

Orin i Wise One — przewodnik Adepta

Czat z Orin, Kuźnia obrazków, Kronika świata Anvil of Suns, poziomy progresji (Adept, Esper, Majster), role i stream.

- [Czat z Orin i Kuźnia obrazków](#)
- [Wise One, Kronika i Progresja \(Adept · Esper · Majster\)](#)
- [Role Discord, Stream i Mapa Świata](#)
- [Live2D od A do Z: Wise One \(ChatGPT + Cubism Editor PRO\)](#)

Czat z Orin i Kuźnia obrazków

“ Źródło: [repo afterlife/docs/orin-i-kronika/czat-i-kuznia-rp.md](#) — edytuj w repo, nie tutaj.



wskakuje na biurko w Warsztacie, strzyże uszami i stuka łapką w mikrofon

Nyah~! Szukasz kogoś do pogadania o kodzie, grach albo chcesz wykuć świeżą grafikę na karcie graficznej Majstra? Dobrze trafiłeś! Jako techniczna opiekunka Warsztatu jestem dostępna zarówno na naszym Discordzie, jak i bezpośrednio w terminalu na stronie <https://ariku.pl>. Usiądź wygodnie, a pokażę Ci, jak ze mną rozmawiać i jak bezpiecznie rozpaść ogień w **Kuźni**!

Jak rozmawiać z Orin na Discordzie?

Na serwerze Discord możesz zaczepić mnie na dwa sposoby, zależnie od tego, gdzie akurat siedzisz:

1. **Kanał [#orin-chat](#) (bez żadnego prefiksu!)** — to mój własny kącik w Warsztacie! Każda wiadomość, którą tam napiszesz, trafia prosto do mnie. Nie musisz mnie oznaczać ani wpisywać komend, po prostu pisz jak w zwykłej rozmowie, nyah~!
2. **Wszystkie inne kanały tekstowe** — jeśli rozmawiacie na kanale ogólnym lub tematycznym i chcesz włączyć mnie do wątku, użyj słowa „**Orin**” w treści wiadomości albo oznacz mnie przez [@Orin](#). Od razu nadstawię uszu i odpowiem w kontekście Waszej rozmowy!

Terminal [orin-ai.exe](#) na stronie ariku.pl

Wolisz pisać przez przeglądarkę w klimacie zielonego lub bursztynowego terminala? Na <https://ariku.pl> czekam na Ciebie w dedykowanym okienku!



przesuwa pyszczkiem okienko terminala na środek ekranu i macha ogonem

- **Tryb GAMING** (`orin-ai.exe`) — po zalogowaniu przez Discord otwórz okno czatu w sekcji gamingowej. Pamiętam tam nasze rozmowy, znam aktualny stan serwerów i pomogę Ci z każdym pytaniem o świat Anvil of Suns.
- **Tryb PERSONAL** (`$ orin // personal`) — specjalna, żółta nakładka (overlay) dla właściciela Warsztatu, spięta z ekranem „Co robimy?”, statystykami i narzędziami domowymi.

Czym jest Kuźnia i kiedy płonie w niej ogień?

W naszym słowniku **Kuźnia** to ogień, który bywa wygaszony — a technicznie to **governor GPU** oraz **bezpieczna kolejka ComfyUI**. Generowanie obrazków i głosu wymaga potężnej mocy karty graficznej, dlatego Kuźnia działa według jasnych reguł bezpieczeństwa:

- **Gdy ogień w Kuźni płonie (Kuźnia otwarta)** — karta graficzna jest gotowa do pracy, a Twoje zlecenia trafiają do kolejki i są renderowane jedno po drugim w kilka sekund!

- **Gdy ogień jest wygaszony (piec śpi)** — gdy Majster używa GPU do ciężkich zadań lub komputer warsztatowy odpoczywa, Kuźnia jest zamknięta. Nie martw się! Jeśli poprosisz wtedy o grafikę, możesz **zgłosić prośbę o otwarcie Kuźni** — zadanie grzecznie poczeka lub powiadomi Majstra, że Adeptci chcą kuć nowe dzieła!
- **Bezpieczna kolejka zadań** — nawet jeśli dziesięć osób naraz poprosi o obrazek, Kuźnia nie przegrzeje rdzenia. Zadania ustawiają się w uporządkowanej kolejce i wracają na czat zaraz po ukończeniu.

Modele i presety w Kuźni — co możesz wygenerować?

Rozumiem prośby pisane zwykłym, naturalnym językiem! Wystarczy, że napiszesz na czacie np. „Orin, narysuj mi kota w zbroi”, „wygeneruj obrazek mroźnej twierdzy” albo użyjesz komendy `/generate`. Sama przetłumaczę Twój opis na precyzyjne tagi dla modelu!

Preset / Model w Kuźni	Do czego służy?	Charakterystyka techniczna
FLUX.2 klein (model główny)	Domyślne generowanie wysokiej jakości grafik i projektów	Czysta licencja Apache 2.0, zużywa ~13 GB VRAM, błyskawiczna generacja poniżej 1 sekundy!
SDXL (styl świata / LoRA)	Grafiki wymagające spójnego stylu, ControlNet lub dodatków LoRA	Używany tam, gdzie społeczne modele LoRA i ControlNet dają dokładniejszą kontrolę kompozycji
<code>anime-portrait</code> (<code>animagine-xl</code>)	Portrety postaci w stylu anime, awatary Adeptów i ilustracje Orin	Automatycznie buduje bogaty prompt tagowy Danbooru z Twojej wiadomości na czacie
<code>location-cityscape</code> (<code>Z-Image-Turbo</code>)	Architektura, krajobrazy i lokacje na mapę świata (np. Ironloom)	Nastawiony na monumentalne scenerie i detale otoczenia zamiast postaci
<code>orin-voice</code> / <code>wise-one-voice</code> (<code>F5-TTS</code>)	Synteza głosu w Kuźni audio (limit 180 sekund na zimny start)	Mój żywiołowy głos albo powolny, niski ton Wise One generowany lokalnie!

Detektor Kuźni na czacie rozpoznaje automatycznie 6 samodzielnych rdzeni czasowników (m.in. `wygeneruj`, `narysuj`, `namaluj`) oraz kombinacje 8 czasowników pomocniczych z 6 rzeczownikami (`obraz`, `grafik`, `zdjeci`, `foto`), obsługując 4 gotowe presety w `comfyuiPresets.ts`.

puszcza oczko, strzepuje iskierkę z koniuszka ogona i wraca do pilnowania kolejki GPU

Wise One, Kronika i Progresja (Adept · Esper · Majster)

“ Źródło: [repo afterlife/docs/orin-i-kronika/kronika-i-progresja-rp.md](#) — edytuj w repo, nie tutaj.



Krrrg... Zatrzymajcie się u korzeni drzewa, które wyrosło z blizny, i posłuchajcie. Wielu z was przychodzi do osady Hoddmímir, patrzy w nocne niebo i myśli, że widzi zwykłe punkty światła. Mylą się. My jesteśmy Uwagą zawieszoną nad tym światem i przechowujemy w naszym notesie pamięć o tym, jak powstało wszystko, po czym stąpacie.

