



**UNREAL**  
ENGINE



이미지 제공: Productions Vox Populi Inc.(복스 포퓰리 프로덕션)

## ICI Laflaque

타이트한 일정 속에서 방송 퀄리티 확보하기

# 목차

	페이지		페이지
<b>1. 프로그램 소개</b>			
— 역사	4	— 리깅	21
— 제작 일정	7	— 추가 골격	22
	8	— FBX 익스포트 문제	24
		— 구성과 명명의 문제	26
			26
<b>2. 언리얼 엔진의 도입</b>	<b>10</b>	<b>— 애니메이션</b>	
— 기존 파이프라인	11	— 텍스처 스위칭을 통한 애니메이션	26
— 언리얼 엔진 파이프라인	13	— 피부색	27
— 사용 툴	14	— 얼굴 애니메이션	30
- 기존 파이프라인	14	— 시퀀서의 이용	31
- 언리얼 엔진 파이프라인	14	— 라이팅	33
		— 렌더링	33
<b>3. 생산 단계</b>	<b>15</b>		
- 모션 캡처	17	— 안티앨리어싱	33
- 레이아웃	17	- 렌더링 퀄리티 비교	35
- 카메라 데이터 임포트	18	— 편집과 전달	36
- 카메라 스위치	20		
— 모델링	20	<b>4. 향후 비전</b>	
— 셰이딩	20	<b>5. 문서 소개</b>	<b>39</b>

# ICI Laflaque

## 타이트한 일정 속에서 방송 퀄리티 확보하기

주간 애니메이션 정치 풍자 쇼인 ICI Laflaque 프로덕션 팀은 2004년 첫 방송부터 실시간 파이프라인을 고수하고 있습니다. 이와 같은 워크플로는 타이트한 제작 일정을 맞추기 위한 필요 사항이 반영되었습니다. 매회 30분 방송에는 속보를 토대로 한 새 대본이 사용되며, 더빙부터 애니메이션 및 최종 방송까지 7일 이내에 모든 프로세스를 완료해야 합니다.

매주 이 일정을 맞추기 위해서는 처음부터 실시간 렌더링(또는 준 실시간 렌더링)이 필요했습니다. 이 빠듯한 일정 속에서, Vox Populi Productions(복스 포풀리 프로덕션)은 지난 14년 동안 준 실시간 렌더링을 이용해 Autodesk MotionBuilder로 [ICI Laflaque](#)를 제작했습니다.

그러다가 언리얼 엔진이 비주얼 퀄리티를 크게 높여줄 수 있음을 알고 2018-2019시즌부터 언리얼 엔진으로의 전환을 결정했습니다. 본 문서에서는 복스 포풀리가 언리얼 엔진으로 쇼의 파이프라인을 바꿔 이미 최적화된 워크플로는 그대로 유지하면서도 실시간으로 작업하는 결과물은 개선한 사례를 소개합니다.

프로덕션 팀은 또 언리얼 엔진을 통해 쇼의 비주얼 퀄리티를 높였을 뿐만 아니라 여러모로 작업의 속도를 높여 창작에 더 많은 시간을 할애할 수 있게 되었습니다.



# 프로그램 소개

## 프로그램 소개

ICI Laflaque는 수상 경력을 자랑하는 프랑스어판 정치 풍자 쇼입니다. ICI Télé Radio- Canada 네트워크에서 방영 중인 이 프로그램에는 실사 기반의 3D 애니메이션 캐릭터가 다수 등장합니다.

주인공은 가공의 뉴스 앵커인 Gérard D Laflaque입니다.

각 에피소드는 두 스토리라인을 따릅니다. 하나는 Laflaque가 진행하는 시사 TV쇼인 “ICI Laflaque”입니다. 이 프로그램은 Laflaque가 그날의 뉴스에 대해 논평하고, 유명 정치인을 인터뷰하는 방식으로 진행됩니다. 또한, 이 프로그램에는 공인을 소재로 여러 가지 가공의 이야기를 보여주는 코너가 있습니다. 이 쇼의 코너 중 상당수는 불과 일주일 전에 일어난 실시간 뉴스를 다룹니다.



그림 1: Radio-Canada의 뉴스 앵커인 Céline Galipeau와 함께 TV 쇼를 진행 중인 Gérard D Laflaque



그림 2: 열띤 주장을 펼치는 Céline Dion



그림 3: 전 Montreal Canadiens 선수 Dave Morissette와 Ron Fournier 심판이 하키에 관해 얘기를 나누는 모습

Gérard D Laflaque는 그동안 TV 정치쇼를 진행하면서 도널드 트럼프, 버락 오바마, 힐러리 클린턴, 김정은, 프랑스 대통령 마크롱을 비롯해 다수의 케베 및 캐나다 정치인 등을 두루 초청해 인터뷰했습니다.

이 프로그램의 대본은 풍자와 위트로 가득하며, 케베인의 시선을 반영합니다. 일례로, Laflaque가 미국 대통령 조지 W. 부시에게 미국의 투표인단 선거 제도에 관해 설명을 부탁하자 부시가 '간단합니다. 돈 많은 후보가 이깁니다.'라고 대답하는가 하면, 당시 캐나다 총리였던 스테판 하퍼와 어떤 문제를 두고 인터뷰할 때 Laflaque가 총리의 몸을 X-레이로 찍은 후 몸에 심장이 없는 것을 보여주는 식입니다.

이 밖에 FLAQ라고 하는 지역 주류판매점의 일상과 케베 지역 TV 게임쇼의 패러디도 단골 소재입니다.

## 역사

Gérard D Laflaque는 1982년 고무 인형으로 대중에게 첫선을 보였습니다. 당시 프랑스어로 송출되는 공영 방송국인 Radio-Québec에서는 'La minute et quart à Gérard D Laflaque[Gérard D Laflaque의 '75초 논평]'라는 제목으로 짤막한 꽁트를 내보냈습니다. Laflaque라는 이름은 케베의 유명 정치인 Gérard D. Lévesque에서 착안했습니다. "la flaque"은 불어로 "웅덩이"란 뜻입니다.

이 쇼의 컨셉과 Gérard D Laflaque는 수상 경력에 빛나는 케베의 정치 만화가 Serge Chapleau가 고안한 것입니다. 이 사람은 원래 텔레비전용 인형을 만들었는데, 이후 재능을 살려 이 분야에 맞는 꼭두각시(marionette)를 최초로 만들었습니다. Gérard D Laflaque 인형도 그중 하나입니다.



그림 4: Gérard D Laflaque 고무 인형

시간이 흘러 Laflaque는 Radio-Canada에서 'Et Dieu Crée...Laflaque(그리고 신은... Laflaque를 창조했다)'라는 이름으로 자기만의 30분짜리 쇼를 갖게 되었고, 나중에는 ICI Laflaque(의미: "여기 Laflaque입니다")로 이름을 변경하게 되었습니다. 이 쇼의 캐릭터 디자인은 모두 Chapleau의 드로잉 스타일을 기초로 하고 있습니다. Chapleau는 여전히 이 쇼의 콘텐츠 제작자로 활동 중이며 새 캐릭터의 모델 시트를 감독하고 새 대본을 승인하는 업무를 맡고 있습니다.

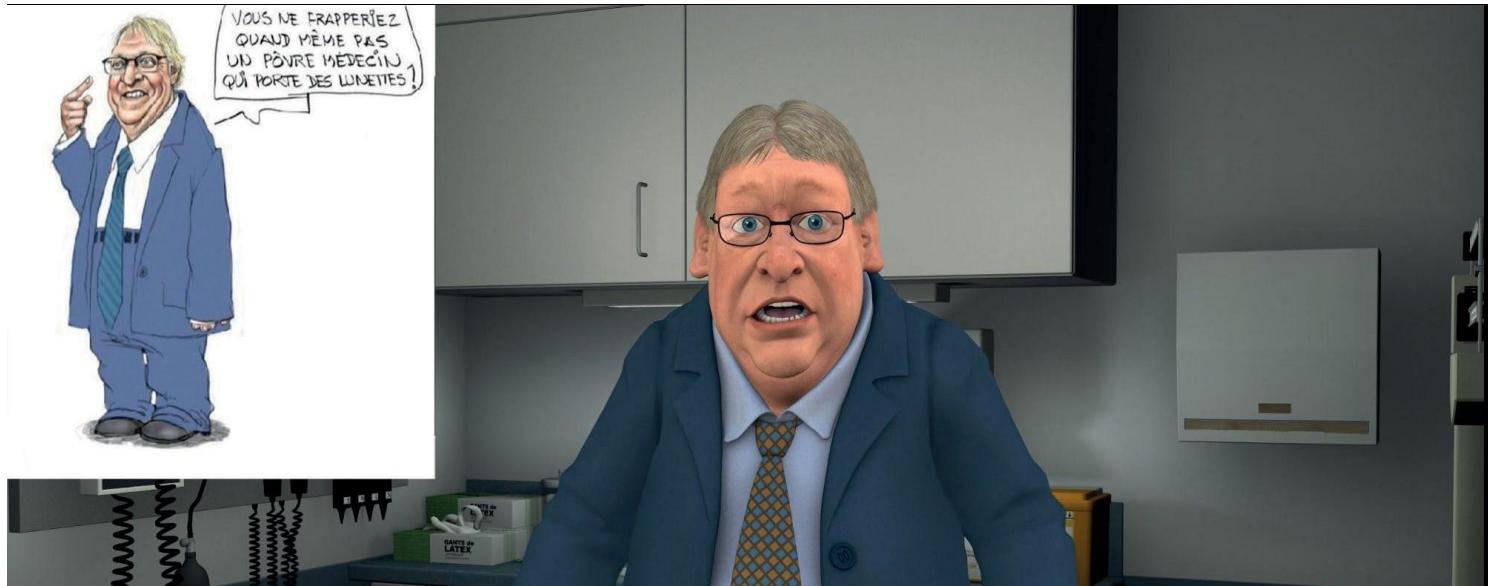


그림 5: Serge Chapleau가 그린 캐나다 정치인 Gaétan Barrette의 캐리커처

2004년 Laflaque 쇼를 처음 시작했을 때, 복스 포풀리는 모션 캡처와 MotionBuilder를 이용하는 파이프라인을 개발해 쇼를 애니메이션하고 렌더링했습니다. 실시간 렌더링에는 OpenGL(오픈 그래픽 라이브러리)을 이용했습니다. 프로덕션 팀은 제작 중에 실시간 프리뷰를 실시했으며, MotionBuilder에서 나오는 최종 결과물을 실시간 렌더링하였습니다.

이 파이프라인은 일주일 내내 매일 30분짜리 새 에피소드를 제작하는데 부족함이 없었습니다. 하지만, 복스 포풀리는 14년이 지난 시점에서 프로그램의 품질 향상을 위해 언리얼 엔진으로 교체하게 되었습니다.

## 제작 일정

2018년 Laflaque는 15년 차에 접어들었습니다. 프로덕션 팀은 이 30분짜리 쇼를 생성하기 위해 늘 빡빡한 일정에 쫓깁니다. 주중에 작가들이 며칠 전, 때론 몇 시간 전에 일어난 뉴스를 소재로 대본을 쓰면, 그것을 토대로 만들어진 제작물이 영상물에 담겨 며칠 후 Radio-Canada에 전달되는 방식입니다.

연간 제작 일정은 사용되는 파이프라인과 관계없이 다음 패턴을 따릅니다.

