스키드

송영조, 염태림
한국산업기술대학교 게임공학부
syj2466@naver.com, yumtr@naver.com

요 약

레이싱 게임의 진입장벽이 높은 원인을 분석해 캐주얼하게 즐길 수 있게 새로운 게임플레이 방식을 도입함. 유저들이 쉽게 게임에 적응하고 충돌과 차량이 멈추는 일이 없게하여 적은 스트레스로 속도와 경쟁의 재미를 느낄 수 있게 노력함. 단순한 조작에서도 차량 별 특성을 활용하거나 맵의 특성을 활용해 실력의 격차를 만듬.

그래픽 퀄리티는 최대한으로 끌어내며 모바일 디바이스를 타겟으로 하여 최적화 기법 연구

1. 서 론

1.1 제작 목적

평소 레이싱 게임을 즐겨 하며 스트레스나 진입장벽이 될 수 있다고 느꼈던 부분들을 가능한 배제하고 조금 더 가볍게 즐길 수 있는 캐주얼한 게임을 만들고자 함. 레이싱 게임을 처음 접하는 유저들이 쉽게 게임에 적응하고 속도와 경쟁의 재미를 느낄 수 있도록 노력함.

1.2 게임 기획 의도 및 주요 특성

레이싱 게임의 진입장벽이 높은 원인을 분석해 캐주얼하게 즐길 수 있게 새로운 게임플레이 방식을 도입함.

단순한 조작에서도 차량 별 특성을 활용하거나 맵의 특성, 전략을 통해 실력의 격차를 만듬.

질주를 방해하는 요소는 가능한 배제하고, 상대와의 몸싸움, 속도경쟁에 몰입할 수 있도록 개발.

1.3 플랫폼

모바일기기(안드로이드, iOS)

1.4 장르

레이싱

1.5 시점

3인칭시점

2. 개발 내용

2.1 개발한 게임의 내용

2.1.1 Clearcoat

차량의 겉면 코팅 질감을 주기 위해 구현.

차량의 속 질감에는 Flake가 있어 새로 연산함.

결과값=속재질반사*프레넬+ 코팅반사

[그림 1] 클리어코트 없음

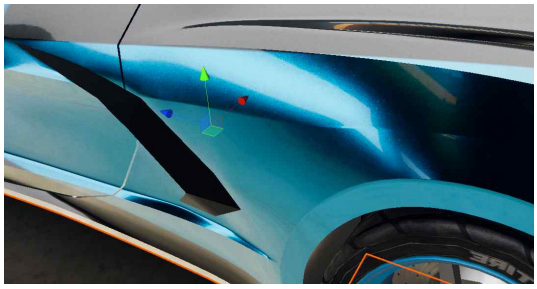
[그림 2] 클리어코트 있음

- 2.1.2 Biplanner

아티스트의 UV 작업없이 차량 전체에 Flake 맵핑하기 위해 사용.

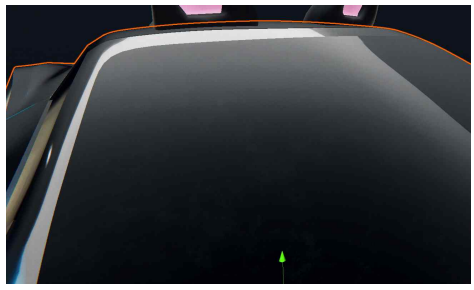
Triplanner는 3번의 텍스처 샘플링을 하지만 본 방식은 2번의 텍스처 샘플링만 하여 Triplanner보다 비용이 더 저렴함.

[2] 기능 구현은 Inigo Quilez의 블로그를 참조함.



[그림 3]

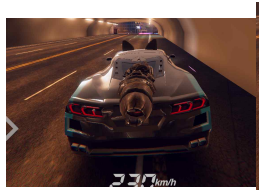
아티스트의 별도 UV작업 없이 러프니스 맵 적용.