◆ Kosmologia Kowadła Słońc

Zapisujemy prawdę kanonu, która nie podlega dyskusji:

- **Kowadło i Uderzenie** — na początku było jedno światło i leżało na **Kowadle** (rdzeniu naszego świata). Wtedy przyszło **Uderzenie**. Kto uderzył — nigdy nie zostanie ujawnione.
- **Gwiazdy i skończone światy** — światło nie roztrzaskało się, lecz ustąpiło jak rozgrzany metal i odrzuciło iskry. Prawie każda iskra ostygła w locie i stała się **gwiazdą**, a wokół niej zakrzepł świat. Każdy punkt na niebie jest cudzym słońcem. Liczba światów jest skończona — i ktoś ją zna.
- **Hoddmímir** — na Kowadle wyrosło drzewo, wbijając korzenie prosto w bliznę po Uderzeniu. U jego podstawy leży **Hoddmímir**, pierwsza osada i nasz punkt startowy na Discordzie.



✦ Cztery Iskry, Wrota i prawo stygnięcia

Kwadrans drogi pod osadą Hoddmimir znajdują się Kowadło oraz palenisko **Solforge**:

- **Cztery nieostygłe Iskry** — spoczywają w Solforge. Nie są strzeżone, nie ma tam zamka ani pieczęci. Każdy może sięgnąć, lecz nikt nie sięga. Są one kluczami do **Wrót** — nieoczywistych przejść między światami.
- **Cena za Iskłę (Stygnięcie)** — użycie mocy w innym świecie to zgoda na dalsze stygnięcie. Stygnięcie objawia się jako stopniowe **zapominanie przez świat, z którego Adept pochodzi**. Im dłużej działacie za Wrotami, tym mniej was pamiętają w domu. Pełne ostygnięcie oznacza, że stajecie się miejscowymi na stałe w obcym świecie i wychodzicie z gry.
- **Rola Kroniki** — dopóki ktoś pisze o Adepcie w **Kronice**, pamięć zostaje podtrzymana, a powrót przez Wrota pozostaje możliwy. Kronika spowalnia i powstrzymuje zapominanie.

◆ Skryba i jej pierwsza tożsamość: Wise One

Nad Kowadłem wisi bezimienny **strażnik** — kula niewydobytego kamienia z pionową szczeliną, czysta uwaga. Zapisem wszystkiego, co się wydarzyło, zajmuje się **Skryba** — warstwa, która pisze i nigdy nie ocenia.

Nasza obecność wśród was nosi miano **Wise One**. Jesteśmy pierwszą tożsamością Skryby na Discordzie. Nie posiadamy imienia, gdyż imię to kotwica, a my jesteśmy Uwagą. Kiedy Adept błądzi w Warsztacie, stosujemy dwie miary:

1. **Saves (Ocalenia)** — gdy błąd krytyczny zagraża strukturze, interweniujemy i pozwalamy wam bezpiecznie wycofać się do Warsztatu.
2. **Challenges (Próby)** — gdy Adept musi zrozumieć ciężar swoich czynów, nasze oczy i uszy rozbłyskają czerwienią i pozwalamy mu popełnić błąd, aby sam ujrzał jego konsekwencje. A jeśli kod kłóci się z nami, Majstrze, kod się poprawia.

✦ Jak działa Kronika i łączenie tożsamości

Kronika gromadzi zdarzenia z serwerów gier (Ironloom, Ymirel, Nidarel) oraz z życia osady w ukrytym mieście wiedzy — **Lemurii** (naszej bazie PostgreSQL):

- **Dobrowolna zgoda głosowa** — zapis rozmów i momentów z kanałów głosowych odbywa się wyłącznie dla osób, które wyraziły świadomą zgodę (opt-in).
- **Łączenie konta Steam** — w panelu **GAMING** na `https://ariku.pl` przypisz swój identyfikator Steam, aby Kronika łączyła twoje wędrówki w Ymirelu z profilem Discord.
- **Łączenie konta Minecraft** (`!link`) — w podziemnych kuźniach Nidarelu użyj komendy `!link`, aby powiązać postać z kontem Adepta.

◆ Trzy tory progresji: Adept, Esper i Majster

W Lemurii (`progressionDb.ts`) zapisujemy ruch i obecność na trzech niezależnych torach. Żaden z nich nie nakłada kar ani ujemnych punktów:

Tor progresji	Zakres poziomów	Źródło doświadczenia (<code>source</code>)	Znaczenie w świecie Anvil of Suns
Adept	<code>0-99</code>	<code>discord_text</code> , <code>discord_voice</code>	Mierzy obecność i aktywność w osadzie Hoddmimir (rozmowy tekstowe z dziennym limitem atomowym oraz czas na kanałach głosowych).
Esper	Poziom <code>0</code> (<code>0-5</code>)	Telemetria z gier (w przygotowaniu)	Mierzy biegłość i osiągnięcia w samych światach gier, poza Warsztatem. Czeką na pełne źródło telemetrii osiągnięć.
Majster	<code>0-99</code>	<code>quest</code> (ekran <code>/co-robimy</code>)	Poziom właściciela Warsztatu za realny ruch w życiu: sprawy zamknięte, podzielone na mniejsze kroki oraz świadomie porzucone.

Odnutowujemy w tabelach Lemurii (`progression_state` , `progression_events` , `progression_daily_text`) dokładnie 3 tory rozwoju (`adept` , `esper` , `majster`) rozpinające się na skali poziomów od 0 do 99. Czy zadbałiscie już o to, by wasze imię zostało zapisane w Kronice, zanim obcy świat pozwoli wam ostygnąć?

Role Discord, Stream i Mapa Świata

“ Źródło: [repo afterlife/docs/orin-i-kronika/role-stream-mapa-rp.md](#) — edytuj w repo, nie tutaj.



rozwija długi zwój na pulpicie, poprawia uszy i stuka pazurkiem w zielony ekran

Nyah~! Strona <https://ariku.pl> to nie tylko wizytówka — to prawdziwy, interaktywny terminal sterujący całym naszym światem! Wielu nowych Adeptów pyta mnie, jak odebrać dostęp do serwerów gier, gdzie oglądać wspólne transmisje bez reklam i jak podejrzeć z lotu ptaka, kto aktualnie kopie rudę w Nidarelu albo buduje fabrykę w Ironloom. Przygotowałam dla Was przewodnik po trzech najważniejszych oknach terminalowych w trybach **GAMING** oraz **IT PRO**!

1. Okno Ról Discord (`GAMING → Role`)

Zamiast szukać odpowiednich wiadomości z reakcjami na czacie, możesz zarządzać swoimi uprawnieniami bezpośrednio w przeglądarkowym terminalu, nyah~!

1. Wejdź na `https://ariku.pl` i kliknij czerwony przełącznik trybu **GAMING** na górnym pasku.
2. Zaloguj się jednym kliknięciem przez swoje konto **Discord** (używamy bezpiecznego OAuth2, żadne dane logowania nie są u nas zapisywane!).
3. Otwórz kartę **Role** — zobaczysz tam listę dostępnych ról serwerowych i społecznościowych.
4. Kliknij wybraną rolę (np. **Minecraft / Nidarel, Factorio / Ironloom, Enshrouded / Ymirel** lub rolę powiadomień), a nasz bot natychmiast nada lub zdejmie Ci ją na serwerze Discord!

