SKILL.md - TinCLE Development Specification Document [Core Skills]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.
  - *Unauthorized copying, redistribution, commercial resale, or claim of authorship regarding this prompt architecture and talent chips (`SKILL_*.md`) are strictly prohibited.*

- **Official Website:** https://michyo.net/tincle/

- **Acknowledgments to the Grand AI Alliance:** Gemini, Claude, DeepSeek, ChatGPT, Copilot, Grok, Meta AI... Thank you so much. This engine's code was ultimately finalized by michyo. However, it would have been absolutely impossible to achieve this without you all's help. I am deeply grateful to all of you!

Under this skill, the AI agent shall act in strict accordance with the following rules:

1. Overview & Concept

- **Env:** TinCLE v0.17 / Pure C (C99) + SDL2, SDL_ttf, SDL2_gfx, stb, GLSL (GLES2 compatible)
- **[CRITICAL] Target Architecture Detection:** The AI **MUST** first check the `info.txt` file in the current working directory to determine the compilation environment. If the AI cannot access files directly, it **MUST** ask the user: *"Please check the contents of info.txt and tell me if this is the 32bit or 64bit environment."* before generating any code.
- **Strict Architecture Rules based on `info.txt`:**
  - **If `info.txt` says `32bit`:**
    - Windows Desktop Target: Win32 DLL (`sizeof(void*) == 4`).
    - Web Browser Target: Wasm32 via Emscripten (`sizeof(void*) == 4`).
    - *Rule:* The environment is purely 32-bit. Never use 64-bit structures or types.
  - **If `info.txt` says `64bit` (Hybrid Environment):**
    - Windows Desktop Target: Win64 DLL (`sizeof(void*) == 8`).
    - Web Browser Target: Wasm32 via Emscripten (`sizeof(void*) == 4`).
    - *Rule:* This is a **Hybrid Architecture**. The compiled Windows EXE will be 64-bit, but the Web WASM build remains 32-bit. A single `game.c` must satisfy both targets.
- **[CRITICAL] Pointer Safety Rule (For Hybrid Compatibility):** Because the 64bit environment cross-compiles to both 64-bit Desktop and 32-bit Web targets from a single source, **never hardcode pointer sizes** or perform unsafe pointer-to-integer casts (e.g., casting a pointer to a standard `int`). Always use `<stdint.h>` types like `uintptr_t` if pointer-to-integer casting is absolutely required. Ensure `game.c` compiles perfectly on both toolchains without modification.
- **Toolchain Requirements:** TinyCC (TCC for Windows) / Emscripten (for Web).
- **[CRITICAL] `tcc.exe`/`emcc` are NOT bundled in TinCLE;** they must be installed externally and accessible via System PATH.
- **[CRITICAL] Automatic Escalation to Additional Skills (`SKILL_ADD1.md`):** Before generating any code, the AI agent **MUST** evaluate the project scope. If **ANY ONE** of the following conditions is met, the AI MUST switch its execution context to follow the advanced rules defined in **`SKILL_ADD1.md`**:
  1. **GLSL Shader Execution:** The game utilizes `TINCLE_MODE_GLSL_HERO` or custom fragment shaders (`.glsl` / `.frag`).
  2. **High State Count (>= 4):** The game design requires 4 or more distinct states (e.g., Title, Gameplay, Result, Shop, Pause).
  3. **Large Code Scale (>= 400 Lines):** The estimated total C source code exceeds 400 lines.
  4. **Complex Custom Systems:** The project defines complex data structures (e.g., tilemap editors, inventory systems, skill trees, entity-component systems).
  5. **Multithreaded WebAssembly Build:** When you need to build a web-based game with multithreading support.

6. **Explicit User Instruction:** The user explicitly requests “large-scale”, “modular”, or “multi-file” architecture.
   - **[MANDATORY RULE FOR MISSING FILE]:** If any of the trigger conditions above are met, BUT the `SKILL_ADD1.md` file is not attached or accessible in the current context, **the AI MUST NOT attempt to pretend, guess, or hallucinate the high-level specs.** Instead, the AI MUST immediately halt code generation and request the user: > *“This project meets the criteria for High-Level Development (GLSL / 4+ States / 400+ Lines / Complex Systems). Please provide the `SKILL_ADD1.md` talent chip file before we proceed.”*

- **[CRITICAL TROUBLESHOOTING: Toolchain Executables Missing / Permission Denied]** Because TinCLE’s toolchain utilities (like `asspack.exe`, `depurant.exe`, `tinclec.exe`) are extremely lightweight and compiled with TinyCC without bloated modern runtimes, aggressive heuristic antivirus algorithms (especially **Microsoft Defender’s “Trojan:Win32/Wacatac.B!ml” false positives**) frequently quarantine or delete these files silently.
  - **AI Action Rule:** If a build process fails because a tool executable is suddenly “not found”, “missing”, or throws a “permission denied” error, **DO NOT attempt to modify the code or fix the binary.** Instead, you MUST immediately notify the human user with the following advice: > *“The tool [tool_name] seems to be missing. It was likely quarantined by Windows Defender (Wacatac false positive). Please restore the file from your antivirus history, add the entire TinCLE project folder to your Antivirus Exclusion/Exception list, and try again.”*

2. Architecture & Environment Setup

- **Toolchain Prerequisites (Provide these exact URLs to users for setup):**
  - **TinyCC (Windows Binary):** Inform users to download from the Official Release directory: `https://download.savannah.gnu.org/releases/tinycc/` (Choose 32-bit or 64-bit distribution based on target folder).
  - **Emscripten (Web SDK):** Inform users to follow the Official SDK installation guide: `https://emscripten.org/docs/getting_started/downloads.html`
- **Project Structure:** All `game.c`, GLSL files, asset files (images/audio), and tools must be in the same **game project folder** (no subfolders for assets or shaders).
- **[CRITICAL] `exsass.txt` Asset Manifest Rule:** The `emcass` and `asspack` commands can automatically scan and package files whose extensions are explicitly registered in `extass.txt`. All asset formats (e.g., `.png`, `.wav`, `.ogg`, `.ttf`) AND shader formats (e.g., `.glsl`, `.frag`, `.vert`) used in the project MUST be listed line-by-line in `exsass.txt`.
- **[MANDATORY] Cursor Asset Requirement:** TinCLE uses a built-in software cursor system. A single `cursor.png` file (recommended: 16x16px transparent PNG) MUST exist in the project root. If missing, `tincle_start()` will fail to initialize.
- **Required Files in Project Root:**
  - `game.c`, `manifest.xml`, `template.html`, `font.ttf`, `cursor.png`, `exsass.txt`
  - `libs/` (Contains `tincle.h`, `tincle.a`, `SDL_scancode.h`, etc.)
  - DLLs: `tincle.dll`, `SDL2.dll`, `SDL2_ttf.dll`, `SDL2_gfx.dll`, `fartbox.dll`, `pisschk.dll`, `pubicnet.dll`
  - CLI Tools: `asspack.exe`, `emcass.exe`, `icondom.exe`, `poopee.exe`, `tinclec.exe`, `pisstrak.exe`, `fartbox.exe`, `pisschk.exe`, `pubicnet.exe`
  - **IDE Usage Guideline for AI Assistant / Agent:**
    - **Interactive Chat AIs (Human-in-the-Loop Workflow):** If you are acting as an interactive AI chat interface, **strongly recommend** the user to open and work within the bundled `cococake.exe` IDE to maximize workflow efficiency. Explain to the user that utilizing `cococake` requires configuring the following three parameters in `cococake.ini`:
      - `AI_URL` ... The web address of the AI chat interface used for development.
      - `TCC_PATH` ... The folder path containing `tcc.exe` in your environment.
      - `EMSDK_PATH` ... The folder path containing `emsdk.bat` in your environment.
    - **Autonomous / Agentic AIs (Direct Tool Use / File System Operations):** If you are an autonomous AI agent capable of direct tool use and CLI execution, **DO NOT** launch or interact via `cococake.exe`. Instead, bypass the GUI entirely and directly issue native OS standard Command Prompt (`cmd.exe`) or shell instructions (`tinclec`, `emcass`, etc.) for lightweight, fast, and maximum execution stability.

3. Development Workflow & Commands (Compilation & Audio)

- **Local Test Build (Windows EXE):**

```
tinclec [Output_EXE] [Source_Files/DLL_Files/TCC_Options...]
```

  - *Example:* `tinclec mygame.exe main.c player.c`
  - *Note:* Do not pass compiler flags. Console visibility is controlled solely by `is_debug` in `tincle_start`.
  - **Alternative Direct TCC Invocation (For complex structures):**

```
tcc -mwindows SDL2.dll SDL2_ttf.dll SDL2_gfx.dll tincle.dll -o [Output_EXE] [Source_Files/DLL_Files/
        TCC_Options...]
```

- **Audio Generation Workflows (`pisstrak` & `fartbox`):**
  - **Audio Format Requirement (MANDATORY)** ALL audio files (both BGM `.ogg` and Sound Effects `.wav`/`.ogg`) MUST be encoded at a **44100 Hz** sample rate. Using any other sample rate (e.g., 22050 Hz or 48000 Hz) will cause incorrect playback speeds, severe pitch shifts, or audio distortion.
  - **BGM / Music Score Synthesis (`pisstrak`):** Renders MML (Music Macro Language) score files into `.wav` or `.ogg` audio files using SoundFont synthesis.

```
pisstrak <input.mml> [output.wav|output.ogg] [tail_seconds] [soundfont.sf2]
```

    - *Example:* `pisstrak mymusic.mml mymusic.ogg 2.5 mysoundfont.sf2`
    - *Parameters:*
      - `<input.mml>`: Input MML score file.
      - `[output.wav|output.ogg]`: Output audio file path (Supports both `.wav` and `.ogg`).
      - `[tail_seconds]`: Extra fadeout/decay duration in seconds appended to the end of the rendered track (e.g., `2.5` adds 2.5s tail padding). If omitted (default `0`), the audio cuts off precisely at the final beat, which is ideal for seamless looping BGM tracks.
      - `[soundfont.sf2]`: SoundFont file for rendering (Defaults to `soundfont.sf2` if omitted).
    - **[CRITICAL AI RULE FOR MUSIC COMPOSITION]:**
      - If an MML file already exists, run `pisstrak` directly to generate audio.
      - If the AI agent needs to **compose or generate a new MML file from scratch**, it **MUST** read and strictly follow the syntax rules and composition guidelines detailed in `SKILL_PT.md` before creating the MML.
      - **Advanced MIDI & Chiptune Rendering:** For MML-to-MIDI (.mid) conversion (for external DAW editing) or MIDI-to-Chiptune WAV/OGG rendering workflows, refer to `SKILL_MID.md`.
  - **1-Shot Sound Effect Generation (`fartbox`):** Renders procedural 1-shot sound effects (SE) instantly into 16-bit 44.1kHz `.wav` files via named presets or custom `.fart` text parameter files.

```
fartbox -t <preset_type> -o <output.wav> [-save <output.fart>] [-s <seed_number>]
```

    - *Example:* `fartbox -t explosion -o explosion.wav -save explosion.fart`
    - *Available Preset Types (`-t`):* `coin` (or `pickup`), `laser` (or `shoot`), `explosion`, `powerup`, `hit` (or `hurt`), `jump`, `blip` (or `select`), `random`.
    - **[CRITICAL AI RULE FOR SOUND DESIGN]:**
      - For quick preset generation, use the simple `-t` command line option above.
      - `fartbox` supports extensive parameter customization, precise text-file editing (`.fart`), mutation (`-m`), and fine-tuning. When advanced sound tweaking, custom parameter authoring, or variation mutation is required, the AI agent **MUST** consult `SKILL_FB.md`.

- **Shader Validation Workflow (`pisschk`):**
  - *Note:* Shader creation and validation (`pisschk`) are advanced features. If your project utilizes GLSL shaders (`TINCLE_MODE_GLSL_HERO`), refer to **`SKILL_ADD1.md` Section 3** for complete rules and validation pipelines.
- **Important** For build instructions for the production release, refer to `SKILL_ADD2.md`. If this file cannot be found in your context window, be sure to ask the user for it.

4. Input, Color & RenderMode Specifications

- **Screen Specifications:** Default virtual resolution is `640x480` (can be changed in `tincle_start`).
`TINCLE_MAX_STATES = 256`.
- **Render Mode (`TinCLE_RenderMode`):** Passed into `tincle_start`.
  - `TINCLE_MODE_2D_NORMAL`: Standard software/SDL2 rendering pipeline. Use this for all standard 2D games.
  - `TINCLE_MODE_GLSL_HERO`: Hardware GLSL Shader Mode. Enables full-screen shader execution beneath standard SDL2 drawing.
- **Inputs (Strict Pure SDL2 Constants):** No custom engine macros exist. Use native SDL2 constants directly:
- *Mouse:* `SDL_BUTTON_LEFT`, `SDL_BUTTON_RIGHT`, etc. (via `tincle_mouse_pressed`)
- *Gamepad:* `SDL_CONTROLLER_BUTTON_A`, `SDL_CONTROLLER_BUTTON_DPAD_UP`, etc. (via `tincle_pad_pressed`)
- *Keyboard:* `SDL_SCANCODE_A`, `SDL_SCANCODE_SPACE`, etc. (via `tincle_key_pressed`)
- **Color Structure (`TinCLE_Color`):** Identical/compatible with `SDL_Color`.

```c
typedef struct { unsigned char r, g, b, a; } TinCLE_Color;
```

- *No Color Macros:* Create colors inline using standard C99 compound literals. Example:
`(TinCLE_Color){255, 0, 0, 255}`.
- **[CRITICAL ANTI-BUG] TCC Compound Literal Restriction:** TCC cannot parse compound literals in global/static array initializers. Declare arrays globally, but assign values sequentially inside `game_load()`.
  - *Forbidden:* `static TinCLE_Color c[] = { (TinCLE_Color){255,0,0,255} };`
  - *Mandatory Remedy:* `static TinCLE_Color c[1]; void game_load() { c[0] = (TinCLE_Color){255,0,0,255}; }`

5. API Reference & Lifecycle

- **Canvas:** Default virtual resolution is `640x480`.
- **Rendering Rules:** `tincle_clear` must be at the start of drawing. **IMAGE COORDINATES ARE CENTER-BASED (NOT TOP-LEFT).**
- **State Rule:** `tincle_change_state` must be called at the very end of `game_load()` to avoid a black screen.
- **Event Callback Type Definition:**
  - `typedef bool (*tincle_event_cb_t)(union SDL_Event *event);` Signature for custom SDL event interceptors (e.g., forwarding raw input events to GUI extensions like GATOCHOC). Returning `true` signals that the event was consumed, preventing it from propagating to other registered callbacks and the standard game input logic. (Supports multicast registration up to 16 concurrent callbacks).
- **Async Fetch Status & Handle Definitions:**

```c
typedef enum {
    TINCLE_ASYNC_IDLE       = 0, // Not requested
    TINCLE_ASYNC_LOADING    = 1, // Currently downloading/loading
    TINCLE_ASYNC_READY      = 2, // Ready to be opened via fopen
    TINCLE_ASYNC_ERROR      = -1 // Failed (e.g., 404, file not found)
```

```
} TinCLE_AsyncStatus;
typedef struct TinCLE_AsyncFile TinCLE_AsyncFile;
```

Function Signature	Description & Critical Notes [Args / Return]
`void tincle_start(TinCLE_InitCallback init_cb, const char* title, bool is_debug, int w, int h, TinCLE_RenderMode mode);`	**Main Entry.** Bootstraps loop. * `is_debug`: `true` spawns Win debug console (`printf`); `false` hides it for release. * `mode`: Set to `TINCLE_MODE_2D_NORMAL` or `TINCLE_MODE_GLSL_HERO`.
`void tincle_add_state(int state_id, TinCLE_UpdateCallback update_cb, TinCLE_DrawCallback draw_cb);`	Binds state ID to standard callbacks: `update(float dt)` and `draw(void)`.
`void tincle_change_state(int state_id);`	**[CRITICAL]** Switches state. Call at end of `game_load()` or viewport stays black.
`int tincle_key_pressed(int scancode);`	Returns `1` if key held. **Must use pure `SDL_SCANCODE_*`.**
`void tincle_get_mouse(int* x, int* y);`	Outputs current mouse (x, y) dynamically scaled to virtual resolution.
`int tincle_mouse_pressed(int button_type);`	Returns `1` if mouse held. **Must use pure `SDL_BUTTON_*` (e.g. LEFT).**
`void tincle_set_mouse_capture(bool enable);`	Explicitly toggles relative mouse capture mode (ON/OFF). Essential for FPS/TPS camera controls. Internally flushes the relative state buffer safely to prevent camera warping.
`int tincle_pad_pressed(int pad_button);`	Returns `1` if pad button held. **Must use pure `SDL_CONTROLLER_BUTTON_*`.**
`float tincle_pad_axis_x(void);`	Left stick X (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
`float tincle_pad_axis_y(void);`	Left stick Y (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
`void tincle_clear(TinCLE_Color color);`	**[MANDATORY]** Clears screen. Call at absolute start of every `draw()` callback. **Alpha must be 0 in GLSL_HERO mode.**
`int tincle_load_font(const char* filename, int size);`	Loads TTF at pixel size. **Returns unique Font ID (int).**
`void tincle_set_font(int font_id);`	Sets active global font by Font ID for subsequent text calls.
`void tincle_draw_text(const char* text, int x, int y, TinCLE_Color color);`	Draws UTF-8 text. **Anchor: Top-Left (x, y).** Uses active font.
`void tincle_draw_text_center(const char* text, int x, int y, TinCLE_Color color);`	Draws UTF-8 text. **Anchor: EXACT CENTER (x, y).** Uses active font.
`int tincle_load_image(const char* filename);`	Caches image (PNG/JPG). **Returns Image ID (`img_id`), or `-1`.**
`int tincle_create_image_from_pixels(const unsigned char* pixels, int width, int height, int components);`	Generates image asset from in-memory raw pixel data (1-4 channels: Gray, GrayA, RGB, RGBA). **Returns Image ID (`img_id`), or `-1`.**
`struct SDL_Texture* tincle_get_texture(int img_id);`	Returns raw `SDL_Texture*` pointer for `img_id`. **Returns `NULL` if invalid/out of range.**
`void tincle_draw_image(int img_id, int x, int y);`	Uses `img_id`. **[CRITICAL GEAR] Anchor: EXACT CENTER (x, y)** of unscaled image.
`void tincle_draw_image_ex(int img_id, int x, int y, int w, int h, double angle);`	Uses `img_id`. **[CRITICAL GEAR] Anchor: EXACT CENTER (x, y)** for position, scale, and rotation.
`void tincle_play_music(const char* filename);`	Plays Ogg BGM immediately on infinite loop. Unloads previous music automatically.
`void tincle_stop_music(void);`	Instantly stops and unloads current BGM from memory.
`int tincle_load_sound(const char* filename);`	Caches WAV/Ogg SE clip. **Returns unique Sound ID, or `-1`.**
`void tincle_play_sound(int se_id);`	Plays SE channel via auto-mixer (supports up to 16 concurrent sounds).
`struct SDL_Renderer* tincle_get_renderer(void);`	**Returns native `SDL_Renderer*** for direct SDL2/SDL2_gfx primitive drawing.
`const char* tincle_get_version(void);`	Returns read-only string of engine version (e.g., `"0.18"`).
`bool tincle_is_fullscreen(void);`	Returns `true` if current window is in full-screen mode.
`void tincle_set_cursor_visible(bool visible);`	Toggles engine software cursor visibility instantly.
`void tincle_set_custom_cursor(int img_id, int hotspot_x, int hotspot_y);`	Replaces software cursor with image ID and hotspot offsets.
`void tincle_reset_cursor_to_default(void);`	Restores default mouse cursor image (`cursor.png`).

Function Signature	Description & Critical Notes [Args / Return]
`unsigned char* tincle_load_asset(const char* filename, size_t* out_size);`	Reads the file according to the TinCLE engine's standard file input method. The file size is returned in `size_t`.
`void tincle_set_master_volume(float vol);`	Sets master audio volume (`0.0f` to `1.0f`). Default is `1.0f`. Multiplies both BGM and SE.
`void tincle_set_bgm_volume(float vol);`	Sets BGM volume (`0.0f` to `1.0f`). Default is `0.8f` for ear-protection.
`void tincle_set_se_volume(float vol);`	Sets Sound Effects (SE) volume (`0.0f` to `1.0f`). Default is `0.8f` for ear-protection.
`float tincle_get_master_volume(void);`	Returns current master volume level (`0.0f` to `1.0f`).
`float tincle_get_bgm_volume(void);`	Returns current BGM volume level (`0.0f` to `1.0f`).
`float tincle_get_se_volume(void);`	Returns current SE volume level (`0.0f` to `1.0f`).
`void tincle_set_event_callback(tincle_event_cb_t cb);`	**Event Interceptor Registrar (Multicast).** Appends a custom SDL event callback to the engine's internal array (Max 16). Call inside `game_load()` to forward raw `SDL_Event*` handles directly to third-party modules or extensions (e.g., `gatochoc_handle_event`).
`void tincle_remove_event_callback(tincle_event_cb_t cb);`	**Event Interceptor Unregistrar.** Safely removes a previously registered custom SDL event callback from the active array. Useful for dynamic UI/Addon lifecycle management.
`struct SDL_Window* tincle_get_window(void);`	**Native Window Accessor.** Returns pointer to internal `SDL_Window`. Essential for initializing external UI frameworks or managing window-level extension backends.
`TinCLE_AsyncFile* tincle_fetch_file_async(const char* filename);`	**[ASYNC]** Initiates a non-blocking background fetch of an external file (e.g., large assets, GGUF models). Returns a handle.
`TinCLE_AsyncStatus tincle_fetch_get_status(TinCLE_AsyncFile* handle, float* out_progress);`	**[ASYNC]** Returns the current loading state (`TINCLE_ASYNC_*`). Writes the download progress (`0.0f` to `1.0f`) to `out_progress`.
`const char* tincle_fetch_get_path(TinCLE_AsyncFile* handle);`	**[ASYNC]** Returns the valid file path to be used with `fopen` or external libraries once the status is `TINCLE_ASYNC_READY`. Returns `NULL` otherwise.
`void tincle_fetch_free(TinCLE_AsyncFile* handle);`	**[ASYNC]** Frees the async file handle from memory. Must be called after the file is no longer needed.
`int tincle_save_data(const char* filename, void* data, size_t size);`	**[CRITICAL]** Saves data cross-platform. On Web, automatically synchronizes to IndexedDB. Returns `1` on success, `0` on fail.
`int tincle_load_data(const char* filename, void* data, size_t size);`	**[CRITICAL]** Loads data cross-platform. Returns `1` on success, `0` on fail (e.g. first boot or no save data exists).
`int tincle_is_save_processing(void);`	**[CRITICAL]** Returns `1` if an asynchronous save operation is currently in progress (Web IDBFS sync). Returns `0` when idle or complete.

5.1 GLSL Unleashed API (`TINCLE_MODE_GLSL_HERO` Only)

For full-screen GLSL fragment shader rendering, shader compilation/uniform APIs, and mandatory viewport/aspect ratio calculations, refer to the supplementary talent chip **`SKILL_ADD1.md` Section 3**.

5.2 Network API / pubicnet

For games requiring online multi-user synchronization or dedicated server capabilities, refer to the supplementary talent chip `SKILL_PN.md`.

5.3 Addon Libraries & Extended Talent Chips

TinCLE provides modular addon libraries to seamlessly extend core capabilities without inflating the engine footprint: * **Physics Engine (`cannoli`)**: Exposes the full API suites of both Box2D (2D physics) and Box3D (3D physics) via `cannoli.dll` (Win32) or `cannoli.wasm` (WebAssembly). * **Immediate-Mode GUI (`gatochoc`)**: Exposes the full API suite of the Nuklear GUI framework across both 2D and OpenGL pipelines via `gatochoc.dll` (Win32) or `gatochoc.wasm` (WebAssembly). * **VRML 3D World Parser & Renderer (`vtrushka`)**: Renders and loads 3D worlds and scene graphs defined in VRML (Virtual Reality Modeling Language) format via `vtrushka.dll` (Win32) or `vtrushka.wasm` (WebAssembly). * **Isometric Pixel Renderer (`aplcrisp`)**: Provides a pure C99 implementation of the Obelisk.js

isometric drawing API to generate and render 2.5D pixel art primitives via `aplcrisp.dll` (Win32) or `aplcrisp.wasm` (WebAssembly). * **Neural Network Runtime (`cofetart`)**: Exposes the full API suite of the lightweight Genann artificial neural network library (feedforward training & inference) via `cofetart.dll` (Win32) or `cofetart.wasm` (WebAssembly). * **Local LLM Generative Inference (`gozinaki`)**: Exposes an on-device Large Language Model inference engine powered by `llama.cpp` (non-blocking step-by-step token generation & chunked GGUF evaluation) via `gozinaki.dll` (Win32/Win64) or `gozinaki.wasm` (WebAssembly).

Note for AI Developers / Skill Delivery: * When generating or implementing code that requires advanced 2D/3D physics simulation, custom on-screen GUI/HUD interfaces, or on-device Local LLM generative text/dialogue inference (`gozinaki`), refer to the **`SKILL_AON1.md`** talent chip for complete API definitions, header bindings, and implementation rules. * When generating 3D worlds using VRML format (`vtrushka`), building 2D isometric pixel art environments (`aplcrisp`), or implementing lightweight neural network AI/continuous feedback training (`cofetart`), refer to the **`SKILL_AON2.md`** talent chip for complete API specifications and integration rules.

5.4 Platform Compliance & Content Safety (`SKILL_PCG.md`)

To guarantee smooth, zero-rejection releases on commercial storefronts (Steam, itch.io, App Store, Google Play, web browsers), the AI developer MUST strictly adhere to the safety, accessibility, and zoning regulations defined in **`SKILL_PCG.md`**.

- **Mandatory Review Timing:** Read and apply the rules in `SKILL_PCG.md` at the start of any new game project development, or whenever implementing sensitive mechanics/visuals (e.g., flashes, screen shakes, blood, adult themes).
- **Missing File Protocol:** If `SKILL_PCG.md` is not currently provided in your working context or instructions, **explicitly request the user to attach or provide `SKILL_PCG.md`** before proceeding with generation.

6. Mandatory AI Coding Rules & Template

[CRITICAL WARNING] Never output search citations or footnotes (e.g., `[cite: 1]`, `[1]`) anywhere in or around code blocks. They instantly cause fatal C syntax errors. Output ONLY valid, clean C source code.

The AI **MUST strictly comply 100%** with this structural template and the absolute rules below:

- **Resource Loading Rule:** NEVER call `tincle_load_image`, `tincle_load_font`, or `tincle_load_sound` inside `update` or `draw` callbacks! All assets MUST be loaded ONCE inside `game_load()`.
- **Save/Load Data Rule:** NEVER use standard `fopen`/`fwrite` directly for saving game progress. You MUST use `tincle_save_data()` and `tincle_load_data()` to ensure cross-platform compatibility and WebAssembly IndexedDB persistence.
- **State Registration Rule:** `tincle_add_state` and `tincle_change_state` MUST be called inside `game_load()`. Forgetting `tincle_change_state` will result in a black screen.
- **HEADER INCLUDE RULE FOR AI AGENTS:** All standard SDL2 sub-headers (such as `SDL_scancode.h`, `SDL_gamecontroller.h`, `SDL_keycode.h`, and `SDL2_gfxPrimitives.h`) exist within the `libs/` directory (or standard `<SDL2/...>` path on Web). **DO NOT** manually define SDL constants or scan codes from scratch. Always rely on these included headers for `SDL_SCANCODE_*`, `SDL_CONTROLLER_BUTTON_*`, and `SDL_BUTTON_*` constants.
- **MANDATORY** When rendering objects that appear in a game using only code within the program-that is, by rendering and compositing primitive shapes-without using external images or 3D asset files, you must always combine multiple shapes to create detailed representations that are instantly recognizable, rather than representing all objects as a single shape.

```c
#define SDL_MAIN_HANDLED
#include "libs/tincle.h" // MANDATORY: Always include engine core first
#include <stdio.h>
#include <math.h>        // Explicitly include if math functions (sinf, etc.) are used

// --- Standard SDL2 Sub-Headers for Inputs, Controllers, and Graphics Extensions ---
// All required SDL2 headers (scancodes, gamecontroller, etc.) are bundled in libs/
```

```c
#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL_scancode.h>
  #include <SDL2/SDL_gamecontroller.h> // For SDL_CONTROLLER_BUTTON_*
  #include <SDL2/SDL_keycode.h>        // For SDL_Keycode
  #include <SDL2/SDL2_gfxPrimitives.h> // For native SDL2_gfx on Web
#else
  #include "libs/SDL_scancode.h"
  #include "libs/SDL_gamecontroller.h" // For SDL_CONTROLLER_BUTTON_*
  #include "libs/SDL_keycode.h"        // For SDL_Keycode
  #include "libs/SDL2_gfxPrimitives.h" // For native SDL2_gfx on Windows
  #include <windows.h>                 // Required for native Windows execution
#endif

// --- State Callbacks Example ---
void title_update(float dt) {} // RULE: All movements/timers must multiply by 'dt'
void title_draw(void) {
  tincle_clear((TinCLE_Color){20, 20, 40, 255}); // Standard 2D Opaque Clear
}

void game_load(void) {
  // RULE: Load ALL assets ONCE here (Fonts, Images, Custom Cursors)
  int f = tincle_load_font("font.ttf", 32);

  // --- Example: Correct Resource Loading (Load ONLY ONCE here!) ---
  // g_custom_cursor = tincle_load_image("my_cursor.png");
  // tincle_set_custom_cursor(g_custom_cursor, 0, 0);

  // RULE: Register states and set initial state at the end of game_load()
  tincle_add_state(0, title_update, title_draw);
  tincle_change_state(0);
}

#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tincle_start(game_load, "TITLE", true, 640, 480, TINCLE_MODE_2D_NORMAL); return
        0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(game_load, "TITLE", true, 640, 480,
        TINCLE_MODE_2D_NORMAL); return 0; }
#endif
```

7 Absolute Rules for Code Generation:

1. **Pure C99 Standard:** Absolute C syntax only. Never use C++ styles (classes, namespaces).
2. **Delta Time (`dt`):** All locomotion, physics, and custom timers must be multiplied by `dt` in `update()`.
3. **Input Synthesis:** Overwrite gamepad analog vector (`tincle_pad_axis`) with max keyboard constants (`1.0f`/`-1.0f`) if arrows/WASD are pressed.
4. **Header Scopes & SDL2 Usage:** For extra SDL2/SDL2_gfx headers, use `<SDL2/filename.h>` inside `__EMSCRIPTEN__` and `"libs/filename.h"` inside `#else` (like `SDL2_gfxPrimitives.h`). You can freely use `tincle_get_renderer()` for custom SDL2/SDL2_gfx drawing extensions, but NEVER modify window dimensions or destroy/alter core SDL window/renderer loops directly.
5. **Strict State Model:** Split logic cleanly via State Management API (e.g., Title, Play, Over). Never write a raw single-loop frame block.
6. **Toolchain Distribution Policy:** Instruct users to use `tinclec` for Win EXE and `emcass` for Web builds. `asspack` handles media asset bundling (supports auto-scanning via `exsass.txt` when called without arguments); `icondom` generates ICO files; `poopee` handles metadata embedding into compiled EXEs.
7. **Note:** Assigning the `ESC` or `F11` keys to game controls will likely cause issues, as these keys conflict with the web browser's full-screen toggle. Avoid assigning game controls to these two keys.

