ПРОМТ

Ты монтируешь видео. В текущей папке лежит сырая запись говорящей головы (один спикер, одна камера, много неудачных дублей и пауз) и, возможно, музыкальный трек. Найди их сам: видеофайл — самый большой `.mp4/.mov/.mkv`, музыка — `.mp3/.wav/.m4a`. Если файлов несколько и непонятно, какой брать — спроси один раз и дальше не переспрашивай.

Сделай из этого готовый ролик и проект для правок.

Настройки

```
ЯЗЫК РЕЧИ:      ru          # код для whisper: ru, en, es, de…
СТИЛЬ:          светлый минимал
АКЦЕНТ:         #c8623c     # один акцентный цвет на всю графику
ПЛОТНОСТЬ:      максимальная # максимальная | естественная
СУБТИТРЫ:       точечно     # нигде | точечно | везде
СЛОВА-ПАРАЗИТЫ: оставить    # оставить | убрать
РАЗРЕШЕНИЕ:     как в исходнике
МУЗЫКА:         разнос 15 дБ под голосом
```

Шаг 0. Окружение

Проверь одной командой и доставь недостающее:

```bash
npx --yes hyperframes@latest doctor
```

Она разом показывает Node, ffmpeg, ffprobe, whisper-cpp, память и диск.

Чего может не хватать:

```bash
# ffmpeg  (macOS / Debian-Ubuntu)
brew install ffmpeg          ||  sudo apt install -y ffmpeg

# Node 22+  — нужен для HyperFrames
brew install node            ||  sudo apt install -y nodejs npm

# whisper.cpp
brew install whisper-cpp     # даёт бинарь whisper-cli

# модель распознавания — формула её НЕ тянет, ~1.6 ГБ
mkdir -p .tmp/models && curl -L --progress-bar \
  -o .tmp/models/ggml-large-v3-turbo.bin \
  https://huggingface.co/ggerganov/whisper.cpp/resolve/main/ggml-large-v3-turbo.bin

# проект HyperFrames
npx --yes hyperframes@latest init project --example blank --non-interactive --skip-transcribe
npx --yes hyperframes@latest skills update talking-head-recut embedded-captions
```

Windows: ffmpeg и Node ставятся через `winget`, whisper.cpp — готовым релизом
с GitHub `ggml-org/whisper.cpp`; остальное работает как есть.

Разложи работу так:

```
tools/      скрипты (пишешь на шаге 1)
project/    HyperFrames-проект
.tmp/       промежуточные файлы, одноразовые
out/        то, что отдаёшь
```

Скрипты пиши как отдельные файлы в `tools/`, а не инлайн в командах: их
придётся запускать много раз с разными параметрами.

Шаг 1. Разбор материала

```bash
ffmpeg -i ИСХОДНИК -vn -ac 1 -ar 16000 -c:a pcm_s16le .tmp/audio16k.wav
M=.tmp/models/ggml-large-v3-turbo.bin

# читаемая расшифровка — чтобы понять содержание
whisper-cli -m $M -f .tmp/audio16k.wav -l ЯЗЫК -t 8 -ojf -osrt -of .tmp/transcript

# словные тайминги — для синхрона графики
whisper-cli -m $M -f .tmp/audio16k.wav -l ЯЗЫК -t 8 \
  -dtw large.v3.turbo -ml 1 -sow -oj -of .tmp/words
```

Два обязательных знания про эти тайминги:

- **Без `-dtw` они мусорные** (`t_dtw: -1`), слова разъезжаются на секунды.
- **При `-ml 1` конец слова = начало следующего.** Сегменты идут сплошняком.
  Осмысленны только *стартовые* тайминги.

Дальше построй **огибающую громкости**: RMS по окнам 10 мс, в dBFS, в JSON.
Она будет основой всех решений о резе.

Точки реза берутся из звука, а не из ASR. Тайминги whisper плывут на
±100–200 мс; уровни — измеренный факт. Whisper отвечает «что сказано»,
огибающая — «где именно».

Шаг 2. Отбор дублей → ГЕЙТ

Прочитай расшифровку и выбери **лучший дубль каждой смысловой фразы**.
Это твоя работа, а не скрипта. Признаки:

- спикер сам себя отменяет вслух («не, не пойдёт», «нет,», «стоп») — в мусор;
- сам себя одобряет («вот это хорошо», «этот вариант») — бери **предыдущий** дубль;
- слитная целая фраза почти всегда лучше рваной, даже если в рваной
  формулировка точнее;
- если все дубли рваные — бери самый полный и прикрой стыки графикой (шаг 4).

