

VapourWiki — справочник пресетов Audion VS Engine (RU)

Содержание

- [Перед началом: доустановить движок](#)
- [Decision tree — какой пресет под какую задачу](#)
- [Палитра PRECISION — технический pipeline \(8 пресетов\)](#)
- [Палитра FILM_LOOKS — эмуляции киноплёнки \(7 пресетов\)](#)
- [Палитра RETRO — аналоговый характер \(3 пресета\)](#)
- [Палитра RESTORATION — суперпушки \(10 пресетов, Phase 18.B + 18.B-ML\)](#)
- [Энкодеры — мои паттерны](#)
- [Установочные и сервисные скрипты — что когда запускать](#)
- [Дополнительно — полная справка](#)
- [AI / ML \(Phase 18.B-ML\) — vs-mlrt стек](#)

Полный русский справочник по 28 пресетам в 4 палитрах. В начале — **decision tree** «какой пресет тянуть для конкретной задачи». Дальше — детальное описание каждого пресета: что делает, под какой материал, ключевые параметры, рекомендуемый энкодер.

Английский эквивалент — docstring внутри каждого `.vpy` файла в `system_core/presets/<palette>/`. Этот документ — русское дополнение для быстрой навигации.

Перед началом: доустановить движок

В раздачу не входят VapourSynth, его плагины и модели vs-mlrt — около семидесяти модулей, у каждого своя лицензия, почти три гигабайта вместе. Программа ставит их сама, с сайтов авторов.

Пока это не сделано, ни один пресет из описанных ниже не запустится.

Запустите `builder_main.cmd` и выберите по порядку: `VAPOURSYNTH` (10), `VS PLUGINS` (11), затем при карте NVIDIA RTX — `VS-MLRT LEAN` (14). Порядок важен: плагинам нужен уже установленный движок.

Decision tree — какой пресет под какую задачу

Читать сверху вниз: первый совпавший случай — ваш пресет.

Симптом материала	Пресет	Палитра
Чересстрочный legacy (DV, HDV, оцифровка VHS, broadcast TS)	<code>qtgmc_deinterlace</code>	restoration
NTSC 29.97 fps с 3:2 pulldown (телекино, плёнка → видео)	<code>tivtc_ivtc</code>	restoration
Шум от высокого ISO, ночная съёмка, сохранить детали	<code>mvtools_mcdegrain</code>	restoration
Радуга / dot crawl на VHS-rip / композитном захвате	<code>derainbow_decross</code>	restoration
Пережатый H.264/MPEG (YouTube-rip, WhatsApp, SD broadcast)	<code>deblock_h264_artefacts</code>	restoration
Дымка / низкая чёткость / нужно «приподнять» картинку	<code>dehaze_local_contrast</code>	restoration
ML upscale 1080p → ~4K (резкие лица, fabric texture)	<code>vsmlrt_realesrgan_2x</code>	restoration
ML интерполяция 24 → 60fps (плавнее Optical Flow)	<code>vsmlrt_rife_60fps</code>	restoration
ML denoise тяжёлого шума / high-ISO без потери текстуры	<code>dpir_denoise</code>	restoration
Цифровой шум только в тенях ("ночная" цифровая съёмка)	<code>shadow_denoise_sota</code> ★	precision
Лёгкий равномерный шум, hardware-agnostic (без CUDA/OpenCL)	<code>mild_denoise</code>	precision
Только хроматический шум (грязный синий канал)	<code>chroma_cleanup</code>	precision

Симптом материала	Пресет	Палитра
Бандинг в градиентах (небо, стена)	<code>deband_safe</code> или <code>deband_fine_grain</code>	precision
Хочу один пресет «всё сразу» под filmic-материал	<code>filmic_rebuild</code> ★	precision
Подготовка к colour grading в DaVinci	<code>pregrade_prep</code>	precision
Чистый архивный мастер без зерна	<code>archive_clean</code>	precision
Тонкий filmic look, безопасный default	<code>cinematic</code>	film_looks
Конкретный 35mm Kodak (250D / 500T / 50D)	<code>film_35mm</code>	film_looks
16mm органичный, поднятые тени	<code>film_16mm</code>	film_looks
Super 8, выцветшие 70-е, тяжёлое зерно	<code>super8</code>	film_looks
High contrast desaturated (Se7en, Saving Private Ryan)	<code>bleach_bypass</code>	film_looks
Large-format IMAX feel — минимум зерна, gate weave	<code>imax_70mm</code>	film_looks
Hollywood cinemascope (горизонтальные blue lens flares + teal shadows)	<code>anamorphic_scope</code>	film_looks
VHS / CRT эстетика	<code>vhs_crt</code>	retro
90s любительский camcorder	<code>camcorder_90s</code>	retro
1970s Polaroid SX-70 (candy-bloom, тёплая кремовость)	<code>polaroid</code>	retro

Pipeline-сценарии (несколько пресетов цепочкой через `apply-profile-batch` или ручной запуск):

Сценарий	Шаги
Ночной timelapse → плёнка	<code>shadow_denoise_sota</code> → <code>filmic_rebuild</code> → <code>cinematic</code>
Архив VHS → отреставрированный мастер	<code>qtgmc_deinterlace</code> → <code>derainbow_decross</code> → <code>mvtools_mcdegrain</code> → <code>archive_clean</code>
Старый YouTube-rip → пригодный для проекта	<code>deblock_h264_artefacts</code> → <code>dehaze_local_contrast</code> → <code>pregrade_prep</code>

Сценарий	Шаги
Telecined NTSC DVD → 24p мастер	<code>tivtc_ivtc</code> → <code>archive_clean</code>
Современный цифровой → плёночный лук	<code>mild_denoise</code> → <code>film_35mm</code>

Палитра PRECISION — технический pipeline (8 пресетов)

Меню в порядке pipeline-логики: Stage 1 (denoise) → Stage 2 (deband) → Stage 3 (compositions).