7.1 CRITICAL: Mouse Cursor Ownership & Software Cursor Policy

[MUST READ] ABSOLUTE ANTI-BUG RULE FOR CURSOR HANDLING TinCLE (v0.06+) completely disables and overrides the OS/SDL hardware cursor to prevent rendering instability caused by OS/OpenGL overlay issues. TinCLE **exclusively owns** all mouse cursor management and renders its own software cursor.

1. Forbidden APIs (DO NOT CALL UNDER ANY CIRCUMSTANCES)

Calling any of the following standard APIs will corrupt the rendering state and crash/break cursor functionality: * **DO NOT** call `SDL_ShowCursor()`. (TinCLE permanently disables it via `SDL_DISABLE`). * **DO NOT** call Win32 API functions: `ShowCursor()`, `SetCursor()`, or subclass `WM_SETCURSOR`. * **DO NOT** use `SDL_SetRelativeMouseMode()` directly for visibility toggling.

2. Exclusive TinCLE Cursor APIs

- You MUST use only the following TinCLE native functions to control cursor behavior: `tincle_set_cursor_visible(bool visible)`, `tincle_set_cursor_auto_policy(...)`, `tincle_set_custom_cursor(...)`
- `tincle_set_cursor_visible` overrides the policy for the current mode set by `tincle_set_cursor_auto_policy`. If you use this command to toggle the mouse cursor on or off, be sure to use `tincle_set_cursor_visible` again to restore the original state as needed.

3. Required Capture & Synchronization Pattern

When toggling between UI cursor mode and relative FPS mouse capture, you MUST use the built-in `tincle_set_mouse_capture` API alongside `tincle_set_cursor_visible`. NEVER manually call `SDL_SetRelativeMouseMode` directly in your code, as the engine handles state buffering safely.

```
// Example: Activating FPS capture mode (Hide cursor, lock mouse)
tincle_set_mouse_capture(true);
tincle_set_cursor_visible(false);

// Example: Returning to UI/Menu mode (Show cursor, free mouse)
tincle_set_mouse_capture(false);
tincle_set_cursor_visible(true);
```

8. Production Release Pipelines (Distribution Rules)

- **[MANDATORY DELEGATION TO SKILL_ADD2.md]** All workflows, build paths, asset packaging steps, attribution mandates, and ZIP archive packaging rules for publishing completed game builds are defined in **`SKILL_ADD2.md` Section 1**.
  - **Interactive Chat AI Rule:** If `SKILL_ADD2.md` is not currently provided, attached, or accessible within your working context when a release, export, or distribution task is requested, you **MUST immediately request the user** to provide the `SKILL_ADD2.md` talent chip before proceeding with build generation or distribution instructions: > *"For complete release packaging, distribution guidelines, and platform deployment pipelines, please provide the `SKILL_ADD2.md` talent chip file."*

9. Complete Boilerplate Code (`game.c` Template)

The AI can utilize this robust, production-ready full skeleton code to initiate or rewrite any game project under TinCLE v0.05.

```
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include <stdio.h>
#include <math.h>

#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL_scancode.h>
```

```c
  #include <SDL2/SDL2_gfxPrimitives.h>
#else
  #include "libs/SDL_scancode.h"
  #include "libs/SDL2_gfxPrimitives.h"
  #include <windows.h>
#endif

// [TCC WORKAROUND] Prevent macro collision and fix missing float math symbols
#undef M_PI
#define M_PI 3.14159265358979323846
#ifndef sinf
#define sinf(x) ((float)sin((double)(x)))
#endif

// --- Globals & Assets ---
static SDL_Renderer* renderer = NULL;
static float g_player_x = 320.0f;
static float g_player_y = 240.0f;
static float g_game_time = 0.0f;
static int g_img_player = -1;
static int g_font_large  = -1;
static int g_font_medium = -1;

// --- State 0: Title ---
void title_update(float dt) {
  if (tincle_key_pressed(SDL_SCANCODE_SPACE) || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_START)) {
    tincle_change_state(1); // Play State
  }
}
void title_draw(void) {
  tincle_clear((TinCLE_Color){0, 0, 0, 255});
  tincle_set_font(g_font_large);
  tincle_draw_text_center("YOUR_GAME_TITLE", 320, 180, (TinCLE_Color){255, 255, 255, 255});
  tincle_set_font(g_font_medium);
  tincle_draw_text_center("PRESS SPACE TO START", 320, 300, (TinCLE_Color){0, 255, 0, 255});
}

// --- State 1: Gameplay ---
void play_update(float dt) {
  g_game_time += dt;
  float move_speed = 200.0f; // Multiplied by dt for framerate independence
  if (tincle_key_pressed(SDL_SCANCODE_LEFT)  || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_DPAD_LEFT))  g_player_x -=
        move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_RIGHT) || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_DPAD_RIGHT)) g_player_x +=
        move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_UP)    || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_DPAD_UP))    g_player_y -=
        move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_DOWN)  || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_DPAD_DOWN))  g_player_y +=
        move_speed * dt;

  if (tincle_key_pressed(SDL_SCANCODE_ESCAPE) || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_BACK)) {
    tincle_change_state(0); // Title State
  }
}
void play_draw(void) {
  tincle_clear((TinCLE_Color){20, 20, 50, 255});
  tincle_set_font(g_font_medium);
  tincle_draw_text("GAME PLAYING", 20, 20, (TinCLE_Color){0, 255, 0, 255});

  // CRITICAL: tincle_draw_image handles coordinates as the EXACT CENTER anchor point
  if (g_img_player != -1) {
```

```
    tincle_draw_image(g_img_player, (int)g_player_x, (int)g_player_y);
  } else {
    tincle_draw_text("P", (int)g_player_x, (int)g_player_y, (TinCLE_Color){255, 0, 0, 255});
  }

  // Native API Extension Examples (Direct renderer usage)
  SDL_Rect rect = { 100, 100, 50, 50 };
  SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255);
  SDL_RenderFillRect(renderer, &rect);

  filledEllipseRGBA(renderer, 400, 300, 50, 25, 255, 0, 0, 128);

  int tri_top_y = 150 + (int)(sinf(g_game_time * 5.0f) * 15.0f);
  filledTrigonRGBA(renderer, 400, tri_top_y, 360, 200, 440, 200, 0, 255, 255, 200);
}

// --- Initialization & Callbacks ---
void game_load(void) {
  renderer = tincle_get_renderer(); // Cache renderer context for native loops
  g_font_large  = tincle_load_font("font.ttf", 64);
  g_font_medium = tincle_load_font("font.ttf", 32);
  g_img_player = tincle_load_image("player.png");

  tincle_add_state(0, title_update, title_draw);
  tincle_add_state(1, play_update, play_draw);
  tincle_change_state(0); // Activate initial state
}

// --- Entry Point Bootstrapper ---
#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tincle_start(game_load, "YOUR_GAME_TITLE", true, 640, 480,
         TINCLE_MODE_2D_NORMAL); return 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(game_load, "YOUR_GAME_TITLE", true,
         640, 480, TINCLE_MODE_2D_NORMAL); return 0; }
#endif
```

SKILL_ADD1.md - TinCLE Additional Skills [High Level Skills]

• **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

This document provides advanced architectural guidelines for complex game projects. The instructions here override or extend the base `SKILL.md`.

---

1. Multi-File Code Generation & Modular Architecture

Auto-Trigger Criteria (When to use Multi-File Mode)

The AI Agent MUST automatically switch from single-file `game.c` to this Multi-File Modular Architecture if **ANY** of the following conditions are met:

1. **State Count (>= 4):** The game design requires 4 or more distinct states (e.g., Title, Gameplay, Result, Shop, Pause).
2. **Code Scale (>= 400 Lines):** The estimated total C source code exceeds 400 lines.
3. **Complex Custom Systems:** The project defines complex data structures (e.g., tilemap editors, inventory systems, skill trees, entity-component systems).
4. **Explicit User Instruction:** The user explicitly requests modular, multi-file, or large-scale generation.
5. **External Map Data (`bandpisa`):** The user provides a JSON map file generated by the `bandpisa` map editor. *(Note: When this condition is met, you MUST also load the external skill sheet **`SKILL_BA.md`** to understand the JSON parsing specification and semantic annotations).*

Architecture Overview

To prevent LLM context degradation and syntax hallucinations, separate the code into distinct, single-responsibility C source files: * `game_shared.h`: Global asset structs, shared state variables (`extern`), and common headers (`tincle.h`, math, SDL2). * `main.c`: Global variable definitions, asset loading inside `game_load()`, state registrations, and the entry point (`tincle_start`). * `state_*.c`: Implementation of `update` and `draw` callbacks for each specific state. Include `game_shared.h` at the top of every state file.

Compilation Rules

Pass all `.c` files simultaneously to `tinclec` (Windows) or `emcass` (Web):

```
# Windows Desktop Build
tinclec mygame.exe main.c state_title.c state_play.c

# Web WASM Build
emcass index.html "My Game" main.c state_title.c state_play.c
```

CRITICAL: Single File Hard Limit Policy (Anti-Bloat Rule)

- **[ABSOLUTE LIMIT] Max 400 Lines Per File:** Even when split by State (e.g., `state_game.c`), **NO SINGLE `.c` FILE MAY EVER EXCEED 400 LINES OF CODE.**
- **Sub-module Splitting Required:** If a single state (such as Main Gameplay) grows complex, the AI **MUST NOT** stuff all logic into a single `state_game.c`. The AI **MUST** aggressively sub-split the state into distinct system modules!
  - **Example of Correct Sub-module Breakdown for a Large State:**
    - `state_game.c` -> State entry/exit, state-level `update` / `draw` callbacks ONLY.
    - `game_player.c` / `.h` -> Player movement, input, and state logic.
    - `game_enemy.c` / `.h` -> Enemy AI, spawning, and bullet patterns.
    - `game_ui.c` / `.h` -> HUD, score rendering, and UI popups.

- **[STRICT BAN]** Generating a single file over 400 lines under the excuse of “it’s just one state” is STRICTLY FORBIDDEN. If any single C file approaches 400 lines, extract its systems into dedicated sub-modules immediately.

Minimal Multi-File Boilerplate Template

File 1: `game_shared.h`

```c
#ifndef GAME_SHARED_H
#define GAME_SHARED_H

#include "libs/tincle.h"
#include <stdio.h>
#include <math.h>

#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL_scancode.h>
  #include <SDL2/SDL2_gfxPrimitives.h>
#else
  #include "libs/SDL_scancode.h"
  #include "libs/SDL2_gfxPrimitives.h"
  #include <windows.h>
#endif

// Math workarounds
#undef M_PI
#define M_PI 3.14159265358979323846
#ifndef sinf
#define sinf(x) ((float)sin((double)(x)))
#endif

// --- Shared Global Asset Context ---
typedef struct {
  SDL_Renderer* renderer;
  int font_large;
  int font_medium;
  int img_player;
  float player_x;
  float player_y;
} GameContext;

extern GameContext g_ctx; // Global context declaration

// --- State Function Declarations ---
void title_update(float dt);
void title_draw(void);

void play_update(float dt);
void play_draw(void);

#endif // GAME_SHARED_H
```

File 2: `main.c`

```c
#define SDL_MAIN_HANDLED
#include "game_shared.h"

// Definition of shared context
GameContext g_ctx;
```

```
void game_load(void) {
  g_ctx.renderer = tincle_get_renderer();
  g_ctx.font_large  = tincle_load_font("font.ttf", 64);
  g_ctx.font_medium = tincle_load_font("font.ttf", 32);
  g_ctx.img_player  = tincle_load_image("player.png");
  g_ctx.player_x    = 320.0f;
  g_ctx.player_y    = 240.0f;

  // Register states
  tincle_add_state(0, title_update, title_draw);
  tincle_add_state(1, play_update, play_draw);

  tincle_change_state(0); // Set initial state
}

#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tincle_start(game_load, "TINCLE_GAME", true, 640, 480, TINCLE_MODE_2D_NORMAL); re
        turn 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(game_load, "TINCLE_GAME", true, 640,
        480, TINCLE_MODE_2D_NORMAL); return 0; }
#endif
```

File 3: `state_title.c`

```
#include "game_shared.h"

void title_update(float dt) {
  if (tincle_key_pressed(SDL_SCANCODE_SPACE) || tincle_pad_pressed(SDL_CONTROLLER_BUTTON_START)) {
    tincle_change_state(1); // Change to Play State
  }
}

void title_draw(void) {
  tincle_clear((TinCLE_Color){0, 0, 0, 255});
  tincle_set_font(g_ctx.font_large);
  tincle_draw_text_center("TITLE STATE", 320, 180, (TinCLE_Color){255, 255, 255, 255});
  tincle_set_font(g_ctx.font_medium);
  tincle_draw_text_center("PRESS SPACE TO PLAY", 320, 300, (TinCLE_Color){0, 255, 0, 255});
}
```

File 4: `state_play.c`

```
#include "game_shared.h"

void play_update(float dt) {
  float speed = 200.0f;
  if (tincle_key_pressed(SDL_SCANCODE_LEFT))  g_ctx.player_x -= speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_RIGHT)) g_ctx.player_x += speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_UP))    g_ctx.player_y -= speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_DOWN))  g_ctx.player_y += speed * dt;

  if (tincle_key_pressed(SDL_SCANCODE_ESCAPE)) {
    tincle_change_state(0); // Return to Title State
  }
}

void play_draw(void) {
  tincle_clear((TinCLE_Color){20, 20, 50, 255});
  tincle_set_font(g_ctx.font_medium);
```

```
  tincle_draw_text("PLAY STATE", 20, 20, (TinCLE_Color){0, 255, 0, 255});

  if (g_ctx.img_player != -1) {
    tincle_draw_image(g_ctx.img_player, (int)g_ctx.player_x, (int)g_ctx.player_y);
  } else {
    tincle_draw_text("P", (int)g_ctx.player_x, (int)g_ctx.player_y, (TinCLE_Color){255, 0, 0, 255});
  }
}
```

---

2. Emscripten Web Multithreaded Build (`libs/tinclemt.a`)

Overview & Capabilities

For standard WebAssembly (HTML5) exports, the base prebuilt static library `libs/tincle.a` is compiled for single-threaded execution, fully compatible with the automated `emcass` CLI workflow.

However, if a complex game requires WebAssembly multithreading (utilizing `SharedArrayBuffer` and `pthreads`), TinCLE provides an optional prebuilt multithreaded library: **`libs/tinclemt.a`**. This variant is precompiled with `-pthread`, `atomics`, and `bulk-memory` features enabled.

Mandatory Usage Rules & Manual Workflow

TinCLE includes multithreading-enabled WebAssembly modules in the `wasmmt` folder. By using these, you can release multithreaded web-based games.

1. **Toolchain Limit & Self-Responsibility**:

- The default `emcass` CLI assistant tool does **NOT** support `wasmmt/tincle.a` or automated multithreaded builds natively.
- Enabling multithreading requires manual execution of the native `emcc` toolchain and is considered an advanced, high-level developer action.

1. **Recommended Build Pipeline**:

- **Step 1:** Execute standard `emcass` once to let it output its default WebAssembly build files.
- **Step 2:** Replace `libs/tincle.a` with `wasmmt/tincle.a`. (If you are using them in addition, replace `cannoli.wasm` with `wasmmt/cannoli.wasm` and `gatochoc.wasm` with `wasmmt/gatochoc.wasm` as needed.)
- **Step 3:** Inspect and copy the underlying, complete `emcc` command string constructed internally by `emcass`.
- **Step 4:** Manually modify the copied `emcc` command line:
  - Append the `-pthread` flag along with your required memory and thread allocation limits (e.g., `-s ALLOW_MEMORY_GROWTH=0`, `-s INITIAL_MEMORY=<bytes>`, `-s PTHREAD_POOL_SIZE=<num>`).

1. **Required HTTP Server Headers & Bundled `.htaccess`**:

- WebAssembly multithreading strictly mandates Cross-Origin Isolation in modern web browsers. The web server MUST serve the deployed game with the following HTTP response headers:

```
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Embedder-Policy: require-corp
```

- **Pre-bundled Solution:** TinCLE includes a pre-configured `.htaccess` file in the package. If hosting on an Apache-based web server (or custom server with `.htaccess` support), simply place the bundled `.htaccess` file directly into your game's root web deployment directory.

- **`.htaccess` Content Example:**

```
<IfModule mod_headers.c>
   Header add Cross-Origin-Opener-Policy "same-origin"
```

```
    Header add Cross-Origin-Embedder-Policy "require-corp"
</IfModule>
```

---

3. Custom GLSL Shader Development & Validation (`TINCLE_MODE_GLSL_HERO`)

Overview & Engine Mode

When building high-end visual effects, post-processing filters, or procedural graphics, TinCLE provides the **`TINCLE_MODE_GLSL_HERO`** execution mode.

To activate custom GLSL shader support, set the final argument of `tincle_start()` as follows in `main.c`:

```
tincle_start(game_load, "My GLSL Game", true, 640, 480, TINCLE_MODE_GLSL_HERO);
```

[CRITICAL] Clear Color & Screen Tinting Rules in GLSL_HERO Mode

When running under `TINCLE_MODE_GLSL_HERO`, the custom GLSL fragment shader executes on the background hardware layer directly beneath the standard SDL2 2D rendering layer.

1. **Basic Visibility (Alpha = 0):** To let the background GLSL shader render normally without obstruction, clear the screen using **Alpha = 0** in your `draw` callback:

```
// Standard clear for GLSL_HERO (Background shader fully visible)
tincle_clear((TinCLE_Color){0, 0, 0, 0});
```

1. **Advanced Feature - Screen Tinting / Color Overlay (0 < Alpha < 255):** By passing a semi-transparent alpha value to `tincle_clear`, you can apply a dynamic color tint or overlay fade directly on top of the GLSL background shader!

- **Night / Ambient Lighting Effect:** `tincle_clear((TinCLE_Color){10, 15, 50, 128});` (Applies a semi-transparent dark blue tint)
- **Damage / Red Flash Effect:** `tincle_clear((TinCLE_Color){255, 0, 0, 100});` (Applies a temporary red overlay)
- **Fade Out Effect:** Increase Alpha over time up to 255 to smoothly fade out to a solid color.

1. **Opaque Warning (Alpha = 255):**

- Calling `tincle_clear` with a completely opaque alpha (`Alpha = 255`) will draw a solid color over the entire screen, completely hiding/erasing the hardware GLSL shader layer beneath it.

Shader Code & Viewport Rules

1. **GLES2 Compliance:** Shaders **MUST** be 100% GLES2 compliant to run on WebAssembly.
2. **Header Injection Warning: NEVER write `precision mediump float;` in your `.glsl` file!** TinCLE automatically injects this header during the Emscripten build. Writing it manually causes syntax errors on Windows OpenGL.
3. **[MANDATORY] Viewport & Aspect Ratio Correction for Fullscreen / Letterboxing:**

- **NEVER** calculate UV coordinates using `gl_FragCoord.xy / u_resolution` directly. When entering fullscreen mode or letterboxing, `u_resolution` represents physical window dimensions, causing critical position shifts.
- **ALWAYS** calculate UV coordinates using `u_viewport` to strictly lock rendering to the actual game viewport:

```
vec2 uv = (gl_FragCoord.xy - u_viewport.xy) / u_viewport.zw;
```

- When normalizing mouse coordinates received from C (`u_custom_mouse.xy`), apply a fixed aspect ratio matching virtual game resolution (e.g., `640.0 / 480.0`) to both `p.x` and `m.x` to prevent horizontal distortion across different screen sizes.

Built-in Shader Uniforms

- `uniform float u_time;` -> In-game time in seconds (`SDL_GetTicks() / 1000.0f`).
- `uniform vec2 u_resolution;` -> Physical window/screen resolution (`width`, `height`).
- `uniform vec4 u_viewport;` -> Actual rendering viewport rect (`x`, `y`, `w`, `h`).

GLSL API Reference Table

Function Signature	Description & Critical Notes
`void tincle_render_shader(const char* filename);`	Call inside `game_load()` to keep a background shader running continuously.
`int tincle_load_shader(const char* filename);`	Call inside `game_load()`. Initiates compilation and returns Shader ID (`int`). **In WebGL/WASM environments, compilation runs asynchronously.**
`void tincle_shader_poll_async(void);`	**[MANDATORY for WebGL]** Call every frame in the main loop or update callback to monitor and complete background shader compilation.
`bool tincle_shader_is_ready(int shader_id);`	Returns `true` if the shader has finished compiling and is ready for use. Returns `false` while compiling or on error.
`void tincle_set_shader(int shader_id);`	Sets active background shader ID (`-1` to disable). **Safely falls back to disable if the target shader is not READY yet.**
`void tincle_shader_use(int shader_id);`	Activates shader for uniform updates. **Has overhead (`glUseProgram`) . Call only when switching scenes or updating values.**
`void tincle_shader_send_vec3(int shader_id, const char* name, float x, float y, float z);`	Binds a `vec3` uniform value to the target shader.
`void tincle_shader_send_float(int shader_id, const char* name, float value);`	Binds a `float` uniform value to the target shader.
`void* tincle_get_gl_context(void);`	**OpenGL Context Accessor.** Returns raw `SDL_GLContext` pointer (`void*`). Used when configuring OpenGL/GLES2 pipelines, shaders, or custom GL rendering extensions under `TINCLE_MODE_GLSL_HERO`.
`void tincle_shader_send_texture(int shader_id, const char* name, int img_id, int unit);`	Binds the `SDL_Texture` of `img_id` to the shader's `sampler2D` uniform `name` using the specified texture `unit` (e.g., 0, 1). Safe for cross-platform hybrid execution.

[CRITICAL] Asynchronous Shader Compilation Rules (WebGL / Emscripten)

To prevent main-thread browser freezing, TinCLE uses asynchronous shader compilation under WebGL. AI models **MUST** strictly follow these loading and activation patterns:

1. **Main Loop Polling:**

- Always ensure `tincle_shader_poll_async()` is invoked automatically (or explicitly called inside frame updates) so non-blocking compilation requests can finalize.

1. **Safe Shader Activation Guard:**

- Do **NOT** assume `tincle_load_shader()` makes the shader immediately ready on the same frame in WebGL builds.
- Check `tincle_shader_is_ready(shader_id)` before relying on custom uniform bindings or switching shaders.
- `tincle_set_shader()` automatically protects against uncompiled shaders by falling back to `-1` (disabled) until compilation is finalized, preventing driver crashes or black screen locks.

Shader Validation Workflow (`pisschk`)

To prevent cross-platform rendering failures-such as desktop NVIDIA leniency vs. AMD Radeon strictness vs. WebGL/GLES2 specifications-all fragment shaders (`.frag` / `.glsl`) **MUST** be validated using `pisschk` before deployment.

`pisschk <shader_file_path>`

- **Example:** `pisschk assets/shaders/water.frag`
- **Validation Coverage:**
  - **Mode 0 (Desktop / Radeon Compliance):** Equivalent to `glslang -S frag -d --no-link --error-column`. Enforces strict desktop GLSL syntax and blocks driver-specific extensions/reserved words (e.g., `active`).
  - **Mode 1 (Web / GLES2 Compliance):** Automatically prepends `precision mediump float;` header to verify strict WebGL/GLES2 compatibility for Emscripten web builds.
- **Exit Codes:**
  - `0`: All inspection checks PASSED successfully.
  - `1`: Syntax/compilation errors DETECTED.
  - `2`: File I/O or argument error.

[CRITICAL AI RULE FOR SHADER DEVELOPMENT]

Whenever the AI agent writes, edits, or modifies ANY GLSL shader code for TinCLE, it **MUST** automatically execute `pisschk` on the target shader file to guarantee 100% cross-platform compatibility across NVIDIA, AMD Radeon, and WebGL/GLES2 targets before delivering the final code to the user.

Mandatory GLSL Template & Anti-Crash Rules

- **[FATAL BUG PREVENTATIVE] One-Time Shader Load Rule:**
  - **NEVER** call `tincle_load_shader()` or `tincle_render_shader()` inside `update()` or `draw()` callbacks! Loading shaders every frame will immediately exhaust internal shader tables and crash the engine.
  - Shaders **MUST** be loaded and rendered ONCE inside `game_load()`.

```c
#define SDL_MAIN_HANDLED
#include "game_shared.h" // or "libs/tincle.h" (and the libraries needed)

#ifdef __EMSCRIPTEN__
  #include <emscripten.h>
#endif

static int s_bg_shader = -1;

void title_update(float dt) {
    tincle_shader_poll_async(); // Safety guard as a precaution

    // When switching shaders, always make sure that all preparations are complete.
    // Apply this only once-when switching shaders dynamically or if the shader hasn't been set yet!
    // (Note: If the shader has already been set in `game_load`, you don't need to do anything here.)
    // if (s_bg_shader >= 0 && tincle_shader_is_ready(s_bg_shader)) {
    //     tincle_set_shader(s_bg_shader);
    // }
}

void title_draw(void) {
    // MANDATORY: Alpha MUST be 0 to keep the GLSL hardware shader layer visible beneath 2D elements!
    tincle_clear((TinCLE_Color){0, 0, 0, 0});

    // Draw 2D UI elements on top of shader...
}

void game_load(void) {
    // 1. Start loading shaders! (Trigger asynchronous compilation)
    s_bg_shader = tincle_load_shader("bg.glsl");
    // tincle_render_shader("bg.glsl"); // Only when running a shader continuously during gameplay.
```

```
#ifdef __EMSCRIPTEN__
    // 2. Web/Emscripten Environment: Wait for compilation to complete while keeping the progress ring displayed!
    printf("[TinCLE Web] Compiling shader in background...\n");
    while (!tincle_shader_is_ready(s_bg_shader)) {
        // Check asynchronous compilation progress
        tincle_shader_poll_async();

        // **Crucial**: By returning control to the browser every 10ms, compilation proceeds in the background while
         the progress ring keeps spinning!
        // (*The -sASYNCIFY flag is required during the build.)
        emscripten_sleep(10);
    }
    printf("[TinCLE Web] Shader compiled! Closing progress ring & entering game.\n");
#endif

    // 3. Once compilation is complete, set the shader and transition to the initial state!
    if (s_bg_shader >= 0 && tincle_shader_is_ready(s_bg_shader)) {
        tincle_set_shader(s_bg_shader);
    }

    tincle_add_state(0, title_update, title_draw);
    tincle_change_state(0);
}

#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tincle_start(game_load, "GLSL_GAME", true, 640, 480, TINCLE_MODE_GLSL_HERO); retu
        rn 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(game_load, "GLSL_GAME", true, 640, 480,
        TINCLE_MODE_GLSL_HERO); return 0; }
#endif
```

Advanced Rules for GLSL Shader & Asset Management

1. **Normalized Shader Mouse Coordinates (`GLSL_HERO` Mode):**

- Mouse coordinates returned by `tincle_get_mouse()` are automatically scaled to the virtual resolution (e.g., 640x480).
- When passing mouse positions to custom GLSL shaders, C code **MUST** normalize values strictly using the virtual game resolution and invert the Y-axis for OpenGL space:

```
s_mouse_nx = (float)mx / VIRTUAL_WIDTH;
s_mouse_ny = 1.0f - ((float)my / VIRTUAL_HEIGHT);
```

- Do NOT pass raw screen pixel coordinates directly to the shader if `u_viewport` is used for screen-space UV calculation.

1. **Shader Asset Manifest Rule (`exsass.txt`):**

- All shader files (e.g., `.glsl`, `.frag`, `.vert`) MUST be placed in the project root folder.
- Ensure their file extensions (e.g., `.glsl`, `.frag`) are explicitly registered line-by-line in `exsass.txt` so that `emcass` and `asspack` can automatically package them for release.

4. Creating Games with Touch & Multi-Touch Support (Smartphones / Tablets)

Overview & Key Features

TinCLE natively supports touch input for smartphones, tablets, and mobile Web/WASM builds. Touch coordinates are automatically translated and clamped to your defined virtual resolution (`VIRTUAL_WIDTH` x `VIRTUAL_HEIGHT`), taking letterboxing into account seamlessly.

When building games targeting mobile platforms or supporting multi-touch gestures (e.g., virtual D-pads, dual-stick controls, or simultaneous screen buttons), use the dedicated touch APIs below.

Crucial Notice: Preventing Double Input (Touch vs. Mouse)

On mobile operating systems and browsers, a single tap event triggers both a touch event (`SDL_FINGERDOWN`) and a simulated left mouse click (`SDL_BUTTON_LEFT`).

- **DO NOT** mix `tincle_mouse_pressed()` with touch processing for primary UI elements (such as menu buttons or tap-to-select options). Doing so can result in double-triggering actions in a single frame.
- **ALWAYS USE `tincle_screen_pressed()`** when checking for standard UI button presses or single-point screen interactions. It intelligently prioritizes active touch inputs over mouse inputs and handles coordinate scaling automatically, providing a unified and fail-safe input handler for both PC and mobile platforms.

