겜녹이 제작 완료 보고서

Game Voice Recorder (GVR) · M0 기술 검증부터 M8 파인튜닝까지 전 단계



작성일	2026년 10월 15일
저장소	https://git.tkrmagid.kr/tkrmagid/game_voice_recorder (main, 커밋 24585e3)
사이트	https://gvr.tkrmagid.kr (운영 중)
검증 환경	개발 서버(리눅스, RTX 5050 8GB) · Windows 11 VM(win-ci) · 디스코드 F&L 서버 "음성" 채널
결과물	GameVoiceRecorder-1.0.0-portable.exe (단일 파일) · GameVoiceRecorder-1.0.0-win-x64.zip (폴더) · 사이트 서버(Docker)

M0 ✓ M1 ✓ M2 ✓ M3 ✓ M4 ✓ M5 ✓ M6 ✓ M7 ✓ M8 ✓

1 요약

계획서의 M0~M8 아홉 단계를 모두 만들었습니다. 단계가 끝날 때마다 Gitea 저장소에 push하고 README 진행표를 고쳤습니다. 단계마다 자동 테스트를 돌렸고, 실제 디스코드 음성채널, Windows VM, 이 서버 GPU에서도 직접 확인했습니다.

3.6%
실제 디스코드 음성 28문장 글자 오류율(CER), GPU large-v3-turbo

1.05초
말이 끝난 뒤 글자가 나오기까지 (중앙값, 최대 1.86초)

1.8%
파인튜닝 모델 CER (기본 모델 4.8%, 사전 없이)

3/3
Windows 롤 채팅 OCR 예시 3줄 모두 인식

단계	만든 것	확인 방법 · 결과
M0 기술 검증	구조 설계, STT 워커(faster-whisper), 앱 코어, 게임 PC 진단 기능	DAVE 음성 연결 1.06초, 음량 정규화·잡음 판정·사전 교정 단위 테스트
M1 녹음 + STT 기본	따라가기·명령 참여, TTS 안내, 유저별 Ogg 녹음, 대기열 STT, /녹음제외	실제 음성 E2E 통과 (3문장 → 3발화, 녹음 제외 즉시 반영)
M2 사이트 서버 · 사이트 v1	디스코드 로그인(서버 멤버만), 게임 PC 등록·중계·잠금, 타임라인·검색·실시간·따라 읽기	gvr.tkrmagid.kr 배포, 서버 통합 테스트, 실서버 WebSocket 연결
M3 발로란트 연동	경기 감지, 킬·라운드·채팅, 라운드 시간 + 경기 경과, 경기 상세(미니맵·킬피드·스코어보드)	가짜 Riot 서버 테스트, 22라운드 전체 스택 시연
M4 롤 연동 · 채팅 OCR	게임 클라이언트 API 이벤트, LCU 결과·타임라인, 채팅 OCR·r3dlog, 롤 경기 상세	Windows 실기: OCR 3/3, Windows 전용 문제 3개 수정
M5 exe · 설정 · 모델	포터블 exe, 트레이, 설정 창 8탭, 첫 실행 마법사 5단계, GPU 추천·모델 다운로드, DPAPI 암호화	Windows 빌드·실행 스모크 전 항목 통과
M6 단어사전 · UI/UX	기본 사전 약 2,000단어 + 교정 규칙, 사이트 사전 화면, 게임 패치 이름 자동 추가, 테마·효과음	사전 적용 시 CER 4.8% → 3.9%
M7 실사용 테스트	GPU 실음성 시험, 모델별 인식률·속도 측정, 시험 중 찾은 문제 수정	실음성 CER 3.6%, 지연 1.05초
M8 파인튜닝	교정 문장 + 합성 음성 LoRA 학습 → 오류율 비교 → 자동 교체·되돌리기	GPU 학습 6.5분, CER 4.8% → 1.8%

게임 PC에서만 확인할 수 있는 것

실제 발로란트·롤 경기 중 게임 API 응답, 게임 프레임 영향, RTX 5070 Ti CUDA DLL 적재는 이 서버나 VM에서 잴 수 없습니다. 모두 게임 PC 앱의 **진단** 탭에서 버튼 하나로 확인하도록 만들어 두었습니다(10장).