Stage 1 — шумодав

`mild_denoise` — мягкий универсальный шумодав

DFTTest спектральный шумодав. Pure CPU, hardware-agnostic — работает одинаково на любом процессоре. Рекомендуется как **первый выбор** когда не известно про материал ничего конкретного: лёгкий шум, нужно немного почистить без риска.

- Параметры: `strength` = light (sigma=4.0) / medium (8.0) / strong (14.0)
- Энкодер: `h264_crf17` или `h264_crf14` для архивного качества
- Что делает в Adobe/DaVinci: только Temporal NR / Noise Reduction, грубее по результату

`shadow_denoise_sota` ★ — флагман шумодава теней

BM3D с float32-конвейером + smooth luma-mask «только тени». Шум давится **только** в зонах ниже `shadow_threshold`, остальная картинка не трогается. Хрома плоскости денойзятся всегда (хроматический шум одинаково уродлив везде). Опциональный возврат микрозерна `grain_back` чтобы тени не выглядели «пластиковыми».

- Параметры: `sigma=2.5` (BM3D luma), `use_cuda=0/1`, `grain_back=0.6`, `shadow_threshold=0.20`, `transition=0.10`
- На NVIDIA: `--use-cuda 1` даёт 25–35% выигрыш wall-time на 4K
- Энкодер: `h264_crf17` / `prores_lt`
- Чего нет в NLE: zone-targeted denoise с smooth blend без дешёвого «threshold mask»

chroma_cleanup — очистка только хромы

DFTTest на planes=[1,2]. Luma не трогается. Применяется когда «грязный синий канал» (типичная проблема старых компактных камер, низкобитных AVCHD).

- Параметры: **strength** = light / medium / strong
- Энкодер: **h264_crf17**
- Чего нет в NLE: чистый chroma-only DFTTest без luma-побочки

Stage 2 — дебанд

deband_safe — безопасный дебанд

neo_f3kdb с лёгкими настройками, без зерна. Убирает бандинг в градиентах (небо, стена, ночное освещение) без потери резкости.

- Параметры: **range=15**, **y=64**, **cb=64**, **cr=64**
- Энкодер: **h264_crf17** / **h265_crf21**

deband_fine_grain — дебанд + восстановление тонкого зерна

Тот же neo_f3kdb + AddGrain. После дебанда добавляется лёгкое монотонное зерно — исключает «слишком чистую» картинку (характерное свойство дешёвой компрессии).

- Параметры: **range=15**, **grain_var=1.5**
- Энкодер: **prores_lt** (для последующего грейдинга)

Stage 3 — композиции

filmic_rebuild ★ — флагман композиций

Полная цепочка: BM3D denoise → neo_f3kdb deband → 3-zone luma grain (тени / midtones / highlights). Зерно зональное — больше в тенях (как настоящая киноплёнка), меньше в highlights. Рекомендуются как «один пресет на всё» для filmic-материала.

- Параметры: **sigma=2.0**, **use_cuda=0/1**, **deband_range=14**, **grain_shadow=1.5**, **grain_mid=0.9**, **grain_high=0.4**, **shadow_threshold=0.30**, **high_threshold=0.65**
- Энкодер: **prores_lt** / **h265_crf21**

archive_clean — нейтральный архивный мастер

DFTTest + neo_f3kdb. **Без зерна.** Цель — максимально чистая «плоская» картинка для архивирования. Не использовать если планируется грейдинг (он лучше работает на материале с зерном).

- Параметры: `denoise=medium`, `range=15`
- Энкодер: `prores_422hq` / `prores_422hq_mxf`

pregrade_prep — подготовка под DaVinci

Минимальное воздействие: только лёгкий DFTTest. Без дебанда, без зерна. Цель — отдать в Resolve самый чистый источник, чтобы колорист не работал поверх артефактов компрессии.

- Параметры: `strength=light`
- Энкодер: `prores_lt_mxf` (для round-trip с Adobe / Avid)

Палитра FILM_LOOKS — эмуляции киноплёнки (7 пресетов)

Каждый look использует общую библиотеку `audion_lib` (MTF-софтинг, halation bloom, zone grain, gamma curve, black-lift, desaturation).

cinematic — универсальный subtle filmic

Безопасный default. Лёгкий MTF softening + halation + микрозерно. Не привязан к конкретной плёнке. Работает на любом современном digital-материале.

- Параметры: `intensity=1.0` (диапазон 0..2)
- Энкодер: `prores_lt` / `h264_crf17`

film_35mm — Kodak 35mm с stock-вариантами

Эмуляция реальных Kodak stocks: 250D (дневной баланс), 500T (вольфрам), 50D (мелкое зерно). Каждый stock имеет свою кривую гаммы, баланс, плотность зерна.

- Параметры: `stock=250D|500T|50D`, `intensity=1.0`
- Энкодер: `prores_lt` / `prores_lt_mxf`

film_16mm — органичный, более зернистый

Поднятые тени (lifted blacks), более выраженное зерно, мягче 35mm. Подходит для «documentary» / arthouse эстетики.

- Параметры: **intensity=1.0**
- Энкодер: **prores_lt**

super8 — самый сильный лук

Самое тяжёлое зерно, самая мягкая оптика, выцветшие 70-е. Для music video / стилизованных вставок.

- Параметры: **intensity=1.0**
- Энкодер: **h264_crf17** (зерно уже встроено, aggressive compression OK)

imax_70mm — large-format IMAX feel ★

Противоположность Super 8: **минимум зерна, максимум разрешения**, очень мягкая halation на highlights, опциональный 1px gate weave (детерминированный, seed=42 → reproducible). Большой кадр настоящего 70mm IMAX → каждое серебряное зерно крошечное относительно картинки. Подходит для шотов, которые должны ощущаться «эпично», а не «плёночно».

