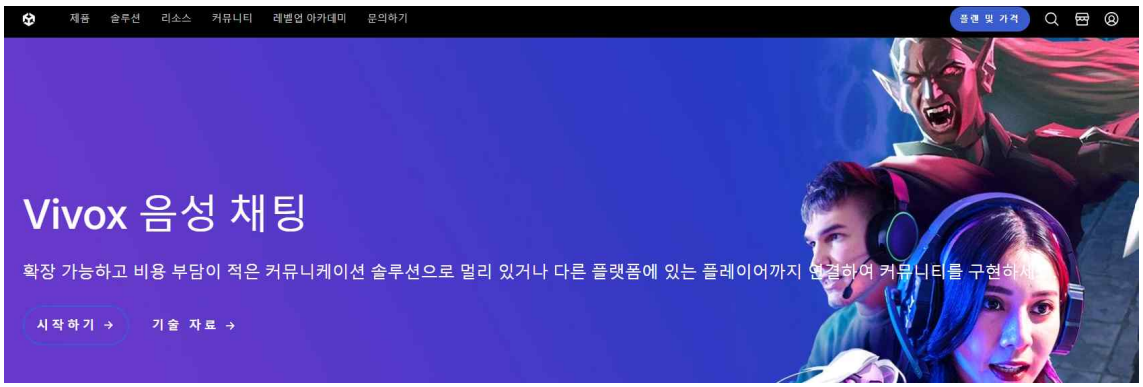
〈Vivox〉

1. Vivox 소개

- Vivox는 멀티플레이어 게임 내 음성/텍스트 채팅 서비스.
- Unity Gaming Services(UGS)와 연동해 로그인, 음성 채널 접속, 3D 포지셔널 오디오 지원.
- 사용처: 실시간 협동 게임, MMO, 파티 기반 게임 등.



2. 준비 사항

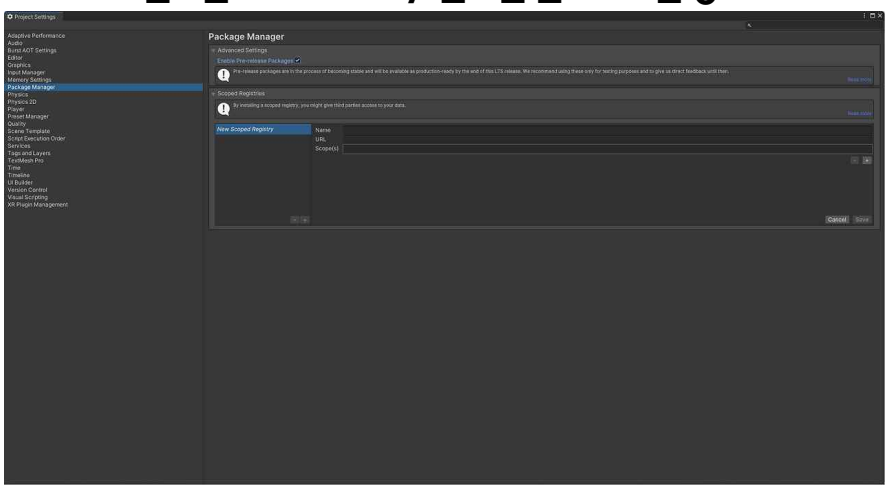
- Unity 프로젝트와 Unity Services 연결.
- Unity Dashboard – Vivox Voice and Text Chat 활성화.
- Package Manager에서 'com.unity.services.vivox' 설치.
- Authentication 패키지 설치 (익명 로그인 or 사용자 계정 기반 로그인).

3. Vivox 초기화 및 로그인

```
using UnityEngine;
using Unity.Services.Core;
using Unity.Services.Authentication;
using Unity.Services.Vivox;
using System;
using System.Threading.Tasks;
public class VivoxController : MonoBehaviour
{    public static VivoxController Instance { get; private set; }
    private void Awake()
    {
        DontDestroyOnLoad(gameObject);
        Instance = this;
    }
    private async void Start()
    {
        await InitializeVivoxAsync();
        await LoginAsync();
    }
    private async Task InitializeVivoxAsync()
    {
        await UnityServices.InitializeAsync();
        await AuthenticationService.Instance.SignInAnonymouslyAsync();
        await VivoxService.Instance.InitializeAsync();
    }
    private async Task LoginAsync()
    {
        var options = new LoginOptions { DisplayName = Guid.NewGuid().ToString() };
        await VivoxService.Instance.LoginAsync(options);
    }
}
```

– ‘DontDestroyOnLoad’으로 유지.

– 로그인 완료 후 UI/씬 전환 시 활용.



4. 그룹 음성 채팅

```
await VivoxService.Instance.JoinGroupChannelAsync("lobby", ChatCapability.AudioOnly);
```

- 채널이 없으면 자동 생성.
- 씬 전환 시 로그인 완료 이벤트와 연계.

5. 3D 위치 기반(포지셔널) 음성 채팅

- Channel3DProperties 설정:
  - ‘audibleDistance’: 최대 청취 거리
  - ‘conversationalDistance’: 정상 대화 거리
  - ‘audioFadeIntensityByDistance’: 거리 감쇠 강도
  - ‘audioFadeModel’: 감쇠 모델(예: InverseByDistance)

```
var props = new Channel3DProperties(32, 5, 1, AudioFadeModel.InverseByDistance);
await VivoxService.Instance.JoinPositionalChannelAsync("field", ChatCapability.AudioOnly, props);
```

- 주기적으로 플레이어 위치 업데이트:

```
VivoxService.Instance.Set3DPosition(transform.position, transform.forward, Vector3.up);
```

6. 코드 구조 및 Best Practices

- Singleton + DontDestroyOnLoad 패턴으로 중앙 관리.
- 이벤트 기반: 로그인 완료 → 채널 접속.
- 채널/닉네임 설정은 Inspector나 Config 파일에서 분리 관리.
- 위치 업데이트 주기는 성능 고려 (예: 0.1\~0.5초 간격).

7. 보안 및 운영 고려사항

- 로그인: 데모는 익명 로그인, 실제 서비스는 계정 연동 필요.
- 플랫폼 권한: 마이크 접근 권한(iOS/Android) 필수.
- 네트워크/비용: 채널 수, 접속자 수, 대역폭에 따라 관리.
- 예외 처리: 접속 실패, 네트워크 끊김, 재접속 로직 필요.

8. 테스트 & 디버깅

- 에디터 + 빌드 환경 모두 확인.
- 여러 클라이언트 동시 접속 테스트.
- 포지셔널 채팅 거리값 실시간 테스트.
- 로그와 상태 UI로 Vivox 상태 추적.

9. 배포 체크리스트

- Unity Dashboard에서 Vivox 활성화 확인.
- 마이크 권한 요청 처리.
- DisplayName 정책 준수.
- 플랫폼별 성능/배터리 최적화 검증.
- 포그라운드/백그라운드 상태 동작 확인.

[참고자료: 티스토리]

1. 사용 준비
2. 초기화와 로그인
3. 음성 채팅 구현
4. 3D 위치 음성 채팅

[Unity 공식사이트]

1. 공식 매뉴얼
2. 공식 페이지

[추가 질문]

1. 개념
2. 튜토리얼