2 전체 구조

구성	역할	기술
게임 PC 앱 GameVoiceRecorder.exe	디스코드 봇(음성 수신·TTS 안내), 녹음 저장, STT 대기열, 게임 연동(발로란트·롤), 기록 DB, 설정 창·트레이, 사이트 중계 연결. 녹음과 기록은 게임 PC에만 저장	Electron + Node.js(TypeScript), discord.js + @discordjs/voice(DAVE), SQLite(FTS5)
STT 워커	음량 정규화 → VAD 잡음 판정 → Whisper 변환 → 사전 교정, 채팅 OCR, r3dlog 읽기, 모델·CUDA 런타임 다운로드	Python(PyInstaller exe), faster-whisper(CTranslate2), Windows OCR
파인튜닝	교정 문장 + 합성 음성으로 LoRA 학습, 학습 전후 CER 비교, CTranslate2로 내보내기	PyTorch, PEFT, Edge TTS (별도 환경)
사이트 서버	디스코드 로그인(서버 멤버 확인), 게임 PC 등록·연결 확인·잠금, 기록·음성 중계(저장하지 않음)	Fastify, WebSocket, Docker(.5 서버)
사이트 화면	타임라인·검색·필터·따라 읽기, 경기 상세(발로란트·롤), 유저별 기록, 단어사전, 교정	React + Vite

음성 한 마디가 사이트에 나오기까지

1. 겜녹이 봇이 음성채널에 들어가 사람마다 따로 Opus 패킷을 받아, 말이 끊기면 발화 하나로 묶어 Ogg로 저장(재인코딩 없음)
2. 실시간 모델(large-v3-turbo)이 바로 변환하고, 경기가 끝나면 정밀 모델(large-v3)이 다시 변환(GPU 여유에 따라)
3. 사람마다 음량을 맞추고, 말소리가 없으면 Whisper를 돌리지 않음(환각 방지). 사전 핫워드·교정 규칙 적용
4. 게임 이벤트(킬·라운드·채팅)와 같은 시간축에 저장 → 사이트 서버를 거쳐 실시간으로 표시
5. 음성 파일은 7일 뒤 자동 삭제. 글자와 경기 기록은 남음

코드 규모와 자동 테스트

부분	코드 줄 수	자동 테스트 (마지막 실행)
앱 (코어 + 설정 창)	약 8,800	`app: npm test` 9개 통과, 타입 검사 오류 0
STT 워커 + 파인튜닝	약 2,900	`worker: pytest` 42개 통과 (Windows 41개 통과, 1개는 리눅스 전용이라 건너뜀)
사이트 서버	약 730	`site/server: npm test` 통합 테스트 통과
사이트 화면	약 4,300	타입 검사·빌드 오류 0, 헤드리스 동작 확인 18개

3 M0 · M1 – 기술 검증, 녹음 + STT

- 디스코드 DAVE(종단 간 암호화) 음성 연결: 겜녹이 봇이 1.06초 만에 연결, 유저별 음성 수신 확인
- 음량 정규화(-30dB 작은 목소리 +24dB 보정 후 같은 문장 인식), 잡음 판정(키보드·백색 잡음·무음 → Whisper 생략)
- 게임 PC 진단 기능: GPU·STT 속도·발로란트/롤 로컬 API·LCU 타임라인·채팅 OCR·r3dlog를 버튼 하나로 점검

실제 디스코드 음성 E2E (M1)

F&L 서버 "음성" 채널에서 테스트봇이 한국어 문장을 말하고, 겜녹이가 따라 들어와 녹음·변환했습니다.

```
겜녹이 따라 들어옴 (follow): 음성
  [테스트봇] 3.9s realtime: 레이즈 공 빠졌어 비사이트로 태하자
  [테스트봇] 3.4s realtime: 오늘 저녁에 치킨 먹을래
  [테스트봇] 2.6s realtime: 나우퍼레이터 살게
녹음 제외 후 새 발화: 0개 (0 이어야 함)
VOICE E2E PASS
```

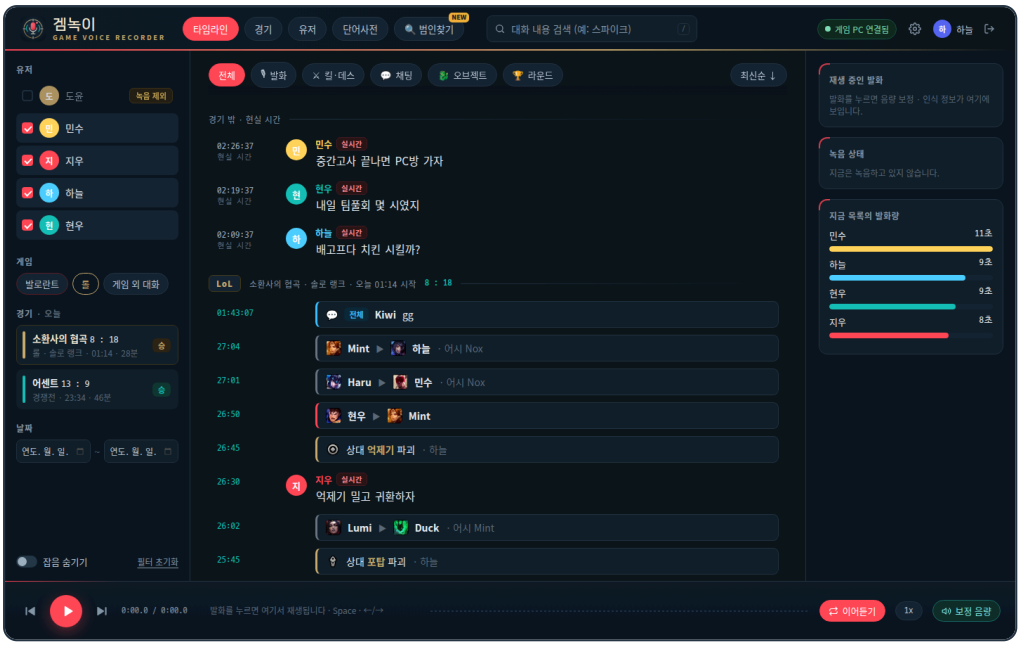
M1은 CPU small 모델이라 틀린 글자가 있습니다. 같은 시험을 GPU large-v3-turbo로 다시 한 결과는 M7(8장)에 있습니다.