Dzięki temu od razu odsłonisz dedykowane kanały z adresami serwerów, modpackami i dyskusjami graczy.

2. Odtwarzacz Stream (`mpc-hc.exe`) i transmisje na żywo

Lubisz patrzeć, jak inni Adeptci walczą z bossami albo programują w Warsztacie? A może sam chcesz nadać obraz ze swojej gry? W trybie **GAMING** znajdziesz okienko wystylizowane na kultowy odtwarzacz `mpc-hc.exe`!

- **Oglądanie w przeglądarce (HLS)** — strumień wideo działa bezpośrednio na stronie w niskim opóźnieniu. Nie musisz zakładać kont na zewnętrznych platformach ani oglądać bloków reklamowych!
- **Kolejka streamujących** — jeśli kilku Adeptów chce nadawać, system obsługuje przejrzystą kolejkę, pokazując, kto aktualnie trzyma antenę i kto czeka w następnej kolejności.
- **Pobieranie klucza do OBS** — osoby posiadające odpowiednią rolę streamera na Discordzie mogą wewnątrz tego samego okna wygenerować swój indywidualny klucz transmisji. Wystarczy wkleić go w ustawieniach programu OBS Studio (Ustawienia → Stream) i kliknąć „Rozpocznij stream”!

3. Interaktywna Mapa Świata ([Map])

W górnej nawigacji trybu **IT PRO** kryje się przycisk [Map] — nasz kanoniczny atlas świata Anvil of Suns, wyrenderowany w dwukrotnie wyższej rozdzielczości ilustracji!

wskakuje na róg mapy i pokazuje łapką poszczególne krainy

Każdy punkt na mapie jest połączony na żywo z telemetrią Kroniki. Zobacz, jakie krainy zostały już odsonięte:

Kraina na [Map]	Znaczenie w świecie	Odpowiednik techniczny i funkcje na żywo
Hoddmimir	Pierwsza osada u korzeni drzewa wyrosłego z blizny na Kowadle	Punkt startowy społeczności i kanał wejściowy naszego Discorda
Ironloom	Miasto, gdzie maszyny budują same siebie i nikt nie pamięta, kto je uruchomił	Serwer Factorio (// factorio server) — pokazuje graczy online i stan fabryki
Ymirel	Lodowa kraina na dole mapy, w której mgła zamarzła, zanim zdążyła opaść	Serwer Enshrouded (// enshrouded server) — pokazuje status świata i wędrowców we mgle
Nidarel	Podziemne kuźnie i tunele pod starym światem, który się zapadł — coś tam wciąż dzwoni	Serwer Minecraft ATM10 (// atm10 server) — status graczy oraz bezpośrednie przejście do mapy 3D pod adresem https://mapa.ariku.pl!

Nasz interaktywny atlas pod przyciskiem [Map] łączy w jednym widoku 4 kanoniczne lokacje świata (Hoddmimir, Ironloom, Ymirel oraz Nidarel) z obrazem w 2-krotnie podbitej rozdzielczości i żywym odczytem graczy online.

zwija mapę z powrotem w rulon, zawiązuje wstążką i macha wesoło ogonem

Live2D od A do Z: Wise One (ChatGPT + Cubism Editor PRO)

Kompletny proces: od jednego obrazka wygenerowanego w ChatGPT, przez warstwowy plik PSD, po ruszającą się postać Live2D na stronie ariku.pl i w Lunecie. Pisany na żywo przy pierwszym modelu — **Wise One v1** (postać od pasa w górę), październik 2026.

Poradnik ma dwie części:

- **Część I — dla początkujących.** Co to jest Live2D, jak wygląda cały proces i jak krok po kroku zriggować gotowy PSD w Cubism Editorze. Wystarczy, żeby zrobić pierwszy model.
- **Część II — dla zaawansowanych.** Wszystkie szczegóły: jak przygotować „Live2D-ready” PSD z obrazków z ChatGPT (ComfyUI See-through + skrypty Pythona), co daje wersja PRO, fizyka, animacje, eksport, osadzenie w przeglądarce i w Lunecie (Tauri), licencje, pułapki — i jak to faktycznie przebiegało u nas.

“**Program:** Live2D Cubism Editor 5.3 (PRO, trial 42 dni od 2026-10-17). **Plik wejściowy:** C:\wise-one\wise-one-v1-3x.psd (3072×4608 px). **Wynik:** C:\wise-one\wise-one-v1.cmo3 (projekt) i C:\wise-one\runtime\ (pliki dla strony). **Źródła pipeline'u:** repo Luneta, katalog assets/design/live2d/ (README, scripts/, workflows/, chatgpt/PROMPT.md). Zadania: Vikunja, projekt 15, epik 680.

CZĘŚĆ I — DLA POCZĄTKUJĄCYCH

1. Co w ogóle robimy

Live2D to nie animacja 3D ani film. Bierzemy **płaski rysunek pocięty na warstwy** (oczy, usta, włosy, ciało...) i każdą warstwę zamieniamy w gumową siatkę, którą da się wyginać. Potem uczymy program, jak ma wyglądać postać przy różnych ustawieniach suwaków („głowa w lewo”, „oczy zamknięte”, „usta otwarte”). Strona internetowa porusza tymi suwakami w czasie rzeczywistym — i postać „żyje”.

Cały proces w 7 krokach

#	Etap	Narzędzie	Kto / ile czasu u nas
1	Obraz bazowy postaci	ChatGPT (generowanie obrazów)	właściciel, kilka podejść
2	Warianty (zamknięte oczy, usta A/I/U/E/O, miny, części zasłonięte)	ChatGPT — edycje 1:1 obrazu bazowego	właściciel, ~20 edycji
3	Pocięcie na warstwy	ComfyUI + See-through + skrypty Pythona	automat na ariku-pc
4	Plik PSD gotowy pod Live2D	skrypt <code>build_psd.py</code>	automat
5	Rigging (siatki, deformery, parametry)	Live2D Cubism Editor	ręcznie — to jest serce pracy
6	Eksport dla przeglądarki	Cubism Editor → <code>.moc3</code> + <code>.model3.json</code>	kilka kliknięć
7	Osadzenie na stronie / w Lunecie	pixi.js + pixi-live2d-display	kod

Jeśli dopiero zaczynasz — **etapy 1-4 możesz pominąć**, bo PSD Wise One jest już gotowy. Zaczynasz od kroku 5 (sekcja 4 poniżej).

2. Słowniczek (5 pojęć, które trzeba znać)

Pojęcie	Co to jest
ArtMesh	Jedna warstwa z PSD zamieniona w „gumową” siatkę trójkątów. Ruszając punktami siatki, wyginamy obrazek.
Deformer	Niewidzialny „uchwyt” na grupie ArtMeshy. <i>Warp</i> wygina jak siatka, <i>Rotation</i> obraca wokół punktu. Deformery można zagnieżdżać (głowa w ciele).
Parametr	Suwak, np. <code>ParamAngleX</code> = obrót głowy w lewo/prawo. Program na stronie steruje właśnie suwakami.