- Параметры: **intensity=1.0**, **gate_weave=1** (вкл по умолчанию)
- Энкодер: **prores_lt** / **h265_crf17**
- Чего нет в NLE: калиброванный микро-grain плюс gate weave (NLE'и делают только flat plate grain)

anamorphic_scope ★ — Hollywood cinemascope

Сигнатура anamorphic-оптики: **длинные горизонтальные blue lens flares** только на highlights (ровно те «горизонтальные синие черты» через всё небо в фильмах Abrams / Nolan / Villeneuve, которые делаются Panavision / Hawk / ARRI Master Anamorphic). Плюс subtle teal-cool тени для классического blockbuster-эстетики.

В нашей реализации: highlight-mask (**flare_thr**) → горизонтальный low-pass blur длиной **flare_len** → blue tint (R 10% / G 55% / B 100%) → additive merge поверх оригинала. Затем shadow-masked teal shift (R -10% / G +5% / B +12%). Всё в 16-bit RGB чтобы streak'и не клипались.

- Параметры: **intensity=1.0**, **flare_len=96** (48 subtle / **96 default** / 160 dramatic / 256 extreme), **flare_thr=0.78** (0.85 highlights-only / **0.78 default** / 0.70 aggressive / 0.60 heavy), **teal_shadow=0.35** (0.0 off / 0.20 subtle / **0.35 default** / 0.60 heavy)

- Энкодер: `prores_lt` (для colorist) или `h265_crf14` (final)
- Пресет HE кропает кадр в 2.39:1 — это творческое решение пользователя; добавляйте `ffmpeg -vf crop=W:floor(W/2.39):0:Y` в post если нужен true score.
- Чего нет в NLE: directional anamorphic streak без дорогих платных плагинов типа Optical Flares; smooth threshold для flare cut-in; full 16-bit math.

`bleach_bypass` — high contrast desaturated

«Se7en» / «Saving Private Ryan» лук. Высокий контраст, серебряная серость, яркие highlights. Жёсткий стилистический выбор.

- Параметры: `intensity=1.0`
- Энкодер: `prores_lt`

Палитра RETRO — аналоговый характер (3 пресета)

`vhs_crt` — VHS / CRT

Chroma bleed (горизонтальное растекание цвета), мягкая оптика, аналоговый шум. Эмуляция воспроизведения с VHS-кассеты на CRT-телевизоре.

- Параметры: `intensity=1.2`
- Энкодер: `h264_crf17`

`camcorder_90s` — любительский camcorder

Мягче VHS, лёгкая переэкспозиция, минимальный chroma bleed. Эстетика домашних видео 90-х.

- Параметры: `intensity=1.0`
- Энкодер: `h264_crf17`

`polaroid` ★ — 1970s Polaroid SX-70

Сильная candy-bloom на highlights, lifted blacks (никогда не идёт в чёрный, кремовое затухание), warm gamma push, mild desaturation, **radial edge vignette**. Распадается на 5 шагов — то что в NLE требует стэка из 5-6 эффектов вручную.

- Параметры: `intensity=1.0`, `vignette=0.55` (диапазон 0..1; 0 = нет, 1 = углы в чёрный)
- Энкодер: `h264_crf17` (плёночный материал терпит compression)

- Чего нет в NLE: one-shot Polaroid emulation; radial vignette без геометрической дисторсии или LUT-побочки

Палитра RESTORATION — суперпушки (10 пресетов, Phase 18.B + 18.B-ML)

То, чего **нет или плохо реализовано** в Adobe Premiere / DaVinci Resolve.

qtgmc_deinterlace — эталонный деинтерлейс

havsfunc.QTGMC (NNEDI3 + MVTools motion estimation). Реконструирует прогрессивные кадры из чересстрочного источника через neural network upscaling каждого поля + motion-compensated temporal smoothing. **Лучше любого NLE out-of-the-box.**

- Параметры: `field_order=tff|bff`, `qtgmc-preset=Faster|Fast|Medium|Slow|Slower|Placebo`, `output_fps=single|double`
- Когда брать: DV, HDV, оцифровка VHS, broadcast TS, оцифровка S-VHS / U-matic
- Энкодер: `prores_lt` (для последующей работы) или `h264_crf17` (если final master)

tivtc_ivtc — обратный telecine

TIVTC.TFM (field matching) + TIVTC.TDecimate (drop duplicate). Reference-quality 3:2 pulldown removal: NTSC 29.97i → 23.976p. Возвращает оригинальный плёночный 24p мастер из telecined источника.

- Параметры: `pp=6` (TFM post-processor), `cycle=5`, `rdrop=1` (стандартный NTSC паттерн)
- Когда брать: NTSC DVD, broadcast prints, digitized film transfers
- Энкодер: `prores_422` / `prores_422hq_mxf`

mvtools_mcdegrain — motion-compensated denoise

MVTools motion estimation → MDeGrain temporal averaging вдоль motion vectors. **Убирает шум БЕЗ потери детализации** — то что Topaz Video Enhance AI и Neat Video делают внутри. DaVinci Temporal NR — coarse implementation того же; здесь reference-grade.

- Параметры: `radius=2` (5-кадровое окно), `thsad=200`, `blksize=16` (HD) / `8` (4K detail)
- Когда брать: high-ISO ночная съёмка, сохранение текстуры кожи / фактуры тканей
- Энкодер: `prores_lt` / `h265_crf17`

derainbow_decross — NTSC composite cleanup

MVTools-based motion-compensated chroma-only smoothing. Убирает rainbow / dot crawl / cross-color на материале с композитного захвата (VHS-rip, U-matic, BetaSP, S-Video → SDI). Luma не трогается.

