헬로, 안드로이드

15주차 – OpenGL의 3D 그래픽 (2)

강대기
동서대학교 컴퓨터정보공학부

학습 목표

- OpenGL ES에서 모델을 만드는 방법을 알아본다.
- 조명 및 카메라 설정에 대해 알아본다.
- 모델을 회전시키는 방법에 대해 알아본다.
- 질감을 적용하는 방법에 대해 알아본다.
- 모델을 투명하게 하는 방법에 대해 알아본다.

차례

- 모델 만들기
- 조명, 카메라
- 액션!
- 질감 적용하기
- 훔쳐보기
- 요약
- 퀴즈
- 연습문제

모델 만들기

- 프로페셔널한 환경에서 3차원 모델은 그래픽 도구 (Maya, 3D Max, POV-Ray 등)로 만들어서 파일에 저장하고, 프로그램에서 불러들임
- 본 예제에서는 간단한 정육면체 모델을 정의해 봄 (GLCube)
  - GLRenderer의 onDrawFrame()에서 호출함
- GLCube.java에서 vertices 배열은 사각형의 모서리를 고정소수점 모델 좌표에 의해 정의함
- 육면체의 각 면은 두 개의 삼각형으로 이루어진 사각형으로, OpenGL의 그리기 모드 중 일반적인 삼각 스트립 (triangle strips) 사용
- 각 지점은 세 개의 좌표로 x,y 와 화면에서 우리의 눈쪽을 향하는 (norm) z 가 있음
- gl.glDrawArrays() 를 호출하여 그림

고정 소수점과 부동 소수점

- OpenGL ES 인터페이스는 고정 소수점과 부동 소수점 인터페이스를 지원함
- 고정 소수점 메서드는 x로 끝나고, 부동 소수점 메서드는 f로 끝남
- 고정 소수점은 4 바이트인 2의 32승(65,536)으로 나타내짐
- 고정 소수점 32,768은 부동 소수점 0.5f 와 대응됨
- 필수 부분은 2 바이트의 Most Significant Bytes (MSB)
- 이렇게, 안드로이드의 2D 라이브러리들에서 일반적으로 정수를 사용하는 방법과는 상당히 다름
- 일부 저가 기기에선 고정 소수점이 부동 소수점보다 더 빠르므로, 프로그래밍하기 수월한 부동 소수점으로 프로그램을 작성한 후, 나중에 필요하면 고정 소수점으로 최적화 시키면 좋음

조명, 카메라

- 실제 생활에서는 수많은 광원이 있음
  - 태양, 헤드라이트, 횃불, 용암 등등
- OpenGL에서는 장면에서 최대 8 개의 광원을 정의하게 해줌
- 조명에는 빛 자체와 빛이 비춰지는 부분이 있음
- 조명의 종류로는
  - ambient – 광원이 멀리 떨어진 경우와 같이 전체적인 장면에 비춰지는 빛
  - diffuse – 형광 패널에서 나오는 것과 같이, 부드럽지만 방향성이 있는 조명
  - specular – 반사광과 같이 주로 밝은 지점에서 오는 반짝이는 빛. 빛나는 물질과 결합해 사실감을 더해주는 하이라이트를 만듦
- 예제에서 사용된 광원은 흰 색의 전방향성 빛으로 밝은 diffuse 구성 요소와 흐린 주변 구성 요소를 가짐
- 또한, 빛은 금속, 플라스틱, 종이 등의 다양한 재료에 따라 다르게 반사되므로, 육면체의 재료를 정의하여야 함
- 예제에서는 종이로 만든 듯 무딘 느낌을 주었고, 육면체의 오른쪽 위 모서리에서 빛이 비치게 함

액션!