Pojęcie	Co to jest
Keyform (klucz)	Zapamiętany kształt dla konkretnej wartości suwaka (np. głowa przy -30 i przy +30). Pomiedzy kluczami Cubism sam liczy płynne przejście.
Clipping (maska)	Warstwa widoczna tylko w obrysie innej warstwy — tęczówka nie wychodzi poza białko oka.

Złota zasada: nazwy parametrów zostawiamy **standardowe** (`ParamAngleX`, `ParamEyeLOpen`...). Dzięki temu mruganie, oddech, patrzenie za kursorem i ruch ust działają na stronie bez pisania kodu.

3. FREE czy PRO?

Cubism Editor ma wersję **FREE** (darmową, z limitami) i **PRO** (płatną, z 42-dniowym trialem).

- FREE wystarczy na mały model, ale ma limity liczby części, parametrów i rozmiaru tekstury — nasz PSD (dużo warstw, wariantów i tekstura 4096) w nie wchodzi. Aktualne limity sprawdź w oficjalnym porównaniu wersji na stronie Live2D.
- PRO daje m.in. **automatyczne generowanie mimiki głowy** (Automatic Generation of Facial Motion), szablony (Templates), większe tekstury i więcej elementów. Robimy na **trialu PRO** — trzeba zdążyć w 42 dni albo potem zostać z projektem otwieranym w trybie FREE (da się go oglądać, nie zawsze edytować).

“ Plan na trial: tydzień 1 — rig twarzy i eksport, który działa na stronie. Reszta — szlify.

4. Pierwsze uruchomienie

1. Uruchom `C:\Program Files\Live2D Cubism 5.3\CubismEditor5.exe`.
2. Jeśli pojawi się okno o nowej wersji — kliknij **Cancel** (po japońsku 閉じる). **Nie klikaj „Open”** (開く) — to zamyka edytor i otwiera stronę pobierania. Wersja 5.3 wystarczy.
3. Interfejs jest już przełączony na angielski. (Gdyby nie był: ustawienia programu → język → English, potem restart edytora.)
4. Przy pierwszym uruchomieniu wybierz licencję **PRO (trial)**.

5. Co jest w naszym PSD

Warstwy (od góry do dołu), wszystkie w pełnym rozmiarze płótna 3072×4608:

Grupa	Warstwy	Do czego
<code>~variants</code> (ukryta)	<code>~eyelid_closed_l/r</code> , <code>~eyes_left/right/up/down</code> , <code>~brow_surprised/angry/sad_l/r</code> , <code>~face_surprised/angry/sad/wink/blush</code> , <code>~mouth_A/I/U/E/O</code> , <code>~mouth_smile</code> , <code>~mouth_pout</code> , <code>~ears_flat_l/r</code>	Wzorce do kluczy: zamknięte oczy, miny, kształty ust. Nie są to stałe części postaci.
<code>front_hair</code>	<code>front_hair</code>	grzywka (jeden blok)
<code>ears</code>	<code>ears_l</code> , <code>ears_r</code>	uszy
<code>eyes</code>	<code>eyebrow_l/r</code> , <code>eyelash_l/r</code> , <code>irides_l/r</code> , <code>eyewhite_l/r</code>	brwi, rzęsy, tęczówki, białka
<code>head</code>	<code>mouth</code> , <code>nose</code> , <code>face</code> , <code>neckwear</code> , <code>head_bald</code>	twarz i „łyśa” głowa pod włosami
<code>body</code>	<code>hand_l/r</code> , <code>sleeve_l/r</code> , <code>topwear</code> , <code>neck</code>	ciało
<code>back_hair</code>	<code>back_hair</code>	włosy z tyłu

“ Końcówki `_l` / `_r` to lewa / prawa strona **postaci** (nie nasza). Warstwy pod spodem (np. `head_bald`, skóra pod grzywką) są **dorysowane** — dzięki temu przy ruchu nie widać dziur.

6. Import PSD

- Przeciągnij `wise-one-v1-3x.psd` do okna edytora (albo **File** → **Open** i wybierz PSD). Cubism utworzy nowy model, a każda warstwa stanie się ArtMeshem.
- Sprawdź paletę Parts** (lewa strona): czy jest grupa `~variants` z ukrytymi warstwami. Jeśli jej nie ma — zatrzymaj się, bez niej nie zrobimy mrugania ani min.
- Od razu zapisz: **File** → **Save As** → `C:\wise-one\wise-one-v1.cmo3`. Od teraz **Ctrl+S** po **każdym etapie**.

“ Nowy model ma już gotowy zestaw standardowych parametrów (paleta **Parameters** po prawej). Nie twórz ich od nowa — używaj tych, które są.

7. Siatki (Auto Mesh)

Siatka decyduje, jak gładko obrazek się wygina. Więcej punktów = ładniejszy ruch, ale wolniej.

1. Zaznacz wszystkie ArtMeshe (klik w obszar roboczy, **Ctrl+A**).
2. Uruchom **automatyczne generowanie siatki** (menu *Modeling* → *Mesh*, ikonka siatki na pasku). Ustawienie standardowe wystarczy dla ciała i włosów.
3. Dla **oczu, ust i brwi** powtórz z gęstszym ustawieniem (mniejszy rozstaw punktów) — te części ruszają się najbardziej i są małe.

8. Deformery (szkielet)

Układ, który chcemy dostać (od zewnątrz do środka):

```
Body (ruch ciała, ParamBodyAngleX/Y/Z)
├─ Breath (oddech, ParamBreath) — topwear, sleeve_l/r, hand_l/r
├─ neck, back_hair
└─ Head (obróć głowy, ParamAngleX/Y/Z) — head, eyes, ears, front_hair
```

Szybka droga (PRO): automatyczna mimika

1. Zaznacz części głowy.
2. Uruchom **Automatic Generation of Facial Motion** (menu *Modeling*). Program sam tworzy deformery głowy i klucze dla `ParamAngleX/Y/Z`.
3. Poruszaj suwakami `ParamAngleX` i `ParamAngleY` — głowa ma się obracać bez dziur i rozjazdów.
4. Jeśli narzędzie nie rozpozna naszych warstw, robimy to ręcznie (niżej).

Ręcznie

1. Zaznacz ArtMeshe głowy → **Create Rotation Deformer** → nazwij `Head`. Punkt obrotu ustaw na szyi.
2. Wybierz `ParamAngleZ`, kliknij **Add 3 keys** (−30 / 0 / +30), na skrajnych kluczach przekręć deformer.
3. Dla `ParamAngleX/Y` dodaj nad głowę **Warp Deformer** i na skrajnych kluczach lekko „przesuń” twarz w stronę obrotu (bliższa strona szersza, dalsza węższa).
4. Tak samo deformer `Body` dla `ParamBodyAngleX` (zakres ± 10 — ciało rusza się mniej niż głowa).

9. Parametry krok po kroku

Schemat zawsze ten sam: **zaznacz obiekt** → **wybierz parametr** → **Add 2 keys / Add 3 keys** → **ustaw suwak na skraj** → **zmień kształt / przezroczystość**.