4 M2 · M3 – 사이트, 발로란트 연동

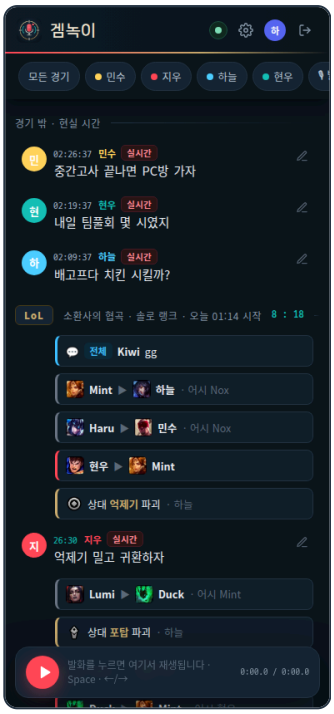
- https://gvr.tkrmagid.kr: 디스코드 로그인(M&L 서버 멤버만), 게임 PC가 연결되어 있을 때만 기록을 보여 주고 끊기면 2.5초 안에 잠금
- 타임라인: 무한 스크롤, 유저·게임·경기 필터, 검색, 실시간 추가, 재생 중 단어 강조(따라 읽기), 이어듣기, 단축키, 문장 교정
- 발로란트: 경기 감지 → 라운드·킬·채팅 → match-details 적용. 시간은 `R7 · 0:38`(라운드 시간)과 경기 경과 시간을 같이 표시



발로란트 경기 상세 — 라운드 띠, 미니맵 킬 위치(게임 좌표 → 지도 변환), 라운드 대화 + 킬피드, 스코어보드(ACS), 라운드별 크레딧. 실제 코어·STT 워커·사이트 서버 전체 스택에 22라운드 데이터와 TTS 대화를 넣어 촬영