- 모델 물체을 이동시키기 위해, **GLSurfaceView.Renderer** 를 구현한 GLRenderer 의 onSurfaceCreated() 와 onDrawFrame() 변경
- onSurfaceCreated() – 시간 초기화
  - startTime = System.*currentTimeMillis();*
- onDrawFrame() – 시간에 근거하여 회전
  - **long elapsed = System.*currentTimeMillis() - startTime;***
  - gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
  - gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
- onDrawFrame()에서 매 프레임마다 모델을 조금씩 회전시킴
- 안드로이드는 여러 다양한 장치에서 운용될 수 있음
- 매 프레임마다 회전 각도를 기억하여 루프를 반복할 때마다 각도를 증가시키게 되면, 빠른 장치에서는 빠르게 회전하고 느린 장치에서는 느리게 회전함
- 위에서처럼 경과된 시간에 따라 이동시키면, 모든 장치에서 예측가능한 움직임을 얻을 수 있음. 다만 빠른 장치에서는 더 부드러운 움직임을 볼 수 있음

질감 적용하기

- 실생활의 모든 것들은 실은 벽돌담이나 정원 자갈길의 거친 표면처럼 질감이 있음 (세상에 완벽히 매끄러운 물체가 있을까?)
- 래미네이트하듯이, 사진을 물체 위에 덮어씌우는 작업을 함으로써, 질감을 나타낼 수 있음
- 안드로이드 Portable Network Graphics (PNG) 형식의 파일을 OpenGL이 이해할 수 있는 형식으로 변환 (GLCube.loadTexture() ← GLRenderer. onSurfaceCreated()에서 호출됨)
  - Bitmap bmp = BitmapFactory.*decodeResource(*context.getResources(), resource);
  - GLUtils.*texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);*
  - gl.glTexParameterx(GL10.*GL_TEXTURE_2D,* GL10.*GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);*
  - gl.glTexParameterx(GL10.*GL_TEXTURE_2D,* GL10.*GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);*
  - bmp.recycle();

  - GLCube.*loadTexture(gl, context, R.drawable.android); //* GLRenderer. onSurfaceCreated()
- GLCube()의 생성자에서 텍스춰를 저장하기 위한 자바 NIO 버퍼 (mTextureBuffer) 설정
  - *// …*
  - ByteBuffer tbb = ByteBuffer.*allocateDirect(texCoords.length * 4);*
  - tbb.order(ByteOrder.*nativeOrder());*
  - mTextureBuffer = tbb.asIntBuffer();
  - mTextureBuffer.put(texCoords);
  - mTextureBuffer.position(0);
- GLCube().draw()에서 텍스춰 버퍼를 로드함
  - gl.glTexCoordPointer(2, GL10.*GL_FIXED, 0, mTextureBuffer);*
- R.drawable.android 는 res/drawable/android.png 파일

GLCube.java (1)

```
...
class GLCube {
  private final IntBuffer mVertexBuffer;
  private final IntBuffer mTextureBuffer;

  public GLCube() {

    int one = 65536;
    int half = one / 2;
    int vertices[] = {   /* FRONT */   -half, -half, half, half, -half, half,           -half, half, half, half, half, half,
        /* BACK */           -half, -half, -half, -half, half, -half,           half, -half, -half, half, half, -half,
        /* LEFT */           -half, -half, half, -half, half, half,           -half, -half, -half, -half, half, -half,
        /* RIGHT */           half, -half, -half, half, half, -half,           half, -half, half, half, half, half,
        /* TOP */          -half, half, half, half, half, half,           -half, half, -half, half, half, -half,
        /* BOTTOM */           -half, -half, half, -half, -half, -half,           half, -half, half, half, -half, -half, };
    int texCoords[] = {          /* FRONT */          0, one, one, one, 0, 0, one, 0,
        /* BACK */          one, one, one, 0, 0, one, 0, 0,
        /* LEFT */          one, one, one, 0, 0, one, 0, 0,
        /* RIGHT */           one, one, one, 0, 0, one, 0, 0,
        /* TOP */          one, 0, 0, 0, one, one, 0, one,
        /* BOTTOM */           0, 0, 0, one, one, 0, one, one, };

    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
    vbb.order(ByteOrder.nativeOrder());
    mVertexBuffer = vbb.asIntBuffer();
    mVertexBuffer.put(vertices);
    mVertexBuffer.position(0);
    // ...
    ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
    tbb.order(ByteOrder.nativeOrder());
    mTextureBuffer = tbb.asIntBuffer();
    mTextureBuffer.put(texCoords);
    mTextureBuffer.position(0);

  }
```

