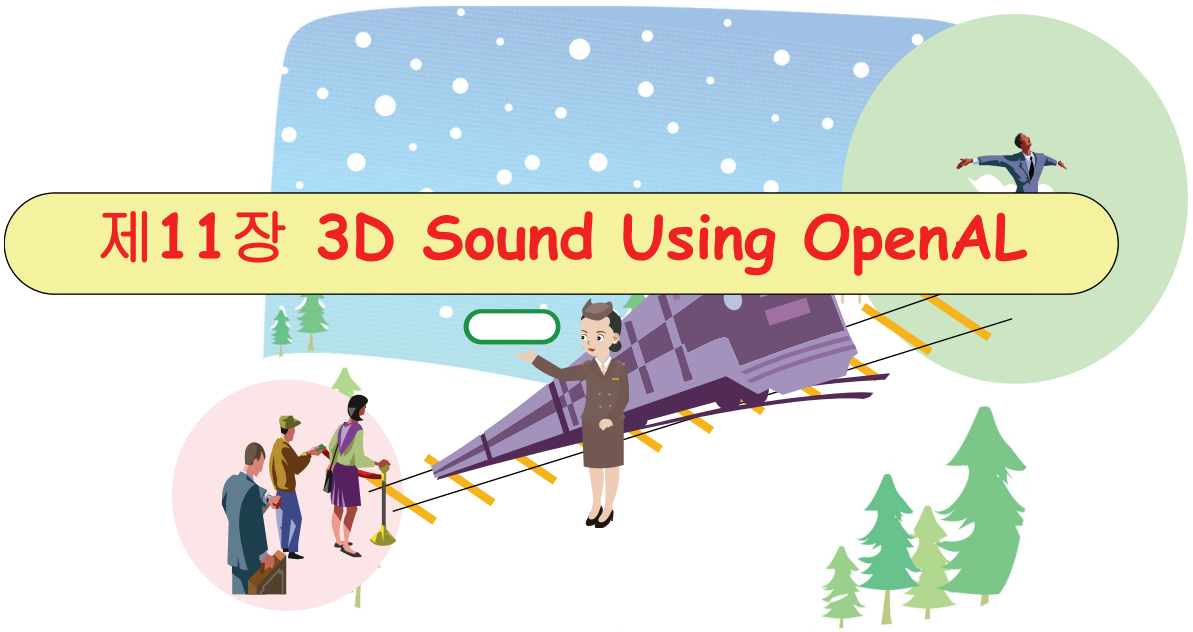
Sound Programming



제11장 3D Sound Using OpenAL

Principles of 3D Sound

- 3D 사운드가 무엇인지 알려면, *인간의 뇌가 소리를 어떻게 해석하는지 먼저 이해해야* 함.
- 소리의 방향을 탐지하기 위해 인간이 의지하는 단서들
  - Loudness
    - The louder the sound, the closer your brain thinks it is.
  - Interaural intensity difference(양 귀간 도달 강도 차이)
    - 음원과 가까운 쪽 귀에 더 강한 음이 들림으로써 음원의 방향을 탐지
  - Interaural time difference(양 귀간 도달 시간 차이)
    - 음원과 가까운 쪽 귀에 더 먼저 음이 들림으로써 음원의 방향을 탐지
  - Muffling
    - 우리 귀의 모양은 정면 음원을 더 잘 들을 수 있도록 생김.
- **OpenAL**에서는 위 요소들을 모두 지원.
  - *Listener & Sources*를 통해 "음원 방향"과 "귀의 위치"를 지정하기만 하면 위의 요소들을 감안한 3D 사운드를 재생해줌.

A Listener and Sources

- ***Listener: 사운드를 듣는 주체, Sources : 사운드를 생성하는 주체들***
  - 보통 Source는 여러 개이고, Listener는 한 명임.
  - Listener 속성과 Source 속성에 따라 재생 상태는 매번 새로 계산되어야 함.
  - 3D 사운드 시스템에서는 여러 소리가 존재하더라도 Listener가 없으면 소리는 안 나는 것임!
- **3D 환경에서 위치 별 사운드 처리의 예**
  - Listener (0, 0, 0), Source (-10, 0, 0) ➔ 소리는 대부분 왼쪽 스피커에서 나야 함.
  - Source 위치가 (-20, 0, 0) 이라면 역시 왼쪽 스피커에서 소리가 들리되 전보다 gain은 더 줄어 있어야 함.
  - 그렇다고 오른쪽 스피커에서 소리가 안 나면 안 됨.
    - 왼쪽에 비해 ***아주 잠깐의 지연 후***부터 소리가 나야 하고,
    - 왼쪽에 비해 ***gain도 미미하나마 낮아야*** 한다.

