Capturing User's Speech (Utterance) via Microphone in Unity

120220354 윤수인

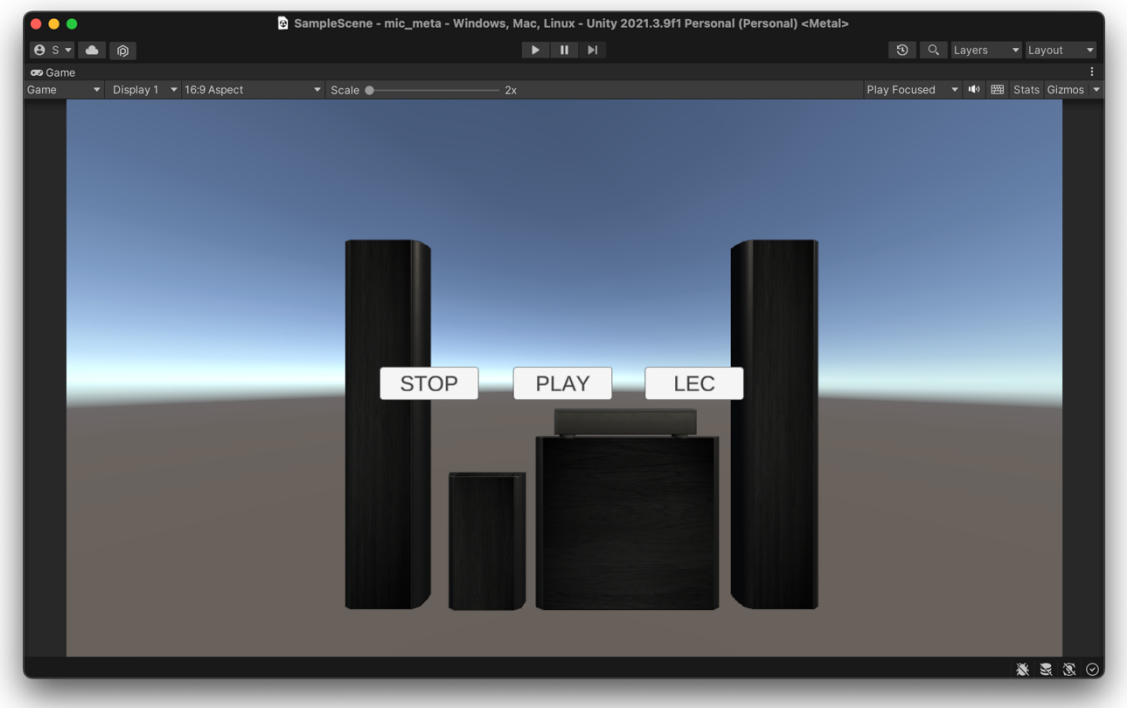
0. 목차

- GUI
- 구조
- Script 분석

1. GUI

: GUI에서 각 버튼을 통해 USER가 직접 음성을 입출력할 수 있도록 하였다.

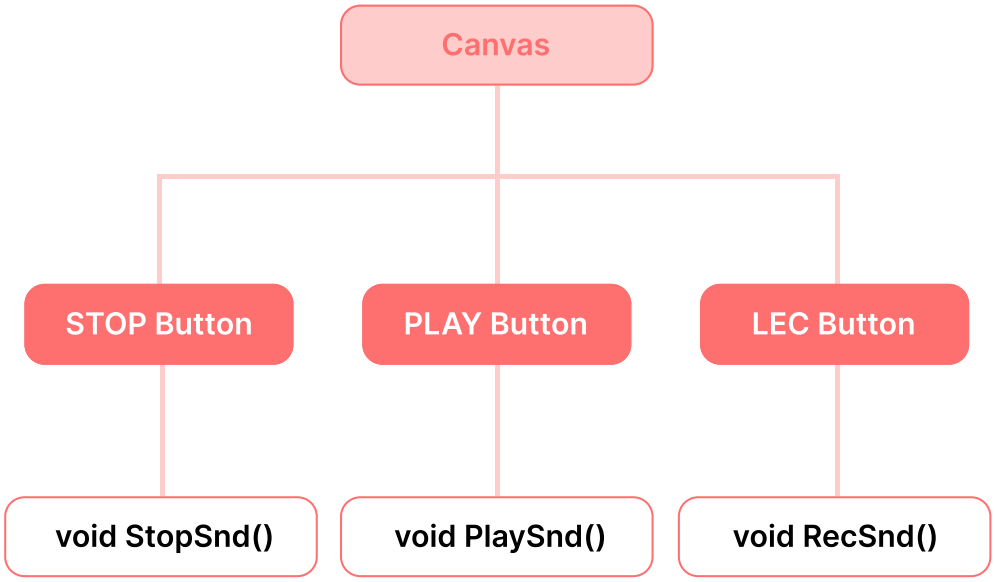
- STOP Button : 음성 입출력 처리를 일시 정지 시킨다.
- PLAY Button : 실시간으로 Mic에서 음성을 입력받아 Audio로 출력한다.
- REC Button : 음성입력을 시작하고, 초기화한다.



2. 구조

하단의 이미지와 같은 구조로 Button들이 Script의 함수에 접근할 수 있도록 하고 유니티에서 각 Button들에 OnClick을 걸어주고 해당 메소드와 연결시키면 Button 클릭 시 Event가 발생하도록 하였다.