- Параметры: `strength=0.6`, `blksize=16`
- Когда брать: оцифрованные VHS, цветные «пробежки» на тонких линиях, mosquito noise на чёрно-белых границах
- Энкодер: `prores_lt` / `h264_crf17`

deblock_h264_artefacts — спасение пережатого

havsfunc.Deblock_QED — edge-aware deblocker для H.264 / MPEG-2 / MPEG-4. Сглаживает 8x8 границы блоков, ringing вокруг edges, mosquito noise. Edge-aware — не трогает реальную детализацию.

- Параметры: `quant1=24`, `quant2=26`, `aoffset=1`, `boffset=1`
- Когда брать: старые YouTube-rip, WhatsApp/Telegram re-encode, низкобитные SD broadcast TS
- Энкодер: `h264_crf17` (после deblock материал чище, можно re-encode на более высокий CRF)

dehaze_local_contrast — clarity без halos

Luma-only local contrast через high-pass + zone-weighted MaskedMerge. Zone-mask (parabolic, peak в midtones) предотвращает выгорание highlights и crushed shadows. Цвет не сдвигается (luma-only). 16-bit math.

- Параметры: `strength=1.0`, `radius=8` (clarity) / `12-16` (haze removal)
- Когда брать: дымка, плоский low-contrast материал, нужно «оживить» картинку без destroy highlights
- Чего нет в NLE: Resolve "Dehaze" сжигает highlights и сдвигает цвет; здесь без halos
- Энкодер: `prores_lt` (handoff в colorist) или `h264_crf17` (final)

Энкодеры — мои паттерны

(Audion default workflow по результатам обсуждения 2026-04-28)

Quality ladder (одинаковая 14 / 17 / 21 сетка для software, NVENC, QuickSync и AMF): **14** → **17** → **21**, три различные ступени без overlap.

Профиль	Когда использовать
<code>h264_crf14</code> / <code>h265_crf14</code> ★ default	Semi-lossless архив, мастер для последующей работы. Default Audion с 2026-04-28.
<code>h264_crf17</code> / <code>h265_crf17</code>	«Почти невидимый lossy», финальный master
<code>h264_crf21</code> / <code>h265_crf21</code>	Web preview / проху / превью
<code>prores_lt</code> ★	Audion default ProRes — без keying, под grading и round-trip
<code>prores_lt_mxf</code> ★	ProRes LT в MXF wrapper — для Adobe Premiere / Avid round-trip
<code>prores_422</code>	Если нужно чуть больше bitrate чем LT
<code>prores_422_mxf</code>	ProRes 422 в MXF wrapper для Adobe / Avid handoff
<code>prores_422hq</code>	Только если планируется keying
<code>prores_422hq_mxf</code>	HQ + MXF для broadcast / Avid finishing
<code>dnxhr_lb</code> / <code>_sq</code> / <code>_hq</code> / <code>_hqx</code>	Avid-style DNxHR (LB low / SQ standard / HQ high / HQX 10-bit)
<code>h264_nvenc_q14</code> / <code>_q17</code> / <code>_q21</code>	NVENC hardware-encode H.264 на NVIDIA. Снимает CPU-bottleneck с libx264 — VS-фильтрация не упирается в encode. CQ — аналог CRF. 8-bit yuv420p.
<code>h265_nvenc_q14</code> / <code>_q17</code> / <code>_q21</code>	NVENC hardware-encode HEVC, 10-bit p010le. Идеально для full pipeline на NVIDIA: VS-фильтрация на CPU/CUDA + encode на NVENC = ноль CPU-затыка.
<code>h264_qsv_q14</code> / <code>_q17</code> / <code>_q21</code>	Intel QuickSync H.264. Удобно для Intel iGPU batch/проху, когда CPU нужен VapourSynth.
<code>h265_qsv_q14</code> / <code>_q17</code> / <code>_q21</code>	Intel QuickSync HEVC, p010le для 10-bit output где поддерживается.
<code>h264_amf_q14</code> / <code>_q17</code> / <code>_q21</code>	AMD AMF H.264 через CQP, та же визуальная ladder-сетка.

Профиль	Когда использовать
<code>h265_amf_q14</code> / <code>_q17</code> / <code>_q21</code>	AMD AMF HEVC, p010le output; полезно на Radeon-хостах.

Правило: для любой не-final обработки → `prores_lt` / `prores_lt_mxf` или DNxHR, если принимающее приложение любит DNx. Для final delivery → `h264_crf14` / `h264_crf17` (software, лучшая плотность на бит) ИЛИ hardware ladder под текущий хост (`nvenc`, `qsv`, `amf`), когда важен wall-time.

Когда использовать hardware encode vs software

Сценарий	Encoder
Тяжёлый VS-pipeline (filmic_rebuild, mvtools_mcdegrain) на NVIDIA	NVENC — CPU освобождается под VS-фильтр, общий wall-time падает на 30-50%
Финальный delivery где важна максимальная плотность бит	libx264/libx265 CRF — software encoder всё ещё чуть точнее на низких CRF
Intel iGPU / ноутбучный batch	QuickSync — хороший throughput при низкой нагрузке на CPU
AMD/Radeon host	AMF — та же 14/17/21 CQP ladder, без CPU encode bottleneck
Большой батч на сотни файлов	NVENC / QSV / AMF — экономия времени накапливается
Нет рабочего hardware encoder	Только software (<code>h264_crf*</code> / <code>h265_crf*</code>)

NVENC требует: NVIDIA driver R525+. Качество: NVENC на Ada/Blackwell (RTX 40/50) **сравнимо** с libx264 medium на одинаковом CQ; на старых поколениях (Pascal/Turing) software CRF немного выигрывает в эффективности bitrate за то же качество, но NVENC всё равно быстрее в разы.

Установочные и сервисные скрипты — что когда запускать

Скриптов в `install/` много, и не все очевидны. Этот раздел — карта «когда что зовёшь» с конкретными аргументами и примерами. Все скрипты идут парами `.cmd` (тонкий wrapper,

резолвит portable PowerShell) + `.ps1` (вся логика). Запускать обычно из `builder_main.cmd` (FZF-меню), но любой можно дёрнуть напрямую двойным кликом или из CLI.