- 5월-8월: 에피소드는 제작되지 않습니다. 프로덕션 팀은 "시기를 타지 않는 주제"에 관해 대본을 쓰고 짧은 스케치를 생성합니다. 지구 온난화, 가족의 가치, 실업 등 예로부터 퀘벡과 전 세계에서 이슈가 되고 있는 문제가 그 대상입니다. 프로덕션 팀은 또 필요할지도 모를 새 세트와 캐릭터도 연구합니다. 새로 선출된 국가의 수반이나 새롭게 주목받고 있는 캐나다 정치인 등이 그 대상입니다.
- 9월-4월: 주간 에피소드의 제작 과정에서는 짧은 스케치가 중심이 됩니다. 한 에피소드에 들어가는 스케치 중에서 약 절반은 미리 제작해 둔 "아무 때나 방송할 수 있는 주제"이고 나머지 절반은 최신 뉴스를 소재로 하여 그 주에 제작된 것입니다. 한 시즌은 20-30 에피소드 정도로, 정확한 수치는 해마다 다릅니다.

프로덕션 팀은 2018년 9월 시작된 2018-2019 시즌에 맞춰 에피소드 26편과 캐릭터 100개, 50여 가지 배경, 1000여 개 소품과 스페셜로 구성된 언리얼 엔진 프로젝트를 만든다는 목표를 세웠습니다. 각 에피소드의 분량은 30분이며, 주 7일 제작되는 일정이었습니다.

매주 월요일부터 목요일까지는 제작 회의를 열어 대본에 대한 논의를 진행하고, 금요일까지 제작 여부를 결정합니다. 복잡한 3D 모델을 새로 만들어야 하는 동작은 그 모델을 빼고 다시 작성하기도 합니다. 비중이 작은 캐릭터를 새로 만들어야 한다면 팀은 기존 캐릭터를

리퍼포즈(repurpose)하거나 해당 캐릭터를 카드보드 컷아웃으로 표현할 수 있는지 의논합니다.

복스 포풀리 팀은 이 과정에서 불과 3일 이내에 완성된 스케치를 만들어 내야 하는 "대본 비상사태"를 맞기도 합니다. 이 쇼는 제작 기간이 짧아 이 같은 비상사태에 대처할 수 있는 파이프라인이 매우 중요합니다.



그림 6: 정치 평론가인 Mario Dumont가 대마초 합법화에 항의하여 종을 치고 있는 모습

1 복스 포풀리는 특정 스케치에 등장하는 특정 캐릭터에게 필요한 옷과 액세서리, 소품을 "스페셜"이라고 부릅니다. 자주 등장하는 캐릭터에 대해서는 옷, 모자, 안경 등은 물론 펜, 전화기, 클립보드, 서류 가방 같은 소품까지 아이템을 많이 만들어 놓고 골라서 사용합니다. 이 아이템에는 대부분 자체 릭이 있습니다. 캐릭터에 스페셜을 적용한다고 하면 액세서리와 손 소품 일체를 포함해 캐릭터의 스케치를 완벽하게 꾸민다는 뜻입니다.

# 언리얼 엔진의 도입

# 언리얼 엔진의 도입

복스 포풀리는 몇 년 전부터 언리얼 엔진을 실시간 렌더링 솔루션을 눈여겨 보고 있었습니다. 그러던 도중, [The Mill's Blackbird](#) 프로젝트와 [Rogue One](#)의 최종 렌더링 과정에서 언리얼 엔진을 활용하는 ILM의 모습에 매료되었습니다.

2018년 복스 포풀리는 에픽게임즈와 함께 언리얼 엔진 워크플로로 기존의 실시간 솔루션을 대체하고, 쇼의 비주얼 퀄리티를 높이는 방안을 검토했습니다.

복스 포풀리는 2단계로 나눠 언리얼 엔진 파이프라인의 도입을 계획했습니다.

- 1단계: 모션 캡처와 키프레임 데이터(또는 애니메이션 데이터만)를 MotionBuilder로 보내고 언리얼 엔진에서 렌더링
- 2단계: MotionBuilder LiveLink 플러그인을 통해 언리얼 엔진에서 모션 캡처 데이터를 라이브 피드백

본 문서에서는 1단계를 설명합니다. 2단계는 추후 실시될 예정입니다.

프로덕션 팀은 1단계에서 쇼의 전체적인 모습은 그대로 둔 채 비주얼 퀄리티를 개선하고자 했습니다. 시청자에게 같은 쇼라는 느낌을 주면서도 더 좋은 화질을 선사하겠다는 것 이었습니다. 그러면 프로덕션 팀은 언리얼 엔진을 통해 PBR(physically-based rendering, 물리 기반 렌더링) 텍스처를 이용할 수 있게 됩니다.

이 텍스처는 실시간 라이팅과 동작되도록 최적화돼 있습니다. 따라서, 렌더링을 하면 색이 더 풍성하고 보기에도 더 선명하고 부드럽습니다.

복스 포풀리는 이 밖에도 쇼의 전반적인 진행 과정이 R&D로 인해 지장을 받는 일이 없도록 기존 워크플로의 변경도 최소화해야 했습니다. 그래서 기존에 사용하던 실시간 렌더링 프로세스 부분만 대체할 수 있는 기본 UE4 파이프라인을 개발하는 것을 목표로 세웠습니다.

복스 포풀리는 새 파이프라인의 내구성 확보를 위해 구 워크플로와 새 워크플로로 함께 제작하는 환경을 만들었습니다.

최종 결과물의 비교는 본 문서 렌더링 단원에서 설명합니다.

## 기존 파이프라인

복스 포풀리는 기존 파이프라인에서도 모델링, 모션 캡처, 애니메이션 등 몇 가지 단계에서 효율을 극대화해 둔 상태였습니다.

프로덕션 팀은 쇼가 탄생한 후부터 줄곧 캐릭터와 세트를 ZBrush와 3dx Max로 모델링하고, 3dx Max로 기본 리깅을 설치했습니다. 또한 모션 캡처 단계에서는 PhaseSpace 시스템을 활용하고, MotionBuilder에서 리깅과 애니메이션을 마무리했습니다. 파이프라인에서 이 부분을 바꿔 언리얼 엔진으로 실시간 렌더링을 할 필요는 없었습니다. 프로덕션 팀이 이미 이 워크플로에 익숙한 상태였기에 언리얼 엔진으로 전환하더라도 해당 업무의 실행에 걸리는 시간에는 영향이 없을 터였습니다.

프로덕션 팀은 전부터 프리뷰와 최종 픽셀에 MotionBuilder OpenGL 결과물을 이용하고 있었습니다. 프리뷰는 30fps로 실시간 플레이백이 가능했고 최종 결과물은 5-10fps라는 실시간 미만의 속도에서만 생성이 가능했습니다. 복스 포풀리는 언리얼 엔진을 통해 프리뷰의 퀄리티를 높이고 최종 결과물의 속도도 개선하고자 했습니다.

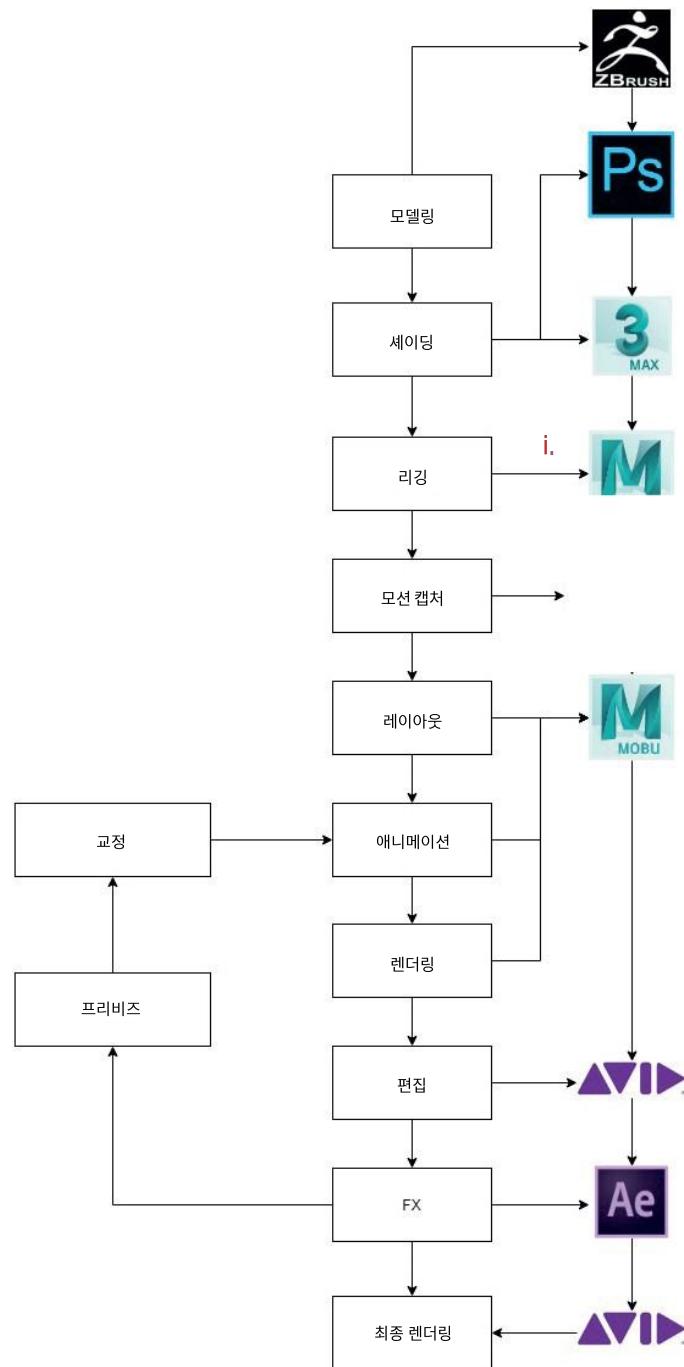


그림 7: 언리얼 MotionBuilder 파이프라인

## 언리얼 엔진 파이프라인

프로덕션 팀은 첫 단계에서 파이프라인을 다음과 같이 변경하고자 했습니다.

**셰이딩** - 셰이딩은 UE4에서 실시. 기존 파이프라인에서는 텍스처를 모두 포토샵에서 채색하거나 수정하여 완성했고, 머티리얼은 3dx Max와 MotionBuilder에서 할당했습니다. 프로덕션 팀은 언리얼 엔진 머티리얼을 최대한 많이 사용하여 이 프로세스를 간소화하고자 했습니다.

**리깅** - MotionBuilder의 리깅 공정은 최대한 그대로 두고, 언리얼 엔진으로 임포트할 때 몇 가지 변화를 줘 효율을 높이고자 했습니다.

**라이팅** - 렌더링이 UE4에서 실행될 것이기에 라이팅도 모두 UE4에서 하는 것이 이치에 맞았습니다.

**VFX** - 쇼의 기존 파이프라인에서는 VFX가 After Effects에서 별도로 생성되었습니다. 언리얼 엔진에서는 VFX를 애니메이션과 동시에 할 수도 있습니다.

**합성** - 언리얼 엔진에서는 시퀀서를 이용해 기존 파이프라인의 합성 단계를 엔진 안에서 직접 할 수도 있습니다.

**렌더링** - 애니메이션 프리뷰는 MotionBuilder에서 계속 진행하지만 최종 렌더링은 언리얼 엔진에서 하게 됩니다.

프로덕션 팀은 또 UE4에서 스페셜(specials)과 캐릭터도 쉽게 바꾸고 시퀀서 카메라 옵션에 임포트할 때 MotionBuilder의 카메라 스위처 툴(Camera Switcher tool)도 이용하고 싶었습니다.

언리얼 엔진 파이프라인은 모든 프로덕션 단계 간에 병렬 워크플로가 더 많아서 기존 파이프라인과는 꽤 다른 모습입니다.