- STOP Button : void Stop()
- PLAY : void PlaySnd()
- REC : void RecSnd()



3. Script (mic_captureSpeech)

```
public class mic_captureSpeech : MonoBehaviour
{
    private AudioSource source; // 오디오 클립 컨트롤에 필요한 오디오 소스 변수 선언
    private AudioClip mic; // 재생할 오디오 클립 파일 변수 선언
    private int lastSample = 0; // 직전 프레임에서의 pos 값 변수
    float[] samples = null; // 오디오 샘플 저장할 공간
    private List<float> readSamples = null; // 데이터 유실을 막기 위한 샘플 데이터 저장
    private int channels = 0; // mic 채널 수
    private int readUpdateId = 0;
    private int previousReadUpdateId = -1;


    public float READ_FLUSH_TIME = 0.5f; // 이전 프레임에서 매 프레임마다 누적된 시간
    private float writeFlushTimer = 0.0f; // readMic 에서 매 프레임마다 누적되는 시간
    private float readFlushTimer = 0.0f; // playMic 에서 매 프레임마다 누적되는 시간
```

오디오 입출력시 필요한 변수 선언

```
void Start() // 스크립트 시작 시 1 회 실행
    {
        readSamples = new List<float>(); // 데이터 유실을 막기 위한 샘플 데이터 저장
        source = new GameObject("Mic").AddComponent<AudioSource>(); // 오디오 소스를
```

받는 게임오브젝트 Mic 정의

```
      mic = Microphone.Start("Microphone (High Definition Audio Device)", true,
100, 44100); // deviceName(기기명), loop(반복 여부), lengthSec(오디오클립 길이),
frequency(samplerate)의 parameters 를 입력하여 녹음을 시작하는 mic 변수 정의
      channels = mic.channels;//mono or stereo, for me it's 1 (k)
   }
```

```
void Update() // 매 프레임당 1 회 실행
   {
      ReadMic(); // 실시간 오디오 샘플 읽는 함수
      PlayMic(); // 오디오 샘플 출력하는 함수

}
```

```
private void ReadMic(){ // 실시간 오디오 샘플 읽기
         writeFlushTimer += Time.deltaTime; // 한 프레임을 진행하는데 걸린 시간 매
프레임마다 writeFlushTimer 에 누적하여 더함
         int pos = Microphone.GetPosition(null); // 가장 최근의 오디오 샘플 위치
         int diff = pos - lastSample; // 한 프레임마다 오디오 buffer 에 새로 추가되는
공간 크기

         if (diff > 0) // 현재 녹음 위치 – 이전 프레임 녹음 위치가 0 보다 크면
         {
            samples = new float[diff * channels]; // 오디오 샘플 저장할 공간
            mic.GetData(samples, lastSample); // lastSample 부터 pos 까지의 샘플을
samples 에 저장

            readSamples.AddRange(samples); // readSamples 에 오디오 샘플 추가
         }
         lastSample = pos; // 다시 update()가 호출되면 직전 최신 위치에서 읽도록
      }
```

```
private void PlayMic(){ // 오디오 샘플 출력
      readFlushTimer += Time.deltaTime; // 한 프레임을 진행하는데 걸린 시간 매
프레임마다 readFlushTimer 에 누적하여 더함

         if (readFlushTimer > READ_FLUSH_TIME) // 기존의 값인 0.5f 보다 크고
         {
            if (readUpdateId != previousReadUpdateId && readSamples != null &&
readSamples.Count > 0)  // readUpdateId 가 이전의 것과 다르고, readSamples 값이 비어있지
않고, 0 보다 크면 입력받은 오디오 샘플 출력
            {
               //Debug.Log("Read happened");
```

```
                previousReadUpdateId = readUpdateId; // 다시 update()가 호출되면 직전
최신 위치에서 출력하도록
                source.clip = AudioClip.Create("Real_time", readSamples.Count,
channels, 44100, false); // source.clip 으로 샘플 수, 채널, samplerate, loop 입력 받음
                source.spatialBlend = 0 // 최소값에 가까우므로 2D(최대값일수록 3D)

                source.clip.SetData(readSamples.ToArray(), 0); // 리스트형
        readSamples 를 Array 로 변환하고 source.clip 에 할당
                if (!source.isPlaying){ // setData false 이므로 모든 샘플을 읽음
                    //Debug.Log("Play!");
                    source.Play(); // 오디오 샘플 출력
                }
                readSamples.Clear(); // readSamples 에 있던 샘플 초기화
                readUpdateId++; // 1 씩 올라감
            }

            readFlushTimer = 0.0f; // 초기화
        }
```

audioClip buffer는 array 타입이라서 크기가 정해져 있다 -> 무한정 녹음할 수 없다
Loop = true 로 하였을 때 circular buffer처럼 작동하여 오디오 샘플들을 buffer의 끝(44100*100)까지 쓰면 다시 처음으로 돌아가 쓰는 방식으로 실시간 녹음을 구현할 수 있다.
그러나 이전에 buffer에 썼던 샘플을 스크립트 코드가 읽기 전에 덮어쓰게 되면 데이터가 유실된다.
데이터가 유실되지 않으려면 다른 공간에 저장해 두어야 한다.(=readSamples)
이 공간은 array 타입인 audioClip buffer와 달리 list 타입으로 크기가 정해져 있지 않다.