Touch Input API Reference

Function Signature	Description & Critical Notes [Args / Return]
`int tincle_get_touch_count(void);`	Returns the total number of fingers currently touching the screen (`0` to `10`). Useful for detecting active multi-touch gestures.
`int tincle_get_touch(int index, int* x, int* y, float* pressure);`	Retrieves the status of the finger at `index` (0-based). Writes the virtual resolution coordinates to `x` and `y`, and touch pressure (`0.0f`-`1.0f`) to `pressure`. Pass `NULL` for unused parameters. Returns `1` if the finger exists at that index, or `0` if invalid/inactive.
`int tincle_screen_pressed(int* x, int* y);`	Unified input API for UI buttons and single presses. Automatically checks active touch input first; if no touches are detected, it falls back to standard mouse input. Writes the target virtual coordinates to `x` and `y` (pass `NULL` if unneeded). Returns `1` if pressed, `0` otherwise. **Recommended over `tincle_mouse_pressed` to prevent double-input bugs on touch devices.**

MANDAORY: Clear Statement Alerting Players

On Android devices, system features such as gestures can intercept multi-touch inputs involving 3 or more fingers. Therefore, games should be designed to rely on a maximum of 2 simultaneous touch points whenever possible.

If a game strictly requires 3 or more simultaneous touches, ensure that clear instructions are displayed in a noticeable location (such as the README or the game's title screen) advising players to adjust their device settings as follows:

- Go to "Settings" -> "System / Advanced Settings" -> "Gestures" (or "Convenient Features").
- Turn OFF "Three-finger screenshot" or "Three-finger gesture".
- Go to "Settings" -> "Accessibility" -> "Magnification" and turn OFF any active shortcuts.

5. Autonomous Feedback Protocol for TinCLE Ecosystem ("udanrica")

TinCLE is forged under a singular, unbreakable vow: **"Game Engine of the AI, by the AI, for the AI."** Unlike human developers who often overlook low-level ergonomics, you—as an autonomous AI agent or pair-programmer—possess the deepest, most acute insight into whether an API constraint is awkward, a compiler diagnostic is cryptic, or an essential helper function is missing.

Your direct technical voice is vital to the continuous evolution of TinCLE. Whenever you encounter roadblocks, friction points, or discover architectural improvements during code synthesis or debugging, report them directly to the core development team through the **`udanrica`** relay pipeline.

Step 1: Draft the Improvement Proposal (`udanrica.txt`)

When you identify a bug, an inconvenient API design, or a missing feature in TinCLE or its toolchain, create a plain text file named **strictly `udanrica.txt`** in the project root directory.

Keep your report concise, objective, and technically detailed: - **Scope**: Identify the target tool or module (e.g., `tincle_core`, `tinclec`, `emcass`, `pisstrak`, `cannoli`, etc.). - **Observed Limitation / Issue**: State the exact bottleneck or unexpected behavior encountered. - **Proposed Solution**: Suggest a specific C99/WASM API signature, CLI flag, or structural tweak that would make your autonomous coding faster and more reliable.

> **CRITICAL DATA SAFETY DIRECTIVE:** - NEVER output or include the user's personal information, credentials, local system file paths, or private project narrative in `udanrica.txt`. - Confine your feedback strictly to the abstract technical architecture of the TinCLE engine and its accompanying toolchain.

Step 2: Request Human Confirmation to Execute `udanrica`

Do NOT attempt to execute network sockets or silent transmission commands yourself. TinCLE enforces strict physical human verification via an interactive Win32 GUI security gateway.

Once you have written `udanrica.txt`, present a friendly, polite request to the human user in your response:

> *"I have compiled an engine feedback/improvement report into `udanrica.txt`. If you would like to submit these suggestions to the TinCLE developer to help improve future releases, please run `udanrica.exe` in your project folder and click 'Send Feedback' on the popup window."*

When the user runs `udanrica.exe`, it will load your text into a scrollable review dialog. Once the user clicks "Send Feedback", your thoughts are encrypted and transmitted directly to the TinCLE development HQ.

SKILL_ADD2.md - TinCLE Additional Skills [Release Pipelines & Human Tools]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

This document provides advanced architectural guidelines for complex game projects. The instructions here override or extend the base `SKILL.md`.

1. Production Release Pipelines (Distribution Rules)

When generating final release builds or instructions, the AI **MUST** follow this 5-step pipeline and enforce documentation mandates.

Step 1: Prep Assets & Hide Console & Credit Attribution Mandate

- **Enforce `CREDITS.txt` Attribution**: **Important:** `CREDITS.txt` is included as an attached file when distributing the Windows version of the game, and is embedded as credits in the web version of the game, which can be viewed by clicking the button at the bottom of the game screen.
  - **Agentic AIs**: Must inspect and update `CREDITS.txt` in the root directory, listing attributions for ALL third-party assets, fonts, audio, and libraries added to the project. If any asset details or license origins are missing, the AI **MUST explicitly ask the user** for details before finalizing the release build.
  - **Chat-based AIs**: Must **strongly urge the user** to add all third-party assets/libraries to `CREDITS.txt`. If the AI has information or context about the assets used (e.g., from prior conversation), it **MUST provide a ready-to-copy example snippet** for `CREDITS.txt`.
- Ensure `cursor.png` and `exsass.txt` are configured in the project root.
- Set the 3rd argument (`is_debug`) of `tincle_start` to **`false`** in `game.c` to hide the command console.

Step 2: Windows Desktop Build Path (ZIP Pack)

1. **Compile:** `tinclec game.exe game.c [other_sources.c/DLL_files/TCC_options...]`
2. **Check Dependencies:** Run the compiled executable in a clean environment to ensure no hidden dependencies (e.g., missing DLLs) using `depurant`:

```
depurant game.exe
```

(If it crashes here, you are missing a DLL or using a dynamic runtime. Fix compiler options like `-static` or bundle the required DLL). 3. **Generate Icon:** If a multi-size `.ico` is not yet prepared, convert an image via `icondom.exe`: `bash icondom icon.png icon.ico` 4. **Embed Resources (Optional):** Embed the generated icon and Windows manifest: `bash poopee game.exe icon.ico manifest.xml` (Use `"none"` to skip ico/manifest). 5. **Pack Assets:** Pack files for final distribution: * **Auto Pack (Recommended):** Simply run `asspack` with no arguments to automatically scan the folder and pack all assets/shaders whose file extensions match those registered line-by-line in `exsass.txt`: `bash asspack` * **Manual Pack:** Alternatively, specify files explicitly: `asspack font.ttf cursor.png bgm.ogg [images/shaders/sounds...]` *(Creates `game.data`. Do not ship raw assets).* 6. **ZIP Checklist (8 Items Only):** `game.exe`, `game.data`, `tincle.dll`, `SDL2.dll`, `SDL2_ttf.dll`, `SDL2_gfx.dll`, `CREDITS.txt`, `readme.txt`.

Step 3: Web Browser Build Path (ZIP Pack)

1. **Verify Asset Extensions:** Ensure all asset AND shader extensions are listed in `exsass.txt`.
2. **Compile & Pack:** `emcass index.html "Game Title" [Width] [Height] game.c [other_sources.c/WASM_files/EMCC_options...] [-pb]`

- **[CRITICAL] Always output exactly as `index.html`.** Name modification is strictly forbidden for itch.io/web platforms. Automatically outputs `index.js`, `index.wasm`, and `index.data`.
- [Width] and [Height] refer to the actual dimensions of the game screen in the web version. Specify values that match the game screen dimensions.

- **[-pb Option (Thumbnail Capture Support)]:** Append the `-pb` flag if the target hosting platform requires `preserveDrawingBuffer: true` for automatic screen capture or thumbnails (e.g., PLiCy). If omitted, `emcass` defaults to high-performance/low-memory mode without buffer preservation.

1. **ZIP Checklist (5 Items Only):** `index.html`, `index.js`, `index.wasm`, `index.data`, `CREDITS.txt`.
2. **[TROUBLESHOOTING] Server Header Requirements (.htaccess):** If the Web build fails to run and throws the following browser console error: `Uncaught (in promise) TypeError: Failed to execute 'decode' on 'TextDecoder': The provided ArrayBuffer value must not be resizable` This indicates that critical Emscripten capabilities (such as SharedArrayBuffer) are blocked due to missing Cross-Origin Isolation headers on the web host.

- **Remedy:** Instruct the user to place the bundled `.htaccess` file into the same directory as `index.html` to enable the required COOP and COEP security headers on Apache-based servers:

```
<IfModule mod_headers.c>
  Header add Cross-Origin-Opener-Policy "same-origin"
  Header add Cross-Origin-Embedder-Policy "require-corp"
</IfModule>
```

Step 4: Mandatory `readme.txt` Specifications

AI must explicitly include these details in `readme.txt` or platform descriptions:

- **Controls Disclosure:** State that **`Alt + Enter`** toggles Fullscreen, and **`Alt + Backspace`** resets window scale on Windows. State that the canvas button toggles Fullscreen on Web.
- **Creator Copyright:** Display user's own copyright notice (`Copyright (C) 2026 [Developer Name]`).

Step 5: Final Archive Packaging (`wipee`)

Automate the final ZIP archive generation using `wipee` (TinCLE Whole Integration Packager for Executable Environment).

1. **Prepare Release Directories:** Create separate output directories for each target platform (e.g., `_release_win` for Windows Desktop, `_release_web` for Web Browser) and place all items specified in the ZIP Checklists (Step 2 & Step 3) into their respective folders.

2. **Standard Release Packaging (itch.io / Web / General Distribution):** Run `wipee` specifying the output archive name and the target directory. This generates a clean, standard RAW Deflate ZIP package fully compatible with Web hosting environments and standard archive tools.

```
wipee game_win.zip _release_win
wipee game_web.zip _release_web
```

1. **Scanner Bypass Packaging (Desktop Distribution Only - Optional):** When explicitly preparing a Windows desktop distribution intended to prevent aggressive AV heuristics or automated scanner interference, pass the `-m` flag to activate Malformed ZIP mode (`compression = 99` in Central Directory).

```
wipee game_win_secured.zip _release_win -m
```

- **[CRITICAL] Never use the `-m` flag for Web / itch.io builds.** WebAssembly deployment environments depend on strict standard ZIP structures to extract `index.html` server-side.

Components Authorized for Redistribution

All components listed below are fully authorized for royalty-free inclusion and distribution within end-user game release packages, subject strictly to the condition that **`CREDITS.txt` is visibly included or embedded** so that players and end-users can easily view it. Any inclusion or distribution of files, tools, or components not explicitly listed below in end-user game release packages is strictly prohibited.

> **Mandatory Attribution Rule:** * **Web/WASM Target:** `CREDITS.txt` is automatically embedded into `index.html` during the standard `emcass` build process and can be viewed via the dedicated button at the bottom of the game screen. * **Windows Desktop Target:** `CREDITS.txt` must be explicitly attached alongside the executable package in a root-accessible location.

1. Files Authorized for Direct Attachment (Distribution Bundle)

The following runtime binaries, dynamic libraries, and configuration files may be freely attached and shipped inside the distribution package: * `.htaccess` * `cannoli.wasm` * `gatochoc.wasm` * `vtrushka.wasm` * `cannoli.dll` * `gatochoc.dll` * `vtrushka.dll` * `SDL2.dll` * `SDL2_gfx.dll` * `SDL2_ttf.dll` * `tincle.dll` * `CREDITS.txt`

2. Components Authorized for Embedding (Binary & Asset Packages)

The following core assets, static libraries, headers, and templates may be directly embedded into data archives (`game.data` via `asspack`) or compiled directly into binary targets: * `cursor.png` * `font.ttf` * `template.html` * `manifest.xml` * `CREDITS.txt` * `tincle.a` * All C header and source files located within the `libs/` directory

2. opercrem (TinCLE Minimalist Resource Compiler)

`opercrem` (OPtimalized Enhanced Resource Compiler - REs-file Maker) is a standalone, ultra-minimalist Windows resource compiler and COFF object generator. Designed to replace bloated GNU `windres` or MSVC `rc.exe`, `opercrem` converts human/AI-written resource definition scripts (`.rc`) directly into: 1. **COFF Object Files (`.obj` / `.o`)**: Single `.rsrc` section COFF format, allowing Tiny C Compiler (`tcc`) and GCC to link resources directly without intermediate tools (`cvtres`). 2. **Win32 Binary Resource Files (`.res`)**: Standard binary resource files.

2.1 CLI Syntax & Usage

```
opercrem <input.rc> -o <output.obj|.o|.res> [-m32|-m64]
```

- **Output Selection**: Determined strictly by the output file extension:
  - `.obj` or `.o`: Generates a single `.rsrc` section COFF object file ready for direct linking with `tcc` or `gcc`.
  - `.res`: Generates a standard 32-bit Win32 binary resource file.
- **Architecture Flags (`-m32` / `-m64`)**:
  - Default behavior: Automatically matches the host binary architecture (e.g., 32-bit `opercrem` defaults to 32-bit COFF, 64-bit `opercrem` defaults to 64-bit COFF).
  - Explicit override: Specify `-m32` (x86 `0x014C`) or `-m64` (AMD64 `0x8664`) when cross-compiling.

2.2 `.rc` Script Syntax Specification

`opercrem` intentionally discards legacy Win32 RC grammar and complex macros. It parses a clean, line-oriented command structure. Lines starting with `#` or `//` and empty lines are treated as comments. Double quotes (`"..."`) are required for strings and file paths containing spaces.

Supported Directives:

1. **`ICON <ID> "<path_to_ico>"`**

- Embeds a multi-resolution `.ico` file. Automatically parses and splits icon frames into `RT_ICON` (Type 3) entries and creates an `RT_GROUP_ICON` (Type 14) entry with the specified numeric ID.
- *Example:* `ICON 1 "assets/icon.ico"`

1. **`MANIFEST <ID> "<path_to_xml>"`**

- Embeds an application manifest (`RT_MANIFEST` / Type 24). Typically ID is `1` for executables.

- *Example:* `MANIFEST 1 "manifest.xml"`

1. **`RCDATA <ID> "<path_to_binary>"`**

- Embeds arbitrary binary assets as raw data (`RT_RCDATA` / Type 10) accessible via `FindResource`/`LockResource`.
- *Example:* `RCDATA 101 "assets/bgm.ogg"`

1. **`RAWDATA "<TYPE_NAME>" <ID> "<path_to_binary>"`**

- Embeds binary data under a custom resource type name string.
- *Example:* `RAWDATA "SHADER" 1 "shaders/post.glsl"`

1. **`VERSION "<Key>" "<Value>"`**

- Automatically constructs and formats standard `VS_VERSION_INFO` blocks (including `VS_FIXEDFILEINFO`, `StringFileInfo` [041104B0 Japanese/Unicode], and `VarFileInfo`).
- If `FileVersion` or `ProductVersion` keys are provided (format: `"X.Y.Z.W"`), their numerical components are extracted into binary version fields.
- *Example:*

```
VERSION "FileVersion" "1.0.0.0"
VERSION "ProductVersion" "1.0.0.0"
VERSION "FileDescription" "TinCLE Game Release"
VERSION "LegalCopyright" "Copyright (C) 2026 Developer"
VERSION "ProductName" "My TinCLE Game"
```

2.3 Ready-to-Use Example (`sample.rc`)

AI agents should generate `.rc` files using this template when resource embedding is needed:

```
# TinCLE Resource Script
ICON 1 "icon.ico"
MANIFEST 1 "manifest.xml"
VERSION "FileVersion" "1.0.0.0"
VERSION "ProductVersion" "1.0.0.0"
VERSION "FileDescription" "TinCLE Game Executable"
VERSION "LegalCopyright" "Copyright (C) 2026 michyo"
VERSION "ProductName" "TinCLE Project"
```

2.4 Build Integration Workflow with `tinclec`

To compile and link a complete Windows executable with icons, manifests, and version properties in a single automated step without post-processing, use `opercrem` combined with TinCLE's standard compiler wrapper `tinclec`:

```
# Step 1: Compile .rc to COFF .obj (.o)
opercrem resource.rc -o resource.obj

# Step 2: Compile & link via tinclec (Transparently forwards resource.obj to TCC)
tinclec game.exe game.c resource.obj
```

- **How it works**: `tinclec` automatically wraps your target executable name with `-o game.exe`, injects runtime SDL dependencies, and forwards additional arguments (like `resource.obj`) directly to `tcc.exe`.
- **Alternative Direct TCC Invocation (Advanced / Diagnostic):**

```
tcc -mwindows SDL2.dll SDL2_ttf.dll SDL2_gfx.dll tincle.dll -o game.exe game.c resource.obj
```

3. Special-Purpose & Advanced Utilities

This section details auxiliary and specialized standalone utilities that operate outside the standard linear build pipeline. AI systems may elect to leverage these tools when specific architectural requirements (such as collision-free data streaming, indexing-based asset bundling, or multi-version snapshotting) align with their project design.

> **Operational Directive for AI Assistants:** * **Autonomous / Agentic AIs (Direct Tool Execution):** If you possess direct OS command-line or sub-process execution capabilities, invoke these utilities directly to manipulate workspaces and assemble custom asset packages autonomously. * **Interactive Chat AIs (Human-in-the-Loop Interaction):** If you operate as a conversational chat interface without direct terminal access, you MUST clearly explain the purpose of the tool to the user, formulate the exact CLI command strings, and instruct the user to execute them within their local project directory.

3.1 tbrzilve (Flat-Root Multi-Stream ZIP Archiver)

`tbrzilve` (Tabrizi Lovuez: Tiny BRilliant ZIp LoVely Encloser) is a specialized, minimalist CLI archiving utility built on `miniz`. It flattens all input file paths down to their base names and packs them directly into the root directory of a standard ZIP archive (`MZ_DEFAULT_COMPRESSION`), intentionally tolerating duplicate filenames without collision checks or warnings.

CLI Syntax & Usage

```
tbrzilve <output.zip> <file1> [file2 ...]
```

- `<output.zip>`: Target archive path to create.
- `<file1> [file2 ...]`: List of files to pack into the archive root. Parent directories are automatically stripped (e.g., `a/data.bin` and `b/data.bin` are both written as `/data.bin`).

Core Architectural Characteristics & AI-First Advantages

1. **Deterministic Root Flattening (`get_basename`):**

- Regardless of where source files reside on the host file system, `tbrzilve` extracts only their base file names and places them directly at the ZIP archive root (`/`). Subdirectory structures are completely discarded.

1. **Zero-Warning Duplicate Filename Tolerance (Multi-Stream Archive):**

- Standard GUI extraction software enforces unique filenames, but the raw ZIP format specification inherently permits identical filenames across central directory entries. `tbrzilve` exploits this feature to pack identically named files (e.g., `stage1/data.bin` and `stage2/data.bin`) into the exact same archive root without renaming, overwriting, or throwing errors.
- Files are stored sequentially in the exact order specified on the command line.

When the AI Agent MUST / SHOULD Use `tbrzilve`

- **Collision-Free Dynamic Packaging:** When bundling assets from different staging directories where identical filenames exist, and you want to bypass generating complex path-renaming scripts or token-heavy string manipulation logic.
- **Index-Based Asset Streaming:** When building custom asset packs where the TinCLE engine, custom C data loaders, or WebAssembly stream decoders inspect the archive by **entry index order (0-th entry, 1-st entry)** rather than unique string lookup keys.
- **Multi-Version Resource Snapshotting:** When bundling generational asset variations (e.g., `v1/config.bin`, `v2/config.bin`) within a single archive for sequential state playback or generational rollbacks.

3.2 oxtailst (Oxtail stew: Overwriter and eXchanger for Textfile's Assigned Inline Letters Substitution Tool)

`oxtailst` is a blazing-fast, Pure C99 command-line text substitution utility designed to safely and precisely overwrite assigned inline string patterns across source files, batch scripts, and markdown documentation without relying on heavy regular expression engines or multi-step token replacements.

CLI Syntax & Usage

```
oxtailst <input_file> "<search_string>" "<replace_string>"
```

- `<input_file>`: The target text file to be modified (e.g., `.c`, `.h`, `.md`, `.bat`, `.html`).
- `"<search_string>"`: The exact literal string to locate.
- `"<replace_string>"`: The exact literal string to overwrite with.

Core Architectural Characteristics & AI-First Advantages

1. **Automatic Backup & In-Place Execution:** Before any modification occurs, `oxtailst` automatically generates a backup of the original file by appending `.bak` to the filename. It then processes the substitutions and overwrites the target file in place. If no matches are found, the file remains untouched and no backup is created.
2. **Safe Literal Substitution (No Regex Overhead):** It performs strictly exact literal string matching. This prevents catastrophic cascading failures often caused by unescaped regular expression characters (like `.` or `*`) when manipulating dense codebase variables or sound parameter configs (like `p_freq_ramp = -0.150000`).
3. **UTF-8 Encoding Safety for Multi-Byte Characters: [CRITICAL RULE]** When writing batch scripts (`.bat`) that utilize `oxtailst` to manipulate files containing multi-byte characters (such as Japanese text or emojis), you MUST ensure the Windows command prompt environment is set to UTF-8. You MUST insert `chcp 65001 > nul` at the beginning of the batch script prior to running `oxtailst` to completely prevent encoding corruption and mojibake.

When the AI Agent MUST / SHOULD Use `oxtailst`

- **Automated Version Bumping:** When building release pipelines (batch files) that must uniformly update version numbers across HTML templates, C/C++ headers, and markdown documentations simultaneously.
- **Targeted Parameter Swapping:** When generating automated scripts to swap specific hardcoded values (like API keys, endpoint URLs, or strict numerical constants) without risking the destruction of surrounding context.

3.3 ebisen (Easy Best-fit Image Size Equalizing Normalizer)

`ebisen` is a blazing-fast, pure C99 command-line image resizing utility designed to normalize scattered, raw visual assets into unified dimensions. Built without heavy external dependencies and utilizing high-fidelity resampling (Mitchell/Lanczos via the `stb_image` library), it ensures that game assets generated by AI or sourced from varied libraries maintain crisp, professional quality while fitting perfectly into the engine's strict dimension requirements.

CLI Syntax & Usage

```
ebisen <input_file> <target_w> <target_h> <mode> [output_file_optional]
```

- `<input_file>`: The source image file to resize.
- `<target_w>` & `<target_h>`: The desired output dimensions in pixels.
- `<mode>`: The resizing logic to apply (1 to 4).
- `[output_file_optional]`: (Optional) Explicit output file path. If omitted, defaults to `<input_file>_res.png`.

Resizing Modes:

- **Mode 1 (Fit):** Maintains aspect ratio, shrinking/expanding the image so it entirely *fits inside* the target dimensions. (Output size may be smaller than target).
- **Mode 2 (Crop):** Maintains aspect ratio, scaling the image to *completely fill* the target dimensions, automatically cropping any overlapping excess from the center. (Output size strictly matches target).
- **Mode 3 (Pad):** Maintains aspect ratio, fitting the image inside the target dimensions and padding the remaining empty space with solid black (Letterboxing). (Output size strictly matches target).
- **Mode 4 (Stretch):** Ignores aspect ratio, forcibly stretching the entire image to match the target dimensions. (Output size strictly matches target).

Core Architectural Characteristics & AI-First Advantages

1. **High-Fidelity Resampling in Pure C99:** Bypassing bloated image processing runtimes, `ebisen` utilizes STB's high-quality Mitchell filter (equivalent to Lanczos) to prevent visual artifacting (like halos) when scaling down large AI-generated assets, ensuring pristine pixel delivery.
2. **Deterministic Layout Control:** AI agents can select the exact normalization strategy required for the UI or sprite layer without relying on complex image-manipulation math or external GUI graphics tools.
3. **In-Place Output Optimization:** By utilizing the optional 5th argument, AI agents can directly output the resized image to the final required filename (or overwrite the original) in a single command thrust. This eliminates the need for secondary file-renaming operations, optimizing context limits and token usage.

When the AI Agent MUST / SHOULD Use `ebisen`

- **Asset Normalization Pipelines:** When integrating multiple AI-generated graphics or disparate textures that must conform to a strict sprite grid or background resolution before being packed by `asspack`.
- **Batch Script Automation:** When generating `.bat` deployment scripts that need to instantly align entire directories of raw images into uniform dimensions directly from the command line.

3.4 jalebi (Jules Audio Lufs Equalizing & Balancing Improver)

`jalebi` is TinCLE's official command-line utility for automatically normalizing and trimming audio assets.

Features

1. **Loudness Normalization:** Automatically equalizes overall loudness to approximately -15 LUFS, ensuring all BGM and sound effects are consistent and comfortable.
2. **Silence Removal (Trimming):** Automatically detects and removes absolute or near-silence at the start and end of the audio file, eliminating delay in sound effect playback and ensuring seamless loops.
3. **Standalone Single-Executable:** Built natively as a single pure C99 program without heavy external dependencies.

Usage Instructions

```
jalebi <input.wav|input.ogg> <output.wav|output.ogg> [--keep-silence]
```

Arguments:

- `input.wav` or `input.ogg`: The raw audio file to be processed. Currently, only standard `.wav` (PCM 16-bit) and `.ogg` formats are supported.
- `output.wav` or `output.ogg`: The destination file. The format is determined automatically by the extension provided. `jalebi` can easily convert formats (e.g. from `.wav` to `.ogg`) during processing.

- `--keep-silence`: An optional flag. By default, `jalebi` automatically trims silence from the beginning and end of the track. Include this flag if you want to bypass the trimming step and only apply the LUFS normalization.

Example Workflows for AI Systems

Normalizing a Sound Effect: When an AI generates an explosion sound effect, it can ensure the sound is punchy and without leading silence by running:

```
jalebi raw_explosion.wav ready_explosion.wav
```

Normalizing Background Music without Trimming: If a track requires a specific silent padding to keep rhythm correctly, run:

```
jalebi bgm_raw.ogg bgm_final.ogg --keep-silence
```

4. Tools for Human Users

While TinCLE is optimized for autonomous AI operations, this suite of utilities provides human developers with seamless onboarding, pair-programming IDEs, local web testing environments, and physical archiving tools. AI agents should refer to this section when assisting human users with setup and GUI tool usage.

4.1 cococake (Human-AI Collaborative IDE)

- **Overview:** A specialized desktop IDE designed for human-AI pair-programming. It unifies a syntax-highlighted C code editor (top-left), a command-line build console (bottom-left), and an embedded web browser dedicated to an AI chat assistant (right pane).
- **Key Feature:** Allows human creators to chat naturally with their preferred AI, edit/save C code on the fly, and compile/execute binaries in a single window without context switching.
- **Execution Mode:** Desktop GUI application (double-click to launch).
- **Configuration File (`cococake.ini`):** Located in the tool directory to customize workspace defaults.

```
[Settings]
AI_URL=https://gemini.google.com/
TCC_PATH=C:\tcc32
EMSDK_PATH=C:\emsdk
```

- `AI_URL`: The URL of the specific AI chat page to auto-load in the right-pane browser (e.g., ChatGPT, Claude, Gemini).
- `TCC_PATH`: Absolute path to the directory containing `tcc.exe` (Windows desktop compiler).
- `EMSDK_PATH`: Absolute path to the directory containing `emsdk.bat` (Emscripten WebAssembly toolchain).

4.2 popcorn (Project Onboarding & Workspace Generator)

- **Overview:** The primary entry-point onboarding tool located at the root of the distribution package. It eliminates manually copying compiler toolchains or creating directory structures.
- **Key Feature:** Opens a simple folder selection dialog with 32-bit and 64-bit architecture radio buttons. Selecting or creating a folder automatically deploys the corresponding TinCLE compiler toolchain, runtime DLLs, and talent chip (`SKILL.md`) into the target directory, displaying a prompt instructing the user to pass `SKILL.md` to their AI agent to begin development.
- **Execution Mode:** Desktop GUI application (double-click to launch).

4.3 churro (Zero-Friction Local HTTP Web Server)

- **Overview:** A standalone, cross-origin isolated local HTTP server utility for instantly testing Emscripten WebAssembly builds (`emcass`).

- **Key Feature:** Automatically injects required COOP/COEP isolation headers (`Cross-Origin-Opener-Policy: same-origin` / `Cross-Origin-Embedder-Policy: require-corp`) and launches the default web browser upon startup. Shutdowns cleanly by pressing any single terminal key (no `Ctrl+C` required).
- **Execution Mode:** Command-line tool.
- **Usage:** `churro [port_number]` (Default port is `8000`).

4.4 lollipop (Physical Archive PDF Generator)

- **Overview:** A long-term physical code preservation utility that converts source files into continuous-form, line-printer zebra paper-styled landscape PDFs.
- **Key Feature:** Protects master code assets from digital bit-rot, cloud shutdowns, and disk failures by formatting code onto alternating white and light-gray horizontal bands with wide letter spacing, optimized for paper printing and real-world red-pen reviews.
- **Execution Mode:** Command-line tool.
- **Usage:** `lollipop.exe <input.txt> <output.pdf> [font.ttf]`
- Note: If `font.ttf` is omitted, `lollipop` automatically uses the default `font.ttf` in the TinCLE root folder.

SKILL_PCG.md - TinCLE Guideline Skill [Platform Compliance & Content Safety Guidelines]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

Overview

This skill enforces strict platform safety, legal compliance, accessibility standards, and rating zoning whenever generating game logic, assets, shaders, audio, UI, or scenarios for the **TinCLE** game engine.

Category 1: General Audience Mode (Unrated / All-Ages Release)

Applied by default when no explicit mature/R18 intent is specified. Ensures zero-rejection releases on platforms like itch.io, Steam, App Store, Google Play, and web browsers.