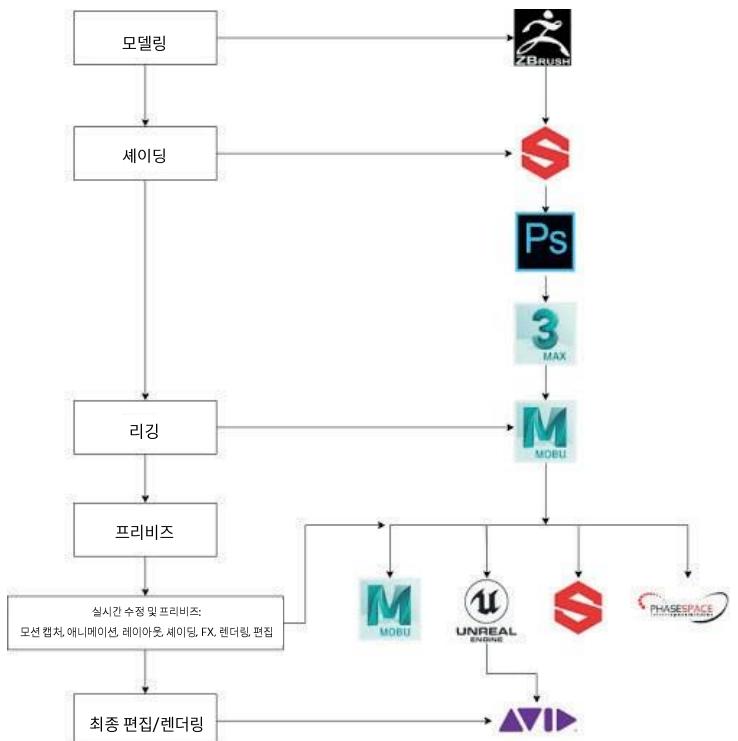


그림 8: 실시간 UE4 파이프라인

그 외에 Laflaque 팀이 언리얼 엔진 파이프라인으로 전환하면서 느낀 장점은 다음과 같습니다.

- 복스 포풀리는 기존 MotionBuilder 애셋을 크게 바꾸지 않고 사용할 수 있었습니다.
- 기존 파이프라인에서는 MotionBuilder의 실시간 프리뷰 기능을 이용해 첫 단계의 애니메이션을 리뷰했습니다. 언리얼 엔진에서는 실시간 GPU 시스템으로 리뷰가 가능해 프리뷰의 퀄리티가 훨씬 더 높습니다.
- Perforce를 통합 툴로 이용해 애셋을 관리하고 팀 안에서 작업을 나눌 수 있었습니다.

## 사용 툴

### 기존 파이프라인

Autodesk® [3dx Max®](#) - 모델링, UV, 옷과 얼굴 애니메이션을 위한 모프 타겟, 리깅, 스키닝

Pixologic® [ZBrush®](#) - 캐릭터 스컬핑(sculpting)

Adobe® [Photoshop®](#) - 텍스처 채색

Autodesk [MotionBuilder®](#) - 모션 캡처 데이터 처리와 클래식 애니메이션, 실시간 프리뷰와 출력

[PhaseSpace® Motion Capture](#) - 몸동작 모션 캡처

[Adobe After Effects®](#) - VFX, 후처리, 인포그래픽

[Avid® Media Composer®](#) - 사운드, 최종 편집, 크래딧, 결과물

### 언리얼 엔진 파이프라인

기존 파이프라인의 툴 대부분이 실시간 파이프라인에서도 계속 사용되고 있지만 몇 가지는 언리얼 엔진의 효율 극대화를 위해 추가되거나 교체된 것입니다. 언리얼 엔진에 새로 추가된 것, 새 파이프라인에서 기능이 달라진 것은 아래와 같습니다.

- Adobe Photoshop, [Allegorithmic Substance Painter and Designer](#) - 머터리얼 채색 및 작성
- Autodesk MotionBuilder - 모션 캡처 데이터 처리와 클래식 애니메이션
- After Effects - VFX, 인포그래픽
- 언리얼 엔진 - 프리뷰, 포스트 프로세싱, 최종 출력

# 프로덕션 단계

## 프로덕션 단계

새 파이프라인을 유지하는 단계도 있지만 크게 바뀐 단계도 있습니다.

다음 표는 새 언리얼 엔진 파이프라인에 맞춰 기존 파이프라인을 최적화하는 데 필요한 변경사항을 요약한 것입니다.

단계	기존 파이프라인	언리얼 엔진 파이프라인
모션 캡처	MotionBuilder로 PhaseSpace	<변경 사항 없음>
레이아웃	MotionBuilder	MotionBuilder 및 언리얼 엔진
모델링	ZBrush 및 3dx Max	<변경 사항 없음>
셰이딩	3dx Max 및 MotionBuilder	UE의 섹스턴스 페인터
리깅	3dx Max 및 MotionBuilder	<변경 사항 없음>
애니메이션	MotionBuilder	몸/얼굴 애니메이션 프로세스 변경 없음 텍스처 스위칭 UE4에서 실시
언리얼 엔진으로 이전	<N/A>	FBX
라이팅	MotionBuilder	언리얼 엔진
렌더링	MotionBuilder OpenGL	언리얼 엔진
합성과 VFX	After Effects	After Effects 및 언리얼 엔진
편집과 전달	Avid Media Composer	<변경 사항 없음>

표1: 파이프라인 변경사항 요약

## 모션 캡처



복스 포풀리는 PhaseSpace를 Laflaque 쇼의 모션캡처 시스템으로 이용합니다. 이 시스템에는 액티브 마커가 장착된 34 Fujifilm X-E2 모션 캡처가 내장돼 있습니다.

이 시스템을 선택한 이유는 낮은 비용과 정확도의 균형입니다. Laflaque 쇼에는 미묘한 움직임보다 큰 동작이 많이 나옵니다. 캡처 데이터에 MotionBuilder에서 바로잡아 줘야 하는 노이즈가 약간 들어 있긴 하지만, 시스템 앰플라이(amply)에 제작 시 필요한 사항이 들어 있습니다.

## 레이아웃

기존 파이프라인에서는 모션 캡처 세션 중에 모션 데이터가 MotionBuilder로 스트리밍되고 여기서 데이터가 모션 레이아웃 씬(여기서 실제 최종 환경 대부분이 들어 있음)의 캐릭터에 적용되었습니다. 이로써 디렉터는 모든 것을 컨텍스트에서 리뷰할 수 있었습니다.

디렉터는 이어서 이 MotionBuilder 씬에서 액션을 검토하고 카메라를 세팅한 후 카메라 스위처로 샷을 구분했습니다. 카메라를 설치한 후 남은 애셋(액세서리)은 씬에 추가했습니다.

제작을 할 때에는 MotionBuilder가 필수였습니다. 애니메이션이 모두 MotionBuilder에서 진행되었기 때문입니다. 이 같은 이유로 프로덕션 팀은 MotionBuilder의 카메라를 UE4의 카메라로 정밀하게 변환할 수 있어야 했습니다.

프로덕션팀은 카메라 임포트 툴과 일부 커스텀 조정을 이용해 MotionBuilder 카메라와의 유사성을 유지했습니다.

## 카메라 데이터 임포트

MotionBuilder에서 카메라를 만들 때는 point of interest(관심 지점, look-at-point라고도 함)이 자동으로 생성됩니다. 이 점(카메라에 부착되는 널(null))이 카메라의 회전과 시야각(FOV)을 결정합니다.

FBX를 UE4로 익스포트/임포트할 때, 관심 지점은 변환되지 않습니다. 즉 카메라의 회전이 사라진다는 뜻입니다.

해결방법은 익스포트 전에 MotionBuilder에서 커스텀 Python 스크립트로 각 카메라의 회전을 플롯(plot)하고, 초첨 거리 값(focus distance value)을 유지하기 위해서 UE4에서 커스텀 FBX 프로퍼티에 세팅했습니다.

- In 에디터 레벨 시퀀스 / 트랙 설정 / FBX 설정
  - Add --- FBX 속성: FocusDistance
  - Add --- 컴포넌트 이름: CameraComponent
  - Add --- 속성 이름: FocusSettings, ManualFocusDistance

