GoForce 3D: 가까운 픽셀에서 곧 개봉합니다

NVIDIA는 휴대용(handheld) 솔루션을 활발하게 개발 중

- 역동적이며 성장 중인 시장
- 세계 수준의 모바일 그래픽 제품 개발에 전력 투구
- 이미 활발하게 개발 중

왜 휴대 장치용 게임을 만들어야 할까?

- 새롭게 부상하는 시장
- 유비쿼터스(ubiquitous)형태의 모바일 장치
  - 전세계 5억 대 이상
  - 벨소리 다운로드만 30억 달러
  - 게이머 6천만명 중 4천 5백만명이 휴대용으로 게임
- 라이트 게이머와 매니아 게이머

왜 휴대 장치용 게임을 만들어야 할까?

- 기술 혁신으로 수준이 상승
  - 실시간 **3D**
  - 무선 연결
- 전 세계 상대의 수익 증대의 잠재력

도전과제...

- 제한된 시스템 자원
- 다른 기종들간의 다양한 개발 공간
  - 다양성

그 밖에 어려움...

- 퍼블리싱과 배포
- 디지털 저작권 관리
- 특이한 시장 역학
  - 서비스 제공업체와 **ARPU**
  - 데이터 및 모바일 엔터테인먼트에 대한 관심 증가

GoForce 3D 소개

- 라이센스식 모바일 장치용 **3D** 코어

- **OpenGL ES / Direct3Dm** 호환
- 절전형 아키텍쳐
- 통합된 단일 **SRAM**
- **VGA** 해상도까지 가능
- 현대적 기능

- **30+Hz** 이상에서 복잡한 게임 실행이 목표

GoForce 3D 특징

- 기하학 엔진 (부동점과 고정점)
- 16비트 색상(16 비트 Z 포함)
- 40비트 색상 (내부)
- 최대 4개의 동시 텍스쳐가 가능한 멀티 텍스쳐링
- 2선형/3선형(bilinear/trilinear) 필터링
- 유연한 텍스쳐 형식
  - 4비트/8비트 팔레트, DXT1 압축 외
- 완전한 원근감 보정 (색상 포함)
- 서브 픽셀의 정확성
- 픽셀별 포그
- 알파 블렌딩

전통적 아키텍쳐

- 깊은 파이프라인 **(200 단계)**
- 항상 모든 단계를 거쳐야만 함
- **OpenGL** 유형의 빠른 텍스쳐링에 최적화 됨
- 파이프라인들은 항상 클럭킹**(clocking)**을 하고 있음
- 빠르지만 전원을 너무 많이 소비
- **100M** 픽셀/초 당 **~750mW**

(~200 파이프 단계)

완전히 새로운 초절전형 아키텍쳐

(~50 파이프 단계)

- 유연한 프래그먼트 **ALU**
- 래스터**(Raster) –** 프래그먼트 생성 및 루프**(loop)** 관리
- 파이프라인들은 활성화시에만 기동됨
- 저 전력
  - **100M** 픽셀**/**초 당 **< 50 mW**
  - 실제 게임플레이 동안
- 높은 확장성 아키텍쳐

Geometry?

- 현재 휴대용 프로세서
  - **Arm 7/Arm 9/+**
  - **Clock rates: 50Mhz – 400Mhz**
  - 부동 소수처리가 없음
  - 호스트 버스가 **DRAM** 버스와 공유를 하고 있음
  - 제한된 시스템 메모리
- **GPU**로 처리과정의 많은 부분을 옮기면
  - 효과적인 파워 관리
  - 보다 나은 성능

절전을 위한 파이프라인 단축

Depth Complexity = 1
텍스쳐 있음
blend 없음
Z 없음

Depth Complexity = 4

1. 텍스쳐, 블렌드 없음
2. 텍스쳐, 블렌드 없음, **Z**
3. 텍스쳐, 블렌드 없음, **Z**
4. 텍스쳐 삼각형, 블렌드, **Z**

단순한 화면은 **fog**, 블렌딩, 알파테스트, 및 모든 트라이앵글을 위한 **depth** 비교를 필요로 하지 않는다.