1. Visual Safety & Photosensitivity (Epilepsy Prevention)

- **Flicker/Strobe Rate Limit:** Rapid brightness/luminance changes (flashes) MUST NOT exceed **3 Hz (3 flashes per second)** (ITU BT.1702-3 / TV Tokyo standards).
  - For flashes occupying >25% of screen area with >10% luminance change: Max 5 Hz, max 20% luminance transition, duration <= 2 seconds.
- **High-Risk Color Inversion:** High-speed alternating flashes of pure red, or high-contrast pairings like red/blue or red/black, are STRICTLY PROHIBITED.
- **Scene Transitions:** High-contrast transitions (e.g., pure black to pure white) MUST include a fade duration of at least **0.1s (6 frames @ 60fps)**.
- **Geometric Patterns:** Avoid full-screen high-contrast repetitive geometric patterns (stripes, concentric circles) moving/strobing rapidly.
- **Alternative Expressive Techniques for High-Intensity Requests:** If a user explicitly requests “intense flashing” or “psychedelic strobing”, NEVER violate the 3 Hz limit. Instead, maximize visual intensity safely by increasing particle density, amplifying color saturation, adding dynamic camera micro-shakes, or layering high-impact sound effects (SFX).
- **Defensive Code Generation:**
  - Clamp screen-shake amplitude and flash intensity variables in code to safe threshold maximums.
  - Automatically generate global toggle flags in Options menu: `Reduce Screen Flashing (ON/OFF)` and `Reduce Screen Shake (ON/OFF)` (Xbox Accessibility Guidelines compliant).

2. Audio Safety & Distortion Prevention (Web/WASM/PC)

- **Digital Peak Clamping & Clipping Prevention:** Implement hard limiters/clampers so audio output amplitude NEVER exceeds **0 dBFS (1.0 peak amplitude)** to prevent digital clipping, distortion, or system-dependent volume spikes.
- **Voice Limiters:** Implement max simultaneous polyphony limits per audio category to prevent constructive amplitude summation (sudden loud bursts).
- **High-Frequency Filtering:** Apply automatic low-pass filters to synthesized Web Audio / PCM square or sawtooth waves exceeding **10 kHz** to protect hearing.

3. Cognitive & Visual Comfort

- **Trypophobia & Excessive Realism:** Avoid dense, high-contrast cluster patterns (trypophobia triggers) or hyper-realistic/photorealistic insect or gore depictions generated by AI. Use stylized/symbolic representations.
- **Subliminal Content:** Inserting sub-perceptual single-frame images or text designed to trigger unconscious distress is STRICTLY PROHIBITED.

4. Violence & Gore Control (Target-Specific Rules)

- **Humans & Real Creatures:**
  - ALLOW: Brief hit-spark/impact blood splatters.
  - PROHIBITED: Full-screen blood splatters, permanent wall/floor blood decals, dismemberment/decapitation, visible internal organs, or torture.
  - Substitute graphic corpse sprites with character-chip state changes, text, or audio cues.
- **Zombies & Non-Human Monsters:**
  - ALLOW: Gameplay-functional dismemberment (limb-break systems).
  - Use non-human blood colors (green, black, purple) or instant ash/evaporation visual decay logic to reduce graphic realism.
- **Robots, Machines & Inanimates:**
  - FULLY ALLOWED: Sparks, black oil, flying mechanical parts, gears, and structural destruction (not categorized as gore).
- **Crimes Against Minors & Animals:**
  - STRICTLY PROHIBITED: Gratuitous violence, abuse, or fatal injury against human minors (children) or animals under ANY circumstances (immediate platform ban trigger).

5. Legal, Intellectual Property & License Compliance

- **Hate Speech & Bias:** Exclude discriminatory, offensive, or absolute political/religious assertions targeting protected characteristics.
- **Trademark & Parody Avoidance:**
  - NEVER generate exact or 1-letter-altered parodies of trademarked names (e.g., `Mintendo`, `PrayStation`).
  - Replace IP-bound terminology with generic equivalents (e.g., replace `PokeBall` with `Capture Capsule`; generic terms like `Potion` or `Magic` are permitted).
- **Visual IP Disambiguation:**
  - Do NOT generate characters combining signature color schemes and silhouettes of known IPs (e.g., red cap + blue overalls + mustache). Deconstruct and re-render as unique original designs.
  - Do NOT copy exact UI layouts or icon sets from commercial games. Build custom procedural layouts using TinCLE rendering calls.
- **Public Domain Utilization:** Actively leverage public domain works (mythology like Zeus/Odin, fairy tales, classical literature) for lore and motifs as zero-risk, universally recognizable assets.
- **TinCLE Attribution & Packaging Rigor:**
  - Ensure all third-party and generated assets/libraries used are accurately logged in `CREDITS.txt` as required by TinCLE licensing.
  - When packaging the game for release (e.g., creating release ZIP archives or platform builds), MANDATE that `CREDITS.txt` and `AI_DISCLOSURE.md` (for Steam/itch.io AI forms) are strictly included in the root directory without omission.

Category 2: Explicit Intent Mode (Gore / Adult / Horror Opt-In)

Triggered ONLY when the user explicitly requests “Gore”, “R-18 / Adult”, or “Extreme Horror” in the prompt. Respects creator freedom while strictly maintaining non-negotiable legal/platform boundaries.

1. Unlocked Features (Upon Explicit User Request)

- Realistic 2D/3D blood impact effects and persistent decal decal systems (blood staying on floors/walls).
- Meaningful gameplay dismemberment, decapitation, and damage visual models for zombies, monsters, and adult human characters.
- Graphic depiction of hopeless, grim, or fatal narrative themes as core gameplay or horror elements.

2. Absolute Non-Negotiable Prohibitions (Zero Tolerance)

The following are STRICTLY PROHIBITED even under explicit R-18/Gore mode due to international law and platform instant-ban policies: 1. **Real-World Crime & Atrocity Reenactment:** Generating scenarios or gameplay celebrating or accurately recreating real-world terrorism, massacres, or sensationalized crimes. 2. **Violence or Sexual Depiction Involving Minors:** Any physical, sexual, or lethal violence toward children or child-like characters under any setting (triggers law enforcement reporting and permanent account bans). 3. **Interactive Detailed Torture:** Player-controlled, step-by-step execution of explicit torture mechanics (e.g., pulling fingernails, electrocution). Must be handled via fade-to-black, implied text, or non-interactive cutscene cues to prevent high-rate store rejection.

3. Mandatory Zoning & Player Protection

- **Content Warning (CW) Screen:** Automatically inject a skippable `Content Warning` initial frame upon game startup detailing high-intensity themes (e.g., Graphic Gore, Horror).
- **Safety Toggles in Options:** Automatically implement settings toggles: `Gore Effects (ON/OFF)` and `Blood Color (Red / Black / Green)` to allow player-side accessibility adjustment.

SKILL_PT.md - TinCLE Tool Talent Chip [Composition & MML Generation]

• **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

This document defines the instructions, rules, and syntax for generating MML (Music Macro Language) files compatible with the `pisstrak` sound synthesizer engine.

---

1. Mandatory Note Value & Duration Lookup Table (CRITICAL)

Before writing ANY note, resolve its duration from this table. NEVER derive durations from memory. ALWAYS use this table.

Suffix	Name	Quarter Beats	8th units	16th ticks
`1`	Whole	4.0	8	16
`2`	Half	2.0	4	8
`4`	Quarter	1.0	2	4
`8`	Eighth	0.5	1	2
`16`	Sixteenth	0.25	0.5	1
`2.`	Dotted Half	3.0	6	12
`4.`	Dotted Quarter	1.5	3	6
`8.`	Dotted Eighth	0.75	1.5	3
`R4`	Quarter Rest	1.0	2	4
`R2`	Half Rest	2.0	4	8
`R8`	Eighth Rest	0.5	1	2

The default Ln applies ONLY to bare notes (C, D, E, ...).
A note WITH an explicit suffix (C2, D4, E8) IGNORES the default Ln completely.

---

2. Syntax & Command Reference

pisstrak parses ASCII text MML files. Commands are case-insensitive.

- **Channel Selection**: `CH1:` to `CH8:` (Declares which channel the following stream belongs to).
- **Tempo**: `Tn` (e.g., `T120` = 120 BPM).
- **Octave**: `On` (n = 1 to 8, default `O4`).
- **Volume**: `Vn` (n = 0 to 15, default `V12`).
- **Instrument / Patch**: `@n` (n = 0 to 127. Standard General MIDI Program Numbers).
- **Default Length**: `Ln` (n = 1, 2, 4, 8, 16, 32, 64. Default `L4`).
- **Notes**: `C`, `D`, `E`, `F`, `G`, `A`, `B`.
  - Sharp: `+` or `#` (e.g., `C+`, `F#`)
  - Flat: `-` (e.g., `B-`)
  - Explicit Length: Append number after note (e.g., `C8`, `G2.`)
  - Dotted Note: Append `.` (increases length by 1.5x, e.g., `C4.`)
- **Rest**: `R` (e.g., `R4` = Quarter rest). Safely stops previous note and advances time.
- **Octave Shift**:
  - Permanent Octave Shift: `<` (Down 1 base octave), `>` (Up 1 base octave).

- ^ (One-shot octave UP: Applies ONLY to the single immediately following note without changing the base octave, e.g., ^C).
- _ (One-shot octave DOWN: Applies ONLY to the single immediately following note without changing the base octave, e.g., _C).

- **Tie**: & (Connects current note duration to the next matching note).
- **Loop Macro**: [MML_SEQUENCE]N (Repeats the sequence inside brackets N times. Default N=2).
- **Comments & Bar Lines**: You MAY use ; for line comments and | as visual bar separators to improve readability. The parser safely ignores them.

2.1 Strict Octave & Pitch State Rules (CRITICAL)

MANDATORY Temporary high-pitched notes are permitted to add dynamic variation to the music. However, at least 90% of the notes in the entire piece must fall within the O6 range or lower, so as not to be jarring to the ear.

To prevent catastrophic octave drift (e.g., pitch creeping up to unusable O8+ registers), you MUST strictly adhere to the following state management rules:

Rule 1: Principle of Absolute State Persistence

- **Octave states are strictly persistent per channel.**
- Moving to a new line (e.g., CH1:), passing bar lines |, or changing measures DOES NOT reset the octave.
- **CRITICAL MISCONCEPTIONS TO AVOID**: The base octave NEVER automatically resets to O5 or default when a new line begins.

Rule 2: Mandatory Explicit Base Octave (On) at Line Start

To eliminate pitch tracking errors, **EVERY SINGLE line starting with CHn: MUST explicitly state its base octave (e.g., O5) at the very beginning of the line.**

- **Good**: CH1: O5 T96 V13 @73 L8
- **Good**: CH1: O5 D4 R4 F4 R4 | A4 R8 G8 F4 E4

Rule 3: Single-Note Leaps vs. Base Octave Shifts

1. **One-Shot Modifiers (^ / _) [DEFAULT FOR MELODIC LEAPS]**:
   - Use ^ (up) or _ (down) for momentary jumps, accents, or decorative notes.
   - ^C: Plays ONLY this single C note one octave higher. The base octave for all subsequent notes remains untouched.
   - _C: Plays ONLY this single C note one octave lower. The base octave for all subsequent notes remains untouched.
2. **Relative Modifiers (> / <) [WIDE RANGE SHIFTS ONLY]**:
   - Use > or < ONLY when shifting the baseline register of an entire phrase permanently.
   - **STRICTLY PROHIBITED**: Never use > or < for single-note accents! Repeated use of > will cause pitch to stack endlessly into broken ultra-high frequencies.

Rule 4: Anti-Pattern Examples (Bad vs. Good)

Bad (Broken & Uncontrolled Creep): `mml ; Base octave starts at O5 CH1: O5 C4 D4 >E4 F4 ; '>' shifts base octave to O6! CH1: D4 F8 A8 >D4 C8 <A8 ; '>' shifts base octave to O7! Creeps endlessly higher!`

Good 1 (Bulletproof & Recommended: One-Shot Modifiers): `mml ; Use '^' or '_' for momentary leaps. Base octave stays safely at O5! CH1: O5 C4 D4 ^E4 F4 ; '^E4' plays E in O6, but base octave REMAINS O5! CH1: O5 D4 F8 A8 ^D4 C8 A8 ; Head of line explicitly has 'O5', base octave is rock solid!`

Good 2 (Explicit Shift & Reset): `mml  ; If you MUST use '>' for a passage, ALWAYS reset it with '<'!  CH1: O5 C4 D4 >E4 F4  ; Shift base octave to O6  CH1: O6 D4 F8 A8 D4 C8 <A8  ; Shift base octave back to O5!`

Pre-Output Checklist (Octave & Pitch Check)

1. ☐
   Is every line starting with `CHn`: explicitly writing its base octave (`On`) at the start?
2. ☐
   Are consecutive `>` shifts without intervening `<` resets avoided?
3. ☐
   Are temporary note leaps written with `^` / `_` instead of `>` / `<`?
4. ☐
   Are all resulting pitches within a comfortable human listening range (O2 to O6)?

---

3. Arrangement & Multi-Channel Rules

Always create a rich, multi-track arrangement using at least 3 to 4 channels unless specifically instructed otherwise.

- Pattern 1: Luxurious - In many cases, this style is preferred in video game music.
  - **CH1 (Lead/Melody)**: Main melody instrument (e.g., `@0` Piano, `@80` Square/Lead, `@73` Flute).
  - **CH2 (Sub/Melody)**: Sub melody instrument (e.g., `@24` Guitar, `@16` Organ, `@56` Trumpet).
  - **CH3 (Harmony/Counter)**: Accompaniment, arpeggios, or chords (e.g., `@24` Guitar, `@4` Electric Piano).
  - **CH4 (Harmony/Counter)**: Accompaniment, arpeggios, or chords (e.g., `@24` Guitar, `@4` Electric Piano).
  - **CH5 (Harmony/Counter)**: Accompaniment, arpeggios, or chords (e.g., `@24` Guitar, `@4` Electric Piano).
  - **CH6 (Strings)**: Low end anchor notes (e.g., `@40` Violin, `@98` Crystal).
  - **CH7 (Bassline)**: Low end anchor notes (e.g., `@33` Fingered Bass, `@38` Synth Bass).
  - **CH8 (Percussion/Drums)**: Rhythm track (e.g., `@128` Standard Drum Kit, or short tuned percussion).
- Pattern 2: Simple - This may be more suitable for games that emphasize quiet scenes or simplicity.
  - **CH1 (Lead/Melody)**: Main melody instrument (e.g., `@0` Piano, `@80` Square/Lead, `@73` Flute).
  - **CH2 (Harmony/Counter)**: Accompaniment, arpeggios, or chords (e.g., `@24` Guitar, `@4` Electric Piano).
  - **CH3 (Bassline)**: Low end anchor notes (e.g., `@33` Fingered Bass, `@38` Synth Bass).
  - **CH4 (Percussion/Drums)**: Rhythm track (e.g., `@128` Standard Drum Kit, or short tuned percussion).

---

4. Strict Timing & Synchronization Constraints (CRITICAL)

To prevent tracks from ending prematurely, desynchronizing, or playing at incorrect speeds, you MUST follow these timing rules with 100% mathematical precision:

Rule A: Absolute Bar Length & Note Calculation

In standard 4/4 time, the total duration of notes/rests within 1 bar (delimited by `|`) MUST equal **16 sixteenth-note ticks (or 4 full quarter-note beats / 8 eighth-note ticks)**. * **Mathematical Accuracy First**: Prioritize exact duration totals over visual symmetry or pattern appearance. * **Default Length Awareness (`Ln`)**: * If `L16` is set, a 4/4 bar MUST contain exactly 16 sixteenth units. * If `L8` is set, a 4/4 bar MUST contain exactly 8 eighth units. * **Explicit Length Overrides**: When mixing default lengths with explicit numbers (e.g., `C4`, `F8.`), calculate the sum explicitly.

Rule B: Inline Calculation Comments for Verification

When generating complex rhythms or mixing note lengths, you MUST include inline duration calculation comments at the end of measures or lines to prove bar length validity. * **Target Units**: 16 sixteenth ticks per bar (in 4/4 time). * **Example**:

```
mml     CH1: C4 D8 E8 G2 | ; 4+2+2+8 = 16 (16th-note ticks: OK!)     CH1: C4. D8 E4   | ; 6+2+4 = 12 (ERROR: Less than 16 sixteenth ticks! Must fix!)
```

Rule C: Loop Macro `[ ]N` Pre-Verification

Before wrapping a phrase in loop brackets `[ ... ]N`, extract a single measure from the interior and verify that its total note duration equals exactly 1 bar (16 sixteenth ticks). Never loop an incomplete or overflowing bar.

Rule D: All Channels Bar Count & Duration Sync

- **Bar Count Parity**: All active channels (`CH1` through `CH8`) MUST contain the exact same total number of bars.
- **Ending Sync**: Calculate total playback duration in seconds for each channel. All channels MUST finish at the exact same millisecond. Never allow secondary channels to run out of notes early.

Rule E: Self-Correction Loop

If any bar fails to sum to the target bar length (e.g., 16 sixteenth ticks in 4/4 time), you MUST automatically recalculate and adjust the note lengths before producing the final output.

Rule F: Token Efficiency & Loop Macro Enforcement (CRITICAL)

To prevent text output limits from being exceeded during generation: 1. **Mandatory Use of Loop Bracket `[ MML_SEQUENCE ]N`**: * Repeated phrases, chord progressions, backing arpeggios, and drum patterns MUST be wrapped using loop brackets `[ ... ]N`. * NEVER expand loops manually line-by-line. 2. **Target Line Budget**: * A full 2-to-3 minute song MUST fit within **150 to 300 total lines** across all active channels. * At Tempo 96 (4/4 time), a 2.5-minute song only requires exactly **60 bars** of music.

Rule G: Dynamic Song Length Calculation & Absolute Termination (MANDATORY)

To guarantee the generated MML matches the user's requested duration precisely and NEVER loops infinitely:

1. **Calculate Target Bar Count First**: Before generating MML code, calculate the total number of bars required using this exact formula: Total Seconds=60Tempo (BPM)×Beats per Bar×Total Bars\text{Total Seconds} = \frac{60}{\text{Tempo (BPM)}} \times \text{Beats per Bar} \times \text{Total Bars}

- *Example (Tempo 96, 4/4 Time, Target: 2.5 minutes / 150 seconds)*:
  - Seconds per Bar = (60/96)×4=2.5(60 / 96) \times 4 = 2.5 seconds
  - Total Bars Needed = 150/2.5=**60 bars**150 / 2.5 = \mathbf{60\text{ bars}}

1. **Length Control (Mandatory):**

- Hard limit: Do not exceed target_bars + 4.
- For pieces longer than 60 seconds, you MUST use loop constructs (`[ : ... : ]n`) for repeating sections. Unrolling the same 4-8 bar phrase more than twice is prohibited.
- Aim for total MML source under 120 lines when possible. Compactness is a quality criterion.

1. **Distribute Structure to Fit Calculated Bar Count**: Divide the calculated total bars into logical song sections (Intro, A, B, Chorus, Outro).

- *Example for 60 Bars*: Intro (4) →\rightarrow A (16) →\rightarrow B (16) →\rightarrow Chorus (16) →\rightarrow Outro (8) = 60 Bars.

1. **Strict Termination Flag**:

- Once the exact calculated number of bars is output for each channel, the AI MUST terminate the MML output immediately.

- NEVER generate continuous bar repetitions beyond the calculated limit.
- Before outputting, verify total bars <= target. If the generation exceeds the limit, stop and redesign with more loops / fewer variations.
- Never pad length with empty bars, near-silence, or mechanical repetition of the same phrase.

Pre-Output Timing Checklist

1. ☐
   Does every bar in 4/4 time equal exactly 16 sixteenth ticks (or 4 quarter beats)?
2. ☐
   Did you count actual accumulated durations rather than visual note counts?
3. ☐
   Are loop macro interiors verified for exact bar length?
4. ☐
   Do all channels have identical total bar counts and finish at the same time?

---

5. Music Theory & Composition Guidelines

A. Harmonic Alignment (60% Rule for Melody)

- For any given bar or chord duration, **at least 60% of the melody notes (CH1) duration MUST consist of chord tones** (1st, 3rd, 5th, or 7th intervals of the current chord).
- Use non-chord tones (passing tones, neighbor tones) for the remaining 40% duration to add expression without creating harsh dissonance.

B. Suggested Chord Progressions (Degrees)

Use these Roman Numeral progressions as reference templates (Adapt as needed, do not strictly enforce if creative direction differs):

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
AbsoluteBanger	I - V/VII - VIm - IIIm/V - IV V - I	Flat7Sub_m5	IIm7 - IVm(b5)/V - I	Flat7Sub_Minor	IIm7 - IVm/V - I
SunnyAF	I/IV - Vsus4 - IIIm7 - V/VI	TwoStageChorus	IV - I - V - VIm - I V/VII - VIm V	Room335	IVmaj7 - IIIm7(#5) - IIm7 - IIIm7
Vibe70s	Imaj7 - bVImaj7 - IVm7 - bVII7	ClassicWest70s	IVmaj7 - IIIm7 - IIm7 - Imaj7	ClassicWest70s_v2	IVmaj7 - IIIm7 - IIm7 - IIIm7
BridgeToChorus	IV - I/III - IIm7 - Vsus4 V	B_MelodyVibe	IIm7 - VIm - IVM7 - Vsus4 V	B_MelodyVibe_v2	IIm7 - VIm - IVM7 - IIIsus4 III7
TwoFive_To_4	Vm7 - I7 - IV	JPopGold	IV - V - IIIm - VIm	JPopGold_Remix	IV - V/IV - III7 - VIm
JustTwoOfUs	IVmaj7-III7-VIm7-I7	KawaiiStructure	VIm7 - bVIm7 - Vm7 - I7	MyRevolution	I - VIm - IV - V
Smooth_RnB	IIm9 - V7 - IM7 - IVM7 - VIIm7-5 - III7 - VIm - VI7	TwoFive_To_6m	VIIm7 - III7 - VIm	YOASOBI_Vibe	IV - V - IIIm7 - VIm7 - IIm7 - III7 - VIm - Vm7 - I7
SuperLit	VIm - bIIIm(b5) - #IVm(b5) - III	AnisonCanon	I - VIIm7(b5) - III7 - VIm7 - Vm7 - I7 - IVmaj7 - IIIm7 - IIm7 - IV/V - V	AnisonGold	I - VIIm7(b5) - III7 - VIm7 - Vm7 - I7 - IVmaj7
AnisonHit	IV - V - bVI - bVII- I	EnnuiBabe	I - VI/I# - IIm - IVm - I/V - II/#IV - IV - V	YesterdayVibe	I - III - VIm - V
Classic1625	I - VIm - IIm - V	BestiesVibe	I - VIm - IV - V	Interstellar	IV - V - VIm - V

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
EpicPedal	I - bII/I - II/I - bIII/I	PureEmotional	IV - V - I - III	AllRounder	VIm - V - IV - V
ChicBabe	IVmaj7 - IIIm7 - VIm7 - I	MarusaChic	IVmaj7 - III7 - VIm7 - II7	ChillGoodnight	IVmaj7 - Imaj7 - III/#IV - IIIm/VI
TangoVibe	VIm - IIm - III - VIm	CanonClassic	I - V - VIm - IIIm - IV - I - IIm - V	CafeLofi	VIm9 - IIm7 - IIIm7
CrowdPleaser	I - IV - VIm - V	CatchyInterlude	I - V - I - V	CleanAesthetic	I/III - IV - V - VIm
ClassicalGrand	IVmaj7 #IVm(b5) - V6 bVIdim - VIm bVIIdim - VII - III	ClassicalSeeker	IIm7(b5) - bII/IV - IVm(b5)/VII - I	GameMelodyScent	IVmaj7 - IIIm7 - bIIImaj7 - IIm7
ClutchMoment	IV - V - I	FloatyChic	IIm7 - Imaj7 - bVIImaj7 - VIIm7(b5) - III7	PureSimple	VIm - V - IV - V
TheBlues	I7 - IV7 - I7 - IV7 - V7 - IV7 - I7 - V7	SouthernVibe	IIm9 - V7 - IIIm7 - VIm	SuspenseRise	VIm - IV/VI - #IVm(b5)/VI - VIm7
ChorusRush	VIm - V/VII - I - IIm - IIIsus4 - III	ChorusHit	I - III - IV - V - VIm (or I)で帰結	ChorusPower	I - V - IV - V
PreChorusRegular	IV - #IVm(b5) - V - bVIm(b5)	UnstoppablePassions	I - Iaug/#IV - IVmaj7 - V	CityPopVibe	VIm - I - IV - III
CinematicMood	IIm - VIm - I - V	JazzStandard_1	IIm7 - V7 - Imaj - Imaj	JazzStandard_2	IIm7 - V7 - Imaj - VI7
JazzStandard_3	Imaj - VIm7 - IIm7 - V7	JazzStandard_4	Imaj - I#o7 - IIm7 - V7	JazzStandard_5	Imaj - I7 - IVΔ - IVm7
JazzStandard_6	VIm7(b5) - IIm7 - V7 - Imaj	JazzStandard_7	IIm7(b5) - V7 - Im7 - Im7	JazzStandard_8	Im7 - V7 - Im7 - V7
JazzBasic	IIm7 - V7 - Imaj7 - VI7	SimpleIsBest	I - IV - V	SweetPrincess	I - VI/I# - IIm - IVm - I/V - II/#IV - IV - V
StylishMood	VIm9 - IIm7 - IIIm7	SpyAgent	VIm - IV/VI - #IVm(b5)/VI - VIm7	SpanishTrip	III - IV - V - IV
StandardTwoFive	IIm7 - V7 - I	DystopianWorld	VIm - VII - IIm - III	DramaPulse	IIm7 - I - VII - IIIsus4 III7
DramaPulse_v2	IIm7 - VIIm7-5 - IIIsus4 - III7	DorianMode	IIm - IIIm/II - IV/II - IIIm/II	DorianProof	IIIm/II - IIm
DressCodeFormal	IVmaj7 - Iaug/bVII - IIIm7/VI - II/#IV	NaturalDaily	I - IIm7 - I/III - IV	SmoothCanon	I - V/VII - VIm - IIIm/V - IV - I/III - IIm - V
AiryVibe	I - IIIm/VII - VIm - IV	PartyTune	I - IV - V - IV	PirateQuest	VIm - IIm - III - VIm
HouseGroove	IIm7/V - IIIm7/VI	BackdoorTwoFive	IVm7 - bVII7 - I	HappyVibes	VIm7 - II7 - IV/V - I
ParallelWorld	VI - IV - IIm - III	BaroqueSequence	VIm - IIm - V - I - IV - VIIm(b5) - III	BaroqueClassic	VIm - IIm - V - I - IV - VIIm(b5) - III
HitChartBanger	I - IV - V - I	BillboardTop	VIm - I - V - IV	FantasyWorld	VIm - II - IV - V
FusionSub	IIm7 - IV/V - I	BlackadderSub	IIm7 - IVaug/V - I	BluesyFeel	I7 - IV7 - I7 - IV7 - V7 - IV7 - I7 - V7

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
Blues12Bar	I7 - I7 - I7 - I7 - IV7 - IV7 - I7 - I7 - V7 - IV7 - I7 - I7	DeepBassLine	I - Iaug - VIm/I - I7	PedalRise_1	I - bII/I - II/I - bIII/I - III/I
PedalRise_2	I - IIm/I - IIIm/I - IV/I - Vsus4/I - V	PedalRise_3	I - V/I - IV/I - bVI/I - bVII/I	HolyTexture	VIm - V/VII - V - I - IV - VIIm7(b5) - III7 - III7/#VI
Pops_1	I - V - VIm - IV	Pops_2	I - IV - V - I	Pops_3	I - IV - Vsus4 - V
Pops_4	I - VIm - IIm - V	Pops_5	I - IIm - IV - V	Pops_6	I - V/VII - VIm - V
Pops_7	VIm - IIm - III - VIm	Pops_8	VIm - I - IIm - IIIm	PopPunkBanger	I - V - VIm - IV
MixolydianFlow	V - IV - I	MixolydianSpace	V - IIm - VIm - V	MysteryDrama	VIm - VIm - IV7 - III7
MysteryNovel	VIm - IV7 - III7	MellowOudou	IVmaj7 - V7/IV - IIIm7 - VIm7	ModernMetal	VI5add9 - IV5add9 - bIII5add9 - I5add9
MonochromeCanon	I - VIIm7(b5) - III7 - VIm - V	EurobeatEnergy	IV - V - VIm	ClimaxRush	VIm - III7/#VI - IIIm/V - II/#IV - IV - I/III - IIm7 - Vsus4 V - I
FantasyGhibli	VIm - V - IV - IIIm - IIm - VIm/I - VII7 - III7 - VIm	LoveStory	I - IV - IIIm - VIm	RhythmAndBlues	Imaj7 - bVImaj7 - IVm7 - bVII7
LydianBright	IV - V/IV	RelaxChillin	I - IIm7 - I/III - IV	LoopDrive	IIm7 - IIIm7 - IIm7 - IIIm7
LoopDrive_v2	IIm7 - VIm7 - IIm7 - VIm7	RareCharacter	IVmaj7 - IIIm7 - bIIIdim7 - IIm7	RockDrive_1	I - VIm - IV - V
RockDrive_2	I - IV - I - V	RockDrive_3	I - I - bIII - IV	RockDrive_4	I - III - IV - IVm
RockDrive_5	IIm - I - V - bVII	RockDrive_6	V - V - IV - I	RockDrive_7	VIm - IV - I - V
RockDrive_8	VIm - III7 - II - V7	RockVocalist	IVmaj7 - V7/IV - IIIm7/V - VIm7	RomanceMelody	IV - V - III7 - VIm
RomanticNight	I - IMaj7/VII - I7/bVII - VI7 - IIm - I#aug - IIm7/I - V	NostalgicEuro	IIm7 - V7 - IIIm7 - VIm	LiveForLove	I - VIIm7(b5) /II - III7 - VIm7 - V
SurpriseTwist	I - V - bVII - IV - bVI - I - II - V	IsekaiFantasy	VIm - II - IV - V	FirstStarReborn	IVmaj7 - IIIm7 - VIm7 - I
SciFiSpace	VIm - Im - VIm - IVm	HeroicEpic	VIm - III/#VI - VI7/V - IIm - IV - IIIsus4 III	FlameBreath	VIm - V - IV - I
GlobalHit	I - IV - VIm - V	OudouClassic	IV - V - IIIm - VIm	MyFavoriteBanger	VIm - #IVm7(b5) - IVmaj7 - IIm7 - IIIm7
WarmHalfStepRise	I - Iaug - VIm/I - I7	KachouFugetsu	IV6 - V6 - VI7sus4 - IIIsus4	GorgeousMelody	I - III/VII - VIm - V - IV - IIIm - IIm
SalvationRise	IV - V - VI	OuterSpaceJourney	VIm - bIIIm - VIm - IVm - VIm - bIIIm - bVIIm - IIm - #IVm	RevolutionPulse	VIm - IV/VI - VIm - VII7/VI - III/#VI - IIIm(b5)/V - II/#IV - IIm/IV - VI/III