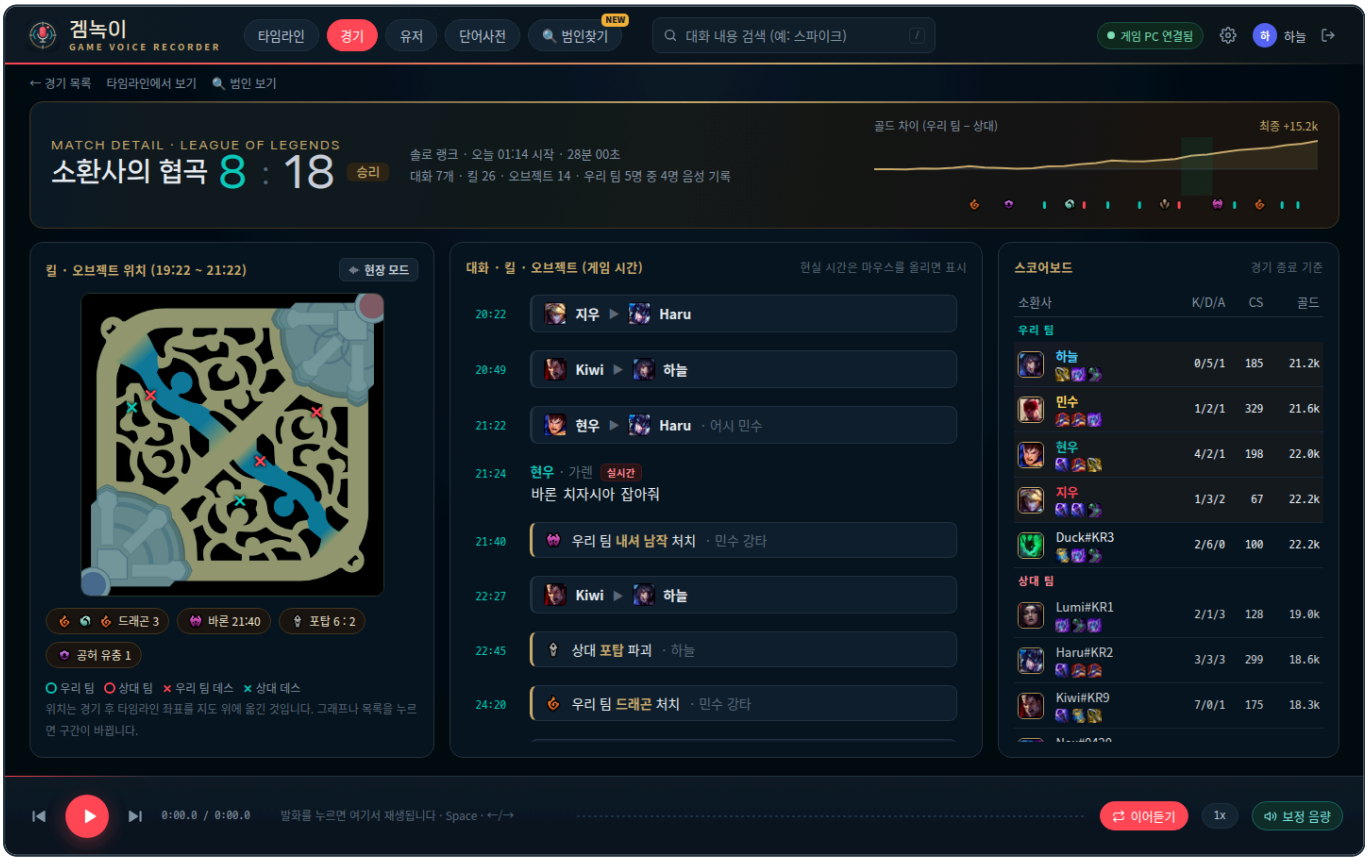
타임라인 — 대화와 게임 이벤트가 한 시간축에



휴대폰 화면

5 M4 — 롤 연동 · 채팅 OCR

- 게임 클라이언트 API(2999): 킬·용·전령·바론·포탑 이벤트, 게임 시간 표시. LCU: 챔피언 선택 채팅, 경기 결과·타임라인(킬 위치·골드 차이)
- 인게임 채팅: r3dlog에 채팅이 남으면 로그를 읽고, 없으면 화면 일부를 Windows 한국어 OCR로 읽음(같은 줄 중복 제거)



롤 경기 상세 — 킬 위치, 골드 차이 그래프, 오브젝트, 스코어보드

Windows 실기 확인 (win-ci Windows 11 VM)

항목	결과
롤 채팅 OCR (Windows 한국어 OCR)	**예시 3줄 중 3줄 인식 — `[팀] 민수 (아리): 드래곤 가자`, `[전체] 상대1 (가렌): gg`, `[팀] 지우 (리 신): 바텀 갱 간다`**
r3dlog 채팅 읽기	**2줄 읽음 (게임 시간·팀/전체·보낸 사람·챔피언·내용)**
STT (CPU tiny)	**한국어 변환 정상, 워커 pytest 41개 통과**

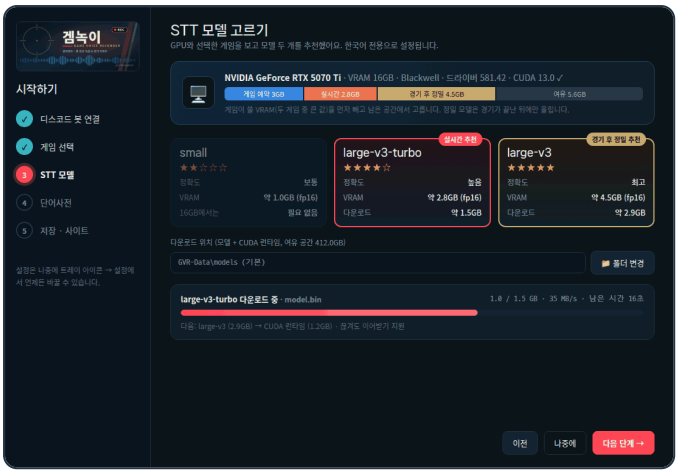
Windows에서만 나던 문제 3개 — 원인을 찾아 고침

증상	원인 (확인 방법)	수정
OCR을 한 번 쓴 뒤 첫 변환에서 워커가 조용히 종료, PyInstaller 분석 실패	pywinrt가 넣어 둔 오래된 `msvcp140.dll`(14.29)이 먼저 올라가면 onnxruntime이 `0xC0000005`로 죽음 (불러오는 순서별 시험)	onnxruntime·ctranslate2를 먼저 적재, 빌드 전 winrt의 msvcp140 삭제
첫 음성 변환이 응답 없이 영원히 멈춤 (pytest 373초 후 실패)	요청 처리 스레드에서 scipy(OpenBLAS DLL)를 처음 불러올 때 멈춤. faulthandler로 스택을 찍어 `scipy.linalg.blas` 적재 중임을 확인, 주 스레드에서 먼저 불러오면 정상인 것도 확인	워커 시작 때 주 스레드에서 scipy·pyloudnorm 미리 적재 → pytest 8초
파인튜닝 요청이 120초 응답 없음	워커가 띄운 자식 프로세스가 워커의 요청 파이프(stdin)를 물려받아, 시작 단계에서 주 스레드의 읽기 대기와 엉켜 멈춤	자식 프로세스에 stdin을 넘기지 않음