Source와 Listener의 속성

- **위치(Position): AL_POSITION**
  - Source와 Listener 모두 위치가 있음.
  - 좌표계 상의 값 세 개 (x, y, z)
  - OpenAL에게 Source와 Listener 모두의 위치를 알려야 함
- **속도(Velocity): AL_VELOCITY**
  - Source와 Listener 모두 속도가 있음.
  - 방향 벡터와 벡터 크기로 표현
    - 방향 벡터는 이전 위치와 현 위치를 알면 구할 수 있음
- **최소/최대 볼륨(Source에만 있는 특성)**
  - AL_MIN_GAIN
    - the minimum gain for this source
  - AL_MAX_GAIN
    - the maximum gain for this source

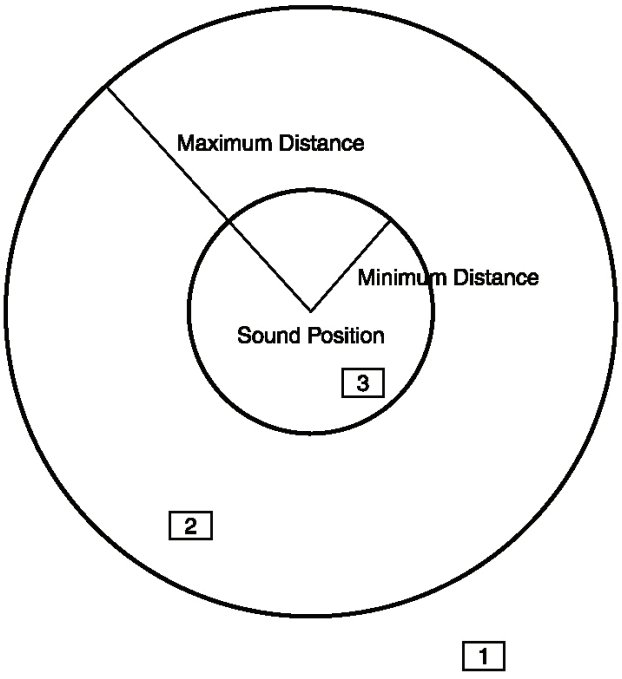
Source와 Listener의 특징 (계속)

- AL_ROLLOFF_FACTOR (Source에만 있는 특성)
  - Source의 소리 감쇠(또는 점증) 수준을 지정한다.
    - Listener가 음원으로부터 멀어질 때 얼마나 빨리 fade out 되는가?
    - Listener가 음원으로 가까워질 때 얼마나 빨리 fade in 되는가?
  - 즉, 소리가 사라지기까지(또는 최고 볼륨까지 올라올 때 까지) 걸리는 시간 수준을 지정한다는 의미
  - Rolloff factor 1.0 : 청취자가 소리에서 멀어져 감에 따른 소리 볼륨의 감쇠 정도를 실제 환경과 동일하게 하라.
  - Rolloff factor 0.5 : 실제 환경보다 감쇠 정도를 절반으로 하라. ➔ 실제에서는 음원에서 청취자가 40미터 떨어져야 안 들리게 되지만, OpenAL에서는 80미터 움직여야 안 들리게 하라.
  - Rolloff factor 0 : 청취자가 Source에서 얼마나 떨어져 있건 간에 Listener가 듣는 볼륨을 변화시키지 마라.
    - Listener는 항상 듣거나 항상 듣지 못할 것임.

Source와 Listener의 특징 (계속)

- 최대 거리(Source에만 있는 특성)
  - AL_MAX_DISTANCE : 이 거리보다 더 멀어지더라도 source의 소리 감쇠가 더 일어나지는 않는다.
- 참조 거리(Source에만 있는 특성)
  - AL_REFERENCE_DISTANCE : source의 볼륨이 절반으로 줄어드는 거리. 단, 최대 거리나 rolloff factor에 의해 영향을 받지 않는 상황에서.