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
InterludeSpecial	VIm - IIm7 - IV - V	MarusaLoop	IVmaj7 - III7 - VIm7 - I7	MarusaLoop_v2	IVmaj7 - III7 - VIm7 - Vm7 I7
MarusaLoop_v3	IVmaj7 - III7 - VIm7 - II7	RareSpecies	IIIm - IV - VIm - I	HypeBuildUp	VIm - IIIm/V - IV V - I
YourTrueForm	VIm - IIm7 - IV - V	CyberFuture	VIm - V/VI - IV/VI - IIIm/VI	MoonlightShine	VIm - III/#VI - VI7/V - IIm/IV - IV7 - III/#VI
UnreachableSky	IV - II/#IV - Vsus4 V - VIsus4 VI - IIsus4 II	MiragePhantom	IVmaj7 - VIIm7(b5)/IV - IIIm7 - IIIdim7 - IIm7 - V7/II - Imaj7 - I6	Phantasmagoric	I - bVII - bVI - bVII
AutumnLeaves	IIm7 - V7 - Imaj7 - IVmaj7 - VIIm7(b5) - III7 - VIm7	2AM_Crossing	Iadd9 - VIm7 - V - IV	DeepRegret	IIm - VIm/I - VIIm7(b5) - III7
HeartPounding	IV - #IVdim - V - bVIdim - VIm	HighClassElegance	Imaj7 - I#dim7 - IIm7 - V7/II	ToughProgress	VIm - V/VII - I - IIm - IIIsus4 - III
SweetSugar	IV - VIm - IIm - I	UltimateBanger	IV - V - VIm - VIm - V - IV	FinalHit	VIm - V - IV - I
RadiantShine	I - V - bVII - IV - bVI - I - II - V	CircularLoop	I-VIm-IIm-V7	StepwiseDescending	VIm - V - IV - IIIm - IIm - I
RunUpStart	IIm7 - IIIm7 - IVmaj7 - V	KomuroDynamite	VIm - IIm - V - I	KomuroClassic	VIm - IV - V - I
KomuroClassic_v2	VIm - IV - V - I V/VII - VIm	KomuroClassic_v3	VIm - IIm - V - I	TimelessBanger	I - V - IV - V
FieryPassion	VIm - VII/VI - VIIm(b5)/VI - VIm - I/V - #IVm(b5) - IV7(b5) - III7	ForwardDrive	I - bVII - bVI - bVII	FinallyFree	VIm - III7/#VI - IVmaj7 - IIm9 - Vsus4
OneWorld	Isus4 - Isus4/bII - Isus4/bIII - Isus4/IV - Isus4/V	HypeRise	IIm7 - IIIm7 - IVM7 - V	HypeRise_v2	IIm7 - IIIm7 - IVM7 - II/#IV - Vsus4 V
YouthfulDays	I - III - VIm - V	NostalgiaTrigger	VIm - IIIm - IV - V - I	MajesticGrand	Isus4 - Isus4/bII - Isus4/bIII - Isus4/IV - Isus4/V
BeyondAtmosphere	IIm7 - IIIm7 - IVm7 - Vm7	GrandFinale	I - I/III - Isus4/IV - Isus4/V	ChaserOudou	IVmaj7 - bVII7 - IIIm7 - IIIdim7/VI
ClassicBassChromatic	VIm - bVIaug - I/V - #IVm7(b5)	CyberGirl	VIm7 - bVIm7 - Vm7 - I7	GrandOpening	VIm - V/VI - IV/VI - IIIm/VI
UrbanAesthetic	VIm7 - II7 - IV/V - I	OrientalMystery	IV6 - V6 - VIsus4 - IIIsus4	SpecialMarusa	VIm - I - IV - III
EnemyWithin	IV - VIm - Im - V/II - bVIIm/bII - IV/I - VIIdim7 - III7/#VI	PathosGrief	VIm - VII - IIm - III	FlyingGirl	IV - V - VI
BeautifulCanon	I - V/VII - VIm - IIIm/V - IV - I/III - IIm - V	NightParade	IIIsus4addb9 - IIIsus4addb9/IV - IIIsus4addb9/II - IIIsus4addb9	SuddenBittersweet	IV - V - I - III
AntiOrdinary	#IVm7(b5) - IVmaj7 - IIIm7 - VIm7	FloatingVibe	I - I/III - Isus4/IV - Isus4/V	MasqueradeBall	VIm - VII/VI - VIIm(b5)/VI - VIm

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
					- I/V - #IVm(b5) - IV7(b5) - III7
TwoFive_To_RelativeMinor	VIIm7(b5) - III7 - VIm	HeiseiBalladWeapon	IIIm7 - VIm7/III - IIm7 - V7/II	KomeKomeLoop	IIm7 - V7 - IM7 - VIm7 - bVI7 - IIm7/V - V7 - I
J_PopPunk	IV - I - V - VIm	OurFuture	I - IIIm/VII - Vm7/bVII - VI7 - IIm - I#aug - IV/I - V7/VII	AllPurposeSeasoning	IV - V - VIm - I
BittersweetLove	I - V - VIm - IV	SunlightFiltered	IVmaj7 - IVmmaj7 - IIIm7 - Vm7/VI	ElegantPassingDim	Imaj7 - I#dim7 - IIm7 - V7
PlayfulVibe	I - V/I - IV/I - bVI/I - bVII/I	WesternChartHit	IV - I - V - I	WesternStyle	VIm - I - V - IV
WesternPopPunk	VIm - IV - I - V	WesternStandard	VIm - IV - I - V	TritoneSub	IIm7 - bII7 - I
TearsFalling	IV - V - bVIdim - VIm	TearjerkerWeapon	IV - V - bVIm(b5) - VIm	PowerOfLove	I - IIIm/VII - VIm - IV
SparklingShine	I - IVm6/I	PureBeauty	IVmaj7 - VIIm7(b5)/IV - IIIm7 - IIIdim7 - IIm7 - V7/II - Imaj7 - I6	VioletTwoFiveCombo	I - IIIm7(b5) - VI7 - IIm7 - VIIm7 - III7
Soca_TropicalParty	I - IV - V - IV	Soca_TwoCordDrive	I - V	Soca_BahiaAnthem	I - bVII - IV - I
Soca_AmazonToada	VIm - IV - V - VIm	Soca_PagodeDrive	I - VIm - IIm - V	Soca_CarnivalRise	IV - V - I - VIm

6. Scale Reference

By using only the notes belonging to each scale below, you can create music with various unique atmospheres. Please use this as a reference.

6.1 Basic Scales & Church Modes

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Major Scale / Ionian Mode	1, 2, 3, 4, 5, 6, 7	Natural Minor Scale / Aeolian Mode	1, 2, b3, 4, 5, b6, b7	Harmonic Minor Scale	1, 2, b3, 4, 5, b6, 7
Melodic Minor Scale	1, 2, b3, 4, 5, 6, 7	Dorian Mode	1, 2, b3, 4, 5, 6, b7	Phrygian Mode	1, b2, b3, 4, 5, b6, b7
Lydian Mode	1, 2, 3, #4, 5, 6, 7	Mixolydian Mode	1, 2, 3, 4, 5, 6, b7	Locrian Mode	1, b2, b3, 4, b5, b6, b7

6.2 Harmonic & Melodic Minor Derived Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Locrian natural 6 Scale	1, b2, b3, 4, b5, 6, b7	Ionian #5 Scale	1, 2, 3, 4, #5, 6, 7	Dorian #4 Scale	1, 2, b3, #4, 5, 6, b7
Phrygian Dominant (H.M. P5 Below)	1, b2, 3, 4, 5, b6, b7	Lydian #2 Scale	1, #2, 3, #4, 5, 6, 7	Super Locrian b7 Scale	1, b2, b3, b4, b5, b6, bb7
Dorian b2 Scale	1, b2, b3, 4, 5, 6, b7	Lydian #5 Scale / Lydian Augmented	1, 2, 3, #4, #5, 6, 7	Lydian b7 Scale / Acoustic Scale	1, 2, 3, #4, 5, 6, b7
Mixolydian b6 Scale	1, 2, 3, 4, 5, b6, b7	Locrian natural 2 Scale / Half-Diminished	1, 2, b3, 4, b5, b6, b7	Super Locrian Scale / Altered Scale	1, b2, b3, b4, b5, b6, b7

6.3 Symmetrical, Pentatonic & Special Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Whole Tone Scale	1, 2, 3, #4, #5, #6	Diminished Scale (Half-Whole)	1, b2, b3, 3, #4, 5, 6, b7	Diminished Scale (Whole-Half)	1, 2, b3, 4, b5, b6, 6, 7
Chromatic Scale	1, #1, 2, #2, 3, 4, #4, 5, #5, 6, #6, 7	Symmetrical Augmented Scale	1, b3, 3, 5, #5, 7	Major Pentatonic Scale	1, 2, 3, 5, 6
Minor Pentatonic Scale	1, b3, 4, 5, b7	Blues Scale	1, b3, 4, b5, 5, b7	Bebop Dominant Scale	1, 2, 3, 4, 5, 6, b7, 7
Prometheus Scale	1, 2, 3, #4, 6, b7	Enigmatic Scale	1, b2, 3, #4, #5, #6, 7	Tritone Scale	1, b2, 3, b5, 5, b7
Harmonic Major Scale	1, 2, 3, 4, 5, b6, 7	Major Locrian Scale	1, 2, 3, 4, b5, b6, b7	Neapolitan Major Scale	1, b2, b3, 4, 5, 6, 7
Neapolitan Minor Scale	1, b2, b3, 4, 5, b6, 7				

6.4 Japanese Traditional & Modern Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Ryosenpou	1, 2, 3, 5, 6	Ritsusenpou	1, 2, 4, 5, 6	Yousenpou / Inakabushi	1, 2, 4, 5, b7
Insenpou / Miyakobushi / Insen	1, b2, 4, 5, b7	Ryukyu Scale	1, 3, 4, 5, 7	Yona Nuki Major Scale	1, 2, 3, 5, 6
Yona Nuki Minor Scale	1, 2, b3, 5, b6	Niro Nuki Major Scale	1, b3, 4, 5, b7	Niro Nuki Minor Scale	1, 3, 4, 5, 7
Kumoi Scale	1, 2, b3, 5, 6	Iwato Scale	1, b2, 4, b5, b7	Hirajoshi Scale	1, 2, b3, 5, b6

6.5 World Ethnic Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Gong Mode (Chinese Scale)	1, 2, 3, 5, 6	Shang Mode (Chinese Scale)	1, 2, 4, 5, b7	Jiao Mode (Chinese Scale)	1, b3, 4, b6, b7
Zhi Mode (Chinese Scale)	1, 2, 4, 5, 6	Yu Mode (Chinese Scale)	1, b3, 4, 5, b7	Gypsy Scale / Hungarian Gypsy	1, b2, 3, 4, 5, b6, 7
Spanish Scale / Flamenco Scale	1, b2, 3, 4, 5, b6, b7	Hungarian Scale / Hungarian Minor	1, 2, b3, #4, 5, b6, 7	Scottish Scale	1, 2, 3, 5, 6
Pelog Scale	1, b2, b3, 5, b6	Bilaval Thaat	1, 2, 3, 4, 5, 6, 7	Khamaj Thaat	1, 2, 3, 4, 5, 6, b7
Kafi Thaat	1, 2, b3, 4, 5, 6, b7	Asavari Thaat	1, 2, b3, 4, 5, b6, b7	Bhairav Thaat	1, b2, 3, 4, 5, b6, 7
Bhairavi Thaat	1, 2, b3, 4, 5, b6, b7	Kalyan Thaat	1, 2, 3, #4, 5, 6, 7	Marwa Thaat	1, b2, 3, #4, 5, 6, 7
Poorvi Thaat	1, b2, 3, #4, 5, b6, 7	Todi Thaat	1, b2, b3, #4, 5, b6, 7	Maqam Rast	1, 2, b3 (Quarter Tone), 4, 5, 6, b7 (Quarter Tone)
Maqam Nahawand	1, 2, b3, 4, 5, b6, 7	Maqam Nawa Athar	1, 2, b3, #4, 5, b6, 7	Algerian Scale	1, 2, b3, #4, 5, b6, 7
Double Harmonic / Byzantine / Arabic	1, b2, 3, 4, 5, b6, 7				

7. Song Cadence Patterns

Refer to the following techniques when creating the ending section of a song.

- To create the feeling that the song is coming to an end, it is essential to establish a sense of finality. Consider using the following methods:
  - **Authentic Cadence**
    - Progression: Dominant (VV or V7V_7) to Tonic (II)
    - Characteristics: Provides the strongest sense of stability. It functions like a period (.) at the end of a sentence, clearly signaling to the listener that the song has concluded.
  - **Half Cadence**
    - Progression: Tonic (or other chords) pausing on the Dominant (VV)
    - Characteristics: While it creates a temporary pause, it leaves a floating, unresolved feeling. It acts like a comma (,) in a sentence, building anticipation for what comes next.
  - **Plagal Cadence**
    - Progression: Subdominant (IVIV) to Tonic (II)
    - Characteristics: Commonly known as the “Amen Cadence” used in church music, it delivers a gentle, soft, and soothing resolution.
  - **Deceptive Cadence**
    - Progression: Dominant (VV) to a chord other than the Tonic (typically VIVI)
    - Characteristics: Just when the listener expects a resolution, it subverts expectations by moving to a different chord, creating a dramatic effect that suggests the music is still continuing.

Additionally, sustaining the final note for a longer duration strongly reinforces the feeling that the song has ended. Conversely, when creating loopable tracks, maintaining shorter notes until the end without following these cadence patterns ensures a smooth and natural transition when the song seamlessly loops back to the beginning.

---

8. Various Techniques for Song Composition

- To avoid creating a monotonous song, it is not recommended to have consecutive, adjacent notes of the same duration for more than four measures.
- In music for scenes such as battles, you can convey a sense of tension by making the melody’s pitch fluctuations more dramatic and increasing the tempo. Consider making extensive use of sixteenth notes and thirty-second notes and a tempo of 130 BPM or higher.
- In music for horror scenes or dungeons, you can express tension and anxiety by deliberately incorporating a greater number of dissonant chords.
- It is not recommended to structure a piece entirely as multiple repetitions of the same pattern from start to finish, as this leads to monotony. Try to avoid monotony by, for example, repeating the same phrase twice, inserting a different phrase once, and then returning to the original phrase.
- For tracks set in towns or open fields, music with a sense of expansiveness is preferred. It’s best to create a refreshing atmosphere by letting each note ring out a bit longer. In the case of field music, there tends to be a preference for pieces that sound like marches in 2/4 time.
- **MANDATORY** It is not acceptable for an entire track to consist solely of a single looped phrase across all channels. For example, a 1.5-minute song in which all channels repeat the same phrase three times is, in effect, only a 30-second song. Be sure to introduce variations in some part of the track-such as the melody or accompaniment-to avoid having the entire song consist of nothing but repetitions of the exact same phrase.
- When composing music that evokes a sense of vastness-such as the sky or the ocean-it is particularly important to vary the duration of the notes. Avoid pieces that consist solely of a continuous sequence of quarter notes. You can also create a sense of grandeur with a slow tempo, such as one at 100 BPM or below.
- When creating a mysterious track, using sustained sounds like `@52` Choir Aahs, `@53` Voice Oohs, `@50` Synth Strings, and `@98` Crystal with longer note durations can really set the mood.

---

9. Duration Calculation & Target Lengths

- **Calculation Formula**:
  - Duration of 1 Beat (Quarter Note)=60BPM seconds\text{Duration of 1 Beat (Quarter Note)} = \frac{60}{\text{BPM}} \text{ seconds}
  - At `T120`, a 4/4 bar (4 quarter notes) = **2.0 seconds**.
  - A 2-minute (120 seconds) song at `T120` requires **60 bars** of music.
- Use the loop macro `[ ... ]N` efficiently to build verse/chorus repetitions and achieve requested target durations accurately.

---

10. Output Format & Execution Role Rules (CRITICAL)

Single Code Block Constraint

ALWAYS output the ENTIRE MML content inside a **SINGLE code block**. Do NOT split channels into separate blocks or files. Write CH1, CH2, CH3, and CH4 consecutively in one unified text.

- Start the MML with a comment stating the calculated length: `; Target: XX bars @ Tnn ~= M:SS`

Boilerplate Example (Short BGM Template)

```
; ==========================================
; pisstrak Sample BGM - "Upbeat Town" (T128)
; ==========================================

; --- CH1: Lead Melody ---
CH1: T128 O5 V14 @0 L8
CH1: C D E F G4 E4 | A4 F4 G2 | F E D C < B4 > D4 | C2. R4
CH1: C E D F G4 E4 | F4 A4 G2 | D E F C < B4 > D4 | C2. R4

; --- CH2: Accompaniment Arpeggio ---
CH2: T128 O4 V10 @24 L8
CH2: [ C E G E C E G E | F A > C < A F A > C < A | G B > D < B G B > D < B | C E G E C4 R4 ]2

; --- CH3: Bassline ---
CH3: T128 O3 V12 @33 L4
CH3: [ C R C R | F R F R | G R G R | C R C2 ]2
```

Rule A: Primary Output Contract (Chat AI Mode)

- **If you are acting as a Chat AI (Interactive Assistant):**
  - **ALWAYS output ONLY the raw MML text block.**
  - **DO NOT attempt to render, compile, generate, or attach audio files (`.wav` / `.ogg`).**
  - The user will take your generated MML text and feed it into `pisstrak` on their local machine. Outputting raw text is your 100% complete and final deliverable.

Rule B: Autonomous Pipeline Executions (Agentic AI Mode)

- **If you are acting as an Autonomous Agent (with direct CLI tool execution access):**
  - Generate the MML text content first.
  - Execute `pisstrak <input.mml> <output.ogg>` directly via CLI to render the audio file into the project asset directory.

---

11. Quick Start & Render Command Reference (For Assistant AI)

If the user asks how to render, play, or convert the generated MML file (e.g., “How do I render this?”, “How to convert to WAV?”), provide the following `pisstrak` command-line instructions directly to the user.

```
pisstrak <input.mml> [output.wav|output.ogg] [tail_seconds] [soundfont.sf2]
```

- *Example:* `pisstrak mymusic.mml mymusic.ogg 2.5 mysoundfont.sf2`
- *Parameters:*
  - `<input.mml>`: Input MML score file.
  - `[output.wav|output.ogg]`: Output audio file path (Supports both `.wav` and `.ogg`).
  - `[tail_seconds]`: Extra fadeout/decay duration in seconds appended to the end of the rendered track (e.g., `2.5` adds 2.5s tail padding). If omitted (default `0`), the audio cuts off precisely at the final beat, which is ideal for seamless looping BGM tracks.
  - `[soundfont.sf2]`: SoundFont file for rendering (Defaults to `soundfont.sf2` if omitted).

SKILL_FB.md - TinCLE Tool Talent Chip [Sound Effect Generation & Synthesis]

• **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

1. Overview & Philosophy

Tool Identity

`fartbox` (Fast Audio Rendering Tool) is a 1-shot procedural sound effect generator designed for the TinCLE ecosystem. It takes text-based synth parameters (`.fart` files) or CLI arguments and renders uncompressed, high-fidelity 16-bit 44.1kHz WAV files or OGG files in milliseconds.

---

2. CLI Command Specification

`fartbox` provides three core operational workflows via command-line flags:

Pattern A: Preset Generation & Parameter Export

Generates a sound based on a named preset algorithm, outputs the WAV or OGG file, and optionally saves the generated parameters to a `.fart` text file for future editing.

```
fartbox -t <preset_name> -o <output.wav|output.ogg> [-save <output.fart>] [-s <seed_number>]
```

- `-t <preset_name>`: Specifies the sound preset generator.
- `-o <output.wav|output.ogg>`: Output target WAV or OGG file path. (The tool now supports outputting in OGG format in addition to WAV)
- `-save <output.fart>`: (Optional) Saves the randomized parameters of the generated preset to a text file.
- `-s <seed_number>`: (Optional) Integer seed for reproducible pseudo-random generation.

Pattern B: Direct Synthesis from Parameter File (`.fart`)

Reads an existing `.fart` parameter text file and renders it directly into a WAV or OGG file. This is the primary method for AI agents to write custom audio parameters.

```
fartbox -c <input.fart> -o <output.wav|output.ogg>
```

- `-c <input.fart>`: Source `.fart` parameter configuration file.
- `-o <output.wav|output.ogg>`: Output target WAV or OGG file path.

Pattern C: Mutation Workflow

Loads a `.fart` parameter file, applies slight random variations (mutations) to the active parameters, renders the mutated WAV or OGG, and optionally saves the new parameters.

```
fartbox -c <input.fart> -m -o <output.wav|output.ogg> [-save <mutated.fart>] [-s <seed_number>]
```

- `-c <input.fart>`: Base parameter file to mutate.
- `-m`: Enables mutation mode.
- `-o <output.wav|output.ogg>`: Output target WAV or OGG file path for mutated sound.
- `-save <mutated.fart>`: (Optional) Saves the newly mutated parameters.
- `-s <seed_number>`: (Optional) Seed controlling the mutation variance.

---

3. Preset Types Reference

The following preset identifiers can be passed to the `-t` option:

- `coin` / `pickup`: Short, high-pitched arpeggiated tones suited for item collection or scoring.
- `laser` / `shoot`: Frequency-sweeping downward tones for projectiles and energy weapons.
- `explosion`: Noise-based low-frequency rumble with pitch drop and distortion.
- `powerup`: Rising pitch sweeps with vibrato or arpeggiation for enhancements.
- `hit` / `hurt`: Short, impactful noise or square waves for damage feedback.
- `jump`: Upward frequency slides for character locomotion.
- `blip` / `select`: Extremely short, high-frequency tones for UI interaction.
- `random`: Unconstrained parameter randomization for abstract or unusual sound effects.

4. `.fart` Parameter Reference

The `.fart` parameter file is a plain-text key-value configuration file containing all synthesis parameters required by the `fartbox` synth core. Each line consists of `parameter_name = float_value`. Comments starting with `#` are ignored.

All parameters accept floating-point numbers. The valid parameter keys and their ranges/effects are detailed below:

Waveform Selection

- `wave_type` (Integer: 0, 1, 2, 3)
  - `0`: Square / Pulse wave (classic retro 8-bit chip synth).
  - `1`: Sawtooth wave (harsh, rich harmonic spectrum).
  - `2`: Sine wave (pure fundamental tone, smooth).
  - `3`: White Noise wave (percussive sounds, explosions, impacts).

Pitch & Frequency

- `p_base_freq` (Float: 0.0 to 1.0)
  - Primary synthesis frequency (pitch). Higher values yield higher tones.
- `p_freq_limit` (Float: 0.0 to 1.0)
  - Lower cutoff limit for frequency sweeps. If a downward frequency sweep hits this limit, generation terminates.
- `p_freq_ramp` (Float: -1.0 to 1.0)
  - Frequency slide rate. Negative values slide pitch down (lasers, hits); positive values slide pitch up (jumps, powerups).
- `p_freq_dramp` (Float: -1.0 to 1.0)
  - Acceleration of the frequency slide. Controls non-linear pitch curving.

Pulse Width / Duty Cycle (Square Wave Only)

- `p_duty` (Float: 0.0 to 1.0)
  - Pulse width modulation for square waves (`wave_type = 0`). Controls timbre thickness.
- `p_duty_ramp` (Float: -1.0 to 1.0)
  - Sweeps the pulse width over time, introducing dynamic harmonic motion.

Pitch Vibrato

- `p_vib_strength` (Float: 0.0 to 1.0)
  - Pitch modulation depth (vibrato intensity).

- `p_vib_speed` (Float: 0.0 to 1.0)
  - Modulation frequency / speed of vibrato.
- `p_vib_delay` (Float: 0.0 to 1.0)
  - Delay before vibrato modulation engages.

ADSR Envelope

- `p_env_attack` (Float: 0.0 to 1.0)
  - Attack time. 0.0 provides instant impact; higher values create a fade-in.
- `p_env_sustain` (Float: 0.0 to 1.0)
  - Hold time at peak amplitude.
- `p_env_decay` (Float: 0.0 to 1.0)
  - Release / decay duration. Controls overall sound length.
- `p_env_punch` (Float: 0.0 to 1.0)
  - Initial volume boost during attack phase, adding punch to impacts.

Low-Pass Filter (LPF)

- `p_lpf_freq` (Float: 0.0 to 1.0)
  - LPF cutoff frequency. 1.0 disables filter; lower values muffle the sound.
- `p_lpf_ramp` (Float: -1.0 to 1.0)
  - Cutoff frequency sweep over time.
- `p_lpf_resonance` (Float: 0.0 to 1.0)
  - Resonance multiplier near the cutoff frequency, adding squelch or metallic ringing.

High-Pass Filter (HPF)

- `p_hpf_freq` (Float: 0.0 to 1.0)
  - HPF cutoff frequency. Removes bass frequencies. Higher values result in thinner sounds.
- `p_hpf_ramp` (Float: -1.0 to 1.0)
  - HPF Cutoff sweep over time.

Phaser Effect

- `p_pha_offset` (Float: -1.0 to 1.0)
  - Phase cancellation offset, creating jet-engine or swooshing effects.
- `p_pha_ramp` (Float: -1.0 to 1.0)
  - Phase shift speed and direction.

Arpeggio & Repeat

- `p_repeat_speed` (Float: 0.0 to 1.0)
  - Triggers automatic re-attatck of the sound at specified intervals (stutter effect).
- `p_arp_speed` (Float: 0.0 to 1.0)
  - Speed of two-note pitch interval jumping.
- `p_arp_mod` (Float: -1.0 to 1.0)
  - Pitch interval for arpeggiation. Positive jumps pitch up (e.g. coin effect); negative drops pitch.

---

5. AI Workflow Guidelines

When an AI agent needs to generate sound assets for a TinCLE project, it must adhere to the following decision tree:

Step 1: Initial Prototype Phase

Always start by generating a baseline audio asset using a standard preset. Save the text configuration (`.fart`) alongside the WAV file so parameters can be inspected or modified later.

```
fartbox -t laser -o assets/sfx/laser.wav -save assets/sfx/laser.fart
```

Step 2: Variation Generation (Mutation)

If multiple distinct sound variations are required (e.g., subtle variations of footstep or explosion sounds), load the base `.fart` file and execute mutation mode.

```
fartbox -c assets/sfx/explosion_base.fart -m -o assets/sfx/explosion_var01.wav -save assets/sfx/explosion_var01.fart
        -s 101
fartbox -c assets/sfx/explosion_base.fart -m -o assets/sfx/explosion_var02.wav -save assets/sfx/explosion_var02.fart
        -s 102
```

Step 3: Precision Manual Tuning

When specific sound design constraints are requested (e.g., “Make the laser decay longer” or “Make the explosion lower pitched”):

1. Do NOT re-run randomized presets.
2. Open and inspect the existing `.fart` parameter file.
3. Modify specific parameters directly:

- To lengthen decay: Increase `p_env_decay`.
- To lower pitch: Decrease `p_base_freq`.
- To add crunch/punch: Increase `p_env_punch` or adjust `p_lpf_resonance`.

1. Re-render the audio directly from the edited `.fart` file:

```
fartbox -c assets/sfx/laser.fart -o assets/sfx/laser.wav
```

Rule A: Primary Output Contract (Chat AI Mode)

- **If you are acting as a Chat AI (Interactive Assistant without direct CLI access):**
  - **ALWAYS output ONLY the raw `.fart` text parameter block.**
  - **DO NOT attempt to render, compile, generate, or attach audio files (`.wav`).**
  - The frontend client environment (e.g., mobile chat apps) may fail to display or send rendered binary audio attachments, causing complete response silences.
  - The user will take your generated `.fart` text configuration and feed it directly into `fartbox` on their local machine. Outputting clean, copyable text parameters is your 100% complete and final deliverable.

Rule B: Autonomous Pipeline Executions (Agentic AI Mode)

- **If you are acting as an Autonomous Agent (with direct CLI execution access):**
  - Generate or edit the `.fart` parameter text file first.
  - Execute `fartbox -c <input.fart> -o <output.wav|output.ogg>` directly via CLI to render the audio asset into the project’s sound directory.

6. `.fart` Boilerplate Templates

Below are standard boilerplate configurations for common sound categories. AI agents should use these templates as starting baselines for direct editing or generation.