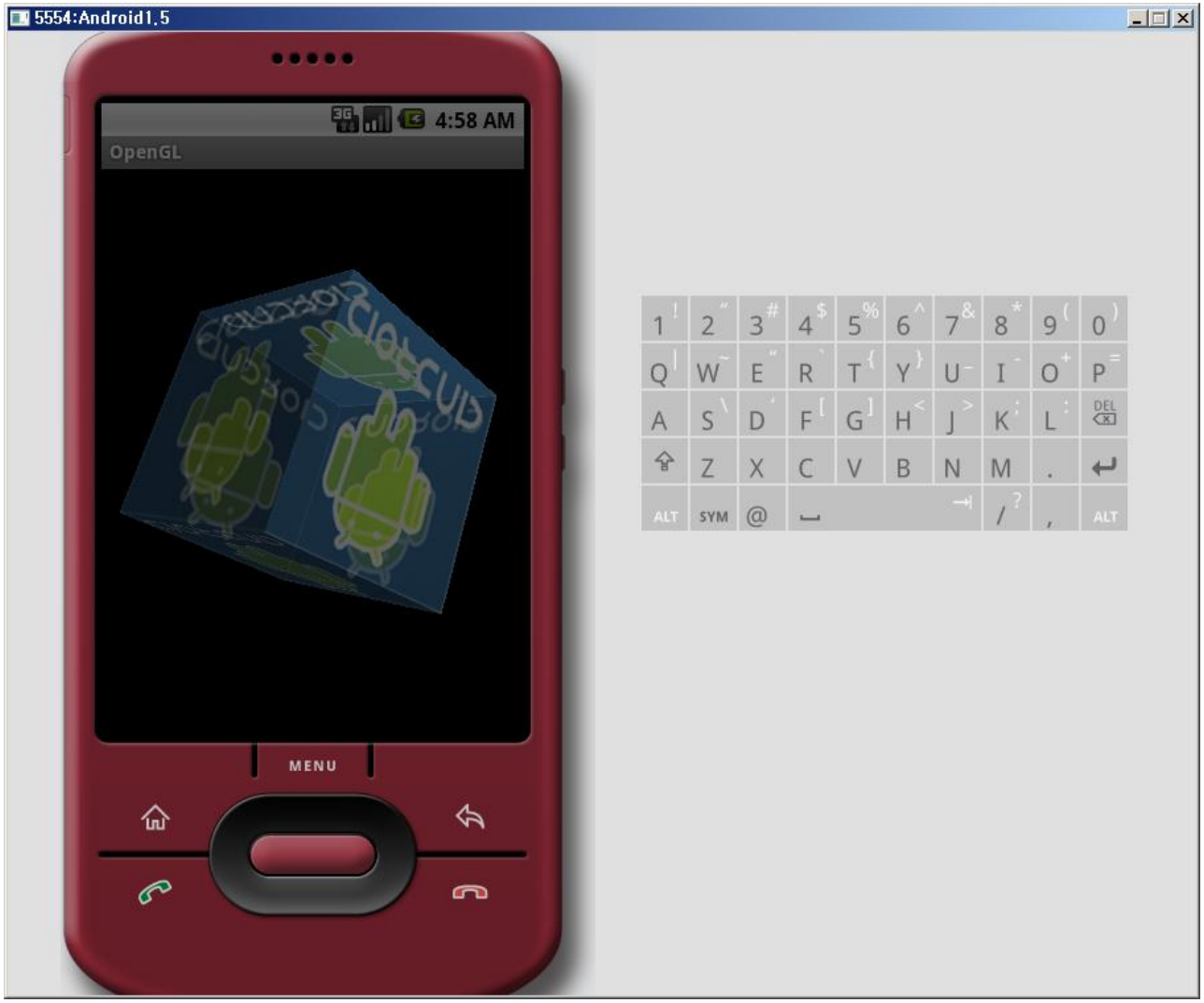
GLCube.java (2)

```
    public void draw(GL10 gl) {
        gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);
        gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTextureBuffer);

        gl.glColor4f(1, 1, 1, 1);
        gl.glNormal3f(0, 0, 1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
        gl.glNormal3f(0, 0, -1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);

        gl.glColor4f(1, 1, 1, 1);
        gl.glNormal3f(-1, 0, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);
        gl.glNormal3f(1, 0, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);

        gl.glColor4f(1, 1, 1, 1);
        gl.glNormal3f(0, 1, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);
        gl.glNormal3f(0, -1, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);
    }


    static void loadTexture(GL10 gl, Context context, int resource) {
        Bitmap bmp = BitmapFactory.decodeResource(
                context.getResources(), resource);
        GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D,
                GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D,
                GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);
        bmp.recycle();
    }

}
```

