Attic

- transcribe_audio.sh — Pre-Context
- transcribe-audio.sh - With Context

transcribe_audio.sh — Pre-Context

```
#!/usr/bin/env bash
set -euo pipefail

# ------------------------------------------------------------
# transcribe-audio.sh
# Post-session transcription on Mac using whisper-cli
# Usage: transcribe-audio.sh [--de|--en|--auto] <audio.wav>
# ------------------------------------------------------------

WHISPER="$HOME/Transkriptionen/whisper.cpp/build/bin/whisper-cli"
MODEL="$HOME/Transkriptionen/whisper.cpp/models/ggml-large-v3-turbo.bin"
VAD_MODEL="$HOME/Transkriptionen/whisper.cpp/models/ggml-silero-v6.2.0.bin"

# ------------------------------------------------------------
# Argument parsing
# ------------------------------------------------------------

LANG_MODE="en"
AUDIO=""

while [[ $# -gt 0 ]]; do
  case "$1" in
    --de)   LANG_MODE="de"; shift ;;
    --en)   LANG_MODE="en"; shift ;;
    --auto) LANG_MODE="auto"; shift ;;
    -*)
      echo "Usage: transcribe-audio [--de|--en|--auto] <audio.wav>" >&2
      exit 2
      ;;
    *)
      AUDIO="$1"; shift ;;
  esac
```

```
done

if [[ -z "$AUDIO" ]]; then
  echo "□ No audio file specified." >&2
  echo "   Usage: transcribe-audio [--de|--en|--auto] <audio.wav>" >&2
  exit 1
fi

if [[ ! -f "$AUDIO" ]]; then
  echo "□ File not found: $AUDIO" >&2
  exit 1
fi

# ------------------------------------------------------------
# Language options
# ------------------------------------------------------------

case "$LANG_MODE" in
  en|de) LANG_OPTS=(-l "$LANG_MODE") ;;
  auto)  LANG_OPTS=() ;;
esac

# ------------------------------------------------------------
# Output path: same dir as audio, .txt extension
# ------------------------------------------------------------

AUDIO_DIR="$(dirname "$AUDIO")"
AUDIO_BASE="$(basename "$AUDIO" .wav)"
TRANSCRIPT_BASE="$AUDIO_DIR/${AUDIO_BASE}_transcript"

echo "□□Audio:      $AUDIO"
echo "□□□ Language:   $LANG_MODE"
echo "□□Transcript: ${TRANSCRIPT_BASE}.txt"
echo "----"

# ------------------------------------------------------------
# Run whisper-cli
# ------------------------------------------------------------
  auto)  LANG_OPTS=() ;;
esac
```

```
# ------------------------------------------------------------
# Output path: same dir as audio, .txt extension
# ------------------------------------------------------------

AUDIO_DIR="$(dirname "$AUDIO")"
AUDIO_BASE="$(basename "$AUDIO" .wav)"
TRANSCRIPT_BASE="$AUDIO_DIR/${AUDIO_BASE}_transcript"

echo "□□Audio:      $AUDIO"
echo "□□□ Language:   $LANG_MODE"
echo "□□Transcript: ${TRANSCRIPT_BASE}.txt"
echo "----"

# ------------------------------------------------------------
# Run whisper-cli
# ------------------------------------------------------------

"$WHISPER" \
  -m "$MODEL" \
  "${LANG_OPTS[@]}" \
  --vad \
  -vm "$VAD_MODEL" \
  --output-txt \
  --output-srt \
  -of "$TRANSCRIPT_BASE" \
  -f "$AUDIO"


# Strip ANSI escape sequences and carriage returns from txt output
sed -i '' 's/\x1b\[[0-9;]*[mGKH]//g; s/\r//g' "${TRANSCRIPT_BASE}.txt"

echo "□ Done: ${TRANSCRIPT_BASE}.txt"
```

transcribe-audio.sh - With Context

```
#!/usr/bin/env bash
set -euo pipefail

# ------------------------------------------------------------
# transcribe-audio.sh
# Post-session transcription on Mac using whisper-cli
# Usage: transcribe-audio.sh [--de|--en|--auto] [--game <slug>]
#                            [--vad-preset tight|default|loose]
#                            [--vt F] [--vspd N] [--vp N] [--et F] [--nth F]
#                            <audio.wav>
# ------------------------------------------------------------

WHISPER="$HOME/Transkriptionen/whisper.cpp/build/bin/whisper-cli"