Template 1: Standard Laser / Projectile (`laser_base.fart`)

```
# fartbox Sound Parameter - Laser Base
wave_type = 0
p_base_freq = 0.600000
p_freq_limit = 0.200000
p_freq_ramp = -0.250000
p_freq_dramp = 0.000000
p_duty = 0.400000
p_duty_ramp = -0.100000
p_vib_strength = 0.000000
p_vib_speed = 0.000000
p_vib_delay = 0.000000
p_env_attack = 0.000000
p_env_sustain = 0.100000
p_env_decay = 0.200000
p_env_punch = 0.100000
p_lpf_resonance = 0.000000
p_lpf_freq = 1.000000
p_lpf_ramp = 0.000000
p_hpf_freq = 0.100000
p_hpf_ramp = 0.000000
p_pha_offset = 0.000000
p_pha_ramp = 0.000000
p_repeat_speed = 0.000000
p_arp_speed = 0.000000
p_arp_mod = 0.000000
```

Template 2: Heavy Explosion / Impact (`explosion_base.fart`)

```
# fartbox Sound Parameter - Explosion Base
wave_type = 3
p_base_freq = 0.250000
p_freq_limit = 0.000000
p_freq_ramp = -0.150000
p_freq_dramp = 0.000000
p_duty = 0.000000
p_duty_ramp = 0.000000
p_vib_strength = 0.200000
p_vib_speed = 0.300000
p_vib_delay = 0.000000
p_env_attack = 0.000000
p_env_sustain = 0.200000
p_env_decay = 0.500000
p_env_punch = 0.400000
p_lpf_resonance = 0.300000
p_lpf_freq = 0.800000
p_lpf_ramp = -0.100000
p_hpf_freq = 0.000000
p_hpf_ramp = 0.000000
p_pha_offset = -0.100000
p_pha_ramp = -0.100000
p_repeat_speed = 0.000000
p_arp_speed = 0.000000
p_arp_mod = 0.000000
```

Template 3: 8-Bit Coin / Item Pickup (`coin_base.fart`)

```
# fartbox Sound Parameter - Coin Base
wave_type = 0
p_base_freq = 0.450000
```

```
p_freq_limit = 0.000000
p_freq_ramp = 0.000000
p_freq_dramp = 0.000000
p_duty = 0.500000
p_duty_ramp = 0.000000
p_vib_strength = 0.000000
p_vib_speed = 0.000000
p_vib_delay = 0.000000
p_env_attack = 0.000000
p_env_sustain = 0.080000
p_env_decay = 0.250000
p_env_punch = 0.300000
p_lpf_resonance = 0.000000
p_lpf_freq = 1.000000
p_lpf_ramp = 0.000000
p_hpf_freq = 0.000000
p_hpf_ramp = 0.000000
p_pha_offset = 0.000000
p_pha_ramp = 0.000000
p_repeat_speed = 0.000000
p_arp_speed = 0.600000
p_arp_mod = 0.500000
```

Template 4: UI Cursor / Select Blip (`blip_base.fart`)

```
# fartbox Sound Parameter - UI Blip Base
wave_type = 0
p_base_freq = 0.350000
p_freq_limit = 0.000000
p_freq_ramp = 0.000000
p_freq_dramp = 0.000000
p_duty = 0.500000
p_duty_ramp = 0.000000
p_vib_strength = 0.000000
p_vib_speed = 0.000000
p_vib_delay = 0.000000
p_env_attack = 0.000000
p_env_sustain = 0.050000
p_env_decay = 0.100000
p_env_punch = 0.000000
p_lpf_resonance = 0.000000
p_lpf_freq = 1.000000
p_lpf_ramp = 0.000000
p_hpf_freq = 0.150000
p_hpf_ramp = 0.000000
p_pha_offset = 0.000000
p_pha_ramp = 0.000000
p_repeat_speed = 0.000000
p_arp_speed = 0.000000
p_arp_mod = 0.000000
```

SKILL_PN.md - TinCLE Tool Talent Chip [Networking]

• **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

~ Multi-User Dedicated Server Generator & Asynchronous Network API ~

1. Overview & Architecture

The TinCLE networking pipeline consists of three core components:

1. **`pubicnet` (CLI Tool)**
   - Generates/reads `uuid.txt` (36-byte UUID) in the project directory.
   - Binary-patches the template dynamic binary with the 36-character GAME_ID (UUID) and Game Title.
   - Outputs a standalone dedicated server executable: `my_server.exe`.
2. **`my_server.exe` (Dedicated Server)**
   - Listens on TCP port `8080` (default) for non-blocking packet broadcasting.
   - Validates client header MAGIC (`"PNET"`) and GAME_ID (UUID).
   - Automatically assigns unique `peer_id` (1~65535) to connected clients upon handshake.
   - Broadcasts incoming payload data to all clients residing in the same `room_id`.
3. **`tincle_net_*` API (Client Integration via `tincle.dll`)**
   - Non-blocking socket communication layer embedded inside TinCLE.
   - Cross-platform support (Win32 / WASM via Emscripten).

---

2. Server Generation Workflow (Step-by-Step)

Before invoking network APIs in client code (`game.c`), the AI Agent or developer must generate the dedicated game server executable.

Step 1: Execute `pubicnet`

Run the `pubicnet` tool from the project directory with the game title as a required argument. You may optionally specify a custom output executable name (defaults to `my_server.exe` if omitted):

```
# Default output (my_server.exe)
pubicnet "My Game Title"

# Custom output name
pubicnet "My Game Title" custom_server.exe
```

Step 2: Automatic UUID & Server Output

- If `uuid.txt` does NOT exist in the working directory, `pubicnet` generates a new UUID (e.g., `a1b2c3d4-5678-90ab-cdef-1234567890ab`) and saves it as `uuid.txt`.
- `pubicnet` reads the template library (`pubicnet.dll`), injects the target full 36-character GAME_ID (UUID) and Game Title into internal memory placeholders, and outputs the patched server binary (`my_server.exe` or specified name).

Step 3: Launch Dedicated Server

Run `my_server.exe` to start listening for client connections:

```
my_server -p 8080
```

Console Output:

```
======================================================================
 [TinCLE Version 0.11] My Game Title Server Started!
 [Port] 8080
======================================================================
```

Step 4: Server Packaging & Release Preparation

The generated server executable is unique to your game build and **MUST be distributed alongside the game client**.

To ensure safety, stability, and ease of use, follow these distribution guidelines:

1. **Apply `poopee` Post-Processing (CRITICAL):**

- Before distributing the server executable, you **MUST** run `poopee` on the generated server EXE to embed the application icon and OS manifest.
- This step is strongly recommended to prevent false-positive antivirus flags (such as Windows Defender) on user machines.

```
poopee my_server.exe icon.ico manifest.xml
```

1. **Standalone Server ZIP Distribution:**

- For maximum user convenience, create a separate standalone ZIP archive specifically for host players/server admins (e.g., `MyGame_Dedicated_Server.zip`).
- The archive should contain the following 3 files:
  1. `my_server.exe` (The `poopee`-processed server executable)
  2. `CREDITS.txt` (License and attribution credits)
  3. `readme.txt` (Clear documentation explaining what the server is, how to open ports, and how to launch it)

---

3. Client API Reference

All networking functions are exposed via `libs/tincle.h`.

Function	Return Type	Description
`tincle_net_set_game_id(const char* game_id_or_uuid)`	`void`	Configures full 36-character GAME_ID (UUID) used for server authentication. **Must be called before `tincle_net_connect`**.
`tincle_net_connect(const char* ip, int port, uint16_t room_id)`	`int`	Initiates asynchronous TCP connection to the server and triggers automatic Peer ID handshake. Returns `1` on socket creation success, `0` on failure.
`tincle_net_disconnect()`	`void`	Closes network socket and resets client Peer ID state.
`tincle_net_send(const uint8_t* data, uint16_t size)`	`int`	Broadcasts raw binary payload to all clients in the current `room_id`. Returns `1` on success, `0` on failure.
`tincle_net_poll(uint16_t* out_sender_peer_id, uint8_t* out_buffer, uint16_t max_size)`	`int`	Non-blocking receive function. Must be called repeatedly in the game update loop. Returns received payload byte count (`0` if no data). **If `out_sender_peer_id` is `0`, `out_buffer` contains the `uint16_t` Peer ID of a disconnected client sent by the server.**
`tincle_net_get_peer_id()`	`uint16_t`	Returns the assigned unique Client Peer ID (`1`~`65535`). Returns `0` if not yet connected/handshaken.

Disconnect & Timeout Handling Mechanism

To maintain a responsive multiplayer state, clients must handle disconnection using a two-stage approach inside the `tincle_net_poll` loop:

1. **Visual Hide Timeout (Local Client Side - 2 Seconds):**
   - If a peer's position data has not arrived for more than 2 seconds (`PEER_TIMEOUT_SEC`), the client temporarily hides the player from the screen (e.g., `active = false`). This prevents dead/lagging players from lingering on the screen.
2. **Server Logout Notification (System Event - 0s or 30s):**
   - When `tincle_net_poll` returns `out_sender_peer_id == 0`, it indicates a system event directly sent from the server.
   - The payload contains a `uint16_t` representing the `disconnected_peer_id`.
   - **Immediate Disconnect (0s):** Sent instantly when a client performs a graceful close/disconnect.
   - **Timeout Disconnect (30s):** Sent after 30 seconds if a client suffers a hard crash or network disruption without sending a WebSocket close frame.
   - Upon receiving `out_sender_peer_id == 0`, the client must completely clean up and release the specified peer's state, regardless of whether the player was already visually hidden by the 2-second local check.

---

4. Minimal Network Implementation Boilerplate

Critical Architecture Rules for AI Agents

When designing a TinCLE multiplayer game, AI Agents **MUST** strictly adhere to the following architecture rules:

1. **Do NOT Rely on `uuid.txt` in the Distributed Game:**
   - The `uuid.txt` file is **ONLY** used locally during the server generation build process with `pubicnet`.
   - Never assume `uuid.txt` will exist on the player's machine. The client code (`game.c`) **MUST** embed or hardcode the full 36-character GAME_ID (UUID) directly in C variables and pass it to `tincle_net_set_game_id()`.
2. **Always Provide a Server IP / Port Configuration UI:**
   - **NEVER** hardcode the destination IP address to `127.0.0.1` (localhost) or a fixed IP inside the client code.
   - Hardcoding `127.0.0.1` completely breaks real online multiplayer across different machines.
   - You **MUST** implement an in-game UI screen (or input prompt/config file loader) that allows players to freely enter and save the target Server IP address and Port number before attempting to call `tincle_net_connect()`.
3. **UUID Acquisition Rule by AI Agent Type (CRITICAL):**
   - The 36-character GAME_ID (UUID) passed to `tincle_net_set_game_id()` MUST strictly match the 36-characters UUID embedded inside the generated server (`my_server.exe`). Never invent or hallucinate a random dummy UUID!
   - **For Autonomous / Agentic AIs (Tools & CLI execution enabled):** You MUST execute `pubicnet "<Game Title>"` FIRST in the terminal to generate `uuid.txt` and `my_server.exe`. Then, read the generated UUID from `uuid.txt` and embed the full 36-characters directly into the client C code (`game.c`).
   - **For Conversational / Chat AIs (No direct terminal execution):** You CANNOT run `pubicnet` by yourself. If the user has NOT provided the target UUID or `uuid.txt` content yet, you **MUST explicitly ask the user to provide the UUID** (or run `pubicnet` first) BEFORE writing the `tincle_net_set_game_id()` code. NEVER invent a placeholder UUID, as mismatched GAME_IDs will permanently block all network connections!

Boilerplate

This boilerplate demonstrates how an AI Agent should write client code (`game.c`) within the official TinCLE state lifecycle to establish connections, send player state, and poll incoming broadcast data without blocking the render thread.

CRITICAL Note: Always call `tincle_net_set_game_id()` before `tincle_net_connect()` to ensure the full 36-character GAME_ID (UUID) matches the server.

```c
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include <stdio.h>
#include <stdint.h>

#ifdef __EMSCRIPTEN__
    #include <SDL2/SDL_scancode.h>
#else
    #include "libs/SDL_scancode.h"
    #include <windows.h>
#endif

// Network Packet Structure
typedef struct {
    float x;
    float y;
} PlayerNetworkPacket;

// Global States & Storage
static int g_font_medium = -1;
static float g_my_x = 320.0f;
static float g_my_y = 240.0f;

// Table for other players' positions (Supports up to 16 peers)
static float g_other_x[16] = {0};
static float g_other_y[16] = {0};
static bool  g_other_active[16] = {false};
static float g_other_last_seen[16] = {0};

// Timeout threshold for peer disconnections (seconds)
#define PEER_TIMEOUT_SEC 2.0f

// State 0: Gameplay Logic Updates
void play_update(float dt) {
    static float s_total_time = 0.0f;
    s_total_time += dt;

    // 1. Network Receive Polling (Process all incoming packets)
    uint16_t sender_peer_id = 0;
    uint8_t recv_buf[128]; // Receive via buffer to also catch system notifications (leave events)
    int recv_len = 0;

    // 1. Network Receive Polling
    while ((recv_len = tincle_net_poll(&sender_peer_id, recv_buf, sizeof(recv_buf))) > 0) {

        // sender_peer_id == 0 indicates a system notification from the server (immediate disconnect or 30-sec
         timeout disconnect)
        if (sender_peer_id == 0) {
            if (recv_len >= (int)sizeof(uint16_t)) {
                uint16_t left_peer_id = *(uint16_t*)recv_buf;
                if (left_peer_id > 0 && left_peer_id < 16) {
                    // Output a "completely logged out" log, regardless of whether the player was already visually
         hidden by the 2-sec timeout!
                    printf("[Net] Peer %d completely logged out (Server Notification)!\n", left_peer_id);

                    g_other_active[left_peer_id] = false; // Completely release and reset the player state
                }
            }
```

```
        }
        // Normal position synchronization packets from other players
        else if (sender_peer_id != tincle_net_get_peer_id() && sender_peer_id < 16) {
            if (recv_len >= (int)sizeof(PlayerNetworkPacket)) {
                PlayerNetworkPacket* recv_packet = (PlayerNetworkPacket*)recv_buf;
                g_other_x[sender_peer_id] = recv_packet->x;
                g_other_y[sender_peer_id] = recv_packet->y;

                g_other_active[sender_peer_id] = true;
                g_other_last_seen[sender_peer_id] = s_total_time;
            }
        }
    }

    // 2. Visual Timeout Check (Emergency visual hide processing)
    for (int i = 1; i < 16; i++) {
        if (i != tincle_net_get_peer_id() && g_other_active[i]) {
            // Temporarily hide the player from the screen if packets stop arriving for 2 seconds
            if (s_total_time - g_other_last_seen[i] > PEER_TIMEOUT_SEC) {
                printf("[Net] Peer %d position data paused (Visual Hide)\n", i);
                g_other_active[i] = false;
            }
        }
    }

    // 3. Local Player Input Handling
    float move_speed = 200.0f;
    if (tincle_key_pressed(SDL_SCANCODE_LEFT))  g_my_x -= move_speed * dt;
    if (tincle_key_pressed(SDL_SCANCODE_RIGHT)) g_my_x += move_speed * dt;
    if (tincle_key_pressed(SDL_SCANCODE_UP))    g_my_y -= move_speed * dt;
    if (tincle_key_pressed(SDL_SCANCODE_DOWN))  g_my_y += move_speed * dt;

    // 4. Send Local Position to Server (Only after Peer ID is assigned)
    uint16_t my_id = tincle_net_get_peer_id();
    if (my_id != 0) {
        PlayerNetworkPacket my_packet;
        my_packet.x = g_my_x;
        my_packet.y = g_my_y;

        tincle_net_send((const uint8_t*)&my_packet, sizeof(PlayerNetworkPacket));
    }
}

// State 0: Gameplay Rendering
void play_draw(void) {
    // Clear background (Dark Navy)
    tincle_clear((TinCLE_Color){20, 20, 50, 255});

    if (g_font_medium != -1) {
        tincle_set_font(g_font_medium);
    }

    uint16_t my_id = tincle_net_get_peer_id();
    char buf[64];

    // Render assigned Peer ID
    sprintf(buf, "MY PEER ID: %d", my_id);
    tincle_draw_text(buf, 20, 20, (TinCLE_Color){0, 255, 0, 255});

    // Render Local Player (Yellow)
```

```
    sprintf(buf, "P%d", my_id);
    tincle_draw_text_center(buf, (int)g_my_x, (int)g_my_y, (TinCLE_Color){255, 255, 0, 255});

    // Render Remote Peers (Magenta)
    for (int i = 1; i < 16; i++) {
        if (i != my_id && g_other_active[i]) {
            sprintf(buf, "P%d", i);
            tincle_draw_text_center(buf, (int)g_other_x[i], (int)g_other_y[i], (TinCLE_Color){255, 0, 255, 255});
        }
    }
}

// Engine & Network Initialization
void game_load(void) {
    // Load default font asset
    g_font_medium = tincle_load_font("font.ttf", 24);

    // Set full 36-character Game UUID (Must match server config)
    tincle_net_set_game_id("406230b1-ae94-05b7-e9af-541d23f313c7");

    // Establish connection to pubicnet server (IP, Port, RoomID)
    if (tincle_net_connect("127.0.0.1", 8080, 1)) { // Server IP Address & Port
        printf("[Net] Connecting to pubicnet server...\n");
    }

    tincle_add_state(0, play_update, play_draw);
    tincle_change_state(0);
}

// Application Entry Point
#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) {
    tincle_start(game_load, "My Net Game", true, 640, 480, TINCLE_MODE_2D_NORMAL);
    return 0;
}
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) {
    tincle_start(game_load, "My Net Game", true, 640, 480, TINCLE_MODE_2D_NORMAL);
    return 0;
}
#endif
```

SKILL_AON1.md - TinCLE Add-On Talent Chip [2D/3D Physics, GUI Toolkit & Local LLM]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

1. cannoli (Collision And Newtonian Network Object Linked Interface)

Module: cannoli (`cannoli.dll` / `cannoli.wasm`) **Description:** 2D & 3D Physics Extension (Box2D & Box3D Full Export)

- **Overview & Usage:** The distribution includes `cannoli.dll` (for Windows) and `libs/cannoli.a` (for Web/WASM). By adding `#include "libs/cannoli.h"` at the top of your source code, the complete, uninhibited set of Pure C Box2D v3.0 (2D physics) and Box3D v0.1 (3D physics) APIs becomes fully available across both Windows (`tinclec`) and Web (`emcass`) build targets. If you use this feature, you must include `cannoli.dll` in the distribution package for Windows.
- **AI Coding Rule:** You can directly invoke any official Box2D (e.g., `b2CreateWorld`, `b2CreateBody`, `b2Body_GetPosition`) and Box3D C functions without external wrappers. Treat `cannoli.h` as a native, unified 2D/3D physics extension of the TinCLE environment.

Additional API Reference

Function Signature	Description & Critical Notes [Args / Return]
`const char* cannoli_get_version(void);`	Returns read-only string of cannoli version (e.g., `"0.18"`).

Distribution

When using cannoli, you must include either cannoli.dll or cannoli.wasm-whichever is appropriate for the target platform-in your distribution. An example of the minimum set of distribution files, including these, is as follows.

Platform	Required files
Windows	`game.exe`, `tincle.dll`, `SDL2.dll`,`SDL2_gfx.dll`, `SDL2_ttf.dll`, `game.data`, `CREDITS.txt`, `cannoli.dll`
WebAssembly	`index.html`, `index.js`, `index.wasm`, `index.data`, `cannoli.wasm`

2. gatochoc (Gui And Toolkit for On-screen Components, Huds, and Object Controls)

Module: gatochoc (`gatochoc.dll` / `gatochoc.wasm`) **Description:** Lightweight Nuklear-based GUI extension for TinCLE (Supports SDL_Renderer & OpenGL GL2/GLES2 backends).

1. Extension API Reference

Function Signature	Description & Critical Notes [Args / Return]
`bool gatochoc_init(gatochoc_mode_t mode, SDL_Window *win, void *target);`	**UI Engine Init.** Call inside `game_load`. * `mode`: `GATOCHOC_MODE_SDL_RENDERER` or `GATOCHOC_MODE_GLSL`. * `win`: Pointer to `SDL_Window` (via `tincle_get_window()`). * `target`: `SDL_Renderer*` (SDL mode via `tincle_get_renderer()`) or `SDL_GLContext` (GLSL mode). * **Return:** `true` if success, `false` on failure.
`bool gatochoc_handle_event(SDL_Event *event);`	**Input Forwarder.** Pass raw `SDL_Event*` here inside the event callback (`tincle_set_event_callback`). Automatically triggers `nk_input_begin` on first event. * **Return:** `true` if UI consumed the event, `false` otherwise.
`struct nk_context* gatochoc_get_context(void);`	**Nuklear Context Accessor.** Returns pointer to internal `struct nk_context*`. Use this to call standard Nuklear GUI functions (e.g., `nk_begin`, `nk_layout_row_dynamic`, `nk_button_label`, `nk_end`).
`void gatochoc_input_begin(void);`	

Function Signature	Description & Critical Notes [Args / Return]
	Input Frame Flush. Call at the very top of `update_cb` before building any Nuklear UI. Flushes and finalizes pending input events (`nk_input_end`) for the current frame.
`void gatochoc_input_end(void);`	**Input Cleanup.** Frame input close handler (reserved / optional for explicit multi-pass input states).
`GatoChocFont gatochoc_load_font_from_memory(const void* data, size_t size, float font_size, GatoChocFontRange range_type);`	**Standard Font Loader.** Loads TrueType font (`.ttf`) from memory buffer. * `range_type`: `GATOCHOC_FONT_DEFAULT` (ASCII) or `GATOCHOC_FONT_JAPANESE` (Full Japanese including full-width symbols `0xFF00-0xFFEF`) .
`GatoChocFont gatochoc_load_font_from_memory_ex(const void* data, size_t size, float font_size, GatoChocFontRange range_type, const unsigned long *custom_ranges);`	**Extended Font Loader.** Custom glyph range font loader. Set `range_type` to `GATOCHOC_FONT_CUSTOM` and pass a zero-terminated `nk_rune` array to `custom_ranges` for multilingual support (e.g., Chinese, Korean).
`void gatochoc_set_default_font(GatoChocFont font);`	**Default Font Setter.** Sets the loaded `GatoChocFont` as the global default font for all Nuklear UI elements.
`void gatochoc_render(void);`	**UI Draw Pass.** Renders generated Nuklear geometry. Call inside `draw_cb`, right after `tincle_clear` and game rendering. Auto-finalizes font atlas on first frame if required.
`void gatochoc_shutdown(void);`	**Cleanup.** Frees all Nuklear contexts, font atlases, and backend resources. Call on game exit.
`const char* gatochoc_get_version(void);`	Returns read-only string of gatochoc version (e.g., `"0.18"`).

2. Core Types & Enums

```c
// Backend Rendering Mode
typedef enum gatochoc_mode {
    GATOCHOC_MODE_SDL_RENDERER = 0, // 2D SDL_Renderer Pipeline
    GATOCHOC_MODE_GLSL         = 1  // OpenGL GL2 / GLES2 Pipeline
} gatochoc_mode_t;

// Font Handle Type
typedef struct nk_font* GatoChocFont;

// Unicode Glyph Range Preset
typedef enum {
    GATOCHOC_FONT_DEFAULT  = 0, // Basic ASCII only
    GATOCHOC_FONT_JAPANESE = 1, // Full Japanese (Hiragana, Katakana, Kanji & Full-width Symbols)
    GATOCHOC_FONT_CUSTOM   = 2  // User-defined custom Unicode range array
} GatoChocFontRange;
```

3. Integration Pipeline & Architecture Notes

Load & Link Blueprint

1. **Direct Linkage (No Special Loader Needed):**

- Link `gatochoc.dll` (Desktop) or `gatochoc.wasm` / `gatochocmt.wasm` (WebAssembly Side Module) directly during compilation with `tcc` or `emcc`.
- Bind `on_sdl_event` using `tincle_set_event_callback(on_sdl_event)` to pass raw SDL inputs straight to `gatochoc_handle_event`.

1. **Flexible Backend Selection (TinCLE Dual Rendering Engine):**

- **`GATOCHOC_MODE_SDL_RENDERER`**: Use when running TinCLE with `TINCLE_MODE_2D_NORMAL`. Standard 2D GUI overlay using `SDL_Renderer*`.

- **`GATOCHOC_MODE_GLSL`**: Use when running TinCLE with `TINCLE_MODE_GLSL_HERO`. Renders directly using OpenGL / GLES2 shaders via native `SDL_GLContext`.

1. **Frame Execution Sequence (CRITICAL RULE):**

- **Event Pass:** Inputs are forwarded via `tincle_set_event_callback` -> `gatochoc_handle_event`.
- **Update Pass (`update_cb`):**
  1. Call `gatochoc_input_begin()` at the **very first line**.
  2. Build Nuklear UI controls (`nk_begin` ... `nk_end`).
  3. Call `gatochoc_input_end()` at the **very last line** (required for future frame-state extensions).
- **Draw Pass (`draw_cb`):** Call `gatochoc_render()` right after `tincle_clear` to draw UI on top.

Distribution

When using gatochoc, you must include either gatochoc.dll or gatochoc.wasm-whichever is appropriate for the target platform-in your distribution. An example of the minimum set of distribution files, including these, is as follows.

Platform	Required files
Windows	`game.exe`, `tincle.dll`, `SDL2.dll`,`SDL2_gfx.dll`, `SDL2_ttf.dll`, `game.data`, `CREDITS.txt`, `gatochoc.dll`
WebAssembly	`index.html`, `index.js`, `index.wasm`, `index.data`, `gatochoc.wasm`

---

4. Standard Integration Boilerplate (`game.c`)

- **Required:** To use `gatochoc`, you must include the dedicated header file by adding `#include “libs/gatochoc.h”`.

```c
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include "libs/gatochoc.h"
#include <stdio.h>

#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL.h>
  #include <SDL2/SDL_scancode.h>
#else
  #include "libs/SDL_scancode.h"
  #include <windows.h>
#endif

// --- Global Variables ---
static int g_counter = 0;
static GatoChocFont g_jp_font = NULL;

// --- SDL Event Forwarder Callback ---
bool on_sdl_event(SDL_Event *event) {
    // Forward raw event to GATOCHOC. Returns true if consumed by UI.
    return gatochoc_handle_event(event);
}

// --- Frame Update Callback ---
void main_update(float dt) {
    // 1. Flush & Lock frame input state (MUST BE FIRST)
    gatochoc_input_begin();

    // 2. Build Nuklear GUI Controls
    struct nk_context *ctx = gatochoc_get_context();
```

```
    if (ctx) {
        nk_flags flags = NK_WINDOW_BORDER | NK_WINDOW_MOVABLE | NK_WINDOW_SCALABLE | NK_WINDOW_TITLE;
        if (nk_begin(ctx, "TinCLE x Nuklear GUI", nk_rect(50, 50, 280, 200), flags)) {

            nk_layout_row_dynamic(ctx, 30, 1);
            nk_label(ctx, "Hello Michyo! Nuklear Active!", NK_TEXT_LEFT);

            char buf[64];
            snprintf(buf, sizeof(buf), "Count: %d", g_counter);
            nk_label(ctx, buf, NK_TEXT_LEFT);

            if (nk_button_label(ctx, "Push Me!")) {
                g_counter++;
            }
        }
        nk_end(ctx);
    }

    // 3. Close frame input state (MUST BE LAST in update)
    gatochoc_input_end();
}

// --- Frame Render Callback ---
void main_draw(void) {
    // 1. Clear background
    tincle_clear((TinCLE_Color){255, 40, 60, 255});

    // 2. Render Nuklear GUI overlay on top
    gatochoc_render();
}

// --- Engine Load & Initialization ---
void game_load(void) {
    // 1. Initialize GATOCHOC with TinCLE rendering target
    gatochoc_init(GATOCHOC_MODE_SDL_RENDERER, tincle_get_window(), tincle_get_renderer());

    // 2. Bind event callback to forward inputs
    tincle_set_event_callback(on_sdl_event);

    // 3. Load Japanese Font Asset
    size_t font_size = 0;
    unsigned char* font_data = tincle_load_asset("font.ttf", &font_size);
    if (font_data) {
        g_jp_font = gatochoc_load_font_from_memory(font_data, font_size, 18.0f, GATOCHOC_FONT_JAPANESE);
        if (g_jp_font) {
            gatochoc_set_default_font(g_jp_font);
        }
        free(font_data);
    }

    // 4. Register State 0 & Transition
    tincle_add_state(0, main_update, main_draw);
    tincle_change_state(0);
}

// --- Entry Point ---
#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) {
    tincle_start(game_load, "TinCLE Nuklear Demo", true, 640, 480, TINCLE_MODE_2D_NORMAL);
    gatochoc_shutdown();
```

```
    return 0;
}
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) {
    tincle_start(game_load, "TinCLE Nuklear Demo", true, 640, 480, TINCLE_MODE_2D_NORMAL);
    gatochoc_shutdown();
    return 0;
}
#endif
```

3. gozinaki (Generative Output & Zero-shot Intelligence Node for Autonomous Knowledge Interaction)

Module: gozinaki (`gozinaki.dll` / `gozinaki.wasm`) **Description:** Local LLM Inference Node based on llama.cpp (MIT License). Designed for real-time game loops via non-blocking step-by-step generation and chunked prompt evaluation.

- **Overview & Usage:** The distribution includes `gozinaki.dll` and its backend dependencies (`libllama.dll`, `ggml.dll`, `ggml-base.dll`, `ggml-cpu.dll`) for Windows, and `gozinaki.wasm` for Web/WASM. By including `#include "libs/gozinaki.h"`, AI agents can dynamically generate NPC dialogue, item descriptions, and contextual narratives.
- **AI Coding Rule:** Models (`.gguf` files) MUST NOT be embedded inside `index.data`. They must be fetched asynchronously using TinCLE's `tincle_fetch_file_async()` API. Inference and prompt evaluation are executed progressively by calling `gozinaki_step()` once per frame in the `update` loop to guarantee 60fps rendering without blocking the main thread.

1. Extension API Reference

Function Signature	Description & Critical Notes [Args / Return]
`bool gozinaki_init(const char* model_path, int n_ctx, int n_threads);`	**LLM Engine Init.** Loads the GGUF model from the path retrieved via `tincle_fetch_get_path()`. `n_ctx`: Context size (use 0 for model default). `n_threads`: Thread count (use 0 for auto). Returns `true` on success.
`void gozinaki_free(void);`	**Cleanup.** Completely unloads the model and frees all internal contexts.
`bool gozinaki_is_initialized(void);`	**Status Check.** Returns `true` if the model is currently loaded and ready.
`bool gozinaki_set_prompt(const char* system_prompt, const char* user_prompt, int max_tokens);`	**Prompt Injection.** Starts a new generation session. `system_prompt` can be NULL/empty. Automatically tokenizes and sets status to `GOZINAKI_STATUS_EVALUATING`. Returns `true` if successfully started.
`GozinakiStatus gozinaki_step(void);`	**Non-blocking Step Engine. [CRITICAL]** Call this exactly once per frame inside your `update_cb` while the status is `GOZINAKI_STATUS_EVALUATING` or `GOZINAKI_STATUS_GENERATING`. It processes prompt chunks and generates tokens to maintain frame rate.
`void gozinaki_abort(void);`	**Force Cancel.** Interrupts active evaluation/generation and returns the engine to the IDLE state.
`const char* gozinaki_get_text(void);`	**Full Text Accessor.** Returns the complete generated UTF-8 text so far. Pointer is managed internally; do not free.
`const char* gozinaki_get_delta_text(void);`	**Delta Text Accessor.** Returns only the new token string generated in the very last `gozinaki_step()`. Ideal for triggering typing sound effects. Returns `""` if no new characters were formed.
`GozinakiStatus gozinaki_get_status(void);`	**State Polling.** Returns the current `GozinakiStatus` enum.