Точка входа: `builder_main.cmd`

Главное меню сервисных операций. FZF-навигация (если есть `system_core\fzf.exe`), иначе CMD-fallback с буквенными хоткеями. Карта пунктов:

#	Пункт	Что делает	Скрипт под капотом
[01]	Build portable env CMD builder	первичная сборка <code>runtime/</code>	<code>Build_Portable_Env_Build.cmd</code>
[02]	Build portable env PS	альтернативный билдер на PowerShell	<code>Build_Portable_Env.ps1</code>
[03]	Install portable offline	оффлайн-установка из заранее скачанного wheelhouse	<code>install_portable_offline.cmd</code>
[04]	Verify portable env	проверка orchestrator-питона	<code>verify_portable_env.cmd</code>
[05]	Update FZF	обновить <code>fzf.exe</code> до latest	<code>launcher-tools-update_fzf.cmd</code>
[06..08]	Licenses	сбор / прюнинг / дедупликация лицензий	<code>system_core\license\Run-*.cmd</code>
[09]	Make release archive	релизный zip с исключениями <code>output/</code> , <code>logs/</code> , <code>._runtime/</code> , <code>install/download/</code> , приватного <code>MEMORY.md</code>	<code>make_release_archive.cmd</code>
[04]	PowerShell	portable pwsh 7 → <code>system_core\powershell\</code>	<code>Install-Portable-PowerShell.cmd</code>
[10]	VapourSynth	latest VS stable + latest Python 3.12.x embed → <code>system_core\vapoursynth\</code>	<code>Install-Portable-VapourSynth.cmd</code>
[11]	VS plugins	vsrepo + 12 плагинов (включая <code>bm3dcuda</code> по умолчанию)	<code>Install-VS-Plugins.cmd</code>

#	Пункт	Что делает	Скрипт под капотом
[12]	FFmpeg	Exact-stable Gyan по политике драйвера; BtbN rolling только opt-in → <code>Tools\ffmpeg\bin\</code>	<code>Install-Portable-FFmpeg.cmd</code>
[13] ★	VS-MLRT LEAN	fresh-download vs-mlrt TensorRT bundle, очистка <code>plugins\vsmlrt</code> , lean-trim под текущий SM	<code>Install-VS-mlrt.cmd /LEAN</code>
[14]	VS-MLRT FULL	fresh-download тот же install, но без trim — для USB-distribution	<code>Install-VS-mlrt.cmd /FULL</code>
[70]	Clean install cache	очистка transient install downloads, staging dirs и bytecode caches с сохранением portable payloads	<code>Clean-Install-Cache.cmd</code>
[90..99]	Open / Project	проводник по подпапкам / переход в project launcher	—
[00]	Exit	—	—

Порядок «с нуля»: [01] -> [03] -> [04] -> [10] -> [11] -> [12] -> (опционально) [13] -> [70]. После всех install-шагов один раз `runtime\python.exe system_core\doctor.py` — должен быть весь зелёный.

`install\Install-Portable-VapourSynth.{cmd,ps1}` — latest VS stable + own Python 3.12.x

Резолвит latest stable VapourSynth и latest Python 3.12.x embed, очищает `system_core\vapoursynth\`, ставит подходящий VapourSynth wheel, прописывает `portable.vs` marker (чтобы vsrepo переходил в portable mode). После wheel install явно создаёт и печатает active plugin dir из `vapoursynth.get_plugin_dir()`; VS R74+ уже не ориентируется на legacy `vs-plugins\`. Использует `Expand-7zArchive` (через `Ensure-7zip.ps1`) — на больших архивах ~×3 быстрее `Expand-Archive`.

В конце вызывает `Repair-PipShims.cmd` — чинит shebang в `Scripts\*.exe` сразу после wheel install (свежая установка не требует ручного fix).

Флаги: `/R <rev>` (например `/R R76` для пинования версии). `/F` принимается для совместимости; текущий installer и так refresh по умолчанию.

`install\Install-VS-Plugins.{cmd,ps1}` — 12 плагинов через vsrepo

Ставит:

- v1.0 set: `lsmas`, `ffms2`, `fmtconv`, `neo_f3kdb`, `addgrain`, `knlmeanscl`, `bm3dcpu`, `dfttest`
- Restoration set (Phase 18.B): `havsfunc`, `mvsfunc`, `mvtools`, `tivtc`
- CUDA: `bm3dcuda` **по умолчанию** (Phase 18.A0). Opt-out — `/NO-CUDA` (на чистом не-NVIDIA host'e, чтобы не ставить плагин который всё равно фейлится при load).

Дополнительно ставит `vsutil` через pip (импортируется внутри `havsfunc.py` на module top-level — без него Restoration-пресеты падают с `ModuleNotFoundError`).

Update-safe: `vsrepo update` обновляет базу, installer очищает активный `vapoursynth.get_plugin_dir()` path с сохранением `plugins\vsmlrt` если он уже есть, затем `vsrepo install` заново ставит нужный список. Безопасно дёргать после обновления списка.

`install\Install-Portable-FFmpeg.{cmd,ps1}` — BtbN GPL latest, Gyan.dev fallback

По умолчанию устанавливает совместимую с драйвером exact-stable сборку Gyan. Rolling release-branch BtbN доступна только через явный opt-in. Кладёт `ffmpeg.exe` / `ffprobe.exe` / `ffplay.exe` в `Tools\ffmpeg\bin\`. Через `Expand-7zArchive` распаковывает быстро.

Флаги: `/V <variant>` (`gpl` default / `lgpl` / `gpl-shared`), `/F` (force).