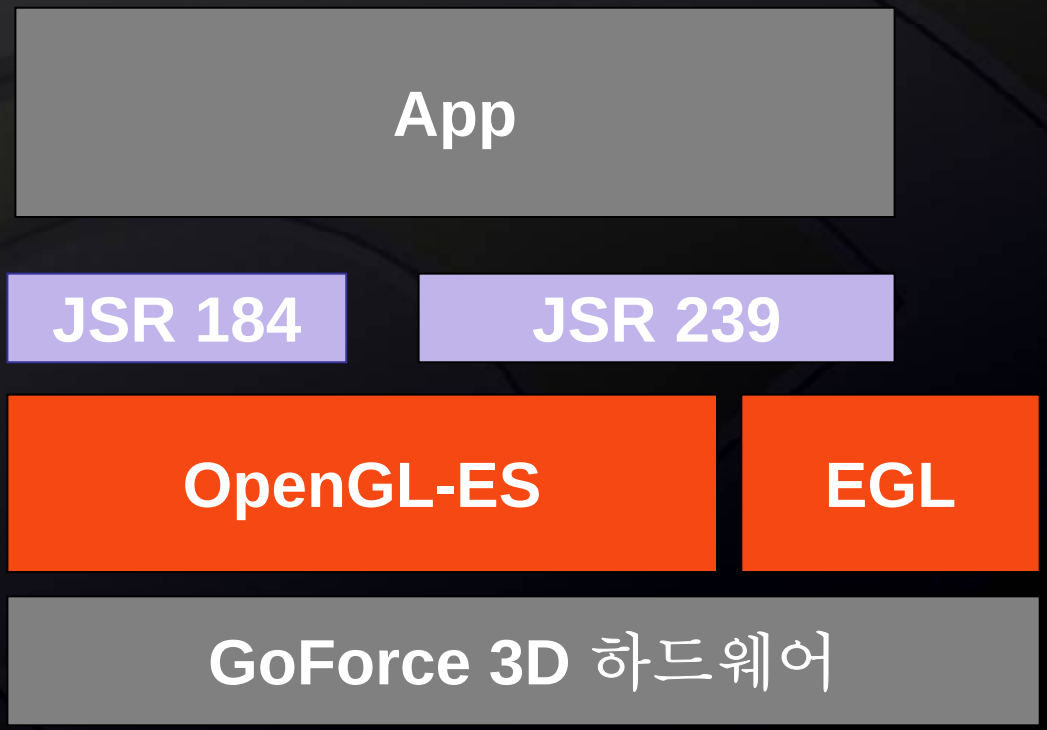
풍부한 2D 기능

- 완전한 단색 채움(color fill)기능
- 소스 복사
- 알파 블랜딩
  - 모든 픽셀에 대한 고정 알파 값
- 16x16 패턴의 채움
- 선 그리기
  - 서브 픽셀단위의 정확성
- 클리핑 및 투명도
  - 내부/외부 클리핑 지원