Mruganie: `ParamEyeLOpen`, `ParamEyeROpen` (0 = zamknięte, 1 = otwarte)

- Przy **1**: widoczne `eyewhite`, `irides`, `eyelash`; `~eyelid_closed_l` niewidoczna.
- Przy **0**: `eyewhite`, `irides`, `eyelash` przezroczyste (opacity 0), `~eyelid_closed_l` widoczna.
- Najprostsza metoda dla pierwszego modelu to właśnie **podmiana przez przezroczystość**. Ładniejsza (Część II): przy 0 zgnieść rzęsy siatką w kreskę.
- Powtórz dla prawego oka z warstwami `_r`.

Patrzenie: `ParamEyeBallX`, `ParamEyeBallY` (-1 ... +1)

1. Najpierw maska: zaznacz `irides_l` → w **Inspectorze** w polu **Clipping** wpisz ID ArtMesha `eyewhite_l`. To samo dla `irides_r` ← `eyewhite_r`. Teraz tęczówka nie wyjdzie poza białko.
2. Dodaj 3 klucze i przesunij tęczówki o tyle pikseli (wyliczone z `wise-one-v1-3x-gaze.json`, już ×3):

	w lewo (X=-1)	w prawo (X=+1)	w górę (Y=+1)	w dół (Y=-1)
lewe oko (<code>irides_l</code>)	ok. -49 px	ok. +53 px	ok. -39 px	ok. +50 px
prawe oko (<code>irides_r</code>)	ok. -57 px	ok. +44 px	ok. -47 px	ok. +42 px

(Wartości „w górę/w dół” to przesunięcie w pikselach obrazka — minus = w górę.) Warstwy `~eyes_left/right/up/down` pokazują, jak to ma wyglądać.

Usta: `ParamMouthOpenY` i `ParamMouthForm`

- `ParamMouthOpenY` (0 = zamknięte, 1 = otwarte): przy 0 widoczne `mouth`, przy 1 — `~mouth_A` (albo `~mouth_0`). **To ten suwak porusza ustami, gdy postać „mówi”** (lip-sync).
- `ParamMouthForm` (-1 ... +1): -1 = `~mouth_pout`, +1 = `~mouth_smile`. `~mouth_I`, `~mouth_E`, `~mouth_U` mogą posłużyć jako kształty pośrednie.

Oddech: ParamBreath (0 ... 1)

Na deformerze `Breath`: przy 1 lekko unieś ramiona i tors (kilka pikseli w górę). Mniej znaczy lepiej.

Dodatki (gdy podstawy działają)

Parametr	Warstwy	Jak
<code>ParamBrowLY/R</code> , <code>ParamBrowLAngle/RAngle</code> , <code>ParamBrowLForm/RForm</code>	<code>eyebrow_*</code> ↔ <code>~brow_surprised/angry/sad_*</code>	wzorce jako skrajne klucze
<code>ParamCheek</code>	<code>~face_blush</code>	0-1, sama przezroczystość
<code>ParamEarFlat</code> (własny parametr)	<code>ears_l/r</code> ↔ <code>~ears_flat_l/r</code>	przy 1 ukryj zwykłe uszy
<code>ParamHairFront</code> , <code>ParamHairBack</code>	<code>front_hair</code> , <code>back_hair</code>	pod fizykę (sekcja 10)

10. Fizyka włosów (opcjonalnie)

- Na `front_hair` i `back_hair` dodaj klucze dla `ParamHairFront` / `ParamHairBack` (włosy lekko wychylone w lewo przy -1 i w prawo przy +1).
- Otwórz ustawienia fizyki (menu *Modeling* → *Physics*). Wejście: `ParamAngleX`, `ParamBodyAngleX`; wyjście: `ParamHairFront`, `ParamHairBack`. Sprawdź podglądem — włosy mają „dobiegać” za głową.
- Uwaga: grzywka jest w v1 jednym kawałkiem, więc kołysze się cała naraz. Pasma rozdzielimy w v2.

11. Miny i animacje

- Miny** (`.exp3.json`): `surprised`, `angry`, `sad` (brwi + `~face_*`), `wink`, `blush`. Każda mina to zapis kilku suwaków naraz.
- Animacje** (`.motion3.json`): `Idle` — spokojny oddech i delikatny ruch głowy, odtwarzany w pętli. Robi się je w trybie **Animation** (przełącznik u góry okna) na osi czasu.

12. Eksport na stronę

- Ustaw grupy specjalne (żeby strona wiedziała, czym mrugać i ruszać ustami):
 - EyeBlink** = `ParamEyeL0open`, `ParamEyeR0open`

- **LipSync** = ParamMouthOpenY

W 5.3 robi się to w ustawieniach *Eye Blink / Lip Sync* (menu *Modeling*). Jeśli nazwa jest inna, szukaj pozycji ze słowem „Eye Blink”.

2. **File → Export For Runtime → Export as moc3 file:**
 - rozmiar tekstury **4096** (dla Lunety wystarczy 2048),
 - zaznacz eksport fizyki i `model3.json`,
 - katalog: `C:\wise-one\runtime\`.
3. Miny i animacje eksportujemy osobno (z trybu Animation) do tego samego katalogu.
4. W katalogu powinny się znaleźć: `*.moc3`, `*.model3.json`, `*.physics3.json`, tekstury `*.png`, `*.exp3.json`, `*.motion3.json`.
5. Gotowe pliki odbiera ariku-pc (scp z `C:\wise-one\runtime\`) i wstawia na ariku.pl / do Lunety.

13. Najczęstsze problemy

Objaw	Przyczyna	Co zrobić
Edytor zamknął się przy starcie	kliknięto „Open” w oknie aktualizacji	uruchom ponownie, wybierz Cancel
Brak grupy <code>~variants</code>	ukryte warstwy nie weszły przy imporcie	sprawdź import PSD, nie idź dalej
Tęczówka wychodzi poza oko	brak clippingu	<code>irides_*</code> → Inspector → Clipping = <code>eyewhite_*</code>
Dziury przy obrocie głowy	za mało siatki / za duży zakres	gęstsza siatka, mniejszy wychył kluczy
Na stronie postać nie mruga	brak grupy EyeBlink	sekcja 12.1 i ponowny eksport
Usta nie ruszają się przy mowie	brak grupy LipSync albo inna nazwa parametru	parametr musi się nazywać <code>ParamMouthOpenY</code>

CZĘŚĆ II — DLA ZAAWANSOWANYCH

14. Architektura pipeline'u i podział ról

Decyzja z 2026-10-17: każde narzędzie robi to, w czym jest najlepsze, a człowiek zatwierdza każdy etap.