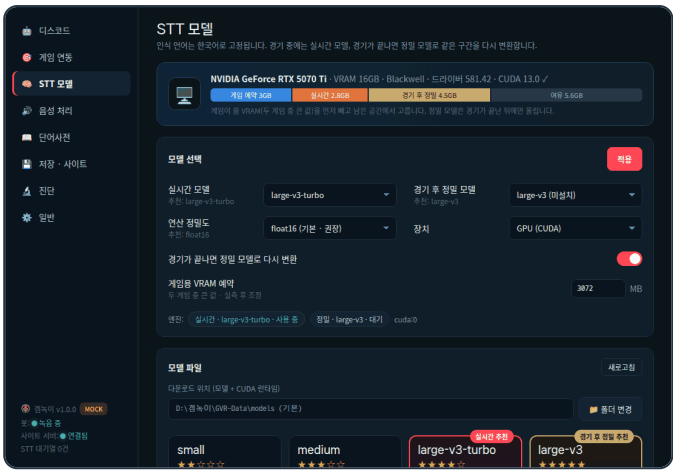
6 M5 — 포터블 exe · 설정 · 모델

결과물 (Windows 빌드)	크기	설명
`GameVoiceRecorder-1.0.0-portable.exe`	168 MB	파일 하나로 실행 (실행할 때 임시 폴더에 풀림)
`GameVoiceRecorder-1.0.0-win-x64.zip`	259 MB	폴더 포터블 (권장: 풀 필요가 없어 시작이 빠름)
내장 STT 워커 (PyInstaller)	325 MB	zip·exe 안에 포함. 모델(1.5~2.9GB)과 CUDA 런타임(1.2GB)은 첫 실행 때 다운로드

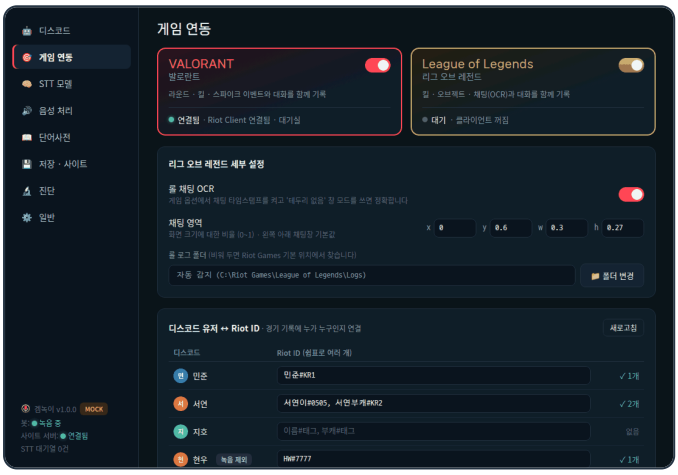
exe 실행 확인 (`GameVoiceRecorder.exe --smoke`): 설정 암호화 DPAPI 왕복 ✓ · 워커 기동 ✓ · GPU 감지(VM은 GPU가 없어 CPU 경고) ✓ · 사전 2,026단어 ✓ · 설정 창 화면 ✓ · 트레이 ✓. 워커 exe 단독 실행도 OCR 3/3·STT 통과.



첫 실행 마법사 3단계 — GPU VRAM을 보고 실시간/정밀 모델 추천, 다운로드(이어받기)