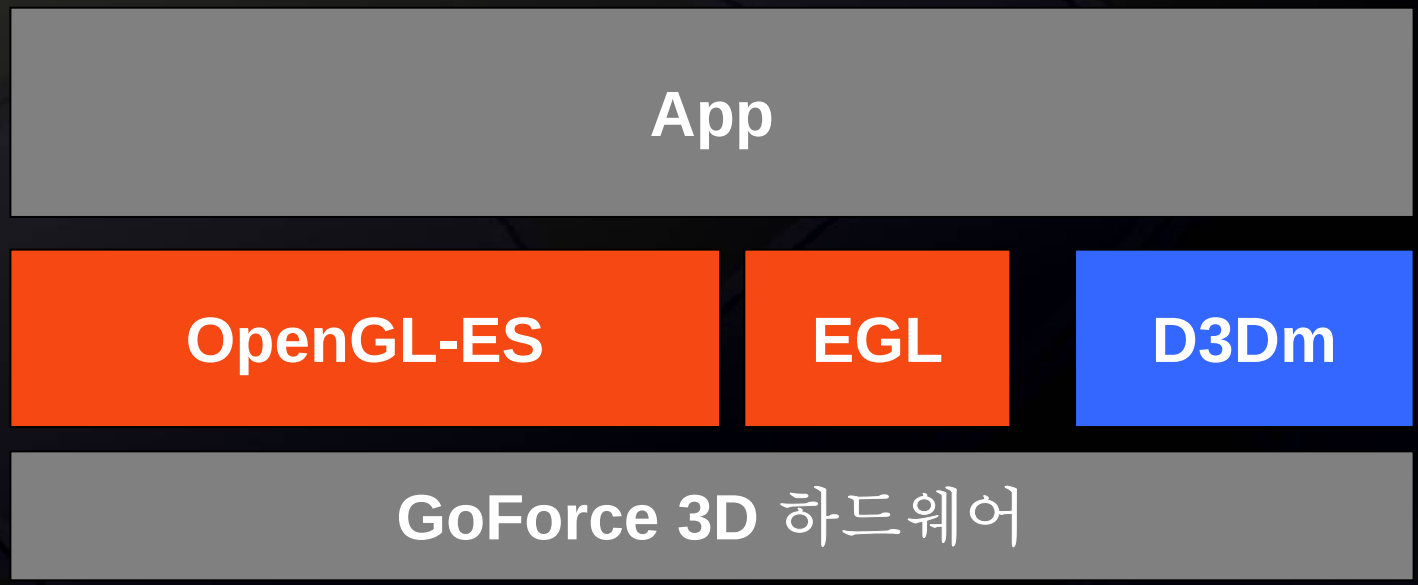
*n*Power 기술

- 미사용 파이프 라인의 자동 절전
- 아키텍쳐 레벨의 전원 관리 – 일반, 대기, 절전모드
- 다양한 레벨의 고급 전원 관리
- 저전력 작동

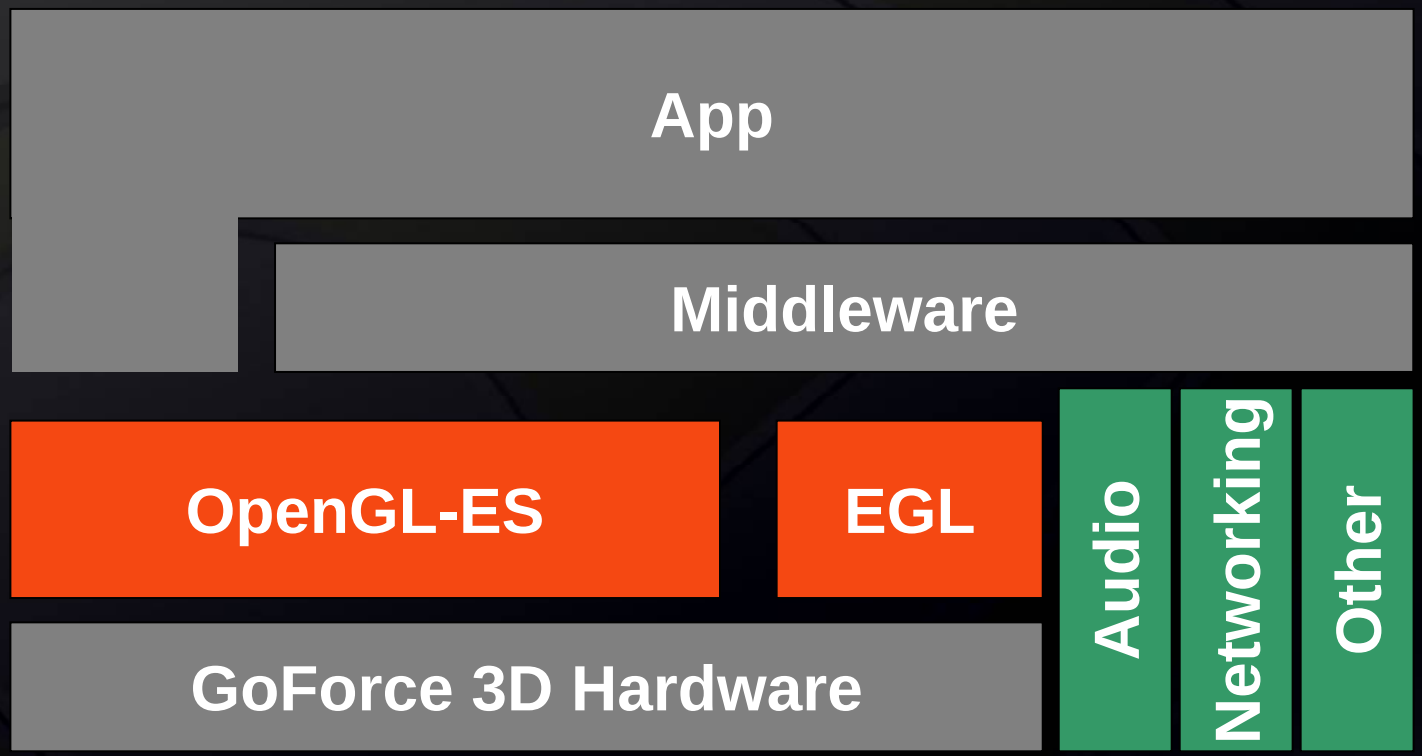
Java 프로그래밍 모델



네이티브 프로그래밍 모델



미들웨어 프로그래밍 모델



OpenGL ES 1.0 대 OpenGL

- **OpenGL 1.3**과 일부 유사
- 누락
  - 디스플레이 목록**(display list)**
  - **glBegin/glEnd**
  - **Texgen**
  - 환경 맵
  - **Evaluators**
- 추가
  - 고정소수점 **type/** 진입점
  - 보다 널리 쓰이는 바이트 타입