훔쳐보기

- 화면에 디스플레이되는 모델 객체을 부분적으로 투명하게 만들 수 있음
- GLRenderer.onSurfaceCreated() 에서 설정함
  - **boolean SEE_THRU = true;**
  -
  - // …
  - **if (SEE_THRU) {**
  - gl.glDisable(GL10.*GL_DEPTH_TEST);*
  - gl.glEnable(GL10.*GL_BLEND);*
  - gl.glBlendFunc(GL10.*GL_SRC_ALPHA, GL10.GL_ONE);*
  - }
- 전경(前景)의 대상 뿐만 아니라, 가려진 부분도 보기 위해 깊이 확인을 비활성화함
- 사물의 투명도가 알파 채널에 기초하게 만드는 블렌딩 모드를 활성화함 – 알파 채널을 질감 안의 RGB의 평균값으로 잡았는데, 3D 그래픽에서 특히 속도가 중요한 상황에서는 색상이 얼마나 밝은지(Intensity)를 대략 측정하는 방법으로 간주될 수 있음

투명하게 한 후, 실행 화면



요약

- OpenGL ES에서 모델을 만드는 방법을 알아본다.
- 조명 및 카메라 설정에 대해 알아본다.
- 모델을 회전시키는 방법에 대해 알아본다.
- 질감을 적용하는 방법에 대해 알아본다.
- 모델을 투명하게 하는 방법에 대해 알아본다.

퀴즈

- 삼각 스트립이란 무엇인가?
- 고정 소수점이란 무엇인가?
- 고정 소수점 메서드의 이름은 무엇으로 끝나고, 부동 소수점 메서드의 이름은 무엇으로 끝나는가?
- Ambient, Diffuse, Specular 조명에 대해 설명하라.
- 자바 NIO 버퍼에 대해 설명하라.
- 물체를 회전시키기 위해서 어떤 함수를 사용하는가?
- 물체에 질감을 적용하기 위해서는 어떠한 일들을 해야 하는가?

연습문제

- 나무껍질 이미지나, 벽돌 이미지와 같은, 다른 이미지를 사용하며 질감을 나타내 보자. 그리고 사람 얼굴을 붙여보자.
- EditText / Button 이나 Spinner와 같은 위젯을 붙여서, 정육면체의 투명도와 회전 속도를 조절할 수 있게 해보자.
- 정육면체의 투명도나 회전 속도를 사용자의 터치 이벤트를 받아들여 조절할 수 있는지 알아보자. 예를 들어, 터치하면 일단 멈추고, 빠르게 드래깅 하면 빠르게 돌아가고, 느리게 드래깅하면 느리게 돌아가게 할 수 있을까?
- OpenAL에 대해 알아보라. 안드로이드에서 OpenAL을 지원할 계획인가? 만일 OpenAL을 안드로이드에 포팅해서 사용하려면 어떻게 해야 할까?