Запиши решения в `edl_spans.json`: по интервалу на фразу, с текстом,

и обязательно **с пометкой, что отвергнуто и почему** — это читает человек.

```json
{
  "source": "raw.mp4",
  "spans": [
    { "id": "S1", "in": 18.40, "out": 23.20,
      "text": "…",
      "note": "4-й дубль, после него автор говорит «хороший вариант»",
      "rejected": ["0.0-8.6 медленно", "9.2-12.4 обрыв"] }
  ]
}
```

Прежде чем зафиксировать интервал — проверь две вещи:

1. Нет ли **внутри** него паузы длиннее секунды. Почти наверняка это шов между двумя заходами на одну фразу, и сжатие пауз склеит их в слышимый повтор.
2. Не начинается ли он речью, за которой идёт длинная пауза. Это хвост предыдущего дубля.

⛔ **СТОП. Покажи человеку `edl_spans.json` и расшифровку. Дальше не иди, пока не подтвердит.** Смысловые акценты вслепую не угадываются.

Шаг 3. Рез

Один проход ffmpeg: по сегменту `trim`/`atrim` + `setpts`/`asetpts`, затем `concat` **фильтром**.

Не concat-демуксером и не сборкой из кусочков: фильтр даёт рез в точный кадр, а не в ближайший ключевой, и режет видео с аудио одним графом — рассинхрон невозможен. На каждом стыке фейд аудио 15 мс, иначе щёлкает.

Выход: `.tmp/cut.mp4`, H.264 CRF 16, разрешение и fps как в исходнике.

Проверка реза — обязательная

```bash
ffmpeg -i .tmp/cut.mp4 -vn -ac 1 -ar 16000 -c:a pcm_s16le .tmp/cut16k.wav
whisper-cli -m $M -f .tmp/cut16k.wav -l ЯЗЫК -osrt -of .tmp/cut_check
```

Расшифровка смонтированного должна читаться как связный сценарий. Но помни ловушку №1 ниже: **эта проверка не видит повторов**. Дополнительно вырежи каждый интервал отдельно и расшифруй коротким фрагментом.

Отдельно измерь остаток тишины и покажи числом. Норма при ПЛОТНОСТЬ=максимальная — не больше ~150 мс в любом месте.

Шаг 4. Графика

Собирай **весь ролик одним HyperFrames-проектом**, включая исходное видео как `<video>`. Не гибрид из альфа-оверлеев с композитом в ffmpeg: рендер 2560×1440 идёт около 11 кадр/с, полторы-две минуты на 35-секундный ролик — гибрид не

нужен и хуже правится.

Словные тайминги сними **заново, со смонтированной дорожки** — иначе всё разъедется.

Тайминги не вбивай руками

Заведи `cues.json`, где каждая метка привязана к произнесённому слову, и генератор, который подставляет числа в композицию — и в блок констант, и в `data-start`/`data-duration` каждого клипа:

```json
{ "cues": {
    "chk1":  { "word": "склейки," },
    "hit2":  { "word": "Claude", "n": 2 },
    "mgIn":  { "word": "добавляет", "offset": -0.10 } },
  "clips": {
    "checks": { "from": "chk1", "pre": 0.35, "to": "chkOut", "post": 0.50 } } }
```

После любой перерезки — один запуск генератора, и вся графика едет за речью. Иначе после каждой правки монтажа руками переставлять три десятка чисел, и что-нибудь обязательно разъедется.

Сопоставление слов делай **без краевой пунктуации и без учёта регистра**: whisper между прогонами пишет то «эти», то «эти.», то «Let's», то «let's».

Правила сцен

1. **Самодемонстрация.** Спикер говорит «добавляет анимацию» — ровно там играет анимация. Говорит «делает субтитры» — ровно там субтитры. Сильнейший приём в таком ролике.
2. **Графика прикрывает джамп-каты.** Где речь рваная и сжатие пауз оставляет заметные склейки — ставь слайд-сцену: спикер уезжает вбок и уменьшается (`scale: 0.42`, сдвиг вправо), с другой стороны выезжает панель. Стыки перестают читаться.
3. **Сторона по свободному месту.** Смотри, где в кадре пусто, туда и ставь.
4. **Не прижимай к низу.** Нижние чипы — примерно на 20% высоты кадра от низа.
5. **Ключевую фразу — на полный экран.** Один-два раза за ролик: цветное поле шторкой, рисуется иконка, слова выходят снизу. Работает как удар.
6. **Hard cut там, где молчание.** Если плавный переход приходится на паузу — заменяй резом по кадру (`tl.set` вместо `tl.to`). Плавность по тишине читается вяло.
7. **Каждый элемент выходит на своём слове**, а не пачкой.