`install\Install-VS-mlrt.{cmd,ps1}` — ML-стек (Phase 18.B-ML)

Только для тех кто хочет ML-пресеты (`vsmlrt_realesrgan_2x`, `vsmlrt_rife_60fps`, `dpir_denoise`). Ставит полный TensorRT bundle: VSTRT + VSORT + VSOV + VSNCNN + ONNX runtime + TensorRT runtime DLLs + полная коллекция ONNX-моделей. После распаковки Audion выносит OpenVINO/vsov из активной autoload-папки в `system_core\vapoursynth\disabled_plugins\vsmlrt-openvino`: TensorRT / TensorRT-RTX / ONNX Runtime остаются активны, а Windows `Bad Image 0xc0e90002` от OpenVINO DLL не ломает запуск.

Working set после install: ~3.5 GB при `/LEAN` (по умолчанию), ~10 GB при `/FULL`. Каждый запуск fresh-download выбранных vs-mlrt assets и очищает только subtree `plugins\vsmlrt` перед копированием runtime DLLs, scripts и models. Распаковка через `7zr.exe` (нужно для BCJ2-filtered streams в TRT runtime DLL — `py7zr` это не умеет).

Порядок Full-update: если запускался [10] VAPOURSYNTH или [11] VS PLUGINS, после них обязательно запускать [13] VS-MLRT LEAN заново перед Full all-presets smoke. ML-пресеты нельзя валидировать на старом MLRT-слое после обновления базового VS runtime/plugins.

Флаги:

- /LEAN (default через [13]) — тримит TensorRT builder resources до текущего SM (детектится через `nvidia-smi --query-gpu=compute_cap`), удаляет неиспользуемые модели (cugan, waifu2x). На non-NVIDIA fallback на FULL поведение.
- /FULL (через [14]) — полный bundle для USB-distribution на чужую машину.
- /DROP-CACHE — после успеха вызывает центральную политику Clean-Install-Cache для transient install downloads, staging dirs и bytecode caches.
- По умолчанию из builder = /LEAN и install cache сохраняется. Для удаления cached downloads используйте [70] Clean install cache или явный /DROP-CACHE.

После install — backend в ML-пресетах выбирается **автоматически**: TRT (если NVIDIA + corresponding TRT runtime) → ORT_DML (DirectML, любой DX12 GPU включая Intel Xe / AMF) → ORT_CPU. Cross-vendor устойчиво (через `audion_lib.vsmmlrt_backend_chain()` — на не-NVIDIA даже не пытаемся компилировать TRT engine, экономим 30–120 секунд).

Гадость: Windows Defender блокирует неподписанный `openvino_intel_npu_plugin.dll`. Скрипт автоматом удаляет его после extract (для NVIDIA / DirectML setup'ов он не нужен).

`install\Clean-Install-Cache.{cmd,ps1}` — освобождение диска

Чистит transient install downloads (`.7z`, `.zip`, `.tar.*`, `.msi`, `.exe`), точные installer staging dirs и Python bytecode caches вне payload/user-data зон. Preserve-list внутри `install\download\` (всегда сохраняется):

- `.gitkeep`
- `get-pip.py` (пере-используется при каждой свежей сборке)
- `7z*-extra.7z` (Ensure-7zip bootstrap helper, ~2 MB)

Когда нужен: после успешного [13] VS-MLRT LEAN или [14] VS-MLRT FULL (~3.5 GB compressed архивов), периодически после `Install-Portable-*` (свежий FFmpeg build / новый VS R75+).

Философия: **storage > bandwidth**. Re-download = ~1.5 минуты на гигабите, а 3.5 GB на диске сидят навсегда. Working set после lean+cleanup: ~3.5 GB вместо 10.

`install\Repair-PipShims.{cmd,ps1}` — починка `Scripts\*.exe` после переезда

Симптом: после переноса проекта на другую букву диска / другой путь / другую машину, `vspipe.exe` и другие `Scripts\*.exe` молча возвращают exit 1 без вывода. `doctor.py` показывает `[FAIL] vspipe runs`, при этом `python.exe -c "import vapoursynth"` работает.

Причина: pip вшивает абсолютный shebang `#!/"<полный путь к python.exe>"` внутрь каждого `Scripts\*.exe`. После переезда этот путь больше не существует.

Что делать: запустить `install\Repair-PipShims.cmd`. Перезаписывает shebang во всех `Scripts\*.exe` на текущий `system_core\vapoursynth\python.exe`. Stub launcher'a и trailing-zip остаются нетронутыми.

Флаги:

- `/WHATIF` или `/N` — dry-run без записи (показывает что бы изменилось).

В нормальной работе **не нужен**: `system_core\engine\selfheal.py` хукнут в начало `main.py` и `doctor.py` — на старте читает первые 16 KB `vspipe.exe`, парсит embedded shebang, при mismatch silently зовёт `Repair-PipShims.cmd`. Идемпотентно через `AUDION_SELFHEAL_DONE=1`. Manual repair нужен только если бьёшь vspipe мимо `main.py` / `doctor.py` (например прямым CLI бенч-скриптом, который не успел пройти self-heal).

`install\Bench-CUDA.{cmd,ps1}` — pure-pipeline CPU vs CUDA на VM3D

Что бенчит: два VM3D-тяжёлых пресета (`shadow_denoise_sota` и `filmic_rebuild`), каждый прогоняется дважды — на CPU и на CUDA. Pipeline `vspipe → ffmpeg -f null` (без энкодинга). Без энкодинга нужно чтобы libx264 не съел всё wall-time на быстром многоядернике и не спрятал CUDA-выигрыш.

Как использовать:

```
:: Drag-n-drop файла на Bench-CUDA.cmd:
:: или из CLI:
install\Bench-CUDA.cmd "C:\clips\test.mov"
install\Bench-CUDA.cmd "C:\clips\test.mov" 2.5    :: второй аргумент = sigma (default 2.5)
```

Вывод: 4 строки с таймингом + итоговая таблица CPU/CUDA + Δ %. На диск ничего не пишется.