설정 창 · STT 탭 — 모델·정밀도·잡음 민감도·음량



설정 창 · 게임 탭 — 발로란트·롤 연동, 채팅 OCR 영역

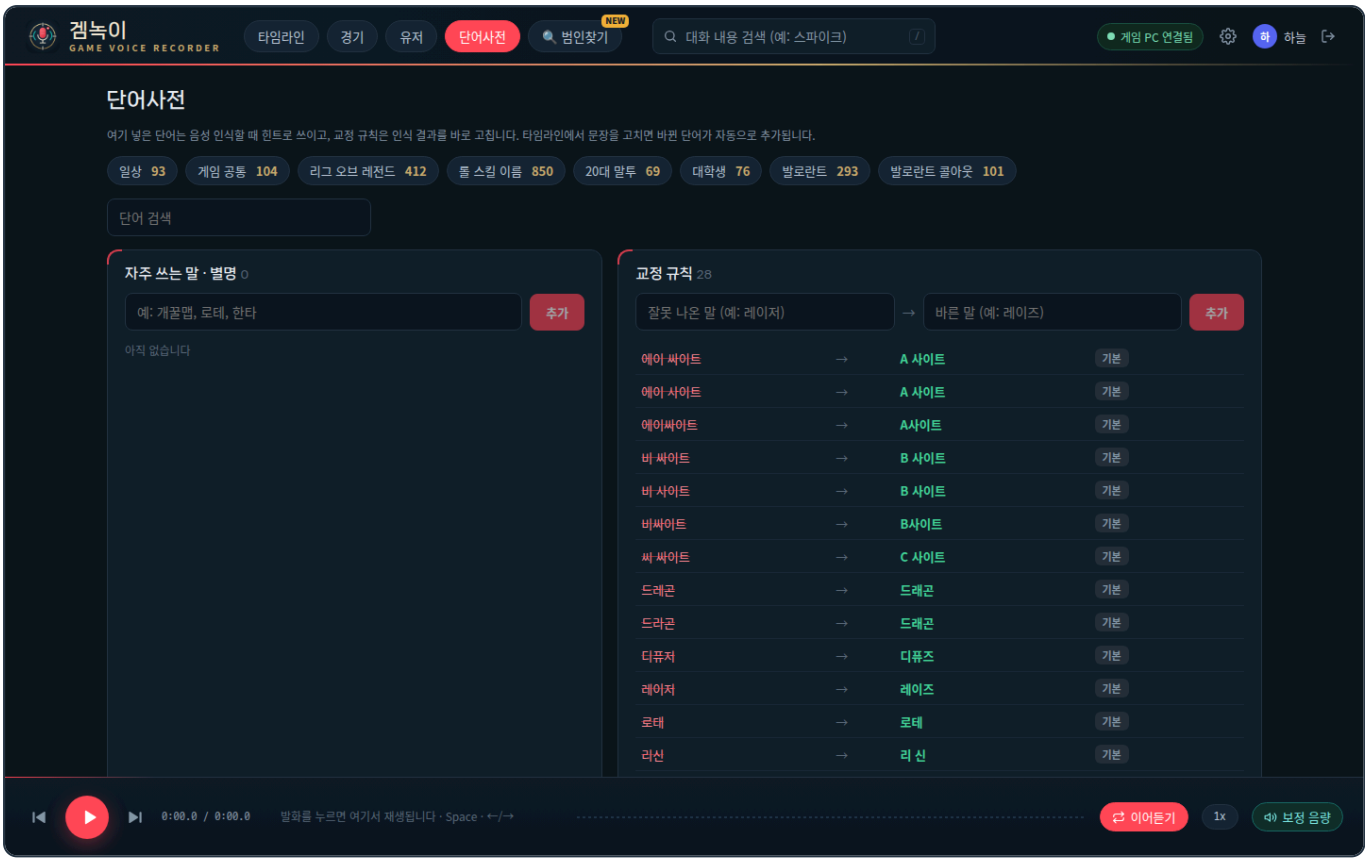


설정 창 · 진단 탭 — 게임 PC 점검

설정 창 화면은 가짜 데이터(RTX 5070 Ti 가정)로 띄워 찍은 것입니다. 실제 exe에서 같은 화면이 뜨는지는 Windows 스모크의 "설정 창 화면 ✓"로 확인했습니다.

7 M6 — 단어사전 · UI/UX

- 기본 사전 8묶음 약 2,000단어: 일상 93 · 게임 공통 104 · 롤 412 · 롤 스킬 850 · 20대 말투 69 · 대학생 76 · 발로란트 293 · 발로란트 콜아웃 101, 교정 규칙 28개
- 경기마다 그 경기에 나온 요원·챔피언·선수 이름을 핫워드 앞쪽에 넣음. 게임 패치 데이터로 새 이름 자동 추가
- 사이트에서 문장을 고치면 바뀐 단어가 교정 규칙으로 자동 추가되고, 같은 문장은 파인튜닝 학습 데이터가 됨
- 발로란트(빨강) × 롤(금색) 테마, 효과음(Web Audio로 합성해 음원 파일 없음, 종류별로 끄고 켬), 애니메이션 11종, 모션 줄이기



사이트 단어사전 화면 — 묶음별 단어 수, 자주 쓰는 말·별명 추가, 교정 규칙(잘못 나온 말 → 바른 말)

8 M7 — 실사용 테스트 (GPU)

이 서버 GPU(RTX 5050 8GB, 다른 서비스가 3GB 사용 중)에서 게임 PC와 같은 GPU 방식(CUDA float16)으로 측정했습니다.