A listener at position 1 will not hear the sound. A listener at position 2 will hear the sound at reduced volume. A listener at position 3 will hear the sound at full volume.



Source와 Listener의 특징 (계속)

- 방향(Orientation)(Listener에만 있는 특성)
  - AL_ORIENTATION
  - Listener는 방향이 있어서, 전면이 어디인지 알아야 함.
  - OpenAL은 Listener 방향을 이용해 후면으로부터의 소리는 억제되어 들리게 해준다.
  - 방향 지정 방법 : 2개의 벡터를 지정하면 됨.
    - 하나의 UP 벡터 + 하나의 전방(AT) 벡터
- 방향(DIRECTION)(Source에만 있는 특성)
  - AL_DIRECTION
  - Source도 방향이 있지만 UP 벡터는 필요 없음

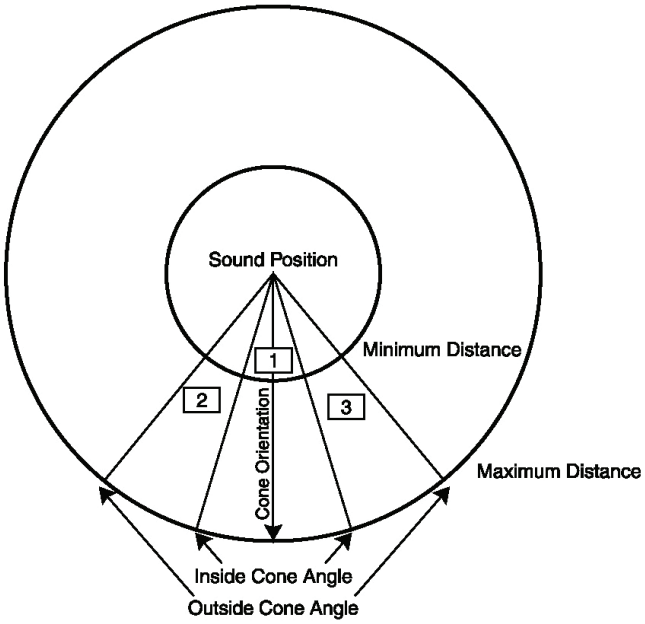
Source와 Listener의 특징 (계속)

- Sound Cones(Source에만 있는 특성)
  - Source는 방향은 없는 대신 소리가 미치는 범위인 sound cone 영역이 있음
  - AL_CONE_INNER_ANGLE
    - Inner cone 안에 있는 청취자는 온전하게 최대 크기로 들을 수 있음 (음원과의 거리만이 영향을 미칠 것임)
  - AL_CONE_OUTER_ANGLE
    - Inner cone 과 outer cone 사이의 청취자 : 음원과의 거리뿐만 아니라 cone 상에서의 위치도 청취 볼륨에 영향을 미침.
    - 디폴트는 360
  - AL_CONE_OUTER_GAIN
    - the gain when outside the oriented cone

Source와 Listener의 특징 (계속)

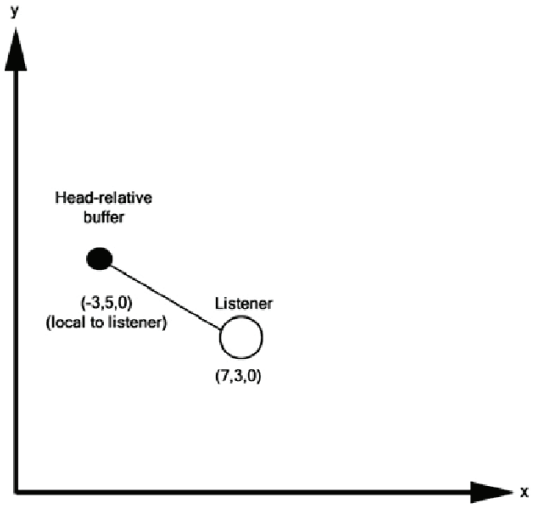
- **Sound Sphere**
  - Outer cone과 inner cone의 각도가 같고,
  - 그 각도가 360도 일 때
  - 즉, 사운드가 방향이 없을 때 Sound Sphere라고 함.