OpenGL ES 1.1 대 OpenGL

- **OpenGL 1.5** 사양이 토대

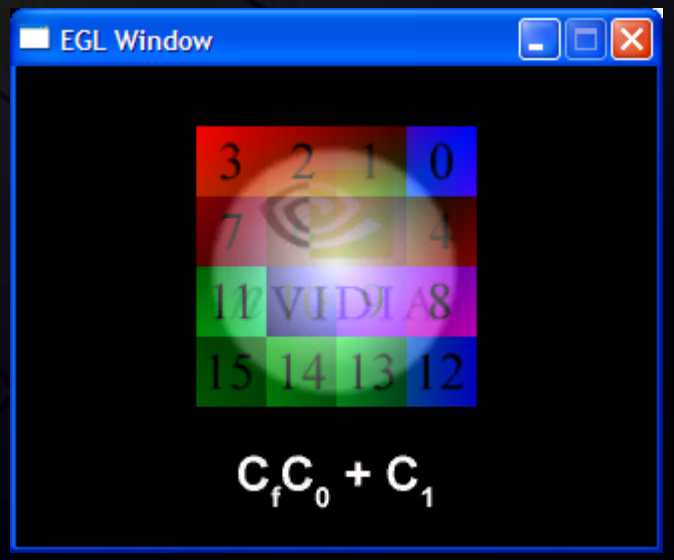
- **ES 1.0**에 기능 추가
  - 버텍스 버퍼 개체
  - 자동 **Mipmap** 생성
  - 향상된 텍스쳐 조합 작업
  - 사용자 정의 클립 평면
  - 포이트 스프라이트 및 포인트 스프라이트 어레이
  - 역동적 상태 **(dynamic state)** 조회

Direct3Dm

- 미발표
- **Microsoft**와 공동 작업

NVIDIA 핸드헬드 SDK

- 데모
  - **OpenGL ES (101)**
  - 기능 데모



- 포팅 레이어
  - **OpenGL-ES**에서 **OpenGL**
  - **PC**에서 실행
  - 임베디드에 익숙치 않은 개발자도 곧바로 제작 및 실행가능

NVIDIA HHDK (계속)

- .NET 데모 마법사
  - x86 Windows 및 ARM 리눅스를 대상으로 하는 뼈대구조의 애플리케이션을 생성
- 도구 및 라이브러리
  - DXT1 압축 도구
  - DXT1 이미지 로딩 라이브러리
  - 고정소수점 라이브러리 – 최적화된 ARM 수학
- 설명서 – GoForce 3D 개요

개발 키트

- 곧 출시!
- **NVIDIA** 휴대용 개발자 프로그램
  http://developer.nvidia.com
- 전자우편
  handset-dev@nvidia.com

사례연구: Bubble

- 원래는 **GeForce 256** 테스크탑 **GPU**용으로 제작**(2000**년 경**)**
- 변형되며 반사되는 표면
  - 스프링 기반의 물리학
  - 환경 매핑

- **GoForce 3D**으로 포팅 됨
  - 목표**: GoForce 3D**에서 네이티브 그래픽 어플리케이션의 적용에 대한 가능성을 이해

사례 연구: **Bubble**

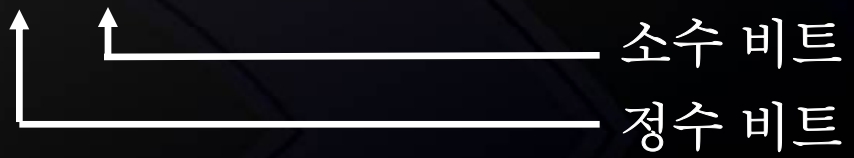
- 데모

Bubble: 개요

- 구(sphere) 모델 – 각종 버텍스와 모서리(edge)
- 각종 힘
  - 순간적으로 “ 찌르는” 힘(Impulse “Poke” Force)
  - “ 재귀적인” 힘 (Homeward Force)
  - 탄성 “ 가장자리” 힘 (Elasticity “Edge” Force)
  - 바깥쪽으로 “ 팽창하는” 힘 (Outward “Swelling” Force)
- 힘은 속도, 위치, 노멀(normal) 관계를 좌우