Субтитры

При СУБТИТРЫ=точечно ставь их **только там, где они по смыслу**: если спикер сам о них говорит — там; иначе на одной ключевой фразе как акцент. Караоке на весь ролик — это СУБТИТРЫ=везде, и просить об этом должен человек.

Стиль

«Светлый минимал» = тёплые белые панели поверх кадра, тонкая типографика, много воздуха, мягкие тени, один акцентный цвет:

```css
--panel:#fcfbf8; --ink:#14161a; --line:rgba(20,22,26,.10);
```

--accent:АКЦЕНТ; --shadow:0 28px 90px rgba(0,0,0,.34);
```

Системный шрифт (`-apple-system, "SF Pro Display", system-ui`) — у него хорошая кириллица и он всегда на месте. Скругления 28px. При кадре 2560 панель ≈1090px, отступ от края 150px.

Тексты на панелях придумывай по смыслу реплики — и **покажи их человеку до рендера**, формулировки почти всегда правят.

---

## Шаг 5. Звук

```
хайпасс 80 Гц → двухпроходный loudnorm → −14 LUFS, истинный пик −1 dBFS
```

Двухпроходный обязательно: однопроходный уплывает от цели. Лимитер по истинному пику — без него подъём до −14 клипует.

Деноиз **не делай**, если пол шума ниже −50 dBFS: на чистой записи он только добавляет артефактов. Сначала измерь.

Музыка

Уровень считай **не процентом от громкости трека** (это ни о чём не говорит), а разносом с голосом:

- подложку нормализуй к абсолютной цели (−24 LUFS при голосе −14);
- приглуши сайдчейном по голосу: ratio 3, threshold 0.10, attack 20, release 380;
- целевой разнос **15 дБ** — музыка отчётливо слышна, голос не вязнет.
  Громче — подложка −22 (13 дБ), тише — −26 (17 дБ).

Сайдчейн обязателен: паузы вырезаны, речь идёт сплошняком, и статичная подложка либо давит голос, либо не слышна.

Режь трек по смыслу, а не с нуля. Построй профиль громкости трека, найди дроп и посади его на финальную фразу: смещение = время дропа минус время фразы. Тогда перед ударом трек истончается под концовку — так это делают руками.

Финальный микс приведи к −14 LUFS.

Шаг 6. Мастер

```bash
npx hyperframes render project -o out/render.mp4 --quality high
# звук подставляется в готовый рендер БЕЗ перекодирования видео:
ffmpeg -i out/render.mp4 -i .tmp/audio_mixed.wav -map 0:v:0 -map 1:a:0 \
  -c:v copy -c:a aac -b:a 256k -ar 48000 -shortest -movflags +faststart \
  out/master.mp4
```

Второй прогон по видео только испортил бы картинку — она уже финальная.

```

Алгоритмы, которые надо реализовать точно

Здесь всё, что отличает хороший результат от посредственного.

Маска речи

```
env      = RMS по окнам 10 мс, dBFS
sdb[i]   = max(env[i-3 .. i+3])          # скользящий максимум ±30 мс
loud[i]  = sdb[i] >= -30 dBFS
речь[i]  = loud[i] И длина непрерывного участка loud ≥ 110 мс
```

Скользящий максимум нужен дважды: он не даёт провалу внутри слова сойти за паузу и подтягивает короткие слова над порогом длительности.

Добор глухих окончаний

```
для каждого участка речи:
   k = конец участка
   пока в пределах 250 мс дальше есть хоп с sdb >= -30:
      расширить участок до него
```

Обязан жить **в маске**, а не в подрезке краёв интервала: резы бывают и внутренние, от сжатия пауз.

Границы интервала

```
вход  = первая речь в интервале − 20 мс
выход = последняя речь в интервале + 30 мс
```

Сжатие пауз

```
пауза = участок без речи длиной ≥ 160 мс
из каждой оставить 50 мс (по 25 мс с каждой стороны, середину выбросить)
все точки резать по сетке кадров
```

⚠️ **Порог 160 мс не опускай ниже 150.** Тихий безударный слог короче этого неотличим от паузы, и слово обрубается: «пока вы спите» → «ка вы спите».