MODEL="$HOME/Transkriptionen/whisper.cpp/models/ggml-large-v3-turbo.bin"
VAD_MODEL="$HOME/Transkriptionen/whisper.cpp/models/ggml-silero-v6.2.0.bin"
META_DIR="$HOME/Syncthing/TranscriptOMatic/meta"

# ------------------------------------------------------------
# Argument parsing
# ------------------------------------------------------------

LANG_MODE="en"
AUDIO=""
GAME_SLUG=""
VAD_PRESET=""   # empty = use language default
VAD_VT=""       # override: voice threshold
VAD_VSPD=""     # override: min speech duration (ms)
VAD_VP=""       # override: padding (ms)
VAD_ET=""       # override: end-of-speech timeout
VAD_NTH=""      # override: noise threshold

while [[ $# -gt 0 ]]; do
```

```
  case "$1" in
    --de)          LANG_MODE="de";    shift ;;
    --en)          LANG_MODE="en";    shift ;;
    --auto)        LANG_MODE="auto";  shift ;;
    --game)        GAME_SLUG="$2";    shift 2 ;;
    --vad-preset)  VAD_PRESET="$2";   shift 2 ;;
    --vt)          VAD_VT="$2";       shift 2 ;;
    --vspd)        VAD_VSPD="$2";     shift 2 ;;
    --vp)          VAD_VP="$2";       shift 2 ;;
    --et)          VAD_ET="$2";       shift 2 ;;
    --nth)         VAD_NTH="$2";      shift 2 ;;
    -*)
      echo "Usage: transcribe-audio [--de|--en|--auto] [--game <slug>]" >&2
      echo "       [--vad-preset tight|default|loose]" >&2
      echo "       [--vt F] [--vspd N] [--vp N] [--et F] [--nth F]" >&2
      echo "" >&2
      echo "VAD presets:" >&2
      echo "  tight    vt=0.30 vspd=250 vp=400 et=2.3 nth=0.40  fast speakers, good mic" >&2
      echo "  default  vt=0.25 vspd=150 vp=200 et=2.8 nth=0.30  mixed tempo, moderate pauses"
>&2
      echo "  loose    vt=0.20 vspd=100 vp=600 et=3.5 nth=0.20  slow speakers, noisy room" >&2
      exit 2
      ;;
    *)
      AUDIO="$1"; shift ;;
  esac
done

if [[ -z "$AUDIO" ]]; then
  echo "□ No audio file specified." >&2
  echo "   Usage: transcribe-audio [--de|--en|--auto] [--game <slug>] <audio.wav>" >&2
  exit 1
fi

if [[ ! -f "$AUDIO" ]]; then
  echo "□ File not found: $AUDIO" >&2
  exit 1
fi

# ------------------------------------------------------------
```

```
# Language options and VAD parameters
# ------------------------------------------------------------

# Language → default preset
case "$LANG_MODE" in
  en)
    LANG_OPTS=(-l en)
    [[ -z "$VAD_PRESET" ]] && VAD_PRESET="tight"
    ;;
  de)
    LANG_OPTS=(-l de)
    [[ -z "$VAD_PRESET" ]] && VAD_PRESET="default"
    ;;
  auto)
    LANG_OPTS=()
    [[ -z "$VAD_PRESET" ]] && VAD_PRESET="default"
    ;;
esac

# Preset base values
#   tight    fast speakers, few pauses, good mic        (EN default)
#   default  mixed tempo, moderate pauses               (DE / auto default)
#   loose    slow/deliberate speakers, noisier room
case "$VAD_PRESET" in
  tight)   _VT=0.30; _VSPD=250; _VP=400; _ET=2.3; _NTH=0.40 ;;
  default) _VT=0.25; _VSPD=150; _VP=200; _ET=2.8; _NTH=0.30 ;;
  loose)   _VT=0.20; _VSPD=100; _VP=600; _ET=3.5; _NTH=0.20 ;;
  *)
    echo "□ Unknown VAD preset: '$VAD_PRESET'. Use tight, default, or loose." >&2
    exit 2
    ;;
esac

# Apply per-parameter overrides
VT="${VAD_VT:-$_VT}"
VSPD="${VAD_VSPD:-$_VSPD}"
VP="${VAD_VP:-$_VP}"
ET="${VAD_ET:-$_ET}"
NTH="${VAD_NTH:-$_NTH}"
```

```
VAD_OPTS=(-vt "$VT" -vspd "$VSPD" -vp "$VP" -et "$ET" -nth "$NTH")

# ------------------------------------------------------------
# Build prompt from YAML meta file (if --game given)
# ------------------------------------------------------------

PROMPT_OPTS=()

if [[ -z "$GAME_SLUG" ]]; then
  echo "⚠  No --game specified. Running without vocabulary prompt."