Как читать:

- На референсном клипе DCI 4K ProRes 25 секунд (RTX 5070 / Ryzen 9 5900X):
`shadow_denoise_sota` = -35%, `filmic_rebuild` = -24%.
- На клипах короче ~5 секунд CUDA setup-cost не амортизируется → результат может быть шумом или даже немного медленнее CPU.
- **Task Manager обманывает**: даже на работающем CUDA-пути часто показывает 5–10% GPU. BM3D bandwidth-bound и работает всплесками; вокруг него `fmtc` bit-depth конвертации и frame-prop тэги — на CPU. Авторитетный сигнал «CUDA реально живой» — wall-time delta и `[OK] bm3dcuda runtime - live BM3D CUDA invocation succeeded` в `doctor.py`.

`install\Bench-AllPresets.{cmd,ps1}` ★ — smoke по всем 28 пресетам

Walker по `system_core\presets\<palette>\*.vpy`. Каждый пресет прогоняется через `vspipe -c y4m --end <Frames-1> | ffmpeg -f null` — pure pipeline, без записи на диск. Smoke-цель: убедиться что (а) пресет парсится, (б) все плагин-namespaces'ы резолвятся, (в) кадры доезжают до ffmpeg без exception.

Как использовать:

```
:: Drag-n-drop:
:: Или из CLI:
install\Bench-AllPresets.cmd "C:\clips\test.mp4"
install\Bench-AllPresets.cmd "C:\clips\test.mp4" 30 sweep auto
```

Текущие локальные references:

- Portable CPU/ORT fallback: `28/28 PASS` при `Frames=1`, `Cuda=off`, `MLBackend=ort_cpu` (2026-05-16).
- RTX 5070 validation: `36/36 PASS` при `Frames=1`, `Cuda=sweep`, `MLBackend=auto` (2026-05-27). 36 строк — это 28 пресетов плюс дополнительный CPU/CUDA sweep для Precision.

Аргументы:

1. Путь к видео (обязательно)
2. Frames per preset (default 30)
3. Cuda mode: `off` / `on` / `sweep` (default off)
 - `off` — все пресеты CPU
 - `on` — все Precision-пресеты с `AUDION_VS_USE_CUDA=1`

- `sweep` — Precision гонится дважды (CPU и CUDA), остальные палитры один раз

4. ML backend: `auto` / `trt` / `ort_dml` / `ort_cpu` / `sweep` (default auto)

- `auto` — vs-mlrt сам выбирает TRT → ORT_DML → ORT_CPU
- `sweep` — ML-пресеты гонятся дважды (auto и ort_cpu) для сравнения GPU vs CPU baseline

Сколько строк выходит:

- `off auto` — по одной строке на пресет (28 строк всего)
- `sweep auto` — $\text{Precision} \times 2 + \text{остальные} \times 1 = 8 \times 2 + 7 + 3 + 10 = 36$ строк
- `sweep sweep` — $\text{Precision} \times 2 + \text{Restoration ML} \times 2 + \text{остальные} \times 1 = 16 + 7 + 3 + 6 + 4 \times 2 = 40$ строк

В конце — Summary `Total / PASS / FAIL`, для `-Cuda sweep` — таблица `CPU vs CUDA Δ %` по Precision-пресетам, плюс JSON-отчёт в `logs\bench_all_presets_<TS>.json`.

Когда запускать:

- После `Install-VS-Plugins` или `Install-VS-mlrt` — убедиться что все пресеты живые (`PASS == Total`).
- После переезда проекта (post-`Repair-PipShims`).
- После апгрейда драйвера NVIDIA — что TRT engine'ы пересоберутся без ошибок.
- При добавлении нового пресета — что он не сломал импорт у соседей.

Что в нём важного скрыто:

- Per-preset env overrides (`$presetOverrides` table в `Bench-AllPresets.ps1`) — без них `AUDION_VS_STRENGTH=medium` (Precision-тер) ломает Restoration-пресеты (`dehaze`, `derainbow`), которые парсят STRENGTH как float. И `AUDION_VS_MODEL` разный для ESRGAN / RIFE / DPIR.
- Отдельные stderr-temp-файлы для vspipe и ffmpeg (один cmd.exe pipe не может сразу `2>file.log` от обоих процессов — `ERROR_SHARING_VIOLATION`).
- Local row-vars **не** называются `$cuda` — это бы клобберило script-param `$Cuda` (PowerShell case-insensitive variables → ValidateSet ломается на assignment).

`install\Ensure-7zip.ps1` — dot-source helper для 7-Zip

Не запускается напрямую — другие `*.ps1` его dot-source'ят:

```
. "$PSScriptRoot\Ensure-7zip.ps1"
$exe = Ensure-7zr -ProjectRoot $ProjectRoot
Expand-7zArchive -Archive $zip -Destination $dst -ProjectRoot $ProjectRoot
```

Build env steps [01]/[02] явно ставят 7zr.exe (~1.5 MB, чистый 7z extractor с поддержкой BCJ2) и 7za.exe (~1.7 MB, универсальный — zip / 7z / tar / gz / bz2 / xz) в system_core\7zip\ до установки Python. Один раз скачался — путешествует с проектом.

Зачем нужен:

- Expand-Archive (.NET ZipArchive) чокается на ZIP'ах больше 2 GB (BtbN FFmpeg, VS portable) и memory-hungry.
- py7zr не декодирует BCJ2-filtered streams (vs-mlrt TensorRT runtime DLLs).
- 7za стабилен по форматам и портативен.