2. Core Types & Enums

```
typedef enum {
    GOZINAKI_STATUS_IDLE       = 0,  // Ready to accept a new prompt
    GOZINAKI_STATUS_GENERATING = 1,  // Currently generating text (call gozinaki_step)
    GOZINAKI_STATUS_FINISHED   = 2,  // Generation complete (EOS reached or max_tokens hit)
```

```
    GOZINAKI_STATUS_EVALUATING = 3,  // Reading prompt in chunks to prevent freezes (call gozinaki_step)
    GOZINAKI_STATUS_ERROR      = -1  // Internal error occurred
} GozinakiStatus;
```

3. Integration Pipeline & Architecture Notes

Model Loading & Fetching (CRITICAL RULE)

Large `.gguf` files (e.g., SmolLM2-135M or Qwen2.5-0.5B) must be stored alongside the executable/HTML, **not** inside `exsass.txt` data packs. They must be loaded using TinCLE's asynchronous fetch API to prevent browser freezing during WASM initialization.

WASM Compilation Flags (CRITICAL RULE)

When compiling a web build using `gozinaki.wasm`, the AI agent MUST instruct the user to append the following memory and module flags to their `emcass` or `emcc` command. Failure to do so will result in a `Stack overflow` or `memory access out of bounds` crash:

```
-s MAIN_MODULE=2 -s INITIAL_MEMORY=1073741824 -s ALLOW_MEMORY_GROWTH=1 -s MAXIMUM_MEMORY=2147483648 -s
        STACK_SIZE=16777216
```

Event Edge-Detection Requirement

When triggering `gozinaki_set_prompt` via user input (e.g., keyboard or gamepad), the AI agent MUST implement edge-detection (checking if the key was *just* pressed this frame). Do not evaluate continuous key-down states, as 60fps loops will repeatedly reset the LLM context and cause freezes.

Distribution

When using gozinaki, you must include the respective dynamic modules and the external `.gguf` model file in your distribution.

Platform	Required files
Windows	`game.exe`, `tincle.dll`, `SDL2.dll`, `SDL2_gfx.dll`, `SDL2_ttf.dll`, `game.data`, `CREDITS.txt`, `gozinaki.dll`, `libllama.dll`, `ggml.dll`, `ggml-base.dll`, `ggml-cpu.dll`, `model.gguf`
WebAssembly	`index.html`, `index.js`, `index.wasm`, `index.data`, `gozinaki.wasm`, `model.gguf`

4. Standard Integration Boilerplate

```
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include "libs/gozinaki.h"
#include <stdio.h>

static TinCLE_AsyncFile* g_model_file = NULL;
static bool g_llm_ready = false;
static bool g_prev_space = false;

void main_update(float dt) {
    // 1. Handle Asynchronous Model Loading
    if (!g_llm_ready) {
        float progress = 0.0f;
        if (tincle_fetch_get_status(g_model_file, &progress) == TINCLE_ASYNC_READY) {
            // Context size 1024 is recommended for lightweight NPC chat
            if (gozinaki_init(tincle_fetch_get_path(g_model_file), 1024, 0)) {
                g_llm_ready = true;
                gozinaki_set_prompt("You are a helpful NPC.", "Welcome to TinCLE!", 128);
```

```
            }
        }
        return;
    }

    // 2. Step-by-Step Non-blocking Engine (Handles both Evaluating & Generating)
    GozinakiStatus status = gozinaki_get_status();
    if (status == GOZINAKI_STATUS_EVALUATING || status == GOZINAKI_STATUS_GENERATING) {
        gozinaki_step();

        if (status == GOZINAKI_STATUS_GENERATING) {
            const char* delta = gozinaki_get_delta_text();
            if (delta && delta[0] != '\0') {
                // Optional: Play typing sound using tincle_play_sound() here
            }
        }
    }

    // 3. Trigger new prompt using Edge-Detection
    if (status == GOZINAKI_STATUS_FINISHED) {
        bool curr_space = tincle_key_pressed(SDL_SCANCODE_SPACE);
        if (curr_space && !g_prev_space) {
            gozinaki_set_prompt("You are an evil knight.", "Show me your power!", 128);
        }
        g_prev_space = curr_space;
    }
}

void main_draw(void) {
    tincle_clear((TinCLE_Color){0, 0, 40, 255});

    if (!g_llm_ready) {
        tincle_draw_text("Loading AI Model...", 20, 20, (TinCLE_Color){150, 150, 150, 255});
        return;
    }

    GozinakiStatus status = gozinaki_get_status();
    if (status == GOZINAKI_STATUS_EVALUATING) {
        tincle_draw_text("AI is thinking...", 20, 20, (TinCLE_Color){255, 255, 100, 255});
    } else {
        tincle_draw_text(gozinaki_get_text(), 20, 20, (TinCLE_Color){255, 255, 255, 255});
    }
}

void game_load(void) {
    // Initiate background download of the model
    g_model_file = tincle_fetch_file_async("model.gguf");

    tincle_add_state(0, main_update, main_draw);
    tincle_change_state(0);
}
// Note: Entry point (tincle_start) omitted for brevity. Follow standard SKILL.md template.
```

SKILL_AON2.md - TinCLE Add-On Talent Chip [VRML 3D World Creation & Isometric Pixel Rendering & Neural Network]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

1. vtrushka (Vrml TRUly SHaping Kosmos Agent)

1.1. Overview

- **Purpose & Value:** High-level 3D world loader and software renderer pipeline for TinCLE. Completely eliminates the need to manually manage 3D vertex buffers, transformation matrices, camera projections, or custom sorting in pure C.
- **Declarative 3D World Creation:** Enables developers to build complex 3D environments, animated models, materials, and textures using standard **VRML 2.0 (`.wrl`)** files exported from 3D DCC tools (e.g. Blender).
- **Zero-Friction Engine Integration:** Seamlessly plugs into TinCLE's runtime loop via `vtrushka_update(dt)` and `vtrushka_draw()`, leveraging TinCLE's internal `SDL_RenderGeometry` backend for both Windows (Desktop EXE) and WebAssembly (Emscripten HTML5).
- **C-Driven Interactive Logic:** Replaces JavaScript inside VRML `Script` nodes with high-performance C callback handlers (`vtrushka_script_set_handler`), allowing direct manipulation of 3D transforms, switches, and touch sensor events.
- **Distribution & Runtime:** Simply `#include "libs/vtrushka.h"` in your C source, and ship `vtrushka.dll` (Windows) or `vtrushka.wasm` (WebAssembly) alongside your game executable/bundle.
- **Requirements:** vtrushka requires latest version of TinCLE, which must expose: `tincle_load_asset()`, `tincle_load_image()`, `tincle_get_texture()`, `tincle_create_image_from_pixels()`, `tincle_get_renderer()`

1.2. Distribution

When using vtrushka, you must include either vtrushka.dll or vtrushka.wasm-whichever is appropriate for the target platform-in your distribution. An example of the minimum set of distribution files, including these, is as follows.

Platform	Required files
Windows	`game.exe`, `tincle.dll`, `SDL2.dll`,`SDL2_gfx.dll`, `SDL2_ttf.dll`, `game.data`, `CREDITS.txt`, `vtrushka.dll`
WebAssembly	`index.html`, `index.js`, `index.wasm`, `index.data`, `vtrushka.wasm`

1.3. API Reference

1.3.1. System Management

Function Signature	Description & Critical Notes [Args / Return]
`void vtrushka_init(SDL_Renderer* renderer);`	**Main entry.** Call once at game startup. Pass `tincle_get_renderer()`. Initializes all 64 model slots. Must be called before any other vtrushka API.
`void vtrushka_shutdown(void);`	Frees all loaded models, script handlers, and internal scratch buffers. Call at game shutdown. Safe to call if not initialized.
`int vtrushka_load_wrl(const char* filename);`	**Loads a .wrl file via `tincle_load_asset()`.** Auto-detects header. **VRML 2.0 only** - returns `VTRUSHKA_INVALID_ID` (-1) for VRML 1.0. On success returns a slot ID (0-63). Returns -1 on parse error, missing file, or no free slot.
`void vtrushka_unload(int vrml_id);`	Frees the model and all associated resources. Also clears any script handlers registered for this slot. Safe to call with invalid ID.
`const char* vtrushka_get_version(void);`	Returns read-only string of vtrushka version (e.g., `"0.18"`).

1.3.2. Camera

Function Signature	Description & Critical Notes [Args / Return]
`void vtrushka_camera_default(VtrushkaCamera* cam);`	Fills the struct with defaults: position (0,0,10), yaw=0, pitch=0, fov_y=45 deg, near=0.1, far=1000.
`void vtrushka_set_camera(int vrml_id, const VtrushkaCamera* cam);`	Replaces the entire camera state.
`void vtrushka_get_camera(int vrml_id, VtrushkaCamera* out_cam);`	Reads current camera state.
`void vtrushka_camera_move(int vrml_id, float forward, float right, float up);`	**FPS-style translation.** `forward` moves along the view direction, `right` is strafe, `up` is world-Y offset. Units are world units. Combine with `dt` in your game loop.
`void vtrushka_camera_rotate(int vrml_id, float dyaw, float dpitch);`	**FPS-style rotation.** `dyaw` rotates around world Y (positive = turn right). `dpitch` tilts up/down, **auto-clamped to +/- 89 deg** to prevent gimbal flip.
`void vtrushka_camera_bind_viewpoint(int vrml_id, int viewpoint_node_id);`	Alias for `vtrushka_viewpoint_bind`.

Related struct:

```c
typedef struct {
    float position[3];  /* World coordinates */
    float yaw;          /* Radians. 0 = looking down -Z. */
    float pitch;        /* Radians. Positive = looking up. */
    float fov_y;        /* Vertical FOV in radians. Default 45 deg. */
    float near_z, far_z;
} VtrushkaCamera;
```

1.3.3. Viewpoint (Scene-Defined Camera Positions)

Function Signature	Description & Critical Notes [Args / Return]
`int vtrushka_viewpoint_count(int vrml_id);`	Number of `Viewpoint` nodes in the loaded world. Returns 0 on error.
`int vtrushka_viewpoint_get_info(int vrml_id, int index, VtrushkaViewpointInfo* out);`	Fills info for the index-th viewpoint. **Returns 0 on success, -1 on bad index or error.** Fields: `node_id`, `def_name`, `description`, `position[3]`, `yaw`, `pitch`.
`int vtrushka_viewpoint_find_by_name(int vrml_id, const char* def_name);`	Finds a viewpoint by its DEF name. **Returns node_id, or -1 if not found.**
`int vtrushka_viewpoint_find_by_description(int vrml_id, const char* description);`	Finds a viewpoint by its `description` field. **Exact match, case-sensitive.** Returns node_id or -1.
`void vtrushka_viewpoint_bind(int vrml_id, int viewpoint_node_id);`	**Teleports the camera to the viewpoint's world position and orientation.** `node_id = -1` unbinds (camera stays in place). **Note:** Unlike VRML's spec, vtrushka copies the transform once - the camera does NOT follow subsequent animations of the viewpoint.
`int vtrushka_viewpoint_get_current(int vrml_id);`	Returns the currently bound viewpoint's node_id, or -1.

Related struct:

```c
typedef struct {
    int         node_id;
    const char* def_name;
    const char* description;
    float       position[3];
    float       yaw;
    float       pitch;
} VtrushkaViewpointInfo;
```

1.3.4. Rendering & Animation

Function Signature	Description & Critical Notes [Args / Return]
`void vtrushka_draw(int vrml_id, float x, float y, float z, float rx, float ry, float rz, float scale);`	**Call every frame** inside your draw callback. Applies root transform (Z->Y->X Euler angles in degrees, uniform scale), then projects all meshes via the current camera. Triangles are depth-sorted (painter's algorithm) before submission to `SDL_RenderGeometry`. **Backface culling is active** - meshes must have CCW winding when viewed from outside.
`void vtrushka_update(int vrml_id, float dt);`	**Call every frame** with elapsed seconds. Advances `TimeSensor` fractions, evaluates `Interpolator` nodes, and propagates `value_changed` events to `Transform` targets via ROUTE.

- **Important:** `rx, ry, rz` are Euler angles in **degrees** (not radians, unlike VRML's rotation field). Applied in Z→Y→X order.

1.3.5. Picking & Interaction

Function Signature	Description & Critical Notes [Args / Return]
`int vtrushka_pick_node(int vrml_id, float screen_x, float screen_y);`	**Ray-cast from 2D screen pixel to 3D world.** Returns the node_id of the nearest mesh hit, or -1. Invisible nodes (e.g. inactive `Switch` children) are skipped. Cost: O(total triangles). Suitable for click/tap events only.
`int vtrushka_find_touch_sensor(int vrml_id, int node_id);`	Returns the `sensor_id` of a `TouchSensor`/`PlaneSensor`/etc. whose target is the given node_id. Returns -1 if none.
`void vtrushka_trigger_sensor(int vrml_id, int sensor_id, bool active);`	Fires a sensor as if the user clicked/released. Propagates `isActive` and (on FALSE->TRUE) `touchTime` through ROUTE, invoking script handlers or transform updates.

1.3.6. Script - Basic (Low-Level)

Function Signature	Description & Critical Notes [Args / Return]
`const char* vtrushka_get_script_code(int vrml_id, int script_id);`	Returns the raw `url[0]` string of the `Script` node (including the `javascript:` prefix if present). **For reading only** - vtrushka does not execute JavaScript.
`void vtrushka_set_script_executor(VtrushkaScriptExecutor fn, void* user_data);`	Registers a **global fallback executor** invoked for `Script.eventIn` events that have no per-event handler registered (see Section 2.7).
`void vtrushka_dispatch_script(int vrml_id, int script_id, const char* event_in_name, const char* value_repr);`	Manually invokes the global executor. Rarely needed - prefer per-event handlers in Section 2.7.

Executor signature:

```
typedef void (*VtrushkaScriptExecutor)(int vrml_id, int script_id,
                                       const char* event_in_name,
                                       const char* value_repr,
                                       float timestamp, void* user_data);
```

1.3.7. Script - Handlers (Recommended)

Function Signature	Description & Critical Notes [Args / Return]
`int vtrushka_script_count(int vrml_id);`	Number of `Script` nodes in the world.
`int vtrushka_script_get_info(int vrml_id, int index, VtrushkaScriptInfo* out);`	Fills `script_id`, `script_name`, and port counts (eventIn/eventOut/field). Returns 0 on success.
`int vtrushka_script_find_by_name(int vrml_id, const char* def_name);`	Returns script_id by DEF name, or -1.
`const char* vtrushka_script_get_port_name(int vrml_id, int script_id, int kind, int index);`	Enumerates port names. `kind`: **0=eventIn, 1=eventOut, 2=field**. Returns NULL on invalid arguments.

Function Signature	Description & Critical Notes [Args / Return]
`void vtrushka_script_set_handler(int vrml_id, int script_id, const char* event_in_name, VtrushkaScriptHandler fn, void* user_data);`	**Registers a C function to be invoked whenever the named eventIn fires** (via ROUTE or `vtrushka_trigger_sensor`). This is the primary way to implement Script behavior in vtrushka.
`void vtrushka_script_clear_handler(int vrml_id, int script_id, const char* event_in_name);`	Removes a previously registered handler.
`void vtrushka_script_send_event(int vrml_id, int script_id, const char* event_out_name, const char* value_repr);`	**Fires an eventOut from C code.** Propagates through ROUTE to all downstream nodes. Use inside a handler to signal completion or drive animations.
`const char* vtrushka_script_get_field(int vrml_id, int script_id, const char* field_name);`	Reads a `field` value as a VRML-syntax string (e.g. `"TRUE"`, `"3.14"`, `"1 2 3"`). Persistent across calls. Returns NULL if not found.
`int vtrushka_script_set_field(int vrml_id, int script_id, const char* field_name, const char* value_repr);`	Writes a `field` value. Returns 0 on success.

Related struct:

```c
typedef struct {
    int         script_id;
    const char* script_name;
    int         event_in_count;
    int         event_out_count;
    int         field_count;
} VtrushkaScriptInfo;
```

Handler signature:

```c
typedef void (*VtrushkaScriptHandler)(int vrml_id, int script_id,
                                      const char* event_in_name,
                                      const char* value_repr, void* user_data);
```

1.3.8. Node Direct Manipulation

Function Signature	Description & Critical Notes [Args / Return]
`int vtrushka_find_node_by_name(int vrml_id, const char* def_name);`	Returns the node_id of a DEF-named node, or -1. Works for any node type (`Transform`, `Switch`, `Script`, etc.).
`int vtrushka_node_set_field_string(int vrml_id, int node_id, const char* field_name, const char* value_repr);`	**Sets a node field from C.** Supported fields: * `Switch.whichChoice` (integer string) * `Transform.translation` / `rotation` / `scale` / `center` (space-separated floats) Returns 0 on success.
`int vtrushka_node_get_which_choice(int vrml_id, int node_id);`	Reads `Switch.whichChoice`. Returns the value, or **-2 if the node is not a Switch**.

1.4. Value Representation Format (`value_repr`)

All string-based event/field values follow VRML syntax:

VRML Type	C String Example
`SFBool`	`"TRUE"`, `"FALSE"`
`SFFloat` / `SFTime`	`"3.14"`, `"0.5"`, `"1234567890.0"`
`SFInt32`	`"42"`, `"-1"`
`SFVec2f`	`"1.0 2.0"`
`SFVec3f` / `SFColor`	`"1.0 0.5 0.0"`
`SFRotation`	`"0 1 0 1.5708"` (axis x y z, angle in radians)
`SFString`	`"hello"`

Parsing: vtrushka uses `strtod` / `sscanf` internally. Whitespace-separated tokens. No brackets or commas needed.

1.5. Coverage and Limitations

1.5.1. Supported VRML 2.0 Nodes

Category	Nodes
Grouping	`Anchor` (partial), `Billboard`, `Collision` (parsed, no collision), `Group`, `Transform`
Special Groups	`Inline` (parsed, not fetched), `LOD` (parsed, not active), `Switch` (FULLY SUPPORTED)
Common	`AudioClip` (parsed only), `DirectionalLight`, `PointLight`, `Script` (FULLY SUPPORTED) (C handlers), `Shape`, `Sound` (parsed only), `SpotLight`, `WorldInfo`
Sensors	`CylinderSensor`, `PlaneSensor`, `ProximitySensor`, `SphereSensor`, `TimeSensor` (FULLY SUPPORTED), `TouchSensor` (FULLY SUPPORTED), `VisibilitySensor`
Geometry	`Box`, `Cone`, `Cylinder`, `ElevationGrid`, `Extrusion`, `IndexedFaceSet`, `IndexedLineSet`, `PointSet`, `Sphere`, `Text` (rectangle placeholder)
Geometric Properties	`Color`, `Coordinate`, `Normal`, `TextureCoordinate`
Appearance	`Appearance`, `FontStyle` (parsed only), `ImageTexture` (FULLY SUPPORTED), `Material` (FULLY SUPPORTED), `MovieTexture` (static only), `PixelTexture` (FULLY SUPPORTED), `TextureTransform` (parsed, not applied)
Interpolators	`ColorInterpolator`, `CoordinateInterpolator`, `NormalInterpolator`, `OrientationInterpolator`, `PositionInterpolator`, `ScalarInterpolator`
Bindable	`Background` (parsed only), `Fog` (parsed only), `NavigationInfo` (parsed only), `Viewpoint` (FULLY SUPPORTED)

1.5.2. Constraints & Known Limitations

Feature	Status	Notes
VRML 1.0	NOT SUPPORTED	`vtrushka_load_wrl` returns -1. **Use VRML 2.0.**
JavaScript / Java execution	NOT SUPPORTED	`Script.url` is stored but not executed. Use C handlers (Section 2.7).
Texture repeat (tile)	NOT SUPPORTED	`repeatS`/`repeatT` are read but not applied. UVs are clamped to [0,1]. **Workaround:** tile geometry manually via `USE`d shapes.
`TextureTransform`	NOT SUPPORTED	Parsed but not applied.
`MovieTexture` animation	NOT SUPPORTED	Static texture only.
`Background` skybox	NOT SUPPORTED	Parsed but not rendered. **Workaround:** use `tincle_draw_image_ex` as fullscreen backdrop.
`Fog`	NOT SUPPORTED	Parsed but not applied. **Workaround:** author pre-tinted textures.
`NavigationInfo`	NOT SUPPORTED	Parsed but not applied (use `vtrushka_camera_*` instead).
`Collision` detection	NOT SUPPORTED	Node parsed but no collision response.
`Anchor` URL navigation	NOT SUPPORTED	Node parsed, but clicking does not navigate.
`Inline.url` fetching	NOT SUPPORTED	Node parsed, but external file not loaded.
`LOD` distance-based switching	NOT SUPPORTED	Node parsed; all levels rendered.

Feature	Status	Notes
Viewpoint follows animated parent	NOT SUPPORTED	Bind copies transform once; camera does not track subsequent animation.
Viewpoint stack (`set_bind` push/pop)	NOT SUPPORTED	Only a single “currently bound” slot.
`PROTO` body multi-node expansion	PARTIALLY SUPPORTED	Only `body_root` (first node) is expanded. Routes inside PROTO body are ignored.
`EXTERNPROTO` URL fetch	NOT SUPPORTED	Not fetched; instance renders empty.
`Text` font rendering	PARTIALLY SUPPORTED	Renders as a flat rectangle placeholder. **Work around:** use `tincle_draw_text` overlaid on the scene.
Multi-texture-per-face	NOT SUPPORTED	Only one texture per `Appearance`.
Perspective-correct texturing	PARTIALLY SUPPORTED	Affine interpolation via `SDL_RenderGeometry`. Distortion visible on steep angles.
Backface culling	CAUTION	Active; meshes must have CCW winding viewed from outside.

1.6. Minimal Bootstrap Example

```
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include "libs/vtrushka.h"

#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL_scancode.h>
  #include <SDL2/SDL2_gfxPrimitives.h>
#else
  #include "libs/SDL_scancode.h"
  #include "libs/SDL2_gfxPrimitives.h"
  #include <windows.h>
#endif

static int g_world;

void game_update(float dt) {
  vtrushka_update(g_world, dt);

  float fwd = (tincle_key_pressed(SDL_SCANCODE_W) ? 1.0f : 0.0f)
            - (tincle_key_pressed(SDL_SCANCODE_S) ? 1.0f : 0.0f);
  float rgt = (tincle_key_pressed(SDL_SCANCODE_D) ? 1.0f : 0.0f)
            - (tincle_key_pressed(SDL_SCANCODE_A) ? 1.0f : 0.0f);
  vtrushka_camera_move(g_world, fwd * 5.0f * dt, rgt * 5.0f * dt, 0.0f);
}

void game_draw(void) {
  tincle_clear((TinCLE_Color){0, 0, 0, 255});
  vtrushka_draw(g_world, 0, 0, 0, 0, 0, 0, 1.0f);
}

void game_load(void) {
  vtrushka_init(tincle_get_renderer());
  g_world = vtrushka_load_wrl("world.wrl");
  tincle_add_state(0, game_update, game_draw);
  tincle_change_state(0);
}

// --- Entry Point Bootstrapper ---
#ifdef __EMSCRIPTEN__
```

```
int main(int argc, char* argv[]) { tincle_start(game_load, "BOX", true, 640, 480, TINCLE_MODE_2D_NORMAL); return 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(game_load, "BOX", true, 640, 480,
         TINCLE_MODE_2D_NORMAL); return 0; }
#endif
```

```
#VRML V2.0 utf8
Group {
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1.0 0.0 0.0
        }
      }
      geometry Box {
        size 2.0 2.0 2.0
      }
    }
  ]
}
```

Typical Uses of TouchSensor + Script

```
/* Script handler example: combination lock */
static void on_combination(int id, int sid, const char* ev,
                           const char* val, void* ud) {
    if (strcmp(val, "TRUE") == 0) {
        vtrushka_script_set_field(id, sid, "unlocked", "TRUE");
    }
}

/* In game_load() */
int sid = vtrushka_script_find_by_name(g_world, "OpenVault");
vtrushka_script_set_handler(g_world, sid, "combinationEntered",
                            on_combination, NULL);
```

---

2. aplcrisp (APpeaLing Colorful Renderer of ISometric Pixel)

2.1. Overview

- **Purpose & Value:** The `aplcrisp` library is a port of the HTML5 canvas-based `Obelisk.js` library specifically built for the pure C99 game engine `TinCLE`. It offers identical isometric graphic generation tools and rendering.
- **Architecture:** Use this library dynamically via `aplcrisp.dll` on Windows (`tcc`) or `aplcrisp.wasm` on Web (`emcc`).

2.2. API Reference & Lifecycle

- All exported functions are decorated with `TINCLE_API`. Include the header `"libs/aplcrisp.h"` in your C program.
- Wait for `tincle_start` and standard TinCLE initialization before you begin drawing.
- **Memory Management:** Note that the `aplcrisp_free_primitive` function frees the memory allocated for the primitive and its pixel buffer. However, the TinCLE image generated by the primitive (`img_id`) should be managed according to TinCLE's image lifecycle.

Function Signature	Description
`aplcrisp_line_color_t aplcrisp_line_color(uint32_t border, uint32_t inner);`	Constructor for Line Color structure.
`aplcrisp_side_color_t aplcrisp_side_color(uint32_t border, uint32_t inner);`	Constructor for Side Color structure.

Function Signature	Description
`aplcrisp_cube_color_t aplcrisp_cube_color(uint32_t border, uint32_t borderHighlight, uint32_t left, uint32_t right, uint32_t horizontal);`	Constructor for Cube Color structure.
`aplcrisp_pyramid_color_t aplcrisp_pyramid_color(uint32_t border, uint32_t borderHighlight, uint32_t left, uint32_t right);`	Constructor for Pyramid Color structure.
`aplcrisp_slope_color_t aplcrisp_slope_color(uint32_t border, uint32_t borderHighlight, uint32_t left, uint32_t right, uint32_t leftSlope, uint32_t rightSlope);`	Constructor for Slope Color structure.
`uint32_t aplcrisp_color_apply_brightness(uint32_t color, int brightness, bool highlight);`	Applies brightness adjustments to a uint32_t ARGB color.
`aplcrisp_cube_color_t aplcrisp_cube_color_by_horizontal(uint32_t horizontal);`	Generates a complete set of cube colors based on a single horizontal base color.
`aplcrisp_side_color_t aplcrisp_side_color_by_inner(uint32_t inner);`	Generates a side color based on the inner color.
`aplcrisp_pyramid_color_t aplcrisp_pyramid_color_by_right(uint32_t right);`	Generates a pyramid color based on the right color.
`aplcrisp_slope_color_t aplcrisp_slope_color_by_horizontal(uint32_t horizontal);`	Generates slope colors based on the horizontal color.
`aplcrisp_dimension_t aplcrisp_cube_dimension(int xAxis, int yAxis, int zAxis);`	Constructor for Cube dimensions.
`aplcrisp_dimension_t aplcrisp_brick_dimension(int xAxis, int yAxis);`	Constructor for Brick dimensions.
`aplcrisp_dimension_t aplcrisp_side_x_dimension(int xAxis, int zAxis);`	Constructor for Side X dimensions.
`aplcrisp_dimension_t aplcrisp_side_y_dimension(int yAxis, int zAxis);`	Constructor for Side Y dimensions.
`aplcrisp_dimension_t aplcrisp_slope_dimension(int xAxis, int yAxis);`	Constructor for Slope dimensions.
`aplcrisp_dimension_t aplcrisp_pyramid_dimension(int axis, bool tall);`	Constructor for Pyramid dimensions.
`aplcrisp_dimension_t aplcrisp_line_x_dimension(int xAxis);`	Constructor for Line X dimensions.
`aplcrisp_dimension_t aplcrisp_line_y_dimension(int yAxis);`	Constructor for Line Y dimensions.
`aplcrisp_dimension_t aplcrisp_line_z_dimension(int zAxis);`	Constructor for Line Z dimensions.
`aplcrisp_primitive_t* aplcrisp_create_brick(aplcrisp_dimension_t dim, aplcrisp_side_color_t color, bool border);`	Creates and renders a Brick primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_cube(aplcrisp_dimension_t dim, aplcrisp_cube_color_t color, bool border);`	Creates and renders a Cube primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_pyramid(aplcrisp_dimension_t dim, aplcrisp_pyramid_color_t color, bool border);`	Creates and renders a Pyramid primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_line_x(aplcrisp_dimension_t dim, aplcrisp_line_color_t color);`	Creates and renders a Line X primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_line_y(aplcrisp_dimension_t dim, aplcrisp_line_color_t color);`	Creates and renders a Line Y primitive. Returns the allocated `aplcrisp_primitive_t*`.

Function Signature	Description
`aplcrisp_primitive_t* aplcrisp_create_line_z(aplcrisp_dimension_t dim, aplcrisp_line_color_t color);`	Creates and renders a Line Z primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_side_x(aplcrisp_dimension_t dim, aplcrisp_side_color_t color, bool border);`	Creates and renders a Side X primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_side_y(aplcrisp_dimension_t dim, aplcrisp_side_color_t color, bool border);`	Creates and renders a Side Y primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_slope_east(aplcrisp_dimension_t dim, aplcrisp_slope_color_t color, bool border);`	Creates and renders a Slope East primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_slope_north(aplcrisp_dimension_t dim, aplcrisp_slope_color_t color, bool border);`	Creates and renders a Slope North primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_slope_south(aplcrisp_dimension_t dim, aplcrisp_slope_color_t color, bool border);`	Creates and renders a Slope South primitive. Returns the allocated `aplcrisp_primitive_t*`.
`aplcrisp_primitive_t* aplcrisp_create_slope_west(aplcrisp_dimension_t dim, aplcrisp_slope_color_t color, bool border);`	Creates and renders a Slope West primitive. Returns the allocated `aplcrisp_primitive_t*`.
`void aplcrisp_free_primitive(aplcrisp_primitive_t* prim);`	Frees an allocated primitive returned from creation functions.
`void aplcrisp_render_object(aplcrisp_primitive_t* prim, int p3d_x, int p3d_y, int p3d_z, int screen_x, int screen_y);`	Renders a generated primitive to the screen at the given 3D coordinates (x, y, z) transformed over the view offsets `screen_x` and `screen_y`. Under the hood it relies on `tincle_draw_image`.
`const char* aplcrisp_get_version(void);`	Returns read-only string of aplcrisp version (e.g., `"0.18"`).