  echo "   Tip: use --game <slug> for better transcription of proper nouns."
fi

if [[ -n "$GAME_SLUG" ]]; then
  META_FILE="$META_DIR/${GAME_SLUG}.yaml"
  if [[ -f "$META_FILE" ]]; then
    # Extract all primary keys from characters, locations, terms, phrases, groups
    # using Python — handles Unicode correctly
    PROMPT_TEXT="$(uv run --with pyyaml python3 - "$META_FILE" <<'PYEOF'
import sys, yaml

with open(sys.argv[1], encoding="utf-8") as f:
    data = yaml.safe_load(f)

keys = []
for section in ["characters", "locations", "terms", "phrases", "groups"]:
    block = data.get(section, {}) or {}
    if isinstance(block, dict):
        for key in block.keys():
            # Strip parenthetical suffixes e.g. "Muiris Doyle (Ó Dubhghaill)"
            clean = key.split("(")[0].strip()
            if clean:
                keys.append(clean)

print(", ".join(keys))
PYEOF
)"
    if [[ -n "$PROMPT_TEXT" ]]; then
      PROMPT_OPTS=(--prompt "$PROMPT_TEXT" --carry-initial-prompt)
      echo "  Game:       $GAME_SLUG"
```

```
      echo "□□Prompt:     $PROMPT_TEXT"
    fi
  else
    echo "⚠□  No meta file found for slug '$GAME_SLUG' in $META_DIR" >&2
  fi
fi

# ------------------------------------------------------------
# Output path: same dir as audio, transcript suffix
# ------------------------------------------------------------

AUDIO_DIR="$(dirname "$AUDIO")"
AUDIO_BASE="$(basename "$AUDIO" .wav)"
TRANSCRIPT_BASE="$AUDIO_DIR/${AUDIO_BASE}_transcript"

echo "□□Audio:      $AUDIO"
echo "□□□ Language:   $LANG_MODE | VAD preset: $VAD_PRESET"
echo "   vt=$VT  vspd=$VSPD  vp=$VP  et=$ET  nth=$NTH"
echo "□□Transcript: ${TRANSCRIPT_BASE}.txt"
echo "----"

# ------------------------------------------------------------
# Run whisper-cli
# ------------------------------------------------------------

"$WHISPER" \
  -m "$MODEL" \
  "${LANG_OPTS[@]}" \
  --vad \
  -vm "$VAD_MODEL" \
  --output-txt \
  --output-srt \
  -of "$TRANSCRIPT_BASE" \
  "${VAD_OPTS[@]}" \
  "${PROMPT_OPTS[@]}" \
  -f "$AUDIO"

# Strip ANSI escape sequences and carriage returns from txt output
sed -i '' 's/\x1b\[[0-9;]*[mGKH]//g; s/\r//g' "${TRANSCRIPT_BASE}.txt"
```

```
echo "□ Done: ${TRANSCRIPT_BASE}.txt"
```