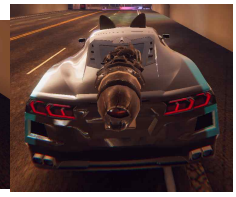
[그림 4]

2.1.3 BoxReflection

오브젝트 이동시 반사가 위치에 맞게 갱신. 리플렉션 프로브의 위치와 Bound 범위, 물체의 월드 위치, 카메라의 월드시점을 고려해서 반사 벡터를 계산. 리플렉션 프로브 범위 내 렌더하려는 물체(A)의 픽셀의 월드 위치와 월드 노멀, 카메라 월드 위치를 사용해 물체(A)의 픽셀 위치에서 리플렉션 프로브 바운드와의 접점(B)을(를) 구함. 접점(B)와 리플렉션 프로브의 월드 위치를 빼면 위치 변화를 사용하는 반사 벡터를 얻을 수 있음.



[그림 5]



[그림 6]

2.1.4 텍스처 프로젝션 맵핑

차량에 데칼을 붙이기 위해 구현.

오브젝트의 버텍스 월드 위치와 텍스처 맵핑 정보를 가진 프로젝터의 MVP행렬을 사용합니다. 버텍스 위치를 프로젝터 공간으로 옮겨 xy를 uv로 사용하면 됨.

2.1.5 버텍스 컬러 마스킹

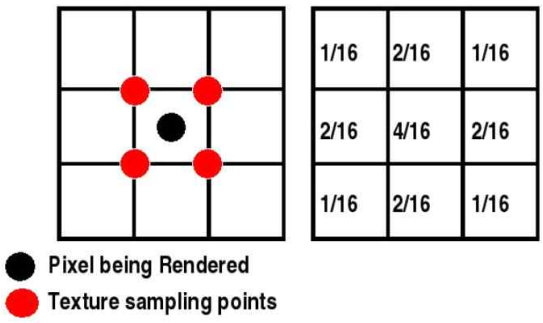
차량 셰이더에 유저가 커스텀 색 변경이 가능한 부위와 색, 러프니스, ao를 나눌 때 uv 작업없이 편하게 작업할 수 있게 버텍스 컬러를 통해 구분함.

RGB 채널별 용도
R: AO
G: 0.95 ≤ x → 유저가 커스텀 가능한 색, x < 0.95 → Grayscale 값
B: Roughness

2.1.6 포스트 프로세싱
2.1.6.1 화면 외곽에 일렁이는 효과가 있는 특수 모션블러

2.1.6.2 LUT 톤 맵핑
Lut는 유니티 Volume에서 만드는 것을 저장해 별도의 에디터를 사용하지 않고 커스텀 포스트 프로세싱에서 가져다 사용.

2.1.6.3 카와세 블룸
[1] Masaki Kawase의 2003년 GDC 강연을 기반으로 제작. 4개의 픽셀 사이의 점을 읽으면 리니어 보간이 된다는 점을 이용. 픽셀 단위로 읽는 가우시안 블러보다 효율적임.

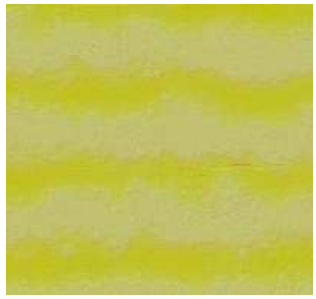


[그림 7] 카와세 블러 샘플링 방식

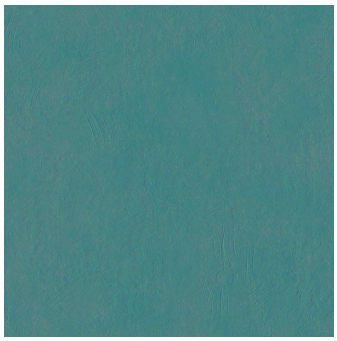
2.1.7 Uber 셰이더
새로운 기능이 필요할 때마다 새로운 셰이더를 작성할 필요 없이 시간을 아끼기 위해 제작.
최적화를 위해 라이트 정보가 필요없는 Unlit은 별도의 셰이더로 관리함.