실제 디스코드 음성 28문장

```
F&L "음성" 채널 · 테스트봇이 시연 대화 28문장을 연달아 말함 → 겜녹이 따라 들어와 녹음 → STT
발화 28개 (말한 문장 28개) · 모델 large-v3-turbo (cuda) · 지연 중앙값 1.05s · 최대 1.86s
CER 3.6% · 완전 일치 21/28
녹음 제외 후 새 발화: 0개 (0 이어야 함)
겜녹이 퇴장: true
VOICE E2E PASS
```

- 28문장을 28개 발화로 정확히 나눔. 지연은 말이 끝난 뒤 사이트에 글자가 생기기까지 걸린 시간(첫 발화 1.86초는 모델 첫 실행)
- 시험 중 발견·수정: 시험용 봇만 있는 방을 "빈 방"으로 보고 30초 뒤 나가던 문제(시험 모드에서 봇도 참가자로 셈)

모델별 인식률·속도 (시연 대화 28문장)

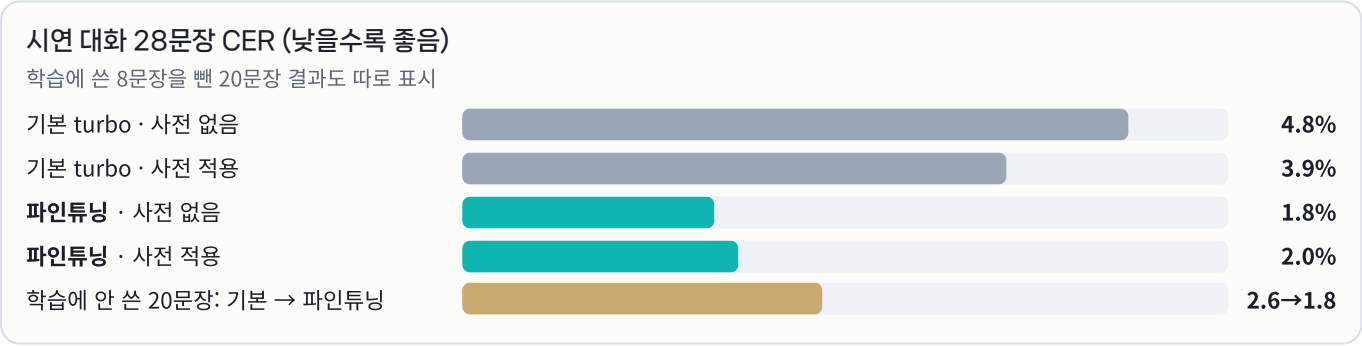
모델 (CUDA float16)	사전 없음 CER	사전 적용 CER	완전 일치	문장당 변환	적재
large-v3-turbo (실시간 기본)	4.8%	**3.9%**	21/28	0.31초	4초
large-v3 (정밀)	4.3%	4.8%	19/28	0.48초	6초
small	10.9%	8.1%	17/28	0.16초	1초 미만

- 남은 오류는 대부분 소리가 비슷한 말입니다: "대시→데스", "칠게→7개", "깔았어→깔아서". 교정 사전과 파인튜닝이 맡을 부분입니다
- 짧은 게임 대화에서는 large-v3가 turbo보다 낫지 않았습니다. 그래서 실시간은 turbo, 정밀 변환은 경기가 끝난 뒤에 하는 지금 기본값을 유지합니다

9 M8 – 파인튜닝

앱의 실제 학습 경로를 그대로 GPU에서 돌렸습니다: 사이트 교정 → 학습 데이터 → 학습 → 학습 전후 오류율 비교 → 좋아졌을 때만 모델 교체 → 되돌리기. 결과는 `FINETUNE E2E PASS`였습니다.

항목	내용
학습 데이터	사이트에서 고친 문장 8개 + 사전 단어로 만든 문장을 Edge TTS 3개 음성으로 합성한 300개(속도·피치 변화, 30% 배경 잡음) → 학습 281 / 검증 27
학습	openai/whisper-large-v3-turbo + LoRA(r=32), 3에폭 54스텝, **394초**, 최대 VRAM **2.7GB**
검증 세트 CER	9.7% → **8.3%** → 자동으로 새 모델 적용(CTranslate2 float16, 1.6GB) → 새 모델로 변환 확인 → 이전 모델로 되돌리기 확인



학습하면서 찾아 고친 문제 2개

증상	원인	수정
첫 학습이 GPU 메모리 부족으로 실패	바뀌지 않는 원본 가중치까지 fp32로 올림(3.2GB). 이 서버 GPU는 다른 서비스가 3GB를 씀	원본은 bf16, 학습하는 LoRA만 fp32 → VRAM 절반(최대 2.7GB)
검증 CER은 좋아졌는데(10.2→7.9%) 실제 변환에서 문장 끝에 "끝", "댕댕댕", "끈의 주인공" 같은 말을 지어냄 (사전 없음 CER 4.8% → 21.8%)	타임스탬프 없는 라벨로만 학습해서, 앱(faster-whisper)이 쓰는 타임스탬프 모드에서 문장을 끝맺는 법을 잊음. 검증도 타임스탬프 없이 해서 걸러내지 못함	라벨 절반을 `<|0.00|>` 문장 `<|말끝|>` 형식으로, 검증도 타임스탬프 모드로 → 사전 없이도 1.8%

> **해석할 때 주의할 점**
> 합성 학습 음성과 시연 대화가 같은 TTS 엔진이라, 실제 사람 목소리에서는 개선 폭이 이보다 작을 수 있습니다. 계획대로 실사용 교정 데이터가 2주쯤 쌓이면 게임 PC(RTX 5070 Ti)에서 다시 학습해 비교합니다. 앱은 오류율이 나아질 때만 새 모델로 바꾸고, 언제든 되돌릴 수 있습니다.

