**HW5 보고서**

20212020 박민준

# 요구사항

본 숙제는 HW3에서 자신이 구현한 내용을 바탕으로 Shading (Gouraud shading 또는 Phong shading), Texture mapping, 그리고 OpenGL shader 작성을 통한 3D 그래픽스 효과를 생성함을 목표로 한다.

**1. 요구사항 및 구현 개요**

## 1. Gouraud / Phong shading 전환

**•** 요구사항:

* 자신이 임의로 선정한 3D 물체에 대해 Gouraud(정점 조명)와 Phong(픽셀 조명)을 모두 구현하고, 실행 중 1·2 키로 곧바로 바꿀 수 있어야 한다.

**•** 구현 여부: **구현 완료** (1번 키 → Gouraud, 2 번 키 → Phong).

**•** 선정한 객체:

* 정적 물체 Bike

**•** 구현 방법:

1. Scene\_Definitions.h와 Scene\_Definitions.cpp에 Shader\_Gouraud와 Shader\_Phong 두 클래스를 새로 선언·정의하고, 각 클래스의 prepare\_shader() 내부에서 대응하는 GLSL 소스(gouraud.vert/frag, phong.vert/frag)를 추가 작성하였다.
2. Scene\_Definitions.cpp 내 Scene::build\_shader\_list()에서 두 프로그램을 생성하여 shader\_ID\_mapper[SHADER\_GOURAUD/PHONG]에 등록하였다.
3. 실시간 전환을 위해 main.cpp 의 keyboard()에서 ‘1’ 키를 누르면 scene.shader\_kind = SHADER\_GOURAUD, ‘2’ 키를 누르면 scene.shader\_kind = SHADER\_PHONG 로 설정하도록 코드에 추가하였다.
4. 실제 렌더 단계에서는 Static\_Object::draw\_object()와 Dynamic\_Object::draw\_object()(각 cpp 파일) 내부에서 scene.shader\_kind 값을 읽어 eff\_shader를 결정한 뒤 glUseProgram()을 호출하도록 분기하였다.

**•** 확인 방법:

* 프로그램을 실행한 뒤 Bike 모델을 바라본 상태에서 ‘1’을 누르면 면 단위 색 띠가 보이고, ‘2’를 누르면 동일 위치에서 부드러운 하이라이트가 나타나 전환이 확인된다.

## 2. 비디오 텍스처와 NEAREST / LINEAR 필터 전환

**•** 요구사항:

* 자신이 임의로 선정한 3D 물체(정적인 물체와 동적인 물체 각각 한 개씩)에 PNG 시퀀스를 재생하는 비디오 텍스처를 적용하고, N·L 키로 필터를 전환해야 한다.

**•** 구현 여부: **구현 완료**

**•** 선정한 객체:

1. 정적 물체: Ironman
2. 동적 물체: Wolf

**•** 구현 방법:

1. VideoTexture.cpp 파일을 새로 작성하였다. load\_frames()는 "video\_frames/ramen\_###.png"를 탐색해 경로를 벡터에 저장하고, 첫 프레임을 GL 텍스처로 업로드한다. advance()는 GLUT 타이머( main.cpp 의 timer\_scene() )에서 0.1 초 간격으로 호출되어 다음 PNG를 glTexImage2D()에 올린다.
2. Scene\_Definitions.h에 VideoTexture vt\_ironman, vt\_wolf 멤버를 추가하고, Scene::initialize()에서 각각 load\_frames()를 호출하였다.
3. Ironman은 Static\_Object::draw\_object()에서 object\_id == STATIC\_OBJECT\_IRONMAN일 때 eff\_shader = SHADER\_PHONG\_TEXTURE로 강제하고, Wolf는 Dynamic\_Object::draw\_object()에서 object\_id == DYNAMIC\_OBJECT\_WOLF일 때 동일 셰이더를 사용하도록 하였다. 해당 셰이더 내부에서 sampler 0이 비디오 텍스처를 참조한다.
4. 필터 전환은 VideoTexture::apply\_filter()에서 수행되며, main.cpp의 keyboard()에 N 키(NEAREST), L 키(LINEAR) 분기를 삽입하였다.