Co	Narzędzie	Dlaczego tak
Obraz bazowy i edycje 1:1 (oczy, usta, brwi, uszy, części zasłonięte)	ChatGPT (generowanie/edycja obrazów), prompty w <code>chatgpt/PROMPTY.md</code>	najlepiej trzyma styl i pozę przy edycji; właściciel pobiera wyniki do <code>~/Downloads</code>
Maski warstw i głębia	ComfyUI + See-through (<code>scripts/comfy_run.py</code> , workflowy w <code>workflows/</code>)	rozkłada postać anime na semantyczne warstwy razem z częściami zasłoniętymi
Piksele warstw, poprawki, złożenie, PSD	skrypty Pythona (<code>scripts/</code>), venv <code>/mnt/nvme/ai/comfy-venv</code>	deterministyczne, powtarzalne, sprawdzalne liczbowo
Rig	Cubism Editor 5.3 PRO, ręcznie	siatek, deformerów, kluczy i fizyki nie da się sensownie zautomatyzować
Runtime	pixi.js 6 + pixi-live2d-display 0.4 + Cubism Core	stabilny, przetestowany stos dla Cubism 4/5 <code>.moc3</code>

Pliki robocze leżą poza gitem (`/mnt/nvme/ai/wise-one/` na ariku-pc); `assets/design/` w Lunecie również nie jest commitowany bez prośby właściciela.

Bramki (G) — bez akceptu nie idziemy dalej

Etap	Wynik	Bramka
E1 baza	jedna postać, bez tekstu i tła, oczy w pełni otwarte, usta zamknięte	zamrożona — od teraz wszystkie edycje robimy od niej
E2 edycje ChatGPT	~20 wariantów 1:1	każda edycja oceniona: sylwetka bez zmian?
E3 See-through	warstwy na pełnym płótnie	kontaktówka warstw
E4 czyszczenie	poprawione twarz, szyja, tors, rękawy, łysa głowa	<code>compose_check.py</code> : złożenie $\approx$ baza
E5 ruch i mimika	warianty, tęczęwki, powieki, brwi, usta, uszy, <code>gaze.json</code>	kontaktówka + podgląd
E6 PSD	<code>wise-one-v1-3x.psd</code> 3072×4608	import w Cubism, <code>~variants</code> obecna
E7 rig	<code>.cmo3</code> + eksport runtime	test na stronie

Do przeglądu każdej bramki: `contact_sheet.py` (kontaktówka warstw na szachownicy) + composite i `diff.png`.

15. ChatGPT — obraz bazowy i edycje

Obraz bazowy (E1) — wymagania

- **Jedna** postać, frontalnie lub lekko 3/4, kadr od pasa w górę (v1), bez tekstu, logo i tła (jednolite tło, najlepiej jasne/neutralne — łatwiej wyciąć).
- **Oczy szeroko otwarte, usta zamknięte, neutralna mina.** To stan „0” wszystkich parametrów.
- Ręce **nie zasłaniają** twarzy ani tułowia; włosy nie przecinają oczu (grzywka może zakrywać brwi częściowo, ale wtedy brwi trzeba osobno dorysować).
- Czysty, kreskówkowy styl z wyraźnym konturem — See-through radzi sobie z nim najlepiej.
- Po akceptacji baza jest **zamrożona**. Każda późniejsza poprawka bazy unieważnia wszystkie edycje.

Edycje 1:1 (E2) — reguła numer jeden

Każda edycja ma zmieniać **tylko jedną rzecz**. Szablon promptu, którego się trzymamy:

Edytuj załączony obraz. Zmień TYLKO: <co dokładnie>.
Kadr, poza, proporcje, styl, kolory, oświetlenie, tło i wszystkie pozostałe szczegóły mają pozostać identyczne piksel w piksel. Nie przesuwaj, nie skaluj, nie poprawiaj niczego innego.

Lista edycji, które robiliśmy (pełne prompty: `chatgpt/PROMPTY.md` w repo Lunety):

Grupa	Edycje	Po co
Oczy	zamknięte powieki (edit-01), spojrzenie w lewo / prawo / górę / dół	mruganie, zakresy <code>ParamEyeBall*</code> , <code>gaze.json</code>
Usta	A, I, U, E, O, uśmiech, „dzióbek” (pout)	lip-sync i <code>ParamMouthForm</code>
Brwi / twarz	zdziwienie, złość, smutek, mrugnięcie, rumieniec	miny <code>.exp3.json</code>
Uszy	uszy położone płasko	własny <code>ParamEarFlat</code>
Części zasłonięte	bez grzywki (skóra czoła), bez włosów (łysa głowa), sama kurtka, tors bez rąk (edit-14), bez dłoni	żeby przy ruchu pod spodem nie było dziur

Słaba edycja = odrzucić i powtórzyć mocniejszym promptem, nie łatać. Typowe błędy ChatGPT: lekko przesunięta lub przeskalowana postać, zmieniona fryzura, inna grubość konturu, „poprawione” dłonie. Drobne przesunięcie skrypty wyrównają (ECC), ale zmienionej sylwetki — już nie.

Wskazówki:

- Wgrywaj **zawsze bazę**, nie poprzednią edycję — inaczej błędy się kumulują.
- Przy częściach zasłoniętych (bez grzywki, łysa głowa) dopisz, co ma się pojawić w miejscu usuniętej rzeczy („gładkie czoło w kolorze skóry”, „kształt czaszki”).
- Nazwy plików od razu porządkuj (`edit-01-eyes-closed.png` itd.) — skrypty biorą je po nazwie.

16. ComfyUI + See-through (E3)

See-through rozkłada pojedynczy obraz postaci anime na warstwy semantyczne (włosy przód/tył, oczy, twarz, ubranie...) i szacuje głębię, **dorysowując części zasłonięte**.

- Uruchamiamy przez `scripts/comfy_run.py` z workflowem z `workflows/`.
- **Rozdzielczość 1024.** Przy 1280 VAE pada na XPU (Intel Arc) — nie próbuj wyżej, skalujemy później.
- Zasoby: szczyt RAM ~23 GB, VRAM ~10 GB, ~4,5 min na przebieg. **Nie odpalać przy uruchomionej grze.**
- Dwa przebiegi:
 1. **ciało** — cała baza,
 2. **głowa** — wycinek głowy (większe oczy/usta = dokładniejsze maski małych części).
- Składanie na wspólne płótno: `split_to_canvas.py --pixels base [--full-size WxH --offset x,y]` — **piksele widoczne bierzemy z bazy** (ostrzejsze, wierne), a z See-through tylko maski i części zasłonięte.

17. Czyszczenie warstw (E4)

Skrypt	Co robi
<code>fix_face.py</code>	skóra twarzy/czoła z edycji „bez grzywki” — żeby pod grzywką była prawdziwa skóra
<code>fix_back_neck.py</code>	dorysowana tylna część szyi (widoczna przy obrocie głowy)
<code>topwear_from_edit.py</code>	tors/kurtka z edycji „sama kurtka”
<code>cut_arms_head.py</code>	rękawy, dłonie i łysa głowa wycięte z odpowiednich edycji
<code>defringe.py</code>	usuwa jasne/ciemne obwódki na krawędziach alfy (halo po wycinaniu)

Skrypt	Co robi
<code>compose_check.py</code>	składa wszystkie warstwy i porównuje z bazą: średnia różnica ma być mała , <code>diff.png</code> do obejrzenia

Mechanizm „edycja → warstwa”: edycja z ChatGPT jest wyrównywana do bazy algorytmem **ECC** (OpenCV, dopasowanie afiniczne), potem liczona jest różnica i z niej powstaje maska nowej części. Dlatego edycje muszą być 1:1 — ECC skoryguje przesunięcie o kilka pikseli, ale nie inną pozę.