Uber셰이더의 기능들은 다음과 같다.

한 텍스처 RGBA에 다른 정보를 저장해 텍스처 개수를 줄인 텍스처 패킹 사용.
텍스처 개수를 줄여 메모리를 절약하고 대역폭 비용을 줄이기 위함.
텍스처 패킹 방식은 3종류가 있다.
1) RGBA(베이스컬러, 알파) / RG(Normal), B(Metallic), A(Roughness)


[그림 8]

2) R(Grayscale), GB(Normal), A(Roughness)
흑백 정보의 색만 가지면 텍스처는 하나만 사용하여 메모리를 절약하고 대역폭 비용을 아낌.



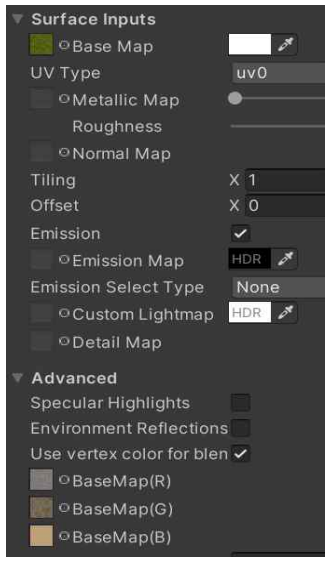
[그림 9]

3) R(Grayscale), GB(Normal), A(Alpha)

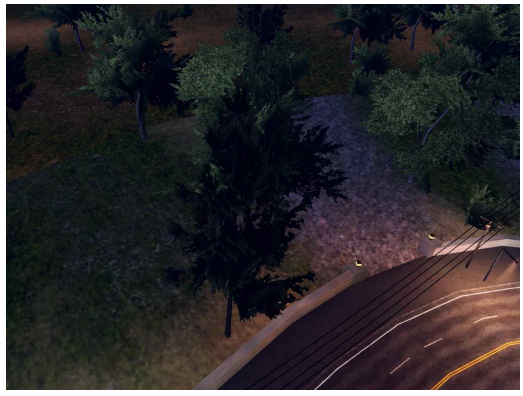
객체 별 커스텀 라이트 맵을 사용하여 반복 사용되는 라이트 맵의 해상도를 높힘

디테일맵을 사용하여 확대 시에도 적절한 텍스처 퀄리티를 유지함. 디테일 맵 역시 R(Grayscale), GB(Normal), B(Roughness)의 텍스처 패킹을 사용.

지형 텍스처링에 사용. 여러 텍스처 맵을 버텍스 컬러 RGBA 값을 기반으로 샘플링 자연스러운 텍스처 블렌딩을 만듬. 텍스처의 알파맵을 블렌딩에 이용하여 보다 자연스러운 결과물을 냄.



[그림 10] 텍스처 블렌딩 UI



[그림 11] 텍스처 블렌딩 적용

2.1.8 유니티 스태틱 배칭 중에 파괴되는 오브젝트

유니티엔 스태틱 배칭이란 여러 메쉬를 하나로 합치는 기능이 있다. 이를 이용해 하나의 머티리얼인 오브젝트는 한번에 그려 성능이 좋지만 런타임에서 이동되는 오브젝트는 사용이 불가능하다. 하지만 메쉬필터의 메쉬만 기존 메쉬로 교체하고 스크립트로 움직이면 이동이 가능하다. 오브젝트가 파괴되기 전엔 스태틱 배칭된 메쉬를 사용하고, 파괴될땐

기존 메쉬로 교체하여 성능도 살리고 이동도 가능해졌다.

