```
attribute vec3 position;
uniform mat4 projectionMatrix;
uniform mat4 modelViewMatrix;

#if NUM_VS_TEXTURES > 0
struct Layer {
    float scale;
    float bias;
    int mode;
    float zmin;
    float zmax;
};

uniform Layer       elevationLayers[NUM_VS_TEXTURES];
uniform sampler2D   elevationTextures[NUM_VS_TEXTURES];
uniform vec4        elevationOffsetScales[NUM_VS_TEXTURES];
uniform int         elevationTextureCount;
uniform float       geoidHeight;

highp float decode32(highp vec4 rgba) {
    highp float Sign = 1.0 - step(128.0, rgba[0])*2.0;
    highp float Exponent = 2.0 * mod(rgba[0], 128.0) + step(128.0, rgba[1]) - 127.0;
    highp float Mantissa = mod(rgba[1], 128.0)*65536.0 + rgba[2]*256.0 +rgba[3] + fl
oat(0x800000);
    highp float Result = Sign * exp2(Exponent) * (Mantissa * exp2(-23.0 ));
    return Result;
}

float getElevationMode(vec2 uv, sampler2D tex, int mode) {
    if (mode == ELEVATION_RGBA)
        return decode32(texture2D( tex, uv ).abgr * 255.0);
    if (mode == ELEVATION_DATA || mode == ELEVATION_COLOR)
#if defined(WEBGL2)
        return texture2D( tex, uv ).r;
#else
    return texture2D( tex, uv ).w;
#endif
    return 0.;
}

float getElevation(vec2 uv, sampler2D tex, vec4 offsetScale, Layer layer) {
    uv = uv * offsetScale.zw + offsetScale.xy;
    float d = getElevationMode(uv, tex, layer.mode);
    if (d < layer.zmin || d > layer.zmax) d = 0.;
    return d * layer.scale + layer.bias;
}
#endif

#ifdef USE_LOGDEPTHBUF
#ifdef USE_LOGDEPTHBUF_EXT
varying float vFragDepth;
varying float vIsPerspective;
#else
uniform float logDepthBufFC;
#endif
#endif
attribute vec2      uv_0;
#if NUM_CRS > 1
attribute float     uv_1;
#endif
attribute vec3      normal;
uniform mat4 modelMatrix;
uniform bool lightingEnabled;
varying vec2 vHighPrecisionZW;
#if MODE == MODE_FINAL
#ifdef USE_FOG
varying float vFogDepth;
#endif
varying vec3        vUv;
varying vec3        vNormal;
#endif
```

```
void main() {
    vec2 uv = vec2(uv_0.x, 1.0 - uv_0.y);
    vec3 transformed = vec3( position );
#if NUM_VS_TEXTURES > 0
    if(elevationTextureCount > 0) {
        float elevation = getElevation(uv, elevationTextures[0], elevationOffsetScal
es[0], elevationLayers[0]);
        transformed += elevation * normal;
    }
#endif

    transformed += geoidHeight * normal;
    vec4 mvPosition = vec4( transformed, 1.0 );
#ifdef USE_INSTANCING
    mvPosition = instanceMatrix * mvPosition;
#endif
    mvPosition = modelViewMatrix * mvPosition;
    gl_Position = projectionMatrix * mvPosition;
#ifdef USE_LOGDEPTHBUF
#ifdef USE_LOGDEPTHBUF_EXT
    vFragDepth = 1.0 + gl_Position.w;
    vIsPerspective = float( isPerspectiveMatrix( projectionMatrix ) );
#else
    if ( isPerspectiveMatrix( projectionMatrix ) ) {
        gl_Position.z = log2( max( EPSILON, gl_Position.w + 1.0 ) ) * logDepthBufFC
- 1.0;
        gl_Position.z *= gl_Position.w;
    }
#endif
#endif
    vHighPrecisionZW = gl_Position.zw;
#if MODE == MODE_FINAL
#ifdef USE_FOG
    vFogDepth = - mvPosition.z;
#endif
#if NUM_CRS > 1
    vUv = vec3(uv_0, (uv_1 > 0.) ? uv_1 : uv_0.y); // set uv_1 = uv_0 if uv_1 is und
efined

#else
    vUv = vec3(uv_0, 0.0);
#endif
    vNormal = normalize ( mat3( modelMatrix[0].xyz, modelMatrix[1].xyz, modelMatrix[
2].xyz ) * normal );
#endif
}
```