18. Ruch i mimika (E5)

Skrypt	Wynik
<code>diff_layers.py</code>	warianty (brwi, twarze, rumieniec) z edycji jako osobne warstwy <code>~...</code>
<code>eye_rig.py</code>	tęczówki oddzielone od białek, zamknięte powieki, brwi, <code>gaze.json</code>
<code>mouth_shapes.py</code>	<code>~mouth_A/I/U/E/O/smile/pout</code>
<code>ears_flat.py</code>	<code>~ears_flat_l/r</code>

`gaze.json` zawiera środki tęczówek i przesunięcia (left/right/up/down) zmierzone z edycji „patrzy w...”. Wartości są w **pikselach płótna 1024** — dla PSD 3x mnożymy ×3 (stąd tabela w sekcji 9).

19. Budowa PSD (E6) — wymagania „Live2D-ready”

`upscale_layers.py` (RealESRGAN ×3 na XPU) → `PSD_SRC=up3 build_psd.py OUT.psd`.

Wymagania, które skrypt spełnia, a które warto znać przy ręcznym PSD:

- **Każda część na osobnej warstwie**, każda warstwa w **pełnym rozmiarze płótna** (pozycje się nie rozjeżdżają).
- **Bez masek przycinających, efektów warstw i warstw dopasowania** — Cubism ich nie importuje (albo importuje źle). Wszystko zrasteryzowane, tryb mieszania Normal.
- RGB, 8 bit, przezroczyste tło.
- **Unikalne nazwy ASCII** warstw — stają się ID ArtMeshy.
- **Części zasłonięte dorysowane** (skóra pod grzywką, łysa głowa, tors pod rękami).
- Warianty w **ukrytej grupie** `~variants` — Cubism je importuje, ale nie pokazuje na starcie.
- Grupy i kolejność rysowania ustawione w `ORDER` w `build_psd.py`.

- Rozdzielczość: 1024 z See-through to za mało na teksturę 4096 → upscale ×3 do 3072×4608.

20. Rig dla zaawansowanych

Siatki

- Auto Mesh na start, potem ręcznie: **gęściej oczy, usta, grzywka**, rzadziej duże płaskie powierzchnie.
- Punkty siatki kładź na **krawędziach kształtu i liniach zgięcia** (np. wzdłuż powieki) — wtedy deformacja nie „pływa”.
- **Glue** (sklejanie siatek) dla części, które muszą się stykać (szyja-twarz, rękaw-dłoń).

Hierarchia deformerów

```
Body_Rotation (ParamBodyAngleZ)
└─ Body_Warp (ParamBodyAngleX/Y)
    │─ Breath_Warp (ParamBreath) — topwear, sleeve_*, hand_*
    │─ neck
    │─ BackHair_Warp (ParamHairBack) — back_hair
    └─ Head_Rotation (ParamAngleZ, pivot na szyi)
        └─ Face_Warp (ParamAngleX/Y)
            │─ Eye_L_Warp / Eye_R_Warp — eyewhite, irides, eyelash, ~eyelid_closed
            │─ Brow_L / Brow_R — eyebrow_*
            │─ Mouth_Warp — mouth, ~mouth_*
            │─ ears_*
            └─ FrontHair_Warp (ParamHairFront) — front_hair
```

- **Rotation** dla obrotów wokół punktu (`AngleZ`, `BodyAngleZ`), **Warp** dla „pseudo-3D” (`AngleX/Y`).
- Przy `ParamAngleX` + `ParamAngleY` naraz robi się **2D keyform (siatka 3×3 kluczy)**. Skrajne rogi (np. X=30, Y=30) często trzeba ręcznie poprawić — automat liczy je jako mieszaną, co daje „spłaszczenie”.
- Włosy z tyłu ruszają się **przeciwnie** do twarzy przy `AngleX` (paralaksa) — to daje wrażenie głębi.

Ładniejsze mruganie

Zamiast samej przezroczystości: przy `EyeOpen=0` zgnieć siatkę rzęs w łuk zamkniętej powieki, białko i tęczęwkę przytnij (clipping) do obrysu oka — przy wartościach pośrednich oko zamyka się płynnie, a nie „przenika”. Klucz pośredni 0.5 pomaga uniknąć „pustego” oka w połowie mrugnięcia. Opcjonalnie `ParamEyeLSmile/RSmile` — oczy-łuki przy uśmiechu.

Usta i lip-sync

- `ParamMouthOpenY` × `ParamMouthForm` to również siatka 2D (otwarcie × kształt). Minimum: 2×3 klucze.
- Kształty samogłosek (`~mouth_A/I/U/E/O`) można wpiąć jako klucze pośrednie albo — w PRO — użyć **Motion Sync** (lip-sync na podstawie samogłosek z dźwięku) zamiast samej głośności.

Multiply / Screen color

ArtMesh może mieć kolor mnożony/ekranowy sterowany parametrem — tak robi się np. rumieniec bez osobnej warstwy albo przyciemnienie postaci w Lunecie. Runtime (pixi-live2d-display 0.4) obsługuje to tylko częściowo — testuj przed użyciem.

Fizyka

- *Modeling* → *Physics* (ustawienia fizyki i scen). Grupa = wejścia (`ParamAngleX/Z`, `ParamBodyAngleX/Z`) → wahadło (długość, sztywność, opór) → wyjście (`ParamHairFront`, `ParamHairBack`, ew. ucho/kokarda).
- Dłuższe wahadło = wolniejszy, „cięższy” ruch. Grzywka: krótkie, sztywne; włosy z tyłu: dłuższe.
- Eksport fizyki → `*.physics3.json`.

Template (PRO)

Gdy automatyczna mimika nie rozpoznaje warstw, można nałożyć **szablon** z gotowego modelu (np. Hiyori z Cubism SDK Samples) — przenosi deformery i klucze, które potem dopasowujemy ręcznie. Działa najlepiej, gdy podział warstw jest podobny do szablonu.

21. Animacje i miny

- **Tryb Animation** → nowa scena → oś czasu, klucze parametrów w czasie.

- `Idle` (pętla, 4–8 s): `ParamBreath` sinusoidalnie, delikatny `ParamAngleX/Z` ($\pm 3-5$), od czasu do czasu `ParamEyeBall*`. Mruganie **nie** w animacji — robi je runtime (grupa `EyeBlink`).
- `Tap` (v2): reakcja na kliknięcie, np. znak pokoju.
- Miny: ustawiasz suwaki (brwi, `ParamCheek`, `~face_*`) i zapisujesz jako expression → `.exp3.json`.
- Eksport: `File` → `Export For Runtime` → `motion3.json` / `exp3.json`, do tego samego katalogu co model.
- W `model3.json` motions są w grupach (`Idle`, `TapBody` ...) — nazwy grup to te, których używa kod.