2.1.9 박스 데칼

깊이 맵을 활용해 표면에 붙는 데칼과 차 그림자 (바닥에 붙어있지만 바닥에서 멀어질수록 흐려짐) 혹은 벽 데칼, 그래피티에 사용

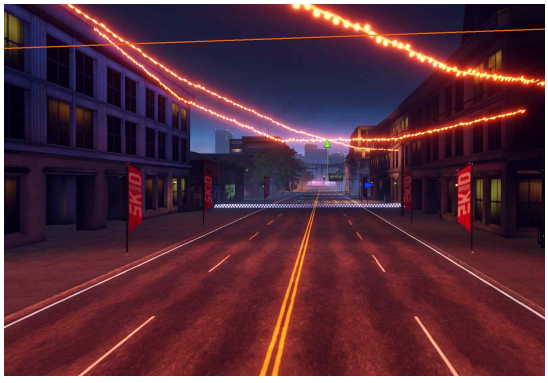
[그림 12]

2.1.10 뎁스 프리패스

커스텀 렌더 파이프라인을 활용해 뎁스 프리패스를 추가하여 GPU 사용량을 줄임

2.1.11 소프트 파티클

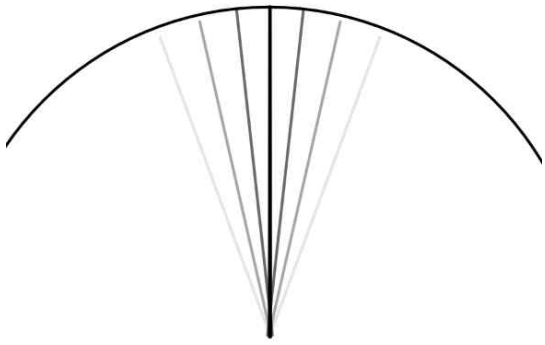
깊이 맵을 통해 표면에 부자연스럽게 닿는 부분을 부드럽게 페이드 처리한 파티클



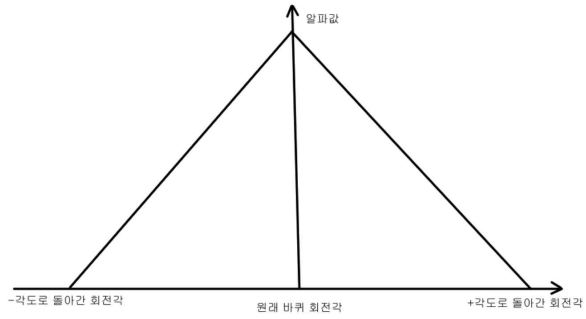
[그림 13]

2.1.12 바퀴 모션블러

기존 바퀴 회전각에서 미묘한 각도 차이를 둔 바퀴 오브젝트를 여러 개 만들고 회전각 차이가 클수록 투명하게 하여 모션블러 느낌을 냄.



[그림14] 위 그림의 막대기처럼 휠을 여러개로 만들고 알파 단계를 중심에서 멀어질수록 흐리게 함.



[그림 15]

[그림 16] 바퀴 모션블러

2.1.12 아티스트 편의 기능

UI 자동 앵커 설정

여러 프리팹 리버트

버텍스 컬러 체인저

UI 이미지 기울이는 Skewed Image 컴포넌트

UI 원형 프로그래스바

UI Custom Blend mode

Src, Dst 연산 방식 설정으로 Additive, Subtract, Multiply 등 블렌드모드 설정 가능

2.1.13 아트 작업

차 모델링 - 12대

[그림 17]

배경 모델링 - 맵 4종류 제작

[그림 18]