A listener at positions 2 or 3 will hear the sound at reduced volume, because even though they're inside the minimum distance, they're outside the inner cone. Only a listener at position 1 will hear the sound at full volume.



Source와 Listener의 특징 (계속)

- **AL_SOURCE_RELATIVE (Source에만 있는 특성)**
  - Source의 위치가 Listener 기준 상대 위치로 지정됨.
  - 디폴트는 AL_FALSE



In head-relative mode, buffer positions are given relative to the listener. In this example, the final position of the buffer is (4,8,0). If the listener were at position (100,200,0), the final position would be (197,205,0).

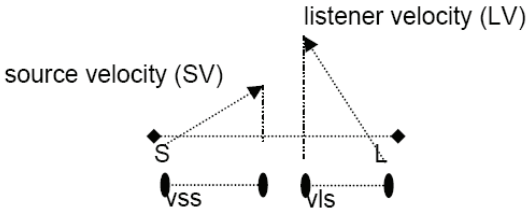
도플러(Doppler) 효과

- 소리 또는 물결이 점점 밀려오거나, 점점 멀어지는 현상
  - Christian Doppler: 1900년대 이 같은 현상을 설명하는 수학적 모델링을 제시한 물리학자 이름
    - 1803 ~ 1853. 오스트리아의 수학/물리학자. 찰스부르크 출생
- 도플러 효과의 예
  - 멀리서 다가오는 기차로부터 들리는 경적소리
  - 멀어져 가는 기차로부터 나오는 경적소리
  - 도플러 효과를 경험할 수 있는 소리 예:
- **OpenAL**에서도 도플러 효과를 지원함
  - ***Source/Listener*의 위치(position)**
  - ***Source/Listener*의 이동 속도(velocity)**

Doppler Shift

- 도플러 효과에 영향을 미치는 요소
  - source와 listener의 속도
  - 소리의 전파 속도
    - 매체에 따라 다를 수도 있음. 물 속? 공기 중?
- 도플러 효과의 정도 == **frequency shift(pitch change)**의 정도
  - Source와 Listener가 각자 방향으로 움직이는 속도와 비례함

Doppler Shift 계산 공식



3D Mathematical representation of vls and vss:

Mag(vector) = sqrt(vector.x * vector.x + vector.y * vector.y + vector.z * vector.z)
DotProduct(v1, v2) = (v1.x * v2.x + v1.y * v2.y + v1.z * v2.z)
SL = source to listener vector
SV = Source Velocity vector
LV = Listener Velocity vector

vls = DotProduct(SL, LV) / Mag(SL)
vss = DotProduct(SL, SV) / Mag(SL)

Dopper Calculation:

vss = min(vss, SS/DF)
vls = min(vls, SS/DF)
f' = f * (SS - DF*vls) / (SS - DF*vss)

Doppler Shift 프로그래밍 API

- 현 OpenAL Context 전체의 도플러 효과에 영향을 미치는 함수 2개
- void alDopplerFactor(ALfloat dopplerFactor);
  - dopplerFactor: 도플러 효과의 강도를 조절하기 위한 단순 보정 값.
  - 이 값이 음수면 무시됨 / 디폴트 값은 1.0
  - 현재 설정된 값은 *alGetFloat{v}* 에 AL_DOPPLER_FACTOR를 주고 얻어 볼 수 있음
- void alSpeedOfSound(ALfloat speed);
  - 소리의 전파 속도를 설정하기 위한 함수.
  - Source와 Listener의 속도는 같은 단위로 설정되어 있어야 함.
  - speed가 음수면 무시됨
  - 디폴트 값은 343.3 (속도 단위는 미터, 전파 매체는 공기 중이라고 가정)
  - 현재 설정된 값은 *alGetFloat{v}*에 AL_SPEED_OF_SOUND를 주고 얻어 볼 수 있음
- OpenAL 응용에서 Doppler 효과를 전혀 사용하고 싶지 않으면, 모든 속도 값을 0으로 해두면 됨.