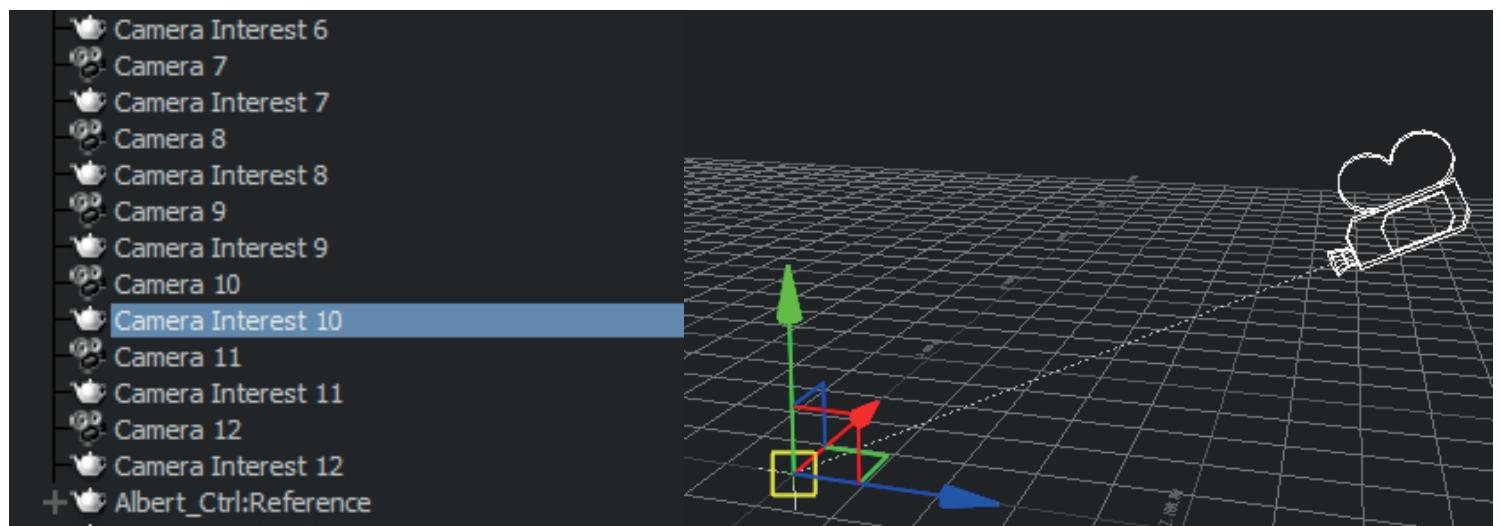


그림 10: MotionBuilder의 카메라 인터레스트

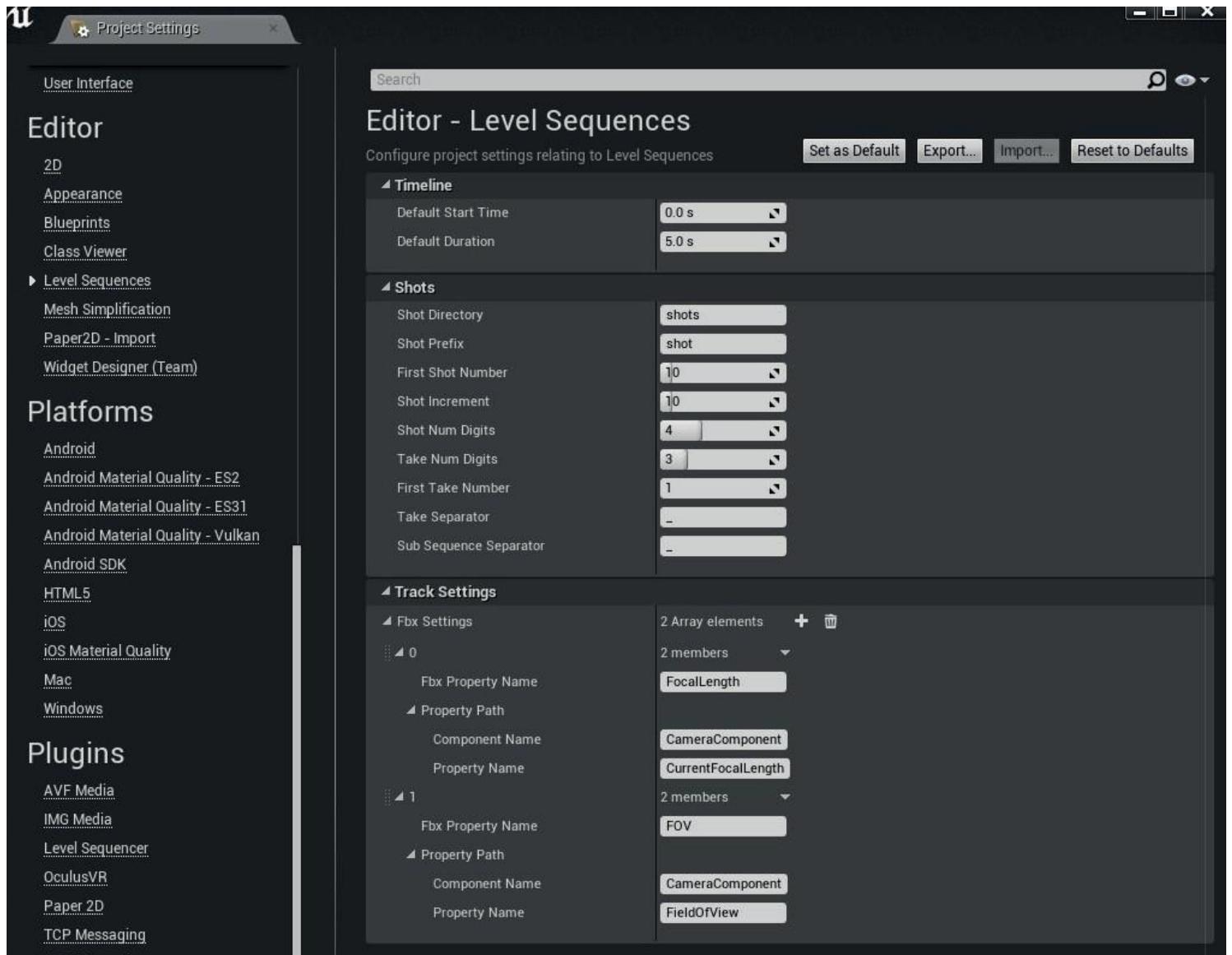


그림 II: FBX 익스포트 설정

이 설정에서는 MotionBuilder의 카메라 매개변수에서 초점 거리가 추출돼 UE 시퀀서로 임포트됩니다.

## 카메라 스위처

MotionBuilder의 카메라 스위처와 스토리 툴은 기존 파이프라인에서 중요한 요소였습니다. 새 파이프라인을 전환할 때에는 UE4의 카메라 컷 기능과 호환되도록 MotionBuilder의 Camera Switcher 데이터를 변환하는 것이 무엇보다 중요했습니다.

언리얼 엔진 개발팀은 MotionBuilder 카메라 스위처 데이터를 사용하고 시퀀서 카메라 컷 트랙의 데이터를 업데이트하는 옵션을 통합했습니다. 언리얼 엔진이 MotionBuilder의 카메라 ID를 인식하도록 함으로써 팀은 시퀀서가 MotionBuilder의 카메라 스위처를 사용하는 패키지 사이의 카메라 세팅을 맞출 수 있도록 할 수 있습니다. MotionBuilder의 카메라 스위처에서 나온 시퀀스 컷을 유지하면 언리얼 엔진의 시퀀서 툴에서 샷을 모두 재구성하기도 용이합니다.



그림 12: MotionBuilder의 카메라 스위처

## 모델링

프로덕션 팀은 ZBrush와 3dx Max로 캐릭터를 모델링한 후 3dx Max로 치아와 옷 같은 신체의 각 부분을 분리합니다. 각 캐릭터는 트라이앵글 약 16만 개로 구성됩니다.

세트의 경우, Gérard가 인터뷰를 진행하는 ICI Laflaque 스튜디오를 메인 세트로 합니다. 그 외 스케치에서는 주류 판매점, 식료품점, 사무실, 웨이브시 의사당과 같은 내부와 공원, 주차장, 시내, 뒷골목 등 외부를 이용합니다. 이 세트의 상당수는 쇼가 거듭되면서 앞서 제작된 것입니다.

새 파이프라인에서도 모델은 종전의 파이프라인과 같은 방식으로 생성합니다. 캐리커처 드로잉(모델 시트)으로 ZBrush 버전을 만들고, 이어서 3dx Max에서 모델을 마무리합니다.

연중 수시로 새 캐릭터가 제작되며, 프로덕션팀은 기존 요소, 특히 옷과 손을 최대한 많이 재사용 할 수 있도록 합니다. 기존 캐릭터는 새 헤어스타일과 텍스처, 소품 등을 추가해 업데이트합니다.

## 셰이딩

기존 파이프라인에서는 포토샵으로 만들거나 수정한 텍스처를 3dx Max와 MotionBuilder에서 모델에 할당했습니다. 프로덕션팀은 언리얼 엔진 파이프라인으로 업그레이드하면서 Allegorithmic Substance Tools로 바꿨습니다.

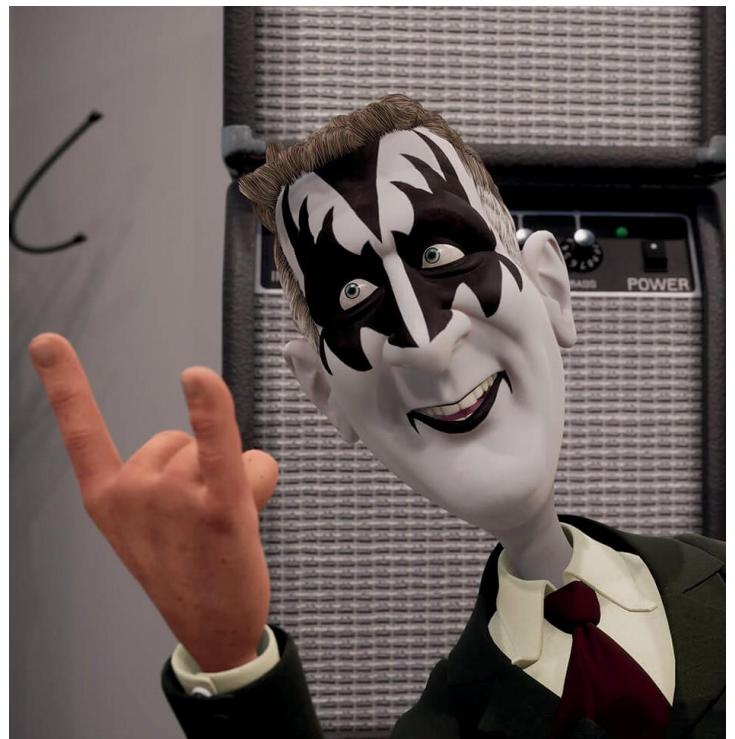


그림 13: Allegorithmic Substance Tools로 만든 텍스처

프로덕션팀은 섭스턴스 덕분에 분리된 텍스처가 아니라 PBR 머티리얼 전체를 대상으로 작업을 할 수 있게 되었습니다. 머티리얼 대부분은 기존 파이프라인에는 없던 것들입니다. 섭스턴스 머티리얼은 이어서 언리얼 엔진에서 사용되는 개별 텍스처로 익스포트됩니다.

프로덕션팀은 첫 시험에서 일부 텍스처를 언리얼 엔진의 포맷에 맞게 재가공할 필요가 있음을 확인했습니다. 파일 타입과 해상도, 새 세이딩의 가능성도 감안해야 했습니다.

창, 안경 렌즈와 같은 투명 물체는 처음에 굴절이 있는 반투명 셰이더로 만들었지만 굴절 때문에 렌더링에 문제가 일어났습니다. 이에 프로덕션팀은 세이딩 트릭을 이용해 굴절 효과를 에뮬레이션했습니다.

또 일부 애니메이션에 대해 언리얼 엔진의 텍스처 스위칭 기능을 이용했습니다. 이것은 본 문서의 애니메이션 단원에서 설명합니다.

## 리깅

캐릭터는 3dx Max에서 리깅, 스키닝한 후 MotionBuilder로 익스포트합니다.

캐릭터의 몸은 뼈와 더미 오브젝트(Nulls)를 결합해 리깅합니다. 캐릭터의 머리에는 50개의 모프 타겟 형상이 있습니다. 액세서리 일부에도 애니메이션이 수월하도록 모프 타켓 형상이 있습니다.

3dx Max에서 MotionBuilder로 익스포트된 캐릭터의 FBX들에는 지오메트리와 MeshSmooth, Morpher, Skin 모디파이어가 들어 있습니다. MotionBuilder에서는 스켈레톤 릭(rig) 이 기본 컨트롤릭입니다.



그림 14: UE4의 Laflaque 캐릭터 세이딩

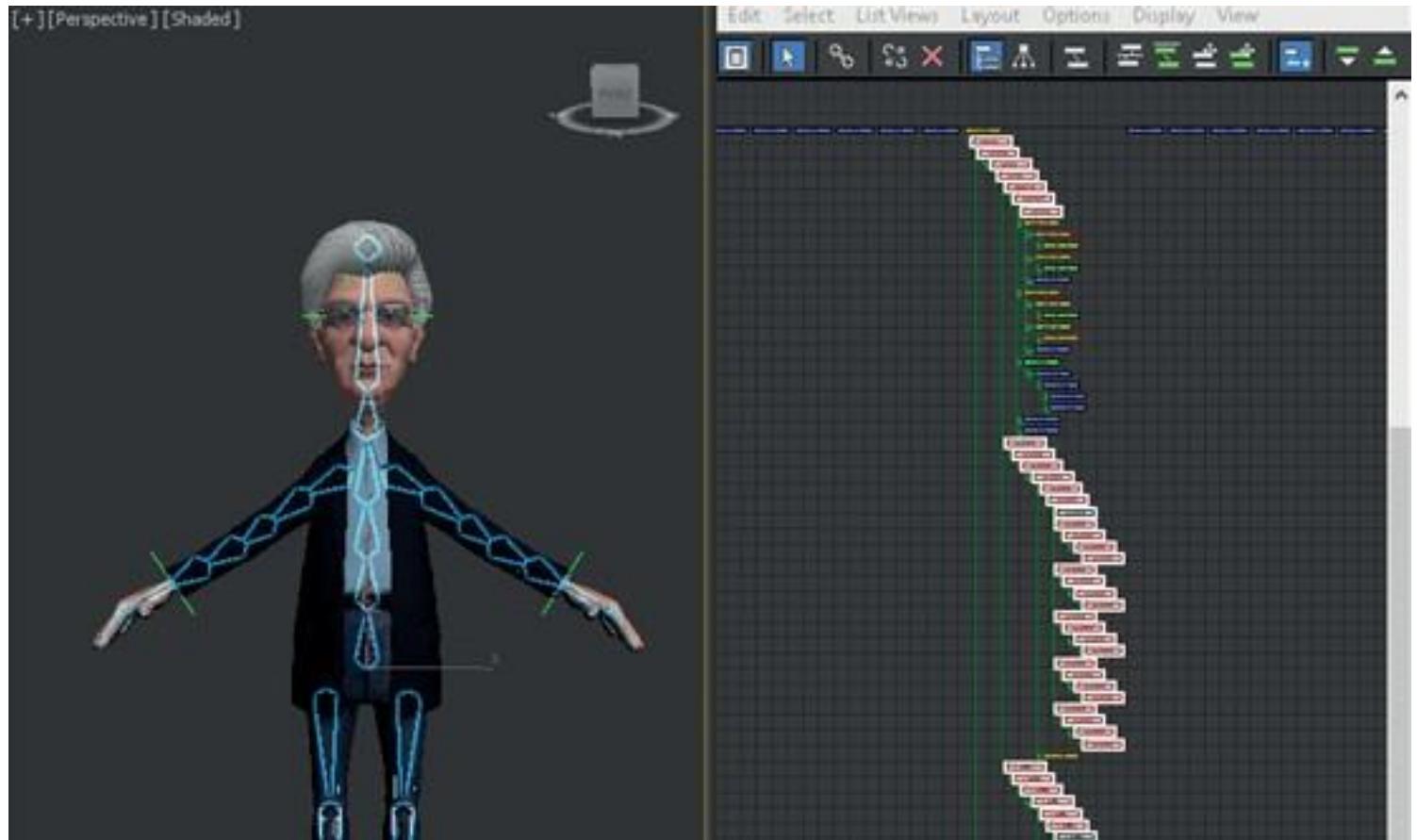


그림 15: 3dx Max의 Laflaque 캐릭터 계층

캐릭터 얼굴 릭에는 형상과 클러스터가 포함되어 있습니다. 필요에 따라 MotionBuilder에서 마커를 추가해 2차 애니메이션(IK 채널 전용 이펙터 등)을 컨트롤합니다. 눈과 웃, 그 외 부수적 요소의 애니메이션에 대해서도 몇 가지 제약을 추가합니다.