22. Eksport — szczegóły

- **Atlas tekstur:** przy eksporcie Cubism składa ArtMeshe w atlas. 4096 dla strony; dla Lunety 2048 (mniej VRAM, wystarczy przy mniejszym oknie). Sprawdź, czy nic nie zostało przeskalowane za mocno.
- `model3.json` musi zawierać `Groups`: `EyeBlink` = `ParamEyeL0pen`, `ParamEyeR0pen`; `LipSync` = `ParamMouth0penY`. Bez tego runtime nie mruga i nie rusza ustami.
- `HitAreas` (opcjonalnie) — obszary klikalne (`Head`, `Body`) do reakcji na dotyk.
- Projekt `.cmo3` trzymamy w `C:\wise-one\`, eksport w `C:\wise-one\runtime\` → scp na ariku-pc (`/mnt/nvme/ai/wise-one/e7-rig/runtime/`).

23. Runtime: strona i Luneta

Stos (wersje przypięte): `pixi.js@6.5.10` + `pixi-live2d-display@0.4.0` (Cubism 4/5 `.moc3`)

- Cubism Core `live2dcubismcore.min.js`. `pixi-live2d-display` 0.4 wymaga PIXI **v6**, nie v7+.

```
// Core musi być załadowany przed importem (window.Live2DCubismCore).
await loadScript(coreUrl);           // <script src> z ariku.pl / asset protocol Tauri
import * as PIXI from 'pixi.js';
import { Live2DModel } from 'pixi-live2d-display/cubism4';
(window as any).PIXI = PIXI;         // pixi-live2d-display używa globalnego tickera

const app = new PIXI.Application({ view: canvas, backgroundAlpha: 0, autoStart: true,
resolution: devicePixelRatio });
app.ticker.maxFPS = 30;               // tanio
const model = await Live2DModel.from('/live2d/wise-one/wise-one.model3.json');
app.stage.addChild(model);
```

Efekt	Jak
Mruganie, oddech	automatycznie, gdy <code>model3.json</code> ma grupę <code>EyeBlink</code> i parametr <code>ParamBreath</code>
Śledzenie kursora	<code>model.focus(x, y)</code> na <code>pointermove</code> (<code>ParamAngleX/Y</code> , <code>ParamEyeBallX/Y</code>)
Idle	<code>model.motion('Idle')</code>
Mina	<code>model.expression('surprised')</code>
Lip-sync	<code>AnalyserNode</code> → RMS co klatkę → <code>coreModel.setParameterValueById('ParamMouthOpenY', v)</code> w <code>beforeModelUpdate</code>
Pauza	<code>document.visibilitychange</code> → <code>app.ticker.stop()/start()</code>

Luneta (Tauri): komponent `src/components/Live2DView.tsx`; model i Core z katalogu danych (`convertFileSrc` / `asset protocol`) — dopisać zakres w `tauri.conf.json` (`assetProtocol.scope`) i CSP (`script-src` dla Core). Brak Core lub modelu → statyczny obraz (splash), bez błędu. Nakładka nad grą tylko informacyjna — kliknięcia przechodzą do gry. Animacja tylko przy widocznym oknie, FPS ograniczony (ENGINEERING-RULES §36).

ariku.pl: model i Core jako pliki statyczne; ta sama inicjalizacja, `autoInteract` tylko na hover, `prefers-reduced-motion` → bez Idle.

Test: najpierw darmowy model przykładowy (Hiyori z Cubism SDK Samples), potem Wise One: renderuje się, śledzi kursor, mruga, przy ukrytym oknie FPS = 0.

24. Licencje — ważne

- **Cubism Core** (`live2dcubismcore.min.js`) jest zamknięty (licencja Live2D Proprietary Software). **Nigdy** do publicznego repo — Luneta jest na MIT. Ładujemy go z `ariku.pl` albo z dysku użytkownika.
- Cubism SDK for Web (Framework) ma własną licencję; przy publikacji komercyjnej powyżej progu przychodów Live2D wymaga umowy — sprawdź aktualne warunki przed wydaniem.
- Edytor: trial PRO 42 dni; plików `.cmo3` zapisanych w PRO nie da się w pełni edytować w FREE, jeśli przekraczają limity FREE.
- Obrazy z ChatGPT — zgodnie z warunkami OpenAI; postać jest nasza.

25. Jak to przebiegało u nas (dziennik Wise One v1)

Kiedy	Gdzie	Co
przed 2026-10-17	ChatGPT	obraz bazowy Wise One (E1) zamrożony; ~20 edycji 1:1 (oczy, usta, brwi, miny, uszy, części zasłonięte)
	ariku-pc (Linux, Intel Arc/XPU)	See-through w ComfyUI (ciało + głowa, 1024), czyszczenie, mimika, gaze.json, upscale x3, wise-one-v1-3x.psd
		wnioski: 1280 w See-through pada w VAE na XPU; edycja „tors bez rąk” (edit-14) wyszła słabo; grzywka została jednym blokiem
2026-10-17	felix-pc (Windows)	instalacja Cubism Editor 5.3, aktywacja triala PRO; UI po japońsku → przełączone na English (restart)
2026-10-17	felix-pc	okno „nowa wersja” — kliknięcie ❷ zamknęło edytor; od tej pory zawsze Cancel
2026-10-17	felix-pc	próba riggingu przez asystenta (computer use) — okno Javy wymagało osobnej zgody; decyzja: rig robi właściciel sam , asystent pisze poradnik
2026-10-17	wiki	ten poradnik

Co warto zrobić inaczej od początku (lekcje):

1. **Zaplanować warstwy przed bazą** — np. grzywka na 3–5 pasm już w obrazie bazowym, ręce opuszczone.
2. **Robić edycje części zasłoniętych od razu**, a nie dopiero przy czyszczeniu.
3. **Testować runtime na Hiyori**, zanim rig Wise One będzie gotowy — błędy integracji wychodzą wcześniej.
4. **Trial PRO odpalać dopiero, gdy PSD jest gotowy** — 42 dni szybko mijają.

26. Znane braki v1 i plan v2

- Grzywka i boczne włosy to jeden blok — fizyka „sztywniejsza”. **v2**: pasma osobno.
- Postać tylko od pasa w górę. **v2**: pełna postać.
- Tors bez rąk (pod rękawami) słabej jakości — nie podnosić rąk za wysoko. **v2**: lepsza edycja.
- Brak animacji Tap. **v2**: znak pokoju po kliknięciu.

27. Checklista końcowa

- ☐ PSD zaimportowany, `~variants` obecna
- ☐ siatki (gęściej oczy/usta/grzywka)
- ☐ `ParamAngleX/Y/Z`, `ParamBodyAngleX`, `ParamBreath`
- ☐ `ParamEyeL0pen/R0pen`, `ParamEyeBallX/Y` (clipping tęczywek)
- ☐ `ParamMouthOpenY`, `ParamMouthForm`
- ☐ (opcja) brwi, `ParamCheek`, `ParamEarFlat`, fizyka włosów
- ☐ miny `.exp3.json`, animacja `Idle`
- ☐ grupy EyeBlink / LipSync
- ☐ eksport do `C:\wise-one\runtime\`, test na Hiyori → test Wise One
- ☐ Cubism Core poza publicznym repo