Using a Listener

- 방해물이 없다면 **Listener**가 **Source** 위치와 가까울수록 더 큰 소리가 들리게 해야 함
- **3D** 사운드를 위한 **Listener** 속성: **Listener**의 위치와 방향
  - Listener가 이동 중이라면 매 loop 마다 위치와 속도(이동 거리), 방향을 변경해줘야 함
  - 위치: Listener가 현 좌표계에서 어디에 있는지.
  - 방향: Listener가 현 좌표계에서 어디를 바라보고 있는지.
- 예
  - Listener가 서쪽을 향하고 있고 Source가 북쪽에서 Listener를 향해 다가오고 있다면, 소리는 오른쪽에서 들려와야 함
  - 이 때 Listener가 방향을 바꿔 북쪽을 바라봤다면, 소리는 정면에서 들려오는 것으로 바뀌어야 함

Using a Listener (계속)

- **alListener()** 계열 함수
  - Listener 속성(위치와 방향 등)을 바꿀 때 사용
  - Listener 위치 바꾸기
    - alListener3f(): 3개의 floating point number를 따로 받아들임
    - alListenerfv(): floating point number들의 배열을 받아들임
    - parameters for the position are simply the x, y and z values for the listener.
  - OpenAL uses a left-handed coordinate system.
    - Negative z values, in OpenAL, descend into the screen.
  - Listener 방향 바꾸기
    - 인자가 6개 필요
    - 첫 3개: Listener가 바라보고 있는 방향 vector, 즉 at 벡터.
    - 다음 3개: vector leading straight up from where the listener is facing, 즉 up 벡터.

Programming 3D Sound

- **main() in 3Dmain.cpp**

```
#define  TEST_WAVE_FILE "beep.wav"          // 테스트에 사용할 모노 음원

// Listener 속성들
ALfloat ListenerPos[] = { 0.0, 0.0, 0.0 };
ALfloat ListenerVel[] = { 0.0, 0.0, 0.0 };
// at벡터(0,0,-1) == 화면 안으로 들어가는 방향, up벡터 (0,1,0) == y축 증가 방향
ALfloat ListenerOri[] = { 0.0, 0.0, -1.0,  0.0, 1.0, 0.0 };
// Source 속성 중 위치와 속도
ALfloat SourcePos[] = { 0.0, 0.0, 0.0 };
ALfloat SourceVel[] = { 0.0, 0.0, 0.0 };

ALuint uiSource;    // Source ID
ALuint uiBuffer;    // Buffer ID
ALubyte ch = ' ';
ALfloat time_unit, rdistance, ldistance, maxPos;  // Source 속도 조절에 사용되는 변수
ALfloat rolloff, maxDistance, deafult_max_distance;      // rolloff 조절과 최대거리 조절
ALfloat refDistance, deafult_ref_distance;      // 참조 거리 조절을 위해
ALfloat dopplerFactor, default_doppler_factor;            // 도플러 효과 조절을 위해
ALfloat soundSpeed, default_sound_speed;      // 소리의 속도 조절을 위해
ALfloat cone_outer_gain, cone_inner_angle, cone_outer_angle; // sound cone 조절
```


main() in 3Dmain.cpp (계속)

```
.....
// Initialize Framework
ALFWInit();
ALFWprintf("3D Sound Sample Application Starts.\n");

if (!ALFWInitOpenAL()) {
    ALFWprintf("Failed to initialize OpenAL\n"); ALFWShutdown();
    return 0;
}

alGenBuffers( 1, &uiBuffer );
if (alGetError() != AL_NO_ERROR)  return AL_FALSE;

if (!ALFWLoadWaveToBuffer((char*)ALFWaddMediaPath(TEST_WAVE_FILE), uiBuffer)) {
    ALFWprintf("Failed to load or attach %s\n", ALFWaddMediaPath(TEST_WAVE_FILE));
    alDeleteBuffers(1, &uiBuffer); ALFWShutdownOpenAL(); ALFWShutdown();
    return AL_FALSE;
}

// Generate source object
alGenSources( 1, &uiSource );
if (alGetError() != AL_NO_ERROR) {
    ALFWprintf("Failed to generate a source.\n");
    alDeleteBuffers(1, &uiBuffer); ALFWShutdownOpenAL(); ALFWShutdown();
    return AL_FALSE;
}
```