При ПЛОТНОСТЬ=естественная поставь `оставить 220 мс`, `порог 400 мс`.

Ловушки — не переоткрывать

Каждая стоила отдельной итерации.

1. Whisper схлопывает повторы. При разборе длинного файла два подряд захода на одну фразу выдаются одной строкой. Расшифровка выглядит чистой, а в звуке фраза звучит дважды. Обратная транскрибация целого файла этого **структурно**

не ловит.
→ Ищи в списке тишины паузы длиннее секунды внутри интервала — это шов между дублями. Проверяй фрагментами до 5 секунд с разбегом из тишины: на 10-секундном клипе повтор всё ещё схлопывается.

2. Дыхание — не тишина. Вдох в близкий микрофон живёт на −36…−42 dBFS, на стыках доходит до −26. Порогом тишины не берётся ни при каком значении: либо пропускает, либо спотыкается о собственный шум и рвёт проверку непрерывности.
→ Отсюда критерий «речь — это то, что держится» выше.

3. Тот же критерий рубит глухие окончания. Финальный глухой слог длится 10–20 мс и бракуется наравне с дыханием.
→ Отсюда добор в маске.

4. Посторонний шум громче порога проходит за речь. Шорох ткани, когда спикер поправляет одежду, держится выше −30 dBFS почти секунду. Все проверки по звуку считают это речью, расшифровка там пустая.
→ Ловится только глазами или по пустому промежутку в расшифровке между репликами. Смотри кадры на стыках.

5. Хвост чужого дубля в начале интервала. Подрезка по речи не спасает — хвост тоже речь. Проверяй структуру интервала до фиксации.

6. `sidechaincompress` отдаёт примерно на секунду меньше, чем получил. Хвост пропадает молча — так исчезает финальный удар музыки.
→ Дополняй оба входа тишиной (`apad`), подрезай выход (`atrim`) и **сверяй длительность результата**.

7. Слова-якоря нестабильны — пунктуация и регистр гуляют между прогонами.
→ Сопоставление без краевой пунктуации и без учёта регистра.

8. Грабли HyperFrames.

Симптом	Решение
`video_missing_muted`	`data-has-audio="true"` на `<video>` — звук несёт само видео
`gsap_exit_missing_hard_kill`	внутренняя обёртка `*-in` + `tl.set(sel,{autoAlpha:0})` в конце твина; твины **никогда** не вешать на сам `.clip`
Конфликт CSS-трансформа и твина	начальное состояние задавать в `gsap.fromTo`, а не в CSS
Столбики диаграммы не видны	`flex:1` растит по ширине — нужен `height:100%`
Контраст падает там, где статика в порядке	аудит смотрит и промежуточные кадры твина; не гонять текст через серый на цветном
Панель полупустая после смены содержимого	скрытые через `autoAlpha` блоки держат высоту — анимировать `height:0` при `overflow:hidden`
`--resolution 1080p` не уменьшает 2560×1440	пресет только повышает DPR; уменьшать отдельным проходом ffmpeg

---

Чек-лист приёмки

Прогони сам, до того как отдать:

1. `npx hyperframes check` — ноль находок по всем пяти аудитам.
2. `npx hyperframes snapshot --at <середины сцен>` и **посмотреть глазами**:
   не порезан ли текст, нормальные ли переносы, не превратились ли буквы в квадраты, сбалансирован ли кадр.
3. Обратная транскрибация мастера — читается как связный сценарий, без

   повторов и обрубленных слов.
4. Остаток тишины измерен и назван числом.
5. `ffmpeg -af ebur128` — −14 LUFS, истинный пик не выше −1 dBFS.
6. `ffprobe` — разрешение и fps как задумано, длительность сходится с суммой
   сегментов, `moov` перед `mdat`.

Порядок общения

- **Гейт после шага 2** — обязателен. Без утверждения отбора дублей не рендерь.
- После сборки отдай превью, дождись правок, и только потом мастер.
- Показывай **измеримое**: сколько тишины осталось, сколько секунд вышло, что
  сказал `check`. Не «стало лучше», а числа.
- Нашёл дефект, о котором не просили, — скажи и почини.

Что отдать

- `out/master.mp4` — мастер
- `project/` — композиция для правок
- `edl_spans.json` и `cues.json` — решения о резе и привязка графики
- `EDIT.md` — раскадровка и где что крутить
- расшифровки исходника и финала