Bubble: 변형

- 부동소수점으로 시뮬레이션 – 매우 느림
- 문제점을 파악하기 위하여 프로파일러 사용
- 정수 수학으로 전환 **(s15.16)**



- 고정소수점 – 범위와 정밀도의 균형
  - 대체 형식 또는 리스케일링

Bubble: 환경 매핑

- 원래는 규브 매핑 및 반사 텍스쳐 생성을 사용
  - **ES 1.0**은 양쪽 다 지원하지 않음
- 이중 포물선 매핑과 수동 텍스쳐 좌표 생성 **(** 고정점**)**

Bubble: 텍스쳐 메모리의 사용

- 장면마다 **8**개의 텍스쳐 사용
  - **2 – 256x256** 텍스쳐 **(mip-mapped)**
  - **6 – 256x256** 텍스쳐 **(non-mipmapped)**

R5G6B5 – 16-bits/texel = 786432 + 349524 = 1.08Mb

DXT1 – 4-bits/texel = 196608 + 87381 = 0.27Mb

뛰어난 품질의 **DXT1** 은 비용도 **R5G6B5**의 **25%** 에 불과

Bubble: 품질

Bilinear

높은 **frequency**

Trilinear

실루엣 근처의
높은 **frequency**

Trilinear w/ LOD clamp

최고 품질 *

*** SGIS_texture_lod**를 사용

NVIDIA 3D 퀄리티 데모

x86/윈도우용의 OpenGL-ES wrapper상에서 실행

GoForce 3D hw 휴대용 그래픽이 생성할 수 있는 것을 이뮬레이트.

NVIDIA 개발자 사이트

- NVIDIA 핸드헬드 개발자 프로그램 등록

developer.nvidia.com

문의사항:

Handset-Dev@nvidia.com

Bubble: 작동 방식

- 구형 모델 – 각종 버텍스와 모서리(Edge)
  - 버텍스
    - 위치
    - 수직
    - 속도
    - 평균속도 – 평균 “ 이웃” 속도
    - 홈 위치 – 버텍스 “ 홈” 정지 위치
  - 모서리(Edge)
    - 버텍스 인덱스의 쌍
    - 홈 길이 – 초기 모서리 길이

Bubble: 작동방식

- 변형 – 힘을 가하여 모델을 업데이트
  - 버텍스
    - 위치
    - 수직
    - 속도
    - 평균 속도
    - 홈 위치
  - 모서리(Edge)
    - 버텍스 인덱스의 쌍
    - 홈 길이

Multi-Step Process...

Bubble: 변형

- 1단계 – 속도 업데이트
- 스프링 힘을 기준으로 조정
  - “Homeward” (안으로 향하는) 힘
  - “Outward” (밖으로 향하는) 힘
  - “가장자리” 힘 (예: 탄성)

버텍스별로 다음과 같음

```
vel += HomeForce( home – pos )
vel += OutwardForce( normal )
```

모서리(Edge)별로 다음과 같음

```
vert[v0].vel += EdgeForce( vert[v0].pos – vert[v1].pos )
vert[v1].vel += EdgeForce( vert[v1].pos – vert[v0].pos )
```

Bubble: 변형

- 2단계 – 속도 필터링
  - 평균 속도 계산
  - 필터 적용 – **vel = 0.9 * vel + 0.1 * avg**
- 3단계 – 위치 업데이트
- 4단계 – 속도에 드래그 적용
- 5단계 – 수직 계산
  - 모서리(Edge) 벡터를 사용하여 모든 삼각형에서 반복

Bubble: 포킹(Poking)

- 즉각적인 속도 업데이트 필요
- **"pick ray"**에 가장 가까운 포인트 찾기
  - 시점 위치: **(0,0,0)**
  - 찍기 광선: **(screen_x,screen_y, -near)**
- 거리에 따라 내향적 펄스**(inward pulse)** 힘 적용

p.vel += PulseForce(distance(closest.pos, p.pos))

위에서 **PulseForce(d) =** k_1 *** Pow(d, -20)**

Bubble: 변형 (다시보기)

- 2단계 – 속도 필터링
  - 속도를 필터링하지 않으면 어떻게 될까?

시뮬레이션이 불안정하게 됨.