**•** 확인 방법:

* Polygon\_Fill mode에서 Ironman 화면을 클로즈업한 뒤 N 키를 누르면 계단 현상이 선명해지고, L 키를 누르면 즉시 부드럽게 보간되는 것이 확인되며 Wolf 모델에도 동일하게 적용된다.

**•** 특이 사항: 두 비디오 모두 10 FPS로 재생되도록 타이머 주기를 100 ms로 설정하였다.

## 3. Global Light 토글

**•** 요구사항:

* 세상 좌표계를 기준으로 최소한 한 개 이상의 광원 배치해야 한다.

**•** 구현 여부: **구현 완료**

**•** 구현 방법:

* Scene\_Definitions.h에 bool lightEnabled를 선언하고 초기값을 true로 설정하였다.
* main.cpp의 keyboard()에 3 키 분기를 넣어 값이 토글되면 printf로 상태를 출력한다.
* Gouraud·Phong·Phong\_Texture 셰이더의 fragment 코드에서 uniform bool uLightOn을 받아 diffuse 계산을 조건문으로 감췄다.

**•** 확인 방법:

* 프로그램 실행 후 3 키를 누르면 실내 모델 전체 조도가 꺼졌다가 다시 켜지는 것을 눈으로 확인할 수 있다.

**•** 특이 사항: 전역 평행광 1개를 월드 좌표 (200, 100, 150)에 배치하였다.

## 4. Eye-Light(카메라 광원) 토글

**•** 요구사항:

* 움직이는 카메라에 대하여 눈 좌표계를 기준으로 광원을 한 개 구현해야 한다.

**•** 구현 여부: **구현 완료**

**•** 구현 방법:

* scene.eyeLightEnabled 플래그를 두고, Scene::draw\_world()에서 현재 ViewMatrix의 역행렬로 카메라 위치를 구해 Eye-Space 좌표 eyePosE를 계산한 뒤 Gouraud·Phong 셰이더에 uniform bool uEyeLightOn과 uniform vec3 uEyePos로 전달하였다.
* keyboard()에 4 키 토글 분기를 추가하였다.

**•** 확인 방법:

* 카메라가 모델 가까이 접근한 상태에서 4 키를 누르면 전면에 specular 스폿이 생겼다 사라지는 것을 확인할 수 있다.

• 특이 사항: Global Light OFF 상태에서도 Eye-Light가 켜져 있으면 카메라 근처 면이 약하게 조명된다.

## 5. Model-Light(동적 물체 로컬 광원) 토글

**•** 요구사항:

* 움직이는 물체에 고정된 광원, 즉 그 물체의 모델링 좌표계에 배치한 광원을 한 개 구현해야 한다.

**•** 구현 여부: **구현 완료**

**•** 선정한 객체:

* 동적 물체 Wolf

**•** 구현 방법:

* Dynamic\_Object::draw\_object()에서 Wolf일 때 localLight(0,0,10)을 ModelMatrix로 변환해 월드-좌표를 얻고 다시 ViewMatrix로 곱해 Eye-Space 위치를 scene.modelLightPosE에 저장하였다.
* scene.modelLightEnabled bool을 만들고 5 키에서 토글한다.
* Shader 세 종류(Gouraud, Phong, Phong\_Texture) 모두 modelLightEnabled, modelLightPosE 변수를 추가해 spot광을 더했다.

**•** 확인 방법:

* Global-Light, Eye-Light를 전부 OFF 한 뒤, Model-Light만 ON/OFF하면 Wolf 몸체가 밝아졌다 어두워졌다 하는 것을 확인할 수 있다.
* Wolf가 이동함에 따라 shading 효과가 적용된 Bike, Ironman의 몸체 밝기가 변하는 것을 확인할 수 있다.