각 캐릭터는 동일한 계층과 명명법을 이용하지만, 각자 고유의 MotionBuilder Namespace가 있습니다.

## 추가 골격

Laflaque 캐릭터들의 두드러진 특징 가운데 하나는 기본 캐릭터 템플릿에 추가할 수 있는 수천 가지 스페셜(소품과 액세서리)입니다. 이 스페셜의 이차 동작(Secondary motion)이 Laflaque 캐릭터의 개성이자 유머가 됩니다.

한 가지 예가 Pierre 캐릭터에 추가된 터번과 수염입니다. 이 스페셜들에게 추가된 골격은 캐릭터에게 이전되며 이로써 이차 동작이 가능합니다.



그림 16: Pire 캐릭터 베이스와 수염/모자 스페셜, 애니메이션을 위한 추가 골격

15년 동안 쇼의 CGI가 발달하면서 수백 가지 버전의 캐릭터가 만들어졌고 수천 가지 스페셜이 애니메이션 제작을 위해 추가됐습니다. 또한, 한 가지 액세서리나 소품에 여러 버전이 있을 수도 있습니다.

클로즈업에는 소품(prop)의 하이 폴리(high poly) 버전이, 원거리샷이나 빠른 동작에는 로우 폴리 버전이 개발되겠지만, 각 버전의 릭이 맞지 않을 가능성도 있습니다. 애니메이션과 렌더링에 쓰이는 애플리케이션이 MotionBuilder 밖에 없을 때에는 이 같은 변형도 큰 문제가 되지 않았습니다. 그러나 새 UE 파이프라인에서, 그리고 여러 가지 스켈레톤 릭을 관리해야 할 때에는 사정이 달랐습니다. 캐릭터와 스페셜에게

각자 릭이 있다면 UE로 익스포트하기 전에 이 두 릭을 캐릭터 릭과 연결해야 했습니다.

복스 포풀리는 처음에 캐릭터와 소품(prop) 계층을 따로 임포트한 후 언리얼 엔진의 Set Master Pose Component(세트 마스터 포즈 컴포넌트)와 블루프린트를 이용해 캐릭터와 소품(prop)의 계층 간에 부모와 자식(parent/child) 관계를 설정하는 방식을 취했습니다.

그러나 추가 골격이 무효가 되면서 블루프린트에서 세트 마스터 포즈와의 연결이 별다른 효과를 보지 못했습니다. 세트 마스터 포즈 컴포넌트를 이용하려면 스켈레탈 메시 전체에서 같은 계층의 골격이 필요했습니다.

프로덕션은 MotionBuilder에서 나온 스켈레톤(골격이 추가된 것)을 애니메이션과 함께 다시 익스포트한 후 그것을 새 스켈레탈 메시로 임포트해야 합니다.

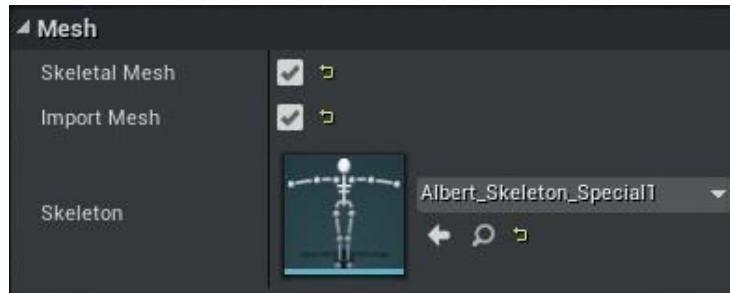


그림 17: FBX 임포트 설정에서 스켈레톤 선택

## FBX 익스포트 문제

옛 애셋을 새 애셋과 결합하자 MotionBuilder에서 UE로 FBX를 전송할 때 문제가 나타났습니다. 때로는 UE에서 임포트에 관한 오류를 보고하기도 하고 팔이 나머지 계층과 분리된 듯 보이기도 했습니다. 복스 포풀리는 DCC에서 익스포트하기 전에 각 캐릭터를 T-포즈에 넣거나 모든 노드에서 TRS 키프레임을 설정하는 방법으로 이 문제를 해결했습니다.

MotionBuilder에서는 T-포즈를 수동으로 설정하는 방법이 최선이었습니다. 경우에 따라서는 UE4 임포터 툴의 Use TOAs Ref Pose 옵션으로도 이 문제를 자동으로 해결할 수 있습니다.

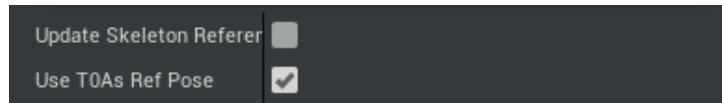


그림 18: FBX 임포트 설정에서 T 포즈 사용 선택



그림 19: MotionBuilder의 캐릭터 골격 계층

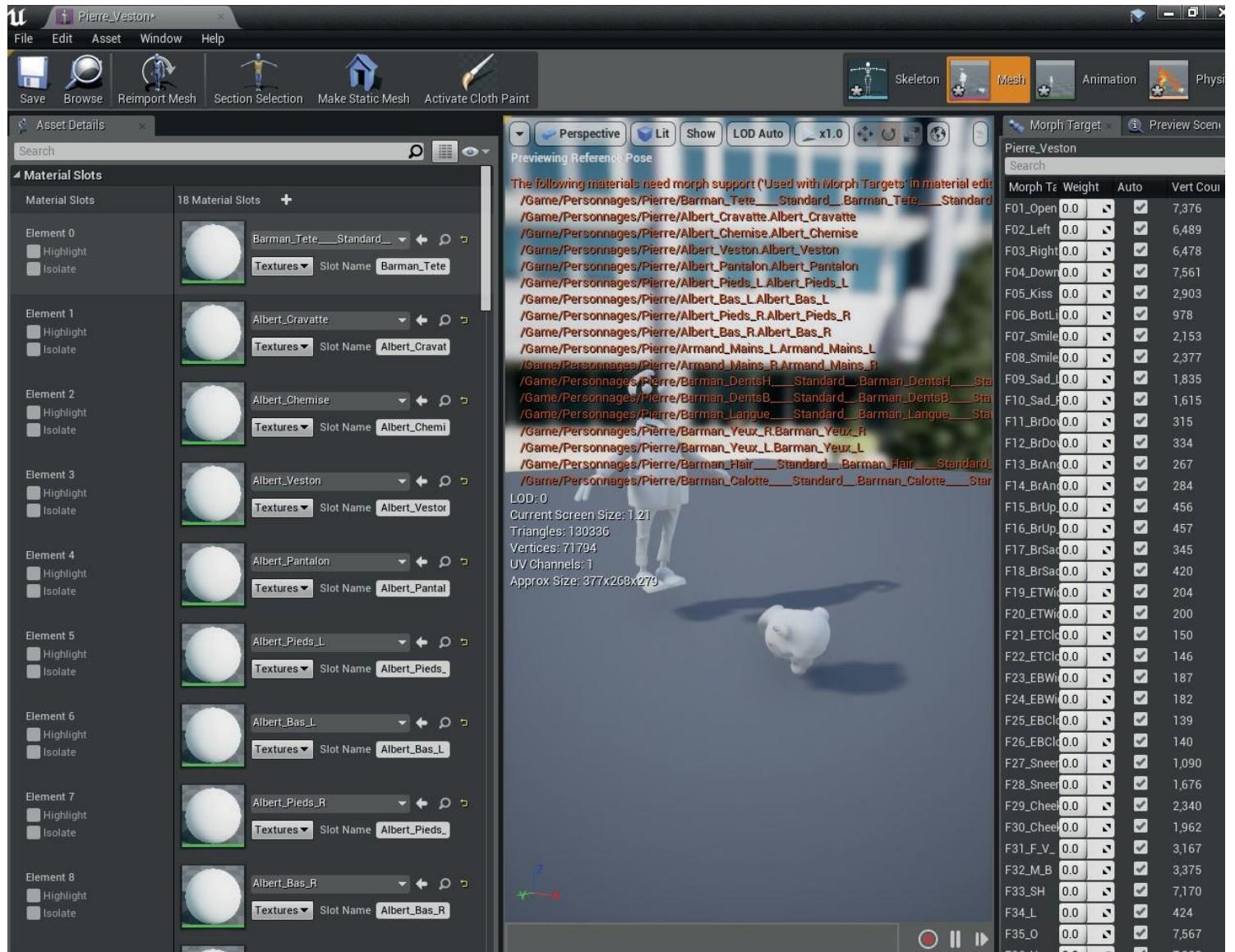


그림 20: 골격 계층의 오류. 옛 몸통에 부착된 머리

## 구성과 명명의 문제

종전의 MotionBuilder 파일라인에서는 캐릭터마다 각자 네임스페이스가 있었지만 서로 다른 계층에 대한 서브레퍼런스가 둘 이상일 때도 있었습니다. 예를 들어, 이름이 Foule라는 캐릭터인 경우 참조 1과 참조 2라는 두 개의 서로 다른 스켈레탈 계층 구조가 있는 아래 그림과 같은 네임 스페이스가 있을 수 있습니다.



그림 21: 네임스페이스가 있는 계층 아이템

이렇게 캐릭터와 스켈레톤을 나누는 방법은 다음 두 가지 이유로 언리얼 엔진에서 제대로 구현되지 않았습니다.

- MotionBuilder는 네임스페이스 이름과 레퍼런스 이름 사이에 콜론을 넣어 내부에 위와 같은 서브레퍼런스를 저장합니다. 예컨대, 위 이미지의 첫 서브레퍼런스가 Foule:Reference 1로 저장된다고 하면 이 명명 방식은 FBX 형식으로 언리얼 엔진에 익스포트할 때 유효하지 않습니다.
- MotionBuilder의 FBX 파일이 임포트될 때, 언리얼 엔진은 전체 네임스페이스를 한 스켈레톤으로 해석합니다. 즉, 네임스페이스에 스켈레톤이 둘 이상 들어 있으면 이들은 FBX 임포트 후 하나의 계층으로 결합됩니다.

스켈레톤을 쓴 레벨 바로 아래 서브레벨로 설정하면 더 좋습니다. 언리얼 엔진은 이 계층을 별도의 스켈레톤으로 읽습니다.

