용어정리

- 실시간 오디오 스펙트럼 처리 용도

- AudioSource.GetSpectrumData

AudioSource.GetSpectrumData(samples, channel, window)

>> samples : 오디오 샘플을 담을 배열. 2의 제곱꼴이어야 한다. 여기서는 배열 한 칸을 bin이라고 한다. 음역대 구분을 더 미세하게 하고 싶은 경우 samplse 배열의 크기 증가

>> channel : 샘플링할 채널, 0으로 하는경우 스테레오 샘플을 평균내서 저장

>> Window : 소리를 퓨리에 변환(FFT)을 통해 각 음역대로 구분할 때, 사용할 변환 방식 설정

AudioSource의 GetSpectrumData를 통해 특정 시간대의 소리로부터 음역대를 분리해낼 수 있다.

이때 GetSpecturmData를 통해 반환되는 음역대의 bin은 실제 주파수 bin의 절반이다.

(48khz일 때 0-24khz는 24-48khz의 미러이기 때문에 0-24khz에 대한 진폭만 반환한다)

>> 따라서 23.43hz마다 하나의 빈에 저장되도록 samplse의 크기가 1024인 경우

>> 2048개의 오디오 샘플이 필요하다

- AudioSettings.outputSampleRate

믹서의 샘플 속도를 반환

오디오 클립의 샘플링 속도가 아니라 이것을 기준으로 해야한다

- AudioSource.clip.frequency

클립의 진동수

정리

>> 오디오로부터 샘플링 속도를 받아온다

>> 해당 샘플링 속도의 절반이 퓨리에 변환을 통해 얻을 범위

>> 샘플링 속도 / 2 / samplse의 길이(여기서는 1024) - bin당 23.47hz 저장

>> bin = samples[i]가 해당 음역대의 볼륨을 나타낸다(아마도)

- Spectral Flux

스펙트럼 데이터의 가까운 두 지점 사이에 bin 당 객체의 수 차이를 구하는 것

현 시간과 이전 시간 사이의 bin 당 값 차이를 통해 해당 음역대의 볼륨이 얼마나 변화하는지를 파악한다

- 이전 프레임의 스펙트럼과 현재 프레임의 스펙트럼 비교

예제에서는 prev스펙트럼과 cur스펙트럼 사이의 각 bin의 차이를 구하고, 차이가 0보다 큰 경우 누적합에 더해간다

해당 누적합을 통해 해당 시간 차이 동안 유의미한 볼륨 증가가 있는지 - 비트로 간주할만한지 - 판단하는데 활용할 것

- 비트 판단하기

전체 스펙트럼 플럭스의 볼륨 증가와 비교하여 유의미한 증가인 경우 비트로 판단한다

판단 기준은

>> 스펙트럼 플럭스의 값의 프레임 당 평균을 계산 > 판단 기준이 되는 상수 -감도승수- 를 곱한다

>> 평균 볼륨 증가보다 크다면 비트로 간주

>> 예제에서는 앞뒤로 15프레임을 비교

>> 구현 -

>> 현재 스펙트럼플럭스를 기준으로 앞 뒤로 x개의 샘플들로부터 스펙트럴 플럭스 누적합 구함

>> 점검한 샘플들의 개수로 나누어 평균 구함

>> 평균에 비트 측정할 상수 곱함

>> 해당 방식의 단점

>> 나는 특정 비트에 노트가 나오도록 하고 싶음

>> 주변 볼륨보다 튀는 볼륨을 노트라고 친다면

>> 해당 방식은 과거의 누적 사운드보다 튀는 경우만 체크 가능함

>> 따라서 전처리 후 현재 위치 기준 앞뒤의 사운드와 비교하도록 해야함

- 전처리 오디오 스펙트럼 용도

- AudioClip.GetData(float[] data, int offsetSamples)

data : float형 배열인 data에 오디오의 샘플 전체를 저장. 전체 샘플 수는 data의 길이에 의해 결정

offsetSamples : 시작위치

- AudioClip.Samples

클립의 채널 샘플의 개수

스테레오(2채널) 샘플이 100개인 경우 AudioClip.Samples의 값은 500

- AudioClip.channels

오디오 클립의 채널 수(스테레오, 모노 등)

>> Samplse와 channels를 종합하면 오디오의 전체 샘플 수는 채널 \* 채널 샘플 수

@@ 스테레오 타입의 경우 채널마다 사운드가 다르다

>> 따라서 두 채널의 평균을 내야 모노 샘플과 유사한 결과값이 나온다

@@ 오디오의 전체 샘플 수

샘플링 레이트(e.g 441000) \* 음악재생시간(초단위) \* 채널수

샘플링 레이트 \* 음악재생시간 = audio.clip.samples