## 6. Dodecahedron 투명도 조절

**•** 요구사항:

* 12면체 이상의 다면체와 같이 다각형의 전후 관계가 알기 쉬운 닫힌 물체에 대하여 투명한 효과를 생성해야 한다.

**•** 구현 여부: **구현 완료**

**•** 구현 방법:

* scene.dodecaTransparent, scene.dodecaAlpha 변수를 추가하고 6, –, = 키 입력을 keyboard()에 구현하였다.
* Static\_Object::draw\_object()에서 Dodeca일 때 glEnable(GL\_BLEND)와 glDepthMask(GL\_FALSE)로 블렌딩을 설정하고, 각 셰이더 fragment에 uniform float uAlpha를 넘겨 투명도를 곱하였다.

**•** 확인 방법:

* Polygon\_Fill Mode에서 6 키를 누르면 Dodeca 내부가 비쳐 보이고 ‘-‘ 키 여러 번으로 거의 유리처럼, ‘=’ 키로 불투명에 가깝게 돌아가는 모습을 확인할 수 있다.

## 7. 재미있는 Shader Effect 구현 (Ink-Edge / RGB-Glitch)

**•** 요구사항:

* ‘스크린 효과’ 또는 ‘블라인드 광원’과 같은 재미있는 쉐이더 효과를 최대 2개까지 구현해야 한다.

**•** 구현 여부: **구현 완료**

**•** 구현 방법:

1. Scene\_Definitions.h / Scene\_Definitions.cpp에 PostFX 구조체를 정의하고 초기화 함수 PostFX::init()을 작성하였다. 이 구조체는 색상용 RGBA8 텍스처 colorTex, 깊이버퍼용 depthTex, 후처리용 FBO fbo, 그리고 2-D 전체 화면 Quad를 담는 VAO quadVAO를 보유한다. init() 내부에서 두 텍스처를 glTexImage2D()로 생성한 뒤 glFramebufferTexture2D()로 FBO에 결합하고, 마지막으로 정점 4개짜리 Quad VBO-VAO를 구성하였다.
2. 전용 셰이더 래퍼 Shader\_Post를 Scene\_Definitions.h에 선언하였다. 이 클래스는 일반 파생 클래스와 달리 매개변수 없는 prepare\_shader()를 비워 두고, 필요할 때마다 load(vspath,fspath)를 호출해 다른 fragment 코드를 읽어들인다. 컴파일 후 uScene (샘플러), uTime (Glitch용 시간), uAlpha 위치를 캐시한다. ­build\_shader\_list()에서 pass-through, Ink-Edge, RGB-Glitch 세 프로그램을 차례로 생성해 shader\_ID\_mapper에 등록하였다.
3. 실제 GLSL 파일을 작성하였다. 공통 정점 셰이더 post\_fullquad.vert는 전달된 (x,y)와 UV를 그대로 넘긴다. post\_passthru.frag는 색을 변형 없이 복사한다. post\_ink.frag는 입력 색을 YUV 가중치로 그레이스케일화한 뒤 fwidth()로 경계를 추적하여 에지 픽셀만 검정으로 바꾼다. post\_glitch.frag는 y축 기준 사인 함수로 작은 오프셋을 만들어 R, G, B 채널을 서로 다른 UV에 샘플링해 색 분리 효과를 준다. 추가로 uTime을 이용해 흔들림이 시간에 따라 진동하도록 했다.
4. Scene::draw\_world()를 2-Pass 구조로 개조하였다. 가장 처음 현재 뷰포트와 깊이·컬링·MSAA 상태를 보존해 둔 뒤, Pass 1에서 FBO를 바인딩하고 창 전체 해상도에 장면(축, 정적·동적 물체)을 렌더링한다. Pass 2에서는 다시 기본 프레임버퍼로 돌아와 선택된 후처리 셰이더를 바인딩한다. 이때 Polygon 모드가 Wire(Line)인 경우 Quad 렌더를 건너뛰어 와이어프레임이 다시 압축되는 문제를 예방하였다. Quad를 그리기 전에 glPolygonMode(GL\_FILL), glDisable(GL\_CULL\_FACE | GL\_MULTISAMPLE | GL\_LINE\_SMOOTH)로 상태를 통일하고, 출력이 끝나면 최초에 저장했던 모든 OpenGL 상태를 복구했다. Glitch 효과가 활성화된 경우 post.time을 프레임당 0.016씩 증가시켜 uTime으로 전달한다.
5. 마지막으로 main.cpp의 keyboard()에 7 키와 8 키 분기를 추가하였다. 7 키를 누르면 scene.post.useInk가 토글되고 동시에 Glitch 플래그를 내려 Ink-Edge가 단독 실행되며, 8 키를 누르면 반대로 Glitch가 토글되고 Ink는 꺼진다. 두 키 모두 동작 후 glutPostRedisplay()를 호출하여 즉시 결과가 반영된다.
6. 결과적으로 Polygon Fill 모드에서 7 키를 누르면 모든 물체 윤곽선이 두꺼운 검정으로 강조되며 만화 같은 느낌을 주고, 8 키를 누르면 화면 전체가 RGB 색분리와 가로 흔들림을 동반한 글리치 효과로 변한다. V 키로 Wire 모드에 진입할 경우 Pass 2가 자동으로 스킵되어 깨끗한 라인 뷰가 유지되고, Fill 모드로 돌아오면 마지막으로 선택한 후처리 효과가 다시 활성화된다.