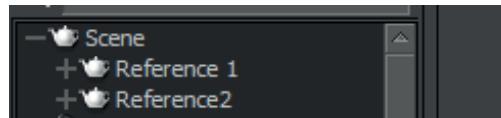


그림 22: 계층 분화된 두 스켈레톤 애셋

## 애니메이션

새 파일라인에서는 애니메이션 전 단계에서 MotionBuilder를 이용합니다. 모캡 애니메이션은 정정 후 FBX 포맷으로 저장돼 언리얼 엔진으로 임포트됩니다. 애니메이션 데이터는 언리얼 엔진에 들어오면 시퀀서 트랙에서 바로 캐릭터에 임베드(embedded) 됩니다.

캐릭터의 몸통은 MotionBuilder에서 모션 캡처 데이터로 애니메이션되고, 얼굴과 손은 수작업으로 애니메이션 합니다. 소품은 수작업으로 애니메이션하거나 모캡을 이용합니다.

한 씬에는 30fps로 200~600프레임이 들어가는데, 대개 씬당 애니메이터 한 명이 애니메이션을 맡습니다.

## 텍스처 스위칭을 통한 애니메이션

일부 샷에서는 샷 전체를 대상으로 또는 애니메이션 시퀀스에서 반복해서 텍스처의 스위칭 아웃이 필요했습니다. Laflaque 팀은 MotionBuilder에서 그런 작업을 수행하는 워크플로가 있었지만 언리얼 엔진에서 이 기법을 시행하고자 했습니다.

## 피부색

피부색의 변화를 표현해야 할 때(예컨대 불에 데어 피부가 검게 변할 때) 기존 파이프라인에서는 MotionBuilder에서 커스텀 슬라이더가 달린 텍스처 블렌딩을 이용했습니다.



그림 23: MotionBuilder의 텍스처 스위처

파이프라인을 변경하면서 언리얼 엔진에서도 비슷한 워크플로가 필요했습니다. Laflaque 팀은 다음과 같은 순서로 머티리얼 매개변수 기능을 이용했습니다.

- 머티리얼 텍스처/머티리얼 매개변수 컬렉션에서 새 머티리얼 매개변수 애셋 생성.

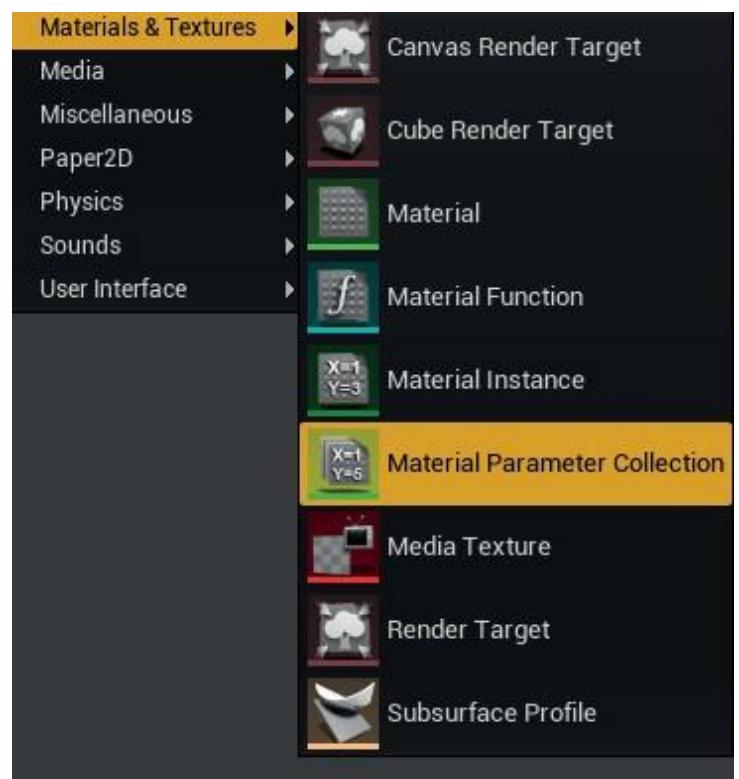


그림 24: 콘텐츠 브라우저에서 머티리얼 매개변수 컬렉션 선택

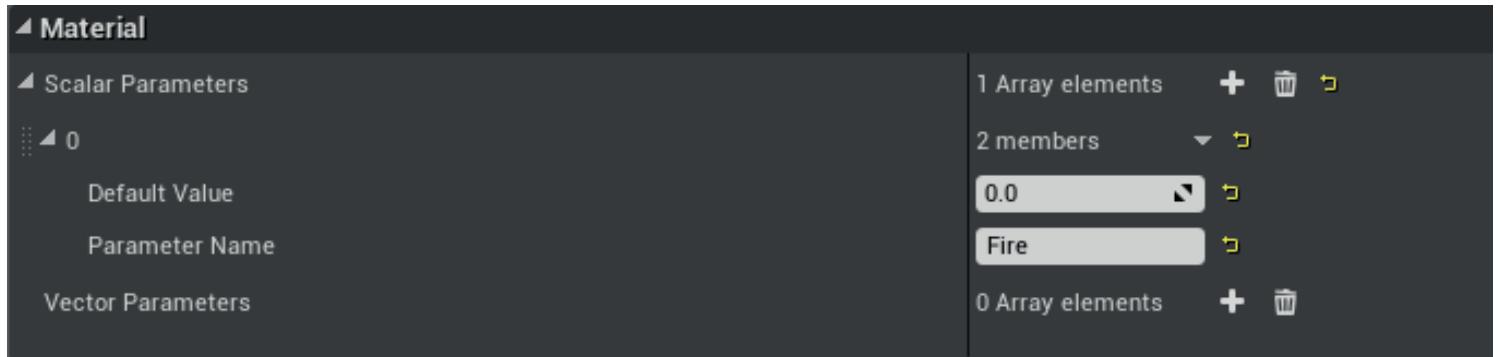


그림 25: 스칼라 매개변수 또는 벡터 매개변수 옆 기호 클릭

- 적당한 변수 이름과 함께 스칼라값 추가
- 해당 머티리얼 에디터로 이동
- 컬렉션 매개변수를 추가한 후 머티리얼 매개변수 컬렉션과 변수 선택

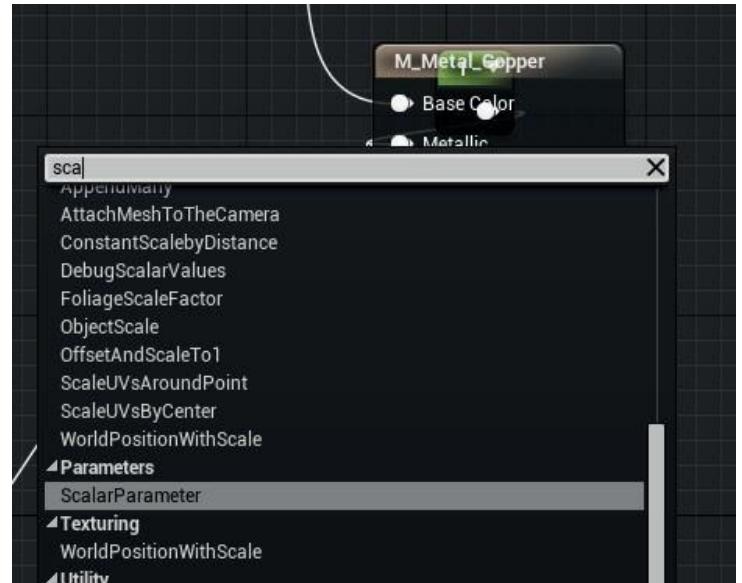
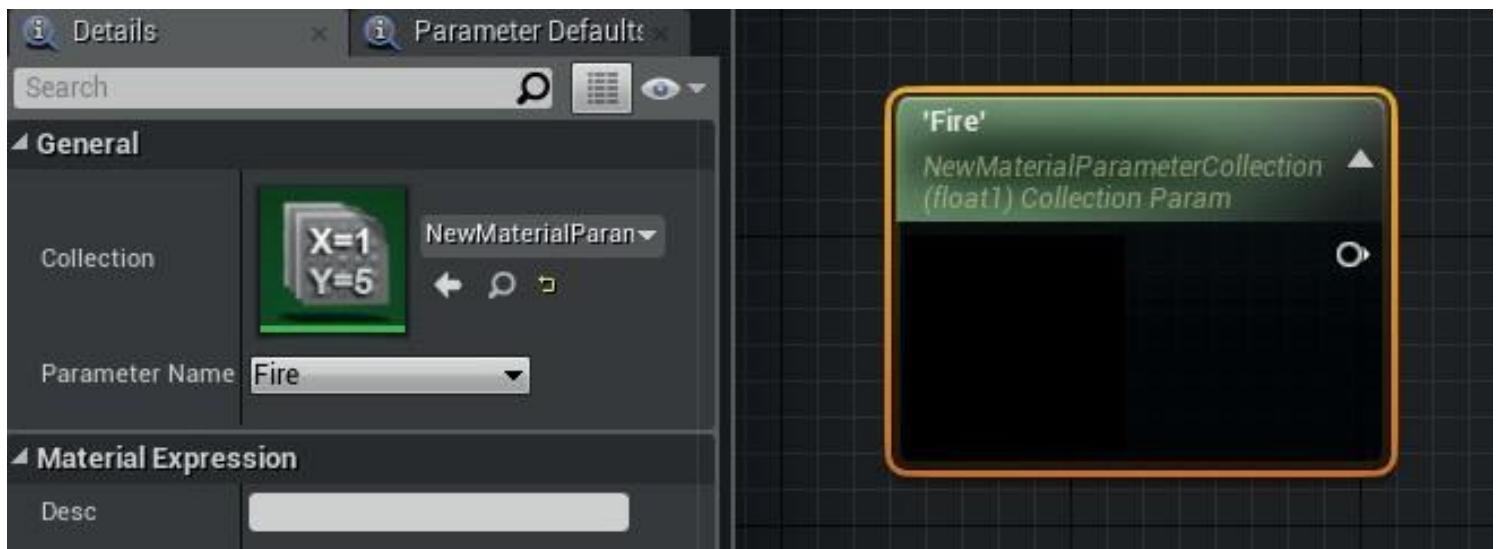


그림 26: 머티리얼 그래프의 스칼라 매개변수 머티리얼 표시 노드

- 선형 보간(Lerp) 노드를 만든 후 컬렉션 매개변수 노드를 알파에 추가. 원하는 블렌드 방법에 따라 Multiply(멀티플라이)나 그 외 노드를 써도 됩니다.



- 시퀀서에서 머티리얼 파라미터와 변수를 추가.

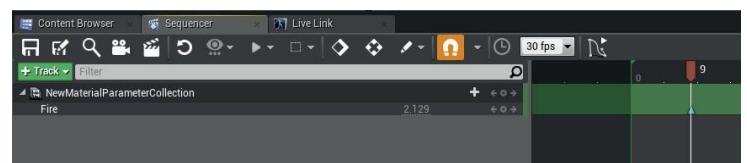
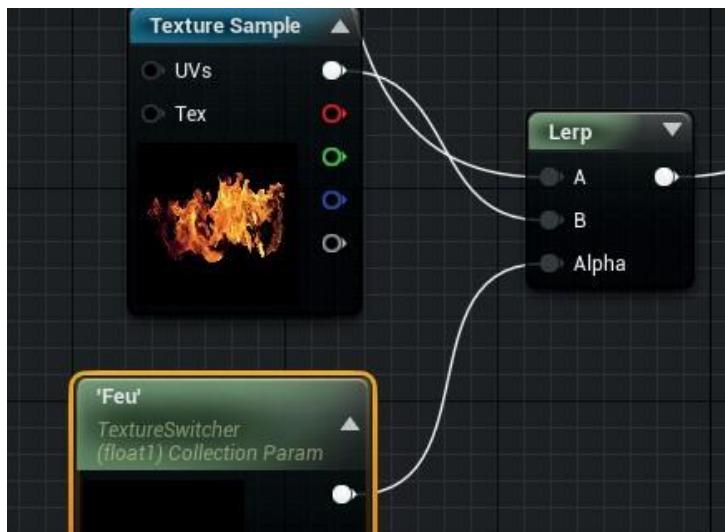


그림 28: 머티리얼 파라미터 컬렉션

그림 27: 머티리얼 그래프의 머티리얼 표시 매개변수 노드

## 얼굴 애니메이션

프로덕션팀은 시퀀스에 따라 텍스처를 스위칭하여 얼굴 표정과 대화를 애니메이션하기도 합니다. 프로덕션팀은 새 언리얼 엔진 파이프라인에서 이런 유형의 애니메이션을 처리하는 방법으로 종전의 파이프라인과 마찬가지로 MotionBuilder에 텍스처 스위칭을 설정하고 언리얼 엔진에서 블루프린트로 다시 설정한 후 애니메이션 곡선만 임포트하는 방식을 택했습니다. 그 결과 익숙한 워크플로를 그대로 유지하면서도 언리얼 엔진의 미세 조정에 애니메이션을 계속 이용할 수 있었습니다.