main() in 3Dmain.cpp (계속)

```
// 3D 사운드를 위한 초기 소스 속성 설정
alSourcei(uiSource, AL_BUFFER,   uiBuffer);
alSourcef(uiSource, AL_PITCH,    1.0f     );
alSourcef(uiSource, AL_GAIN,     1.0f     );
alSourcefv(uiSource, AL_POSITION, SourcePos);        // 초기 Source의 위치
alSourcefv(uiSource, AL_VELOCITY, SourceVel);        // 초기 Source의 속도
alSourcei (uiSource, AL_LOOPING,  AL_TRUE);

if (alGetError() != AL_NO_ERROR) {
   ALFWprintf("Failed to set the source attributes.\n");
   alDeleteBuffers(1, &uiBuffer);
   alDeleteSources(1, &uiSource);
   ALFWShutdownOpenAL();
   ALFWShutdown();
   return AL_FALSE;
}
```


main() in 3Dmain.cpp (계속)

```
// 소스 설정 값 중 디폴트 값을 저장해둘 필요가 있는 것을 얻어둔다.
alGetSourcef (uiSource, AL_MAX_DISTANCE, &deafult_max_distance);
alGetSourcef (uiSource, AL_REFERENCE_DISTANCE, &deafult_ref_distance);
default_doppler_factor = alGetFloat (AL_DOPPLER_FACTOR);
default_sound_speed = alGetFloat (AL_SPEED_OF_SOUND);
alGetSourcef (uiSource, AL_CONE_OUTER_GAIN, &cone_outer_gain);
alGetSourcef (uiSource, AL_CONE_INNER_ANGLE, &cone_inner_angle);
alGetSourcef (uiSource, AL_CONE_OUTER_ANGLE, &cone_outer_angle);

// 3D 사운드를 위한 Listener 속성 설정
alListenerfv (AL_POSITION,    ListenerPos);
alListenerfv (AL_VELOCITY,    ListenerVel);
alListenerfv (AL_ORIENTATION, ListenerOri);

time_unit = 40.0f;  // 40 millisecond. 1000으로 나누면 초 단위가 나올 것임.
rdistance = 0.4f;   // 40 ms 동안 0.4미터 만큼 이동한다는 의미
ldistance = 0.2f;   // 40 ms 동안 0.2미터 만큼 이동한다는 의미
maxPos = 20.0f;     // 20 미터 이동하고 나면 방향을 바꿔라.

// 재생 시작
alSourcePlay(uiSource);
```

main() in 3Dmain.cpp (계속)

```
ALFWprintf("아무 키나 누르시면 Source가 x축 좌우 이동을 시작합니다:");
ALFWGetKey();

while (1) {
   /*
    * Source가 x 축을 좌우로 왔다 갔다 하는 소리를 흉내내보자.
    * 이 때, Listener는 원점 (0, 0, 0)에 있는 것으로 한다.
    */
   // x축 + 방향 소스의 속도를 계산한다. 단위 시간 time_unit을 초로 환산한 시간 동안
   // rdistance 미터 만큼 x축 +방향으로 움직이는 상황을 시뮬레이션 한다.
   SourceVel[0] = rdistance / (time_unit/1000);
   alSourcefv (uiSource, AL_VELOCITY, SourceVel);
   while (1) {
      ALFWprintf("Source Pos: (%.1f %.1f %.1f), Source Vel: (%.1f %.1f %.1f)\r",
                  SourcePos[0], SourcePos[1], SourcePos[2],
                  SourceVel[0], SourceVel[1], SourceVel[2]);
      Sleep((DWORD)time_unit);        // 단위 시간 동안 소리가 충분히 나도록 한다.
      SourcePos[0] += rdistance;      // 단위 시간이 지났으면 Source는 이동했을 것임
      alSourcefv (uiSource, AL_POSITION, SourcePos);
      if (SourcePos[0] >= maxPos)     // 너무 멀리 이동하지 않도록 제한을 가한다.
         break;
   }
   ……
```