10 게임 PC에서 처음 할 일

1. `GameVoiceRecorder-1.0.0-win-x64.zip`을 원하는 폴더에 풀고 `GameVoiceRecorder.exe` 실행 → 첫 실행 마법사(봇 토큰 → 게임 선택 → 모델 다운로드 → 사전 → 사이트 등록)
2. 설정 창 **진단** 탭 → 전체 진단: RTX 5070 Ti STT 속도·VRAM, CUDA 런타임, 발로란트·롤 로컬 API, LCU 타임라인, 채팅 OCR, r3dlog 채팅
3. 사이트 등록 코드를 앱에 넣어 https://gvr.tkrmagid.kr 연결 → 친구들과 디스코드 음성방에서 발로란트·롤 한 판씩
4. 사이트 타임라인에서 틀린 문장을 고치면 교정 사전과 다음 파인튜닝 데이터가 됨

아직 확인하지 못한 것

항목	이유	확인 방법
실제 경기 중 발로란트·롤 API 응답	개발 서버·VM에서 게임 실행 불가 (실제 API와 같은 구조의 가짜 서버로 전 과정 테스트함)	진단 탭 + 실제 경기 1판
게임 프레임 영향	게임 PC에서만 측정 가능	경기 중 프레임 비교. 정밀 변환은 경기 후로 미루는 것이 기본값
Windows CUDA DLL 적재 (RTX 5070 Ti)	VM에 GPU 없음 (GPU 경로 자체는 이 서버 RTX 5050에서 확인)	진단 탭 "STT 속도"
실제 사람 목소리 인식률	시험은 TTS 목소리로 함	실사용 2주 후 교정률·CER

11 부록 — 확인 명령과 커밋

다시 돌려 볼 수 있는 확인 명령 (저장소 루트 기준)

```
# 자동 테스트
cd app && npm test && npx tsc --noEmit -p .          # 앱 9개
cd worker && .venv/bin/python -m pytest -q            # 워커 42개
cd site/server && npm test                            # 사이트 서버 통합
# 실제 디스코드 음성 (F&L "음성" 채널, GPU)
GVR_E2E_SET=demo GVR_E2E_DEVICE=cuda GVR_E2E_MODEL≤large-v3-turbo 폴더> \
  node --import tsx app/scripts/voice-e2e.mts 789051930965835810 934817907479810048
# 모델별 인식률 (GPU)
GVR_EVAL_DEVICE=cuda node --import tsx app/scripts/stt-eval.mts <모델 폴더> large-v3-turbo large-v3 small
# 파인튜닝 전체 경로 (GPU)
GVR_FT_BASE=openai/whisper-large-v3-turbo GVR_FT_STEPS=0 GVR_FT_SYNTH=300 node --import tsx app/scripts/finetune-e2e.mts
# Windows 검증·exe 빌드 (win-ci VM)
tools/win-ci/run.sh full <커밋>
```

단계별 커밋 (Gitea main)

커밋	내용
7c9146c	M0 기술 검증 — 구조 설계, STT 워커, 앱 코어, 게임 PC 진단
2745af8	M1 녹음 + STT 기본 — 실제 디스코드 음성 E2E 통과
9abda88	M2 사이트 서버 · 사이트 v1 — gvr.tkrmagid.kr 배포
5222b3e	M3 발로란트 연동 — 전체 스택 시연·테스트, 핫워드 되풀이 방지
827e9d8 · e6723a4	M4 Windows 워커 크래시(msvcp140)·첫 변환 멈춤(scipy) 수정
5f04ccd · ae2af63 · 8f44f22	M5 포터블 exe(트레이·설정 창·마법사), Windows 자식 프로세스 멈춤·빌드 경로 수정
47aeef9	M6 단어사전 화면, 사이트 재배포
f26f408	M4·M5 완료 기록 (Windows 실기 확인·exe 빌드)
c32236c	M7 GPU 실사용 시험 (실제 디스코드 음성 28문장, 모델별 인식률·속도)
24585e3	M8 GPU 파인튜닝 (large-v3-turbo LoRA, CER 4.8% → 1.8%)

자세한 기록: `docs/dev/test-results.md`(단계별 테스트), `docs/dev/ARCHITECTURE.md`(구조·인터페이스), `docs/gemnoki-project-plan.pdf`(설명서·계획서).