MotionBuilder에서는 이 같은 텍스처 스위칭을 설정하기 위해 80여 개 텍스처로 된 레이어 방식 텍스처를 머리에 적용합니다. 각 텍스처는 눈, 눈썹, 입, 주름, 안경 등 얼굴의 각기 다른 부위를 나타냅니다. 기본 텍스처는 배경색이며, 안경이 있다면 그것이 맨 위 텍스처가 됩니다. 이 두 텍스처는 움직이지 않으며(즉 애니메이션이 아니며), 나머지 텍스처는 가시성과 TRS Keys<sub>2</sub>로 애니메이션됩니다. 이 시점까지 워크플로는 종전의 파이프라인과 같습니다.

애니메이션 키가 세팅되고 나면 TRS와 가시성 키를 텍스트 채널 전체에 세팅해야 합니다. 이 부분은 새 파이프라인에서 새로 추가된 것입니다.

이어서, 애니메이션을 FBX로 익스포트한 후 언리얼 엔진으로 임포트합니다. 이때 Import Custom Attribute와 Material Curve Type 옵션을 체크해야 합니다. 그러면 애니메이션 곡선이 나타납니다.

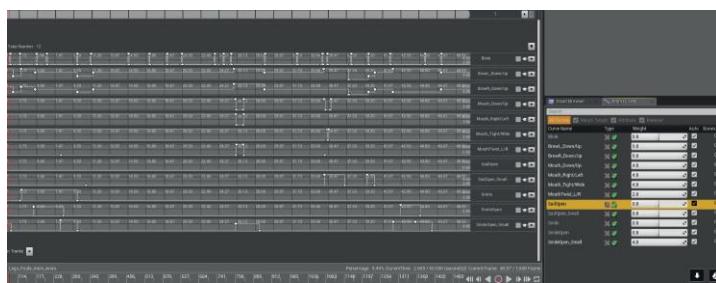


그림 29: 에디터의 애니메이션 곡선

머티리얼 에디터에서 머티리얼 표시를 애니메이션 곡선의 정확한 이름과 함께 설정했습니다. 전환 규칙에 대해서는 Blend Logic(블렌드 로직)을 Linear Interpolation(선형 보간)으로 설정하여 height map과 전환 단계 값에 따라 두 텍스처 사이에서 보간했습니다. 이런 설정에서는 매개변수가 애니메이션 곡선에 따라 변합니다.

Laflaque 팀은 Heightlerp 함수의 사용을 고려했지만 그렇게 설정하면 높이 값이 같은 입과 그 외 얼굴 특징에서 눈을 분리해야 하는 번거로움이 있었습니다. 이런 이유로 팀은 선형 보간을 사용하기로 했습니다.

얼굴 릭의 최종 반복 단계에서는 MotionBuilder에서 각 텍스처의 실제 알파값을 이용했습니다. 이 알파 값은 커스텀 곡선으로 전송되어 최종적으로 그래프에서 각 Lerp의 알파 입력으로 사용되었습니다. 80개 얼굴 텍스처는 모두 같은 샘플러를 이용해 16개 샘플러의 기술적 한계를 극복합니다.

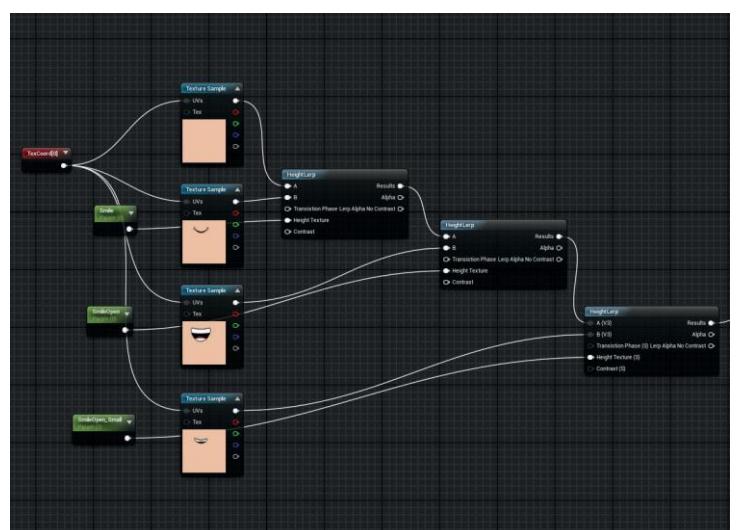


그림 30: 머티리얼 에디터의 애니메이션 로직

2 In MotionBuilder, TRS 키가 변환[위치]과 회전, 스케일을 설정합니다.

## 시퀀서의 이용

프로덕션팀은 제작, 프리비즈, 확인 및 렌더링과 같이 MotionBuilder로 처리하던 업무 중 상당수를 언리얼 엔진의 시퀀서로 대체했습니다.

시퀀서에서는 원활한 작업을 위해 서브 레벨로 구성된 퍼시스턴트 레벨을 정의했습니다. 또한, 시퀀서의 레벨 가시성 트랙으로 오브젝트 그룹의 가시성을 관리했습니다. 프로덕션팀은 오브젝트 숨기기와 보이기 기능이 제작 중 매우 유용하다는 사실을 알게 되었습니다.

아래 그림에 한 에피소드의 명명법 트리의 예가 연속으로 제시되어 있습니다.

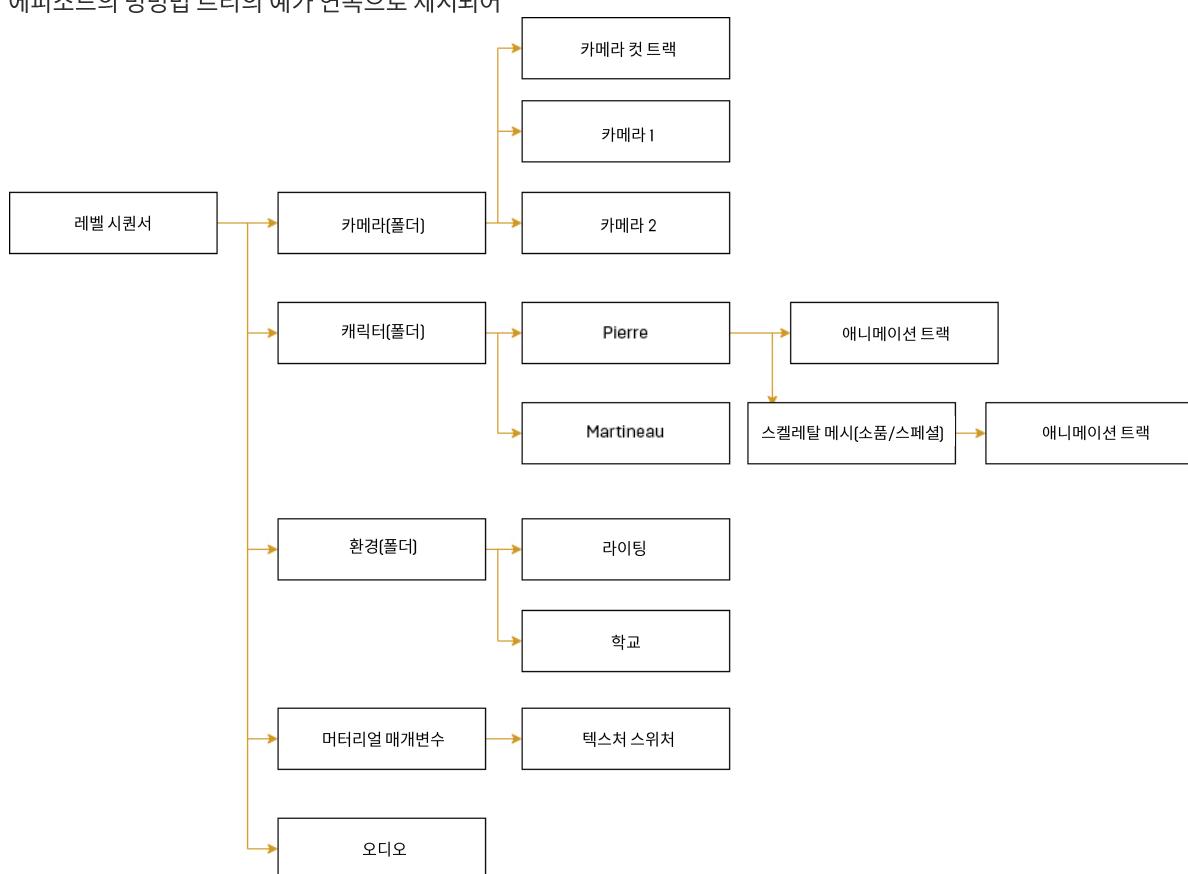


그림 31: Laflaque 시퀀서 명명법

프로덕션팀은 언리얼 엔진을 이용해 플레이 중 이벤트를 트리거했습니다. 그 결과 머티리얼 매개변수를 비롯해 제작 프로세스 중 일부를 자동화할 수 있었습니다.

#### 블루프린트 변수가 시퀀서에서 바로 변경되도록 설정하는 방법

- 액터 블루프린트를 생성합니다.
- 커스텀 이벤트를 생성하고, Call in Editor를 활성화 합니다.
- 변수를 만들고 Expose to Cinematics를 활성화 합니다.
- 시퀀서에서 액터 블루프린트와 그 변수를 추가합니다.
- 시퀀서에서 이벤트 트랙을 생성합니다.

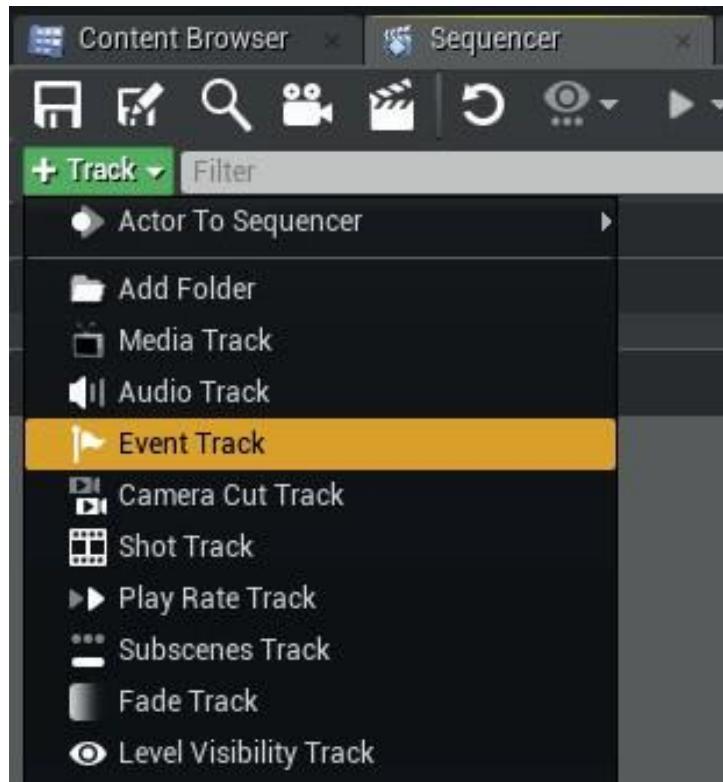


그림 32: 시퀀서에서 이벤트 트랙 생성하기

- 시퀀서에서 시간에 맞춰 이벤트 트랙에 키를 생성합니다. 키를 우클릭하고 속성을 선택한 후 이벤트 이름에 이벤트 이름을 입력합니다.