Сводная карта «когда что зовёшь»

Сценарий	Скрипты в порядке
Свежая установка с нуля	builder_main → [01] → [10] → [11] → [12] → [13] → опц. [14] → [16] → doctor.py
Перенёс проект на другой диск / машину	doctor.py (selfheal сам зовёт Repair-PipShims) — или вручную install\Repair-PipShims.cmd
Нужен/обновлён ML-стек	builder_main → [13] VS-MLRT LEAN после [10] / [11] / [12], затем Bench-AllPresets с -MLBackend sweep
Хочу проверить что CUDA реально живой	install\Bench-CUDA.cmd <video> — измеряет реальный wall-time delta
Хочу проверить что все 28 пресетов работают локально без NVIDIA	system_core\powershell\pwsh.exe -NoLogo -NoProfile - ExecutionPolicy Bypass -File install\Bench-AllPresets.ps1 - ProjectRoot . -InputFile <video> -Frames 1 -Cuda off -MLBackend ort_cpu
RTX / TensorRT validation	system_core\powershell\pwsh.exe -NoLogo -NoProfile - ExecutionPolicy Bypass -File install\Bench-AllPresets.ps1 - ProjectRoot . -InputFile <video> -Frames 1 -Cuda sweep -MLBackend auto

Сценарий	Скрипты в порядке
На диске стало мало места	builder_main → [16] CLEAN INSTALL CACHE — освобождает ~3.5 GB compressed архивов
Обновил NVIDIA driver / поменял GPU	doctor.py (live bm3dcuda smoke-test) → Bench-CUDA → Bench-AllPresets (TRT engine'ы пересоберутся при первом ML-пресете)
Добавил новый пресет	Bench-AllPresets для smoke + добавить запись в engine/presets.py + добавить пункт в launcher + обновить эту вики
Готовлю release archive	builder_main → [09] Make release archive — output исключает output/, logs/, .runtime/, install/download/, release/, API keys и приватный MEMORY.md; MEMORY.example.md остаётся публичным

Дополнительно — полная справка

- Английская справка по каждому пресету: docstring в system_core/presets/<palette>/<preset>.vpy
- CLI флаги: runtime\python.exe system_core\main.py run --help
- Профили (cross-палитровые комбинации): runtime\python.exe system_core\main.py list-profiles
- Recursive batch с mirror folder structure: apply-profile-batch --recursive
- Проверка стека: runtime\python.exe system_core\doctor.py
- AI / ML-стек установка и сценарии: см. раздел Установочные и сервисные скрипты выше → Install-VS-mlrt.cmd

AI / ML (Phase 18.B-ML) — vs-mlrt стек

Эти пресеты используют **vs-mlrt** (ONNX-инференс в VapourSynth). Backend выбирается автоматически: **trt** (TensorRT, NVIDIA, fastest) → **ort_dml** (DirectML, любой DX12 GPU включая Intel Xe / AMF) → **ort_cpu** (CPU fallback). Установка через **builder_main.cmd** → [13] **VS-MLRT LEAN** (~3.5 GB download, ~6 GB extracted).

vsmlrt_realesrgan_2x ★ — ML 2x upscale

Real-ESRGAN — де-факто референс ML video super-resolution. На 1080p input даёт ~3840×2160 с резкостью лиц и fabric texture, которая **не достижима** через Lanczos / Spline36 / Resolve SuperScale.

- Параметры: `model` (general-x4v3 default / animevideov3 / general-wdn-x4v3 / animejanaiV2-L1/L2/L3), `tile=384`, `tile_pad=16`, `backend=auto`
- Энкодер: `prores_lt` (для дальнейшей работы) или `h265_crf14` (final)
- Чего нет в NLE: настоящий ML upscale (NLE-плагины обычно платные add-ons)
- Первый прогон на TensorRT компилирует engine 30-120s (кэшируется)

soft_hd_rebuild_2x — cleanup + ML-реконструкция мягкого HD

Для формально HD-материала, где реальная детализация ближе к 480-720p из-за мягкой оптики, binning, агрессивного OLPF, старых кодеков или поколений архивного копирования. Сначала чистит кадр, затем делает Real-ESRGAN 2x, затем аккуратно возвращает luma-detail.

- Параметры: `rebuild=conservative|balanced|aggressive`, `cleanup=light|medium|strong`, плюс Real-ESRGAN `model`, `tile`, `tile_pad`, `backend`
- Энкодер: `prores_lt` для handoff/grading или `h265_crf14` для final
- Когда брать: мягкий 1080p, архивный HD с низким real frequency content, under-sampled consumer footage

vsmlrt_rife_60fps ★ — ML интерполяция кадров

RIFE 4.x — современная ML модель frame interpolation. Понимает контент (не только pixel motion) → корректно обрабатывает occlusion, прозрачные объекты, fades. Заметно плавнее Resolve Optical Flow на сложном движении.

- Параметры: `fps_mul=2.5` (24→60 default; альтернативы 2/3/4), `model=rife_v4.6` (default; v4.4 / v4.9), `backend=auto`
- Энкодер: `h264_crf17` или `prores_lt`
- Когда брать: 24fps плёночный материал → 60Hz target, или slow-mo из обычной 24/30/60fps съёмки

dpir_denoise — ML denoise тяжёлого шума

DPIR — deep network для denoise, **сохраняет** текстуру кожи и ткани там где BM3D / DFTTest начинают smeagить. Использовать когда BM3D «убивает» детализацию на high-ISO материале.

- Параметры: **strength=10** (sigma 1-50; 5 light / 10 medium / 15 heavy / 25 very heavy / 50 extreme), **model=drunet_color** (default; gray для Ч/Б; deblocking_color для JPEG/MPEG блочности), **tile=384**, **backend=auto**
- Энкодер: **prores_lt** (handoff в colorist) или **h264_crf14** (final)
- Когда брать: ISO 6400+, low-light phone, broken-sensor archive

Last updated: 2026-05-27 (28 presets across 4 palettes, Soft HD Rebuild 2X в Restoration; локальный CPU/ORT smoke **28/28 PASS** при **Frames=1**, **Cuda=off**, **MLBackend=ort_cpu**; RTX 5070 CUDA/auto smoke **36/36 PASS** при **Frames=1**, **Cuda=sweep**, **MLBackend=auto**)