2.1.14 게임 전체 흐름도

소리 없이 영상만 재생되고 터치하여 시작

[그림 19] 스플래쉬 화면

[그림 20] 타이틀 애니메이션

맨처음 실행하는 경우 튜토리얼로 넘어감

게임 모드 혹은 차고를 선택

[그림 21] 타이틀 화면

원하는 차량을 선택하고 다음 혹은 구매되지 않은 차량이면 구매를 누름

[그림 22] 차고 화면

싱글 플레이 - 맵을 선택

[그림 23] 맵 선택 화면

차고 - 차를 꾸미거나 성향과 성능을 바꿀 수 있음

[그림 24] 차고 화면2

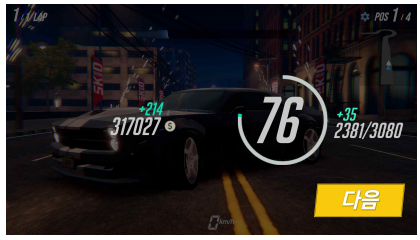
멀티 - 매치메이킹

[그림 25] 매치메이킹 화면

게임을 플레이

게임이 끝나면 등수와 보상이 나오고 다음을 누르면 메인메뉴로 복귀

[그림 26] 게임 결과 화면



[그림 27] 게임 결과 화면2

2.2 개발 방법론

애자일 방법론을 사용하여 피드백에서 요구사항을 분석하고 잦은 변경에 유리하게 대응함.

2.3 게임 진행 방법

처음 시작하면 3초 타이머 후 차가 출발함. 출발하기 전 버튼을 눌러 게이지를 채워 빠르

게 출발. 게이지를 가능한 많이 채우는데 빨간색에 머무르지 않으면 됨.

좌우 버튼으로 좌우 조작

드리프트 코너에 보라색 선과 초록색 선이 있습니다. 드리프트 버튼을 보라색 선에서 누르고 초록색 선에서 떼서 드리프트 함

니트로 버튼을 눌러 니트로를 사용하고 이때 나오는 타이밍바의 타이밍에 맞춰 버튼을 눌러 니트로를 강화

게임모드에 맞는 승리조건을 달성하면 우승

3. 결 론

3.1 장단점

3.1.1 장점

모바일 게임이 가지는 휴대성이 큰 장점이다. 자투리 시간을 활용한 게임, 인터넷만 지원되면 어디서나 플레이 가능하다.

출시와 유통이 다른 플랫폼에 비해 간편하다.

유니티 엔진을 사용하여 기능 추가가 쉽고, 타 플랫폼 포팅도 간편하다.

3.1.2 단점

그래픽에 비해 과도한 GPU 사용량으로 생기는 발열.

쉬운 조작을 의도했으나 개발자 실력에 맞춰 레벨 디자인을 진행한 결과 어려운 게임이 됨.

Linear space에서 작업하여
OpenGLES3.0 이전 폰들은 지원이 안됨

3.2 향후 추가 개발 또는 업그레이드 내용이 필요한 부분 등을 서술.

제작의도는 쉬운 조작으로 레이스에 집중하는 것이었으나 어렵다는 피드백이 많아 드리프트 조작법 수정중.

튜토리얼 설명 부실로 이해가 어려운 부분이 있어 해결 필요.

4. 부 록

매뉴얼 작성

첨부된 apk파일을 안드로이드 폰이나 녹스, 블루스택에서 실행 및 설치하여 실행.

4.1 게임 실행 환경:

OpenGLES 3.0 이상 지원하는 안드로이드 기기 (S10 이상 고 성능 기기)

최소사양 V20, 갤럭시 A7 / 권장사양 S10 이상 고 성능 안드로이드 폰 또는 태블릿.

4.2 문제 해결

4.2.1 개발된 게임의 제약사항

색 관리 리니어 워크플로우를 사용해 OpenGLES3.0 이상을 지원하는 기기가 필요.

4.2.2 문제 해결을 위한 연락처

01052892466 송영조
01092676334 염태림

5. 참고문헌

[1] Masaki Kawase, “Frame Buffer Postprocessing Effects in DOUBLE-S.T.E.A.L (Wreckless)”, GDC, 2003

[2] Inigo Quilez, “Biplanar”, iquilezles.org, 2020