main() in 3Dmain.cpp (계속)

```
   ……
   // x축 - 방향 소스의 속도를 계산한다.
   SourceVel[0] = -ldistance / (time_unit/1000);// x축 - 방향이므로 속도는 음수이어야 함
   alSourcefv (uiSource, AL_VELOCITY, SourceVel);
   while (1) {
      Sleep((DWORD)time_unit);
      SourcePos[0] -= ldistance;      // 왼쪽으로 이동했으므로 x축 위치 값을 감소시킴.
      alSourcefv (uiSource, AL_POSITION, SourcePos);
      ALFWprintf("Source Pos: (%.1f %.1f %.1f), Source Vel: (%.1f %.1f %.1f)\r",
                  SourcePos[0], SourcePos[1], SourcePos[2],
                  SourceVel[0], SourceVel[1], SourceVel[2]);
      if (SourcePos[0] <= -maxPos)
         break;
   }
```

main() in 3Dmain.cpp (계속)

```
// 숫자 키 1 ~ 6 각각에 기능 하나씩 시험해보자.
if (ALFWKeyPress()) {
    static char nextDir = 0;
    static char nextRolloff = 0;
    static char nextMaxdistance = 0;
    static char nextRefdistance = 0;
    static char nextDoppler = 0;
    static char nextCone = 0;

    ch = ALFWGetKey();
    switch (ch) {
    case '1':          // Listener의 Orientation 바꾸기
        nextDir = (nextDir + 1) % 4;
        switch (nextDir) {
        case 0: ListenerOri[0] = 0; ListenerOri[1] = 0; ListenerOri[2] = -1; break;
        case 1: ListenerOri[0] = 1; ListenerOri[1] = 0; ListenerOri[2] = 0; break;
        case 2: ListenerOri[0] = 0; ListenerOri[1] = 0; ListenerOri[2] = 1; break;
        case 3: ListenerOri[0] = -1; ListenerOri[1] = 0; ListenerOri[2] = 0; break;
        }
        alListenerfv (AL_ORIENTATION, ListenerOri);
        // 소스의 위치는 (0, 0, 0)으로 리셋한다
        SourcePos[0] = 0; SourcePos[1] = 0; SourcePos[2] = 0;
        alSourcefv (uiSource, AL_POSITION, SourcePos);
        ALFWprintf("\nListener Orientation: (%.1f %.1f %.1f)\n", ListenerOri[0], ListenerOri[1], ListenerOri[2]);
        break;
```


main() in 3Dmain.cpp (계속)

```
    case '2':       // AL_ROLLOFF_FACTOR 기능 확인하기
        nextRolloff = (nextRolloff + 1) % 4;
        switch (nextRolloff) {
        case 0: rolloff = 1.5f; break; // 거리에 따른 감쇠가 50% 더 많이 일어나게
        case 1: rolloff = 1.0f; break;  // 디폴트
        case 2: rolloff = 0.5f; break; // 거리에 따른 감쇠가 50% 더 적게 일어나게
        case 3: rolloff = 1.0f; break; // 디폴트
        }
        alSourcef (uiSource, AL_ROLLOFF_FACTOR, rolloff);

        // 현재의 rolloff factor 값을 얻어 출력한다.
        alGetSourcef (uiSource, AL_ROLLOFF_FACTOR, &rolloff);
        ALFWprintf("\nRoffoff factor: %.1f\n", rolloff);
        break;
```

main() in 3Dmain.cpp (계속)

```
case '3':       // AL_MAX_DISTANCE 기능 확인하기
    // 현재의 max. distance를 얻는다.
    alGetSourcef (uiSource, AL_MAX_DISTANCE, &maxDistance);
    ALFWprintf("\nAL_MAX_DISTANCE changed: %.1f --> ", maxDistance);
    nextMaxdistance = (nextMaxdistance + 1) % 4;
    switch (nextMaxdistance) {
    case 0: maxDistance = 1.0f; // 1미터가 지나면 더 이상 볼륨 감쇠가 일어나지 않게
            break;
    case 1: maxDistance = 3.0f; // 3미터가 지나면 더 이상 볼륨 감쇠가 일어나지 않게
            break;
    case 2: maxDistance = 5.0f; // 5미터가 지나면 더 이상 볼륨 감쇠가 일어나지 않게
            break;
    case 3: maxDistance = deafult_max_distance; break;      // 디폴트로 환원
    }
    alSourcef (uiSource, AL_MAX_DISTANCE, maxDistance);
    ALFWprintf("%.1f\n", maxDistance);
    break;
```