**•** 확인 방법:

* Polygon\_Fill Mode에서 7 키로 만화 잉크 선이, 8 키로 색 분리 glitch가 적용되는 것을 확인할 수 있다. V 키로 Wire 모드로 전환하면 효과가 사라지고 깨끗한 와이어프레임만 남는다.

**•** 특이 사항:

* Glitch shader 시간 입력은 post.time을 프레임마다 0.016씩 증가시켜 60 FPS 기준으로 움직인다.

**2. 프로그램 조작법 요약**

**1) HW3 카메라 조작**

**• W/S/A/D/Q/E → 이동**

**• Z/X → Roll**

**• 키보드 방향키 → Pitch-Yaw**

**• 마우스 Wheel → Zoom in/Zoom out**

**• F키 → ORTHO(정면·측면·상면)와 CCTV(Main+정적0/1/2+동적) 그룹 전환**

**• I/O → Main Camera/Dynamic CCTV Camera 활성화**

**• P → Tiger-Eye 뷰 토글**

**• T → 모든 카메라 프레임(RGB 축) 표시 토글**

**2) 셰이딩·광원·후처리 단축키**

**• 1 → Gouraud 셰이딩 선택**

**• 2 → Phong 셰이딩 선택**

**• 3 → 전역(Global) 라이트 ON/OFF**

**• 4 → Eye-Light(카메라 광원) ON/OFF**

**• 5 → Wolf 로컬 라이트 ON/OFF**

**• 6 → Dodeca 투명 모드 ON/OFF**

**– / = : Dodeca alpha값 ↓ / ↑**

**• 7 → Ink-Edge 후처리 ON/OFF (Polygon Fill 모드에서만)**

**• 8 → RGB-Glitch 후처리 ON/OFF (Polygon Fill 모드에서만)**

**3) Video Texture Filter**

**• N → NEAREST 필터 적용**

**• L → LINEAR 필터 적용**

**4) Render 상태 토글**

**• ‘C’ → 면 컬링(Cull Face) 모드 토글.**

**• ‘V’ → 폴리곤 렌더 모드(Polygon Mode) 토글.**

**• ‘B’ → 깊이 테스트(Depth Test) 토글.**

**5) 기타**

**• ESC → 프로그램 종료.**