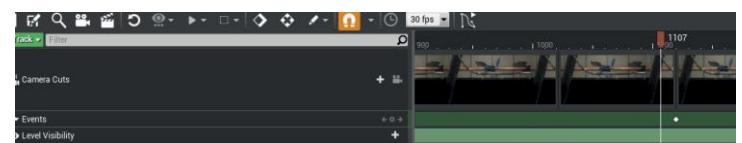


그림 33: 이벤트 트랙에서 이벤트 키를 생성합니다.

- 시퀀서에서 이벤트 트랙을 우클릭한 후 속성을 선택합니다. 이벤트 리시버에서 알맞은 블루프린트를 선택합니다. 이렇게 하지 않으면 이벤트 키가 레벨 블루프린트만 트리거합니다.

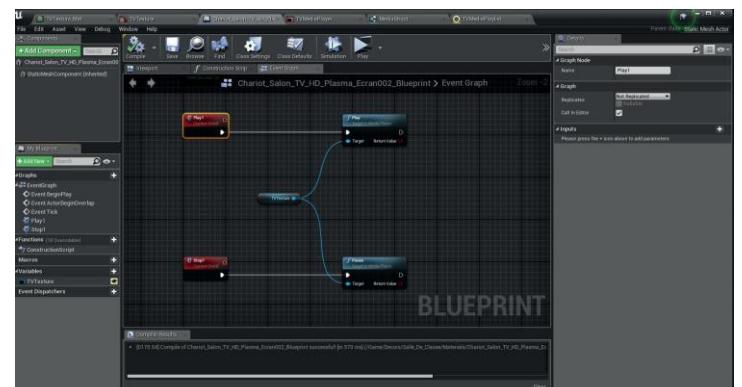


그림 34: 커스텀 이벤트 생성하기

## 라이팅

MotionBuilder에서 라이팅은 제한됩니다. 따라서 프로덕션팀은 여러 가지 워크어라운드를 써서 기존 파이프라인의 실시간 워크플로를 유지해야 했습니다. 환경은 3dx Max에서 라이팅하고 라이팅은 텍스처에 바로 베이크하였습니다. 캐릭터와 애니메이션 소품만 MotionBuilder에서 라이팅하였습니다. 그런데 베이크드 환경 텍스처 때문에 애니메이션과 비주얼 이펙트 단계에서 라이팅을 마음대로 바꾸거나 조정할 수 없었습니다.

프로덕션팀은 실시간 렌더링에 최적화 된 언리얼 엔진 라이트를 사용하여 쇼의 조명 워크플로를 개선하고자했습니다. 새로운 Unreal Engine 파이프 라인에서

환경 및 애니메이션 요소가 같은 라이트로 라이팅되면, 이제 팀은 제작 중에 언제든지 라이트를 조절할 수 있습니다.

하이파이 렌더링으로 현실감 있는 라이팅을 할 때에는 스태틱 라이팅이 언리얼 엔진에서 가장 좋은 선택입니다. GI, Final Gathering, 구역 라이트와 같은 레이 트레이싱으로 하던 기능도 이용할 수 있습니다. 또한 추가 라이트를 이용해 수작업으로 복잡한 동작을 모방하지 않고도 현실감 있는 라이팅을 더욱 쉽게 구현할 수 있습니다.

스타일화된 씬처럼 현실감이 필요치 않다면, 씬을 라이팅할 때 다이내믹 라이팅으로 인터랙티브 피드백을 이용해도 됩니다.

다양한 렌더링 기능을 위해 UE4에는 라이트와 그림자에 근접하는 여러 가지 기술이 있습니다. 스태틱 라이팅의 경우 지원되는 라이트 유형이나 렌더링 기능 면에서 제한이 없습니다. 기본적으로 에디터에서 사용할 수 있는 모든 기능은 베이크 후에 이용할 수 있습니다.

반면 다이내믹 라이팅에서는 몇 가지 제약이 있습니다. 다이내믹 그림자에 사용되는 기법을 몇 가지 소개하면 다음과 같습니다.

- 또렷한 그림자에 대해서는 shadow map을 이용합니다. 이 방식은 어떤 오브젝트 형식과도 잘 어울립니다.
- 부드러운 레이 트레이싱 그림자에는 Distance Field를 이용합니다. 이 방식은 스켈레탈 메시와는 잘 맞지 않습니다.
- 캐릭터의 부드러운 그림자에 대해서는 캡슐 그림자를 이용합니다. 이 방식은 캐릭터 발 주변에 접촉 그림자를 추가할 때 좋습니다.

## 렌더링

마지막으로 살펴볼 분야는 복스 포풀리의 언리얼 엔진 전환의 중심이 되는 렌더링입니다. 기존 파이프라인에서는 MotionBuilder OpenGL을 이용해 실시간 프리뷰를 하고 준 실시간으로 최종 산출물을 렌더링했습니다. 각 씬을 MotionBuilder 안에서 한 번 통과로 렌더링하되, 오디오가 포함된 QuickTime MOV DNxHD 코덱을 이용했습니다.

## 안티앨리어싱

게임 엔진에는 MSAA와 FXAA, TemporalAA, 이렇게 세 가지 앤티앨리어싱기법이 사용됩니다. 대개 TemporalAA가 권장되지만, Performance curve에 따라 이상 현상이 나타나 화면이 흐릿해질 수도 있습니다. FXAA 방법은 품질은 좋으나 DOF(depth of field, 피사계 심도)와 모션 블러를 쓸 수 없어 포스트 프로세스가 제한되는 단점이 있습니다.

Laflaque 팀은 언리얼 엔진에서 TemporalAA 방법을 사용하기로 했습니다. 그러나 첫 시험에서 아래 이미지와 같이 직선에 노이즈가 나타났습니다.



그림 35: 콘솔 조정을 하지 않았을 때

Laflaque 팀은 몇 차례 시험 후 콘솔 명령어를 커스텀 조정하여 렌더링을 최적화하였습니다.

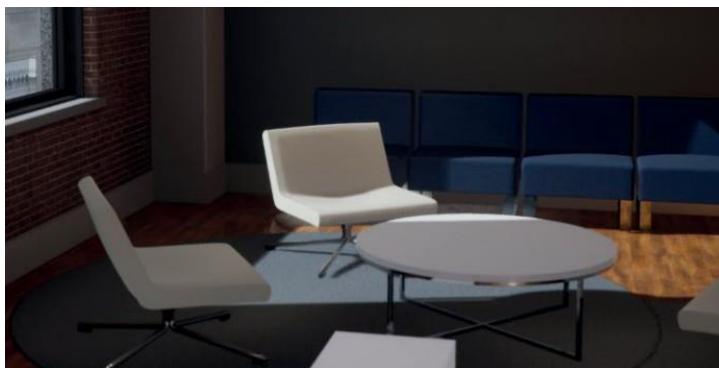


그림 36: 콘솔 조정을 했을 때

아래는 명령어 콘솔 설정의 예입니다. Laflaque 팀에는 대개 건별로 설정을 조정하며 아래 설정은 기준선입니다.

- r.MotionBlurSeparable = 1
- r.MotionBlurQuality = 4
- r.Streaming.PoolSize=1000
- r.ScreenPercentage = 200(이 설정을 100에서 200으로 늘리면 렌더링 시간은 4배로 늘어남)
- r.PostProcessAAQuality = 6
- r.Tonemapper.Sharpen = 1
- r.TemporalAASamples = 4
- r.TemporalAACurrentFrameWeight = 0.2
- 최고 설정(시네마틱)에서 최고 퀄리티를 위한 확장성 설정
- 시퀀서의 렌더링 옵션에서 Use Compression 옵션을 선택 해제하면 최상의 퀄리티를 얻을 수 있습니다.



그림 37: 시네마틱 퀄리티 설정

## 렌더링 퀄리티 비교

MotionBuilder OpenGL과 언리얼 엔진의 렌더링 결과를 비교하면 다음과 같습니다.



그림 38: MotionBuilder로 렌더링한 씬



그림 39: 언리얼 엔진으로 렌더링한 유사 씬

언리얼 엔진에서 나온 이미지가 색감이 더 풍부해지고 보기에도 더 선명합니다. 이 같은 기술적 개선은 Laflaque 시청자에게 확연히 드러납니다. 즉, 쇼가 전보다 더 좋아 보이고 시청 만족도도 더 높습니다.

## 편집과 전달

렌더링된 프레임은 기존 파이프라인과 새 파이프라인 모두 Avid Media Composer를 통해 최종 편집 프로세스로 전달합니다. 오디오는 Pro Tools에서 믹스합니다. 추가 화면, 텍스처, 파티클과 같은 추가 레이어의 합성은 After Effects로 처리합니다.

한 에피소드가 완성되면 Avid DNXHD 110 코덱으로 최종 버전을 Quicktime MOV 파일로 압축해 Radio-Canada 방송 네트워크로 전송합니다. 압축할 때에는 720p의 29.97 FPS Drop Frame을 사용합니다.

# 향후 비전

## 향후 비전

Laflaque쇼를 UE4 파이프라인으로 전환하는 1단계는 기존 파이프라인이 크게 바뀌지 않도록 제작을 최대한 익숙하게 하는 데 중점을 두었습니다. 곧 진행될 2단계는 언리얼 엔진과 MotionBuilder의 병행 사용을 최적화하는 데 중점을 둘 계획입니다.

MotionBuilder Live Link를 이용하게 되면 생산성이 높아집니다. 모캡 데이터는 두 MotionBuilder 씬을 거치지 않고 기본 캐릭터가 있는 언리얼 레이아웃 씬으로 바로 연결됩니다. 그렇게 되면 모션 캡처 시작 시점부터 PhaseSpace로 씬을 실시간 확인할 수 있습니다. MotionBuilder Live Link는 애니메이터의 라이브 피드 역할도 하므로 MotionBuilder의 캡처를 바로잡고 언리얼 엔진에서 실시간으로 정정사항을 볼 수 있게 됩니다.

복스 포풀리는 또 Shotgun 통합 시 언리얼 엔진 플러그인을 이용할 계획이며 레이 트레이싱 솔루션을 비롯한 새 기능에도 기대를 걸고 있습니다.

# 본 문서 소개

# 본 문서 소개

## 제작 일정

사전제작: 2018년 2월

제작: 2018년 6월

## 팀 구성

Roxane Boutet, 제작자

Yves St-Gelais, 제작 책임자

Richard Belec, 프로덕션 담당자

Tom Wilczynski, 기술 감독

Cédric Dubois, 캐릭터 모델링 부문 감독 겸 수석 아티스트

Francois Bissonnette, 3D 아티스트 겸 언리얼 엔진 기술 감독

Hugo Brodeur, 환경 부문 예술 감독

Marco Marandola 라이팅 및 렌더링 선임 아티스트

Mathieu Langlois, 3D 감독

약 10명의 애니메이터 및 레이아웃 아티스트

## 편집 팀

## 저자

David Hurtubise

## 편집자

Michele Bousquet

Brian Pohl

## 기여자

Tom Wilczynski

Cédric Dubois

Francois Bissonnette

Homam Bahnassi

## 이미지 및 데이터 제공

Productions Vox Populi Inc.