main() in 3Dmain.cpp (계속)

```
case '4':       // AL_REFERENCE_DISTANCE 기능 확인하기
    // 현재의 ref. distance를 얻는다.
    alGetSourcef (uiSource, AL_REFERENCE_DISTANCE, &refDistance);
    ALFWprintf("\nAL_REFERENCE_DISTANCE changed: %.1f --> ", refDistance);

    nextRefdistance = (nextRefdistance + 1) % 4;
    switch (nextRefdistance) {
    // 3미터가 지나면 무조건 볼륨이 원 볼륨의 절반으로 떨어지게.
    case 0: refDistance = 3.0f; break;
    case 1: refDistance = 5.0f; break;      // 5미터          ;;
    case 2: refDistance = 7.0f; break;      // 7미터          ;;
    case 3: refDistance = deafult_ref_distance; break;   // 디폴트인 1미터로 환원.
    }

    alSourcef (uiSource, AL_REFERENCE_DISTANCE, refDistance);
    ALFWprintf("%.1f\n", refDistance);
    break;
```

main() in 3Dmain.cpp (계속)

```
        case '5':       // AL_DOPPLER_FACTOR 기능 확인하기
                dopplerFactor = alGetFloat (AL_DOPPLER_FACTOR); // 현 doppler factor 값
                soundSpeed = alGetFloat (AL_SPEED_OF_SOUND);  // 현 sound speed 값
                nextDoppler = (nextDoppler + 1) % 6;
                switch (nextDoppler) {
                case 0: dopplerFactor = default_doppler_factor; break;   // 디폴트인 1.0으로
                case 1: dopplerFactor = default_doppler_factor * 3.0f; break; // 도플러 효과 3배
                case 2: dopplerFactor = default_doppler_factor * 4.0f; break; // 4배로.
                case 3: dopplerFactor = default_doppler_factor * 0.5f; break; // 절반으로.
                case 4: dopplerFactor = default_doppler_factor;
                        soundSpeed = default_sound_speed / 2.0f;         // 소리 속도 감소
                        break;
                case 5: dopplerFactor = default_doppler_factor;
                        soundSpeed = default_sound_speed * 2.0f;         // 소리 속도 증가
                        break;
                }
                alDopplerFactor (dopplerFactor);
                alSpeedOfSound (soundSpeed);
                ALFWprintf("\nAL_DOPPLER_FACTOR: %.1f", dopplerFactor);
                ALFWprintf("\nAL_SPEED_OF_SOUND: %.1f\n", soundSpeed);
                break;
```

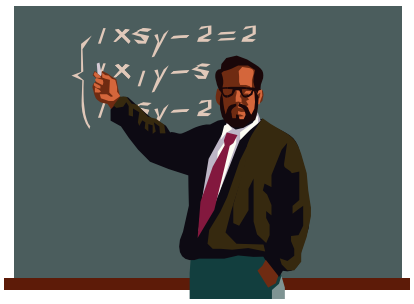

main() in 3Dmain.cpp (계속)

```
        case '6':       // SOUND CONE 기능 확인하기
                ALFWprintf("\nAL_CONE_OUTER_GAIN: %.1f", cone_outer_gain);
                ALFWprintf("\nAL_CONE_INNER_ANGLE: %.1f", cone_inner_angle);
                ALFWprintf("\nAL_CONE_OUTER_ANGLE: %.1f\n", cone_outer_angle);
                nextCone = (nextCone + 1) % 5;
                switch (nextCone) {
                case 0: cone_outer_angle = 360.0f; break;
                case 1: cone_outer_angle = 270.0f; break;
                case 2: cone_outer_angle = 180.0f; break;
                case 3: cone_outer_angle = 90.0f; break;
                case 4: cone_outer_angle = 0.0f; break;
                }
                alSourcef (uiSource, AL_CONE_OUTER_ANGLE, cone_outer_angle);
                ALFWprintf("\nAL_CONE_OUTER_ANGLE: %.1f\n", cone_outer_angle);
                break;
        } // end of switch(ch)
    } // end of if (ALFWKeyPress())
} // end of while (1)
```

11장 보고서

- 11장에서 소개된 3D 사운드 관련 옵션들을 사용해보고 기말 프로젝트의 어느 부분에 적용할 지 설계하시오.

Q & A



인생은 음악처럼

- http://video.nate.com/204114790