정도의 적절한 수준으로 해야 할 지 결정을 해야 했다(반복된 회의와 테스트 결과 8명이 가장 적합한 것으로 잠정적인 결론을 내렸다).

〈그림 9〉 현재 제작중인 온라인 Styx 홈페이지

둘째는 게임의 방식이다. 유저가 클라이언트로 접속하는 기능과 서버로서 동작하는 기능 두 가지를 가지게 하는 방법(Peer Server type)과 하나의 서버를 만들어서 이 서버는 플레이에 참여하지않고 플레이들 간의 메시지 전달과 조정을 담당하는 방법(Remote Server type) 중 어느 것을 선택하느냐가 관건이었다.

여기에 더해서 위의 두 가지 사항이 결정된 이후에도 문제점은 끊임없이 도출되었

다. 슈팅 게임이다 보니 네트워크 신호의 동기화 처리 부분이 상당히 문제가 되었고 캐릭터의 이동 처리, 아이템의 생성위치 동기화 등 조절을 보아야 할 것들이 상당히 많았다. 나아가 실제 게임 서버 외에 접속 서버에서는 게임의 전적 및 유저의 정보 처리 능력을 보다 극대화 시켜야 했다. 즉, 게임 중 전송되어져 오는 유저들의 많은 자료들을 기록하고 읽어 들이는 것과 이를 실시간으로 변경하는 알고리즘을 갖추어야 했다.

결국 산적해 있는 많은 이러한 문제들은 수 차례의 테스트과정을 거치고, 만족할 만한 결과를 도출해낼 때까지 이 과정을 반복하면서 해결할 수 밖에 없었다. 지금 돌이켜 보면 이 때가 가장 힘들었던 때가 아닌가 한다. 기획, 프로그래머, 각 분야를 불문하고 모든 제작 인원들이 각 작업을 일시 중지하고 실제 유저가 되는 테스트과정이란 게 그리 반길만한 사항은 아니었기 때문이다.

5. 맺음말

이 지점까지 오는 동안 많은 장애물들을 헤치고 수많은어두운 틈들을 횡단하면서 여정을 마쳤다. 지금에 와서야 하는 이야기이지만 제작 과정 자체가 썩 순탄치만은 않았던 것 같다. 9개월여 시간동안에 팀원들간의 사기 진작문제, 중간에 작업자가 바뀌면서 작업 방식도 덩달아 바뀌게 되었던 문제, 잦은 작업회의와 그에 따른 시행 착오들, 이 모든 것들이 아마도 첫 시도에 대한 불안감과 경험 부족에서 시작된 듯 싶다. 그렇지만 어찌 되었든 지금은 이렇듯 소기의 목적은 달성했지 않은가. 이제 앞으로의 남은 일은 유저의 요구를 얼마나 넓게 소화할 수 있느냐의 문제가 관건일 것이다. 아무리 게임이 각 기술적 측면으로 잘 되었다 할 지라도 정작 유저에게서 버림받는다면 의미가 없기 때문이다.

〈Styx〉는 이제 오픈 베타를 앞두고 있다. 시험을 치고 결과를 기다리는 수험생의 심정과 다를 바가 없다. 남은 것은 유저들의 따끔한 질책일 것이다. 그로 인해서 우리들은 다시 한번 각자의 한계에서 탈피하게 될 것이며 〈Styx〉는 꾸준히 업그레이드가 될 것이다.

〈Styx〉는 온라인 전용 게임이다. 유, 무료의 구분에 상관없이 유저의 요구를 받아들이며 끊임없이 발전하고 성장해나가야만 한다. 그렇기 때문에 〈Styx〉는 장르의 한계 내에서는 모든 가능성을 열어 두고 있다. 이는 우리 제작자들이 가장 염두에 두고있는 사항이다. 즉, 언제든 기꺼이 즐거운 마음으로 소화할 준비가 되어있다는 뜻이다. 이 얼마나 재미난 일인가. 게임을 만드는 일, 더욱이 온라인 게임을 만드는 일, 가상공간에서의 마법사가 되는 것, 길지 않은 인생에 있어서 마법과 같은 일 아니겠는가.Ⓖ