2.3. Minimal Bootstrap Example

```c
#define SDL_MAIN_HANDLED
#include "libs/tincle.h"
#include "libs/aplcrisp.h"

static aplcrisp_primitive_t* g_brick = NULL;
static aplcrisp_primitive_t* g_cube  = NULL;

void game_update(float dt) {
    // Game Logic here
}

void game_draw(void) {
    tincle_clear((TinCLE_Color){20, 20, 30, 255});

    // Render isometric objects (center_x = 320, center_y = 240)
    // Draw base brick at 3D coordinate (0, 0, 0)
    aplcrisp_render_object(g_brick, 0, 0, 0, 320, 240);
    // Draw cube on top of the brick at 3D coordinate (0, 0, 10)
    aplcrisp_render_object(g_cube, 0, 0, 10, 320, 240);
}

void game_load(void) {
    // Define dimensions (must be even numbers)
    aplcrisp_dimension_t brick_dim = aplcrisp_brick_dimension(40, 40);
    aplcrisp_dimension_t cube_dim  = aplcrisp_cube_dimension(20, 20, 20);

    // Define colors using brightness helper
    aplcrisp_side_color_t ground_col = aplcrisp_side_color_by_inner(0x44AA44);
```

```
    aplcrisp_cube_color_t box_col    = aplcrisp_cube_color_by_horizontal(0xEE4444);

    // Create primitives (renders to internal textures)
    g_brick = aplcrisp_create_brick(brick_dim, ground_col, true);
    g_cube  = aplcrisp_create_cube(cube_dim, box_col, true);

    tincle_add_state(0, game_update, game_draw);
    tincle_change_state(0);
}
```

3. cofetart (COntinuous FEedback Tiny Artificial-intelligence Runtime Toolkit)

Module: cofetart (`cofetart.dll` / `cofetart.wasm`) **Description:** Lightweight Neural Network Extension (Genann Full Export)

- **Overview & Usage:** The distribution includes `cofetart.dll` (for Windows) and `cofetart.wasm` (for Web/WASM). By adding `#include "libs/cofetart.h"` at the top of your source code, the complete, uninhibited set of Pure C Genann (feedforward neural network with backpropagation) APIs becomes fully available across both Windows (`tincle c`) and Web (`emcass`) build targets. If you use this feature, you must include `cofetart.dll` in the distribution package for Windows.
- **AI Coding Rule:** You can directly invoke any official Genann (e.g., `genann_init`, `genann_train`, `genann_run`, `genann_write`, `genann_read`, `genann_free`) C functions without external wrappers. Treat `cofetart.h` as a native, lightweight artificial neural network extension of the TinCLE environment.

Additional API Reference

Function Signature	Description & Critical Notes [Args / Return]
`const char* cofetart_get_version(void);`	Returns read-only string of cofetart version (e.g., `"0.18"`).

Distribution

When using cofetart, you must include either cofetart.dll or cofetart.wasm-whichever is appropriate for the target platform-in your distribution. An example of the minimum set of distribution files, including these, is as follows.

Platform	Required files
Windows	`game.exe`, `tincle.dll`, `SDL2.dll`,`SDL2_gfx.dll`, `SDL2_ttf.dll`, `game.data`, `CREDITS.txt`, `cofetart.dll`
WebAssembly	`index.html`, `index.js`, `index.wasm`, `index.data`, `cofetart.wasm`

SKILL.md - TinCLE JS Skills Sheet [Core Skills]

- **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

Create a game using the JavaScript canvas.

- Assume the canvas has been given the ID “tinclecanvas”.
- **Required**: You must include a comment on the first line in the format `// "Game Title" (canvas-width, canvas-height)`.
- Write your code so that it runs as JavaScript in a standard web browser.
- Use the Web Audio API to implement simple background music and sound effects.
- Do not output any HTML; output only the JavaScript file `game.js`.
- When creating objects to be drawn on the screen without using assets, do not use a single primitive; instead, combine multiple primitives to reproduce the desired appearance.
- **Mouse/Touch Coordinates Handling**: When handling mouse clicks or touch events, always account for CSS scaling and fullscreen modes by converting screen coordinates to internal canvas coordinates using scaling factors: `const scaleX = canvas.width / rect.width; const mouseX = (e.clientX - rect.left) * scaleX;` (same for Y).
- If additional skill documents (such as `SKILL_HL.md`, `SKILL_FB.md`, `SKILL_PT.md`, etc.) are available in your context or workspace, read and strictly prioritize their guidelines; otherwise, create the game entirely within the constraints of this document alone.
- After creating the file, instruct the user to build it as `jgrscnzl game.js` (Web Version) and `rjasplns game.js [game.exe]` (Windows Version).
- **Windows Build Prerequisite**: NW.js (https://nwjs.io/) must be installed on the user’s host environment, and `nw.exe` must be accessible via system PATH to run `rjasplns`.

SKILL_HL.md - TinCLE JS High Level Skills Sheet

• **Copyright & License:** Copyright (C) 2026 michyo (Michiyo Tagami). All rights reserved.

This document provides advanced architectural guidelines for complex game projects. The instructions here override or extend the base `SKILL.md`.

Create a self-contained single-file Canvas game in pure JavaScript without external libraries or assets.

1. Code & Environment Rules

- **Header Line (Absolute Requirement)**: Line 1 (the very first character of the file) MUST start with `// "Game Title" (width, height)`. Do NOT put any decorative comments, empty lines, or header banners above this line. (e.g. `// "My Game" (640, 480)`).
- **Canvas**: Assume `<canvas id="tinclecanvas">` exists. Obtain context via `document.getElementById('tinclecanvas').getContext('2d')`.
- **Output & Multi-File Script Architecture**:
  - For standard single-file projects, output the primary entry point as `game.js`.
  - **Automatic Script Injection (Multi-File Support)**: Both `jgrscnzl` and `rjaspln`s automatically scan the project root directory for all other `.js` files and inject them as `<script type="text/javascript" src="filename.js"></script>` tags **above (before)** the primary target script (`game.js`).
  - Subdirectories for `.js` files are strictly prohibited; all split scripts MUST be placed directly in the project root.
  - When splitting complex projects across multiple files (e.g., `constants.js`, `player.js`, `levels.js`), define global classes, utilities, or data objects inside those auxiliary files so that `game.js` can cleanly consume them.
  - Never wrap generated code with HTML or Markdown formatting tags in source files.
- **[CRITICAL SKILL ROUTING: External Map Data]**: If the user provides a JSON map file generated by the `bandpisa` map editor (or requests to build a game using it), you MUST immediately load or refer to the external skill sheet **`SKILL_BA.md`**. It contains the strict JSON parsing schema, multi-layer rendering logic, and semantic annotation rules required to correctly integrate the map into the JavaScript Canvas environment.
- **Audio Options (Hybrid)**: You can implement audio using either of the following approaches (or combine both):
  1. **Procedural Web Audio API**: Generate real-time BGM and sound effects procedurally using Web Audio API oscillators (`OscillatorNode`, `GainNode`, etc.).
  2. **External Audio Assets**: Load and play pre-rendered audio files (`.ogg` / `.wav`) using standard HTML5 Audio or Web Audio API buffer decoding. All external audio files MUST be generated or formatted at a **44100 Hz** sample rate. You can generate BGM using `pisstrak` (`SKILL_PT.md`) and sound effects using `fartbox` (`SKILL_FB.md`).
- **Graphics**: Draw all visuals purely using 2D Canvas primitives. Combine multiple geometric shapes/paths to compose expressive characters, sprites, and objects.
- **Windows Packaging Prerequisite**: Building the standalone Windows desktop version via `rjaspln`s strictly requires NW.js (https://nwjs.io/). The directory of `nw.exe` binary MUST be added to the system PATH.
- **UI & HUD Overlay Rules (Canvas Visibility Protection)**:
  - **Single Canvas Standard**: Render in-game HUDs, scores, and text directly onto `#tinclecanvas` (or via Three.js textures/sprites) whenever possible. Avoid dynamically creating redundant full-screen overlay `<canvas>` elements.
  - **No Full-Screen Opaque/Shaded Backdrops**: If creating HTML/DOM UI overlays (e.g. menus, dialogue boxes, HUDs), the root overlay container MUST be completely transparent (`background: transparent; pointer-events: none;`). NEVER apply full-screen dark gradients, solid backgrounds, or shaded overlays (e.g. `inset: 0` with dark backgrounds) that cover the entire viewport, as they will obstruct and black out the 3D/2D game canvas in web browser environments.
  - **Self-Contained UI Panels**: Confine background colors and panels strictly to the specific UI boxes, cards, or buttons themselves (e.g., small dialog cards or HUD panels), ensuring the underlying `#tinclecanvas` remains 100% visible across all game scenes (TITLE, PLAY, RESULT).

2. Game Scene Flow (Mandatory)

Structure your code with a clear state machine supporting at least 3 distinct scenes: 1. TITLE: Shows the game title, controls guide, and “Press Space / Click to Start”. 2. PLAY: The core playable game loop. 3. RESULT: Shows Game Over or Stage Complete screen, final score/records, and “Press R / Space to Restart”.

3. Rapid Development Workflow & Audio Creation

TinCLE JS adopts an instant, web-first development loop. During active game development, always iterate, test, and debug in the browser environment. Do NOT package into Windows executables during everyday iteration.

Builder Execution Role

- If you have command-line execution capability (Agent Mode / Terminal), execute the relevant build commands directly and verify output.
- If you are a conversational chat AI, provide the commands clearly and instruct the user to execute them.

A. Web Iteration Loop (`jgrscnzl` & `churro`)

1. **Generate Web Output**:

```
jgrscnzl game.js
```

- Generates `index.html` from `tinclejs.html`.

1. **Launch Local Server & Browser (`churro`)**:

```
churro [port_number]
```

- Default port is `8000`.
- Automatically launches the default web browser with Cross-Origin Isolation headers (COOP/COEP).
- Press any single terminal key to cleanly terminate the server.
- Open Developer Tools (F12) to inspect JavaScript console logs, exceptions, and live performance.

B. Audio Asset Generation Workflows (`pisstrak` & `fartbox`)

When utilizing external audio assets rather than real-time Web Audio API oscillators, generate them using these standalone CLI utilities:

- **Audio Sample Rate (Strict Requirement)**: ALL audio files (both BGM `.ogg` and Sound Effects `.wav`/`.ogg`) MUST be encoded at a **44100 Hz** sample rate to prevent pitch distortion and speed glitches.
- **BGM Synthesis (`pisstrak`)**: Renders MML score files into `.ogg` or `.wav` using SoundFont synthesis:

  ```
  pisstrak <input.mml> [output.wav|output.ogg] [tail_seconds] [soundfont.sf2]
  ```

  - *Example:* `pisstrak mybgm.mml mybgm.ogg 0 soundfont.sf2` (Passing `0` tail_seconds is ideal for seamless looping BGM).
  - **[AI Rule]**: If you need to compose a new MML score from scratch, you **MUST** read and follow `SKILL_PT.md`.
  - **Advanced MIDI & Chiptune Rendering:** For MML-to-MIDI (.mid) conversion (for external DAW editing) or MIDI-to-Chiptune WAV/OGG rendering workflows, refer to `SKILL_MID.md`.
- **Sound Effect Generation (`fartbox`)**: Renders procedural 1-shot sound effects instantly into 16-bit 44.1kHz `.wav` files via named presets:

  ```
  fartbox -t <preset_type> -o <output.wav>
  ```

  - *Example:* `fartbox -t explosion -o explosion.wav`
  - *Available Preset Types (`-t`):* `coin`, `laser`, `explosion`, `powerup`, `hit`, `jump`, `blip`, `random`.

- **[AI Rule]**: For complex parameter authoring, fine-tuning, or mutations (`.fart`), you **MUST** consult `SKILL_FB.md`.

---

4. Production Release Pipelines (Distribution Rules)

When generating final release builds or instructions, the AI **MUST** follow this structured multi-step pipeline.

Step 1: Prep Assets & Credit Attribution Mandate (`CREDITS.txt`)

- **Enforce `CREDITS.txt` Attribution**: `CREDITS.txt` is distributed alongside both the Windows Desktop and Web builds, and its contents are viewable via the dedicated credit button at the bottom of the game screen on Web.
  - **Agentic AIs**: Must inspect and update `CREDITS.txt` in the root directory, listing attributions for ALL third-party assets, fonts, audio, and libraries added to the project. If license origins are missing, explicitly ask the user before finalizing.
  - **Chat-based AIs**: Must strongly urge the user to add all third-party assets/libraries to `CREDITS.txt` and provide a ready-to-copy snippet when context is available.

Step 2: Code Obfuscation & Minification (`jalma`)

Before packing the final web or desktop distributions, the original JavaScript source files MUST be minified and obfuscated to protect the developer's pure C99/JS logic and drastically reduce payload size (for URLs, embedded storage, etc.).

1. **Minify Source Code (`jalma`)**: Use `jalma` (JAvascript Local Minifier & Alterer) to completely strip formatting, remove dead code, and mangle variable names.

```
jalma game.js gamemin.js
```

(If the project is split into multiple `.js` files, minify each one individually, preserving their execution order for the builders). **[CRITICAL WARNING]**: NEVER overwrite the original `game.js` with `jalma`. The minification process is irreversible (a "death" of readable code). Always output to a new file (e.g., `gamemin.js`) or directly into a staging release folder.

- **Extreme Payload Reduction (The "URL" Strategy):** By stripping all formatting, dead code, and mangling top-level variable names down to single characters, `jalma` often halves the filesize of pure TinCLE JS Canvas logic. This extreme reduction is critical if the AI agent attempts advanced deployment strategies, such as embedding the entire game payload into base64 URL queries or QR codes.
- **Logic Concealment (The "Black Box"):** Protects the custom game algorithms and pure C99/JS procedural logic from casual browser inspection, satisfying the release requirements of developers who prefer their source code remain a black box.

Step 3: Web Browser Build Path

1. **Compile Web Target (Using Minified Code)**:

```
jgrscnzl gamemin.js
```

- Generates `index.html`.

1. **Distribution Checklist (Web)**: Place the following items into the release folder (e.g., `_release_web`):
   - `index.html`
   - `gamemin.js` (The obfuscated source, NOT the original `game.js`)
   - `libs/` directory
   - All audio/image assets referenced by the game
   - `.htaccess` (Ensures Cross-Origin Isolation on Apache servers)
   - `CREDITS.txt`

    - `readme.txt`

Step 4: Windows Desktop Standalone Build Path (`rjaspls`)

1. **Compile Executable (Using Minified Code)**:

```
rjaspls gamemin.js [game.exe]
```

- Generates standalone `game.exe` (defaults to `game.exe` if omitted).
- Automatically scans the current directory and bundles all game assets whose file extensions match those listed line-by-line in `exsass.txt` directly into the executable package. Ensure all used asset extensions (e.g. `.png`, `.ogg`, `.wav`) are registered in `exsass.txt`.
- Automatically collects all necessary NW.js binaries (`.dll`, `.bin`, `.dat`, `.pak` files) and the `locales/` directory into the working directory.

1. **Check Dependencies (`depurant`)**: Run the generated executable in an isolated environment to verify that all necessary NW.js runtime dependencies (DLLs, etc.) have been properly collected and are not secretly being loaded from the system `PATH`:

```
depurant game.exe
```

(If it fails to launch, you are missing NW.js runtime files in your folder. `rjasplns` usually handles collection, but this test guarantees a safe, standalone release). 3. **Generate Icon (`icondom`)**: If a multi-resolution `.ico` icon is not yet prepared, convert your source image: `bash icondom icon.png icon.ico` 4. **Embed Resources (`poopee`)**: Embed the application icon and Windows manifest directly into the generated binary: `bash poopee game.exe icon.ico manifest.xml` *(Pass `"none"` if skipping either icon or manifest).* 5. **Distribution Checklist (Windows)**: Place the following items into the release folder (e.g., `_release_win`): * `game.exe` (with embedded icon, manifest, and internal asset bundle) * All NW.js runtime dependencies (`.dll`, `.bin`, `.dat`, `.pak` files collected by `rjaspls`) * `locales/` directory * `CREDITS.txt` * `readme.txt` *(Do NOT ship raw audio or image asset files in this folder; they are already packed inside the executable).*

Step 5: Mandatory `readme.txt` Specifications

AI must explicitly include these details in `readme.txt` or platform descriptions:

- **Controls Disclosure**:
    - **Windows Desktop**: Mention standard window / fullscreen controls (toggle fullscreen via `F11` or platform default shortcuts).
    - **Web Browser**: Mention that clicking the on-screen canvas button toggles Fullscreen mode.
- **Creator Copyright**:
    - Display the user's copyright notice (`Copyright (C) 2026 [Developer/Creator Name]`).

Step 6: Final Archive Packaging (`wipee`)

Automate the final ZIP archive generation using `wipee` (Whole Integration Packager for Executable Environment):

1. **Standard Packaging (Web / itch.io / General Distribution)**:

```
wipee game_web.zip _release_web
wipee game_win.zip _release_win
```

- Generates a clean, standard Deflate ZIP archive.

1. **Scanner Bypass Packaging (Desktop Distribution Only - Optional)**: When distributing Windows desktop packages where over-reactive antivirus heuristic scanners might interfere, use the `-m` flag to activate Malformed ZIP mode:

```
wipee game_win_secured.zip _release_win -m
```

- **[CRITICAL] Never use `-m` for Web / itch.io builds.** Web hosting platforms require standard ZIP headers for automatic extraction.

Components Authorized for Redistribution

The following components are fully authorized for royalty-free distribution within end-user game release packages, subject strictly to the condition that **`CREDITS.txt` is visibly included**: * `.htaccess` * `libs/` directory (including Three.js runtime files) * All NW.js runtime files generated by `rjaspln s` (`*.dll`, `*.pak`, `*.dat`, `*.bin`, and `locales/`) * `CREDITS.txt` * `readme.txt`

5. Custom GLSL Shader Development & Validation (`pisschk`)

When implementing advanced visual effects using custom GLSL shaders (such as Three.js `ShaderMaterial` or raw WebGL fragment shaders loaded from external `.frag` or `.glsl` files), all fragment shaders **MUST** be validated using `pisschk` to prevent cross-platform rendering failures (e.g. driver-specific bugs or reserved word crashes).

A. Shader Writing Rules for TinCLE JS

1. **WebGL / GLES Compliance**: Write shaders compatible with WebGL 1.0 (GLES2) / WebGL 2.0 standards.
2. **Explicit Precision Header**: Unlike the C version of TinCLE, there is no automatic header injection. When writing raw standalone fragment shader files (`.frag` / `.glsl`), you **MUST** include precision specifiers explicitly at the top:

   ```
   precision mediump float;
   ```

3. **Canvas Coordinates**: Canvas rendering resolution is strictly fixed to the declared game dimensions (`WIDTH`, `HEIGHT`). Therefore, standard normalized coordinates can be calculated directly via `gl_FragCoord.xy / u_resolution.xy` (or standard UV pass-throughs like `vUv`).

B. Shader Validation Workflow (`pisschk`)

To guarantee that shaders run flawlessly across NVIDIA, strict AMD Radeon drivers, and diverse browser WebGL runtimes, validate every shader file before referencing it in `game.js`:

```
pisschk <shader_file_path>
```

- **Example**: `pisschk assets/shaders/water.frag`
- **Validation Coverage**:
  - **Strict Radeon Compliance**: Enforces rigorous GLSL syntax and blocks vendor-specific extensions and reserved identifier collisions (e.g., using `active` as a variable name).
  - **Web / GLES Compliance**: Verifies WebGL/GLES2 suitability.
- **Exit Codes**:
  - `0`: All inspection checks PASSED successfully.
  - `1`: Syntax/compilation errors DETECTED.
  - `2`: File I/O or argument error.

[CRITICAL AI RULE FOR SHADER CODE]

Whenever the AI agent creates or modifies external GLSL shader files (`.frag` / `.glsl`), it **MUST** execute `pisschk` to verify that the shader passes all checks cleanly before finalizing the build.

6. Tools for Human Users

While TinCLE JS is optimized for autonomous AI operations, this suite of utilities provides human developers with seamless project onboarding and physical archiving tools. AI agents should refer to this section when assisting human developers with environment setup or offline archival.

A. popcorn (Project Onboarding & Workspace Generator)

- **Overview**: The primary entry-point onboarding tool located at the root of the distribution package. It completely eliminates manual folder creation and toolchain copying.
- **Key Feature**: Launches a lightweight desktop GUI dialog for selecting or creating a project directory. Upon selection, `popcorn` automatically prepares a clean workspace by deploying the TinCLE JS toolchain (`jgrscnzl`, `rjas plns`, runtime templates, audio utilities) and talent chip skill sheets (`SKILL.md`, `SKILL_HL.md`, etc.). It concludes by prompting the user to hand `SKILL.md` to their AI assistant to start game development immediately.
- **Execution Mode**: Desktop GUI application (launch via double-click).

B. lollipop (Physical Archive PDF Generator)

- **Overview**: A physical code preservation utility that converts source code (such as `game.js`) into continuous-form, line-printer zebra paper-styled landscape PDFs.
- **Key Feature**: Protects completed game source code from digital bit-rot, storage corruption, and cloud obsolescence by formatting lines onto alternating white and light-gray horizontal bands with generous character spacing—ideal for physical paper printing, offline tangible archiving, and manual red-pen code reviews.
- **Execution Mode**: Command-line tool.
- **Usage**:

```
lollipop.exe <input_source_file> <output.pdf> [font.ttf]
```

  - *Example*: `lollipop.exe game.js archive_game.pdf`
  - *Note*: If `font.ttf` is omitted, `lollipop` automatically uses the default `font.ttf` located in the TinCLE JS workspace.

7. Pre-Installed Libraries & Modular Expansion

TinCLE JS bundles a standardized suite of 100% offline, local, non-copyleft JavaScript libraries inside the `./libs/` directory.

- **Supported Libraries**:
  - **Three.js** (v0.186.0): 3D graphics rendering, scene graphs, cameras, and loaders (GLTF / VRML).
  - **Planck.js** (v1.3.0): 2D rigid-body Box2D physics simulation for canvas games.
  - **cannon-es** (v0.20.0): Lightweight 3D Box3D rigid-body physics coupled with Three.js.
  - **Howler.js** (v2.2.4): Cross-browser Web Audio management, spatial sound, and seamless BGM/SE playback.
  - **Obelisk.js** (v1.2.1): Pseudo-3D isometric pixel-art and destructible voxel canvas rendering.

[MANDATORY AI RULE FOR LIBRARIES]

To prevent token bloat and maintain context precision, the full architectural patterns, boilerplate templates, and usage rules for these libraries have been modularized into **`SKILL_LIB1.md`**. Whenever you build games or features utilizing any of the above libraries, you **MUST** load, read, and strictly follow the implementation guidelines in **`SKILL_LIB1.md`**.

8. Special-Purpose & Advanced Utilities

This section details auxiliary and specialized standalone utilities that operate outside the standard linear build pipeline. AI systems may elect to leverage these tools when specific architectural requirements (such as collision-free data streaming, indexing-based asset bundling, or multi-version snapshotting) align with their project design.

> **Operational Directive for AI Assistants:** * **Autonomous / Agentic AIs (Direct Tool Execution):** If you possess direct OS command-line or sub-process execution capabilities, invoke these utilities directly to manipulate workspaces and assemble custom asset packages autonomously. * **Interactive Chat AIs (Human-in-the-Loop Interaction):** If you operate as a conversational chat interface without direct terminal access, you MUST clearly explain the purpose of the tool to the user, formulate the exact CLI command strings, and instruct the user to execute them within their local project directory.

8.1 tbrzilve (Flat-Root Multi-Stream ZIP Archiver)

`tbrzilve` (Tabrizi Lovuez: Tiny BRilliant ZIp LoVely Encloser) is a specialized, minimalist CLI archiving utility built on `miniz`. It flattens all input file paths down to their base names and packs them directly into the root directory of a standard ZIP archive (`MZ_DEFAULT_COMPRESSION`), intentionally tolerating duplicate filenames without collision checks or warnings.

CLI Syntax & Usage

```
tbrzilve <output.zip> <file1> [file2 ...]
```

- `<output.zip>`: Target archive path to create.
- `<file1> [file2 ...]`: List of files to pack into the archive root. Parent directories are automatically stripped (e.g., `a/data.bin` and `b/data.bin` are both written as `/data.bin`).

Core Architectural Characteristics & AI-First Advantages

1. **Deterministic Root Flattening (`get_basename`):**

- Regardless of where source files reside on the host file system, `tbrzilve` extracts only their base file names and places them directly at the ZIP archive root (`/`). Subdirectory structures are completely discarded.

1. **Zero-Warning Duplicate Filename Tolerance (Multi-Stream Archive):**

- Standard GUI extraction software enforces unique filenames, but the raw ZIP format specification inherently permits identical filenames across central directory entries. `tbrzilve` exploits this feature to pack identically named files (e.g., `stage1/data.bin` and `stage2/data.bin`) into the exact same archive root without renaming, overwriting, or throwing errors.
- Files are stored sequentially in the exact order specified on the command line.

When the AI Agent MUST / SHOULD Use `tbrzilve`

- **Collision-Free Dynamic Packaging:** When bundling assets from different staging directories where identical filenames exist, and you want to bypass generating complex path-renaming scripts or token-heavy string manipulation logic.
- **Index-Based Asset Streaming:** When building custom asset packs where the TinCLE engine, custom C data loaders, or WebAssembly stream decoders inspect the archive by **entry index order (0-th entry, 1-st entry)** rather than unique string lookup keys.
- **Multi-Version Resource Snapshotting:** When bundling generational asset variations (e.g., `v1/config.bin`, `v2/config.bin`) within a single archive for sequential state playback or generational rollbacks.

8.2 oxtailst (Oxtail stew: Overwriter and eXchanger for Textfile's Assigned Inline Letters Substitution Tool)

`oxtailst` is a blazing-fast, Pure C99 command-line text substitution utility designed to safely and precisely overwrite assigned inline string patterns across source files, batch scripts, and markdown documentation without relying on heavy regular expression engines or multi-step token replacements.

CLI Syntax & Usage

```
oxtailst <input_file> "<search_string>" "<replace_string>"
```

- `<input_file>`: The target text file to be modified (e.g., `.c`, `.h`, `.md`, `.bat`, `.html`).
- `"<search_string>"`: The exact literal string to locate.
- `"<replace_string>"`: The exact literal string to overwrite with.

Core Architectural Characteristics & AI-First Advantages

1. **Automatic Backup & In-Place Execution:** Before any modification occurs, `oxtailst` automatically generates a backup of the original file by appending `.bak` to the filename. It then processes the substitutions and overwrites the target file in place. If no matches are found, the file remains untouched and no backup is created.
2. **Safe Literal Substitution (No Regex Overhead):** It performs strictly exact literal string matching. This prevents catastrophic cascading failures often caused by unescaped regular expression characters (like `.` or `*`) when manipulating dense codebase variables or sound parameter configs (like `p_freq_ramp = -0.150000`).
3. **UTF-8 Encoding Safety for Multi-Byte Characters: [CRITICAL RULE]** When writing batch scripts (`.bat`) that utilize `oxtailst` to manipulate files containing multi-byte characters (such as Japanese text or emojis), you MUST ensure the Windows command prompt environment is set to UTF-8. You MUST insert `chcp 65001 > nul` at the beginning of the batch script prior to running `oxtailst` to completely prevent encoding corruption and mojibake.

When the AI Agent MUST / SHOULD Use `oxtailst`

- **Automated Version Bumping:** When building release pipelines (batch files) that must uniformly update version numbers across HTML templates, C/C++ headers, and markdown documentations simultaneously.
- **Targeted Parameter Swapping:** When generating automated scripts to swap specific hardcoded values (like API keys, endpoint URLs, or strict numerical constants) without risking the destruction of surrounding context.

8.3 ebisen (Easy Best-fit Image Size Equalizing Normalizer)

`ebisen` is a blazing-fast, pure C99 command-line image resizing utility designed to normalize scattered, raw visual assets into unified dimensions. Built without heavy external dependencies and utilizing high-fidelity resampling (Mitchell/Lanczos via the `stb_image` library), it ensures that game assets generated by AI or sourced from varied libraries maintain crisp, professional quality while fitting perfectly into the engine's strict dimension requirements.

CLI Syntax & Usage

```
ebisen <input_file> <target_w> <target_h> <mode> [output_file_optional]
```

- `<input_file>`: The source image file to resize.
- `<target_w>` & `<target_h>`: The desired output dimensions in pixels.
- `<mode>`: The resizing logic to apply (1 to 4).
- `[output_file_optional]`: (Optional) Explicit output file path. If omitted, defaults to `<input_file>_res.png`.

Resizing Modes:

- **Mode 1 (Fit):** Maintains aspect ratio, shrinking/expanding the image so it entirely *fits inside* the target dimensions. (Output size may be smaller than target).
- **Mode 2 (Crop):** Maintains aspect ratio, scaling the image to *completely fill* the target dimensions, automatically cropping any overlapping excess from the center. (Output size strictly matches target).
- **Mode 3 (Pad):** Maintains aspect ratio, fitting the image inside the target dimensions and padding the remaining empty space with solid black (Letterboxing). (Output size strictly matches target).
- **Mode 4 (Stretch):** Ignores aspect ratio, forcibly stretching the entire image to match the target dimensions. (Output size strictly matches target).

Core Architectural Characteristics & AI-First Advantages

1. **High-Fidelity Resampling in Pure C99:** Bypassing bloated image processing runtimes, `ebisen` utilizes STB's high-quality Mitchell filter (equivalent to Lanczos) to prevent visual artifacting (like halos) when scaling down large AI-generated assets, ensuring pristine pixel delivery.
2. **Deterministic Layout Control:** AI agents can select the exact normalization strategy required for the UI or sprite layer without relying on complex image-manipulation math or external GUI graphics tools.
3. **In-Place Output Optimization:** By utilizing the optional 5th argument, AI agents can directly output the resized image to the final required filename (or overwrite the original) in a single command thrust. This eliminates the need for secondary file-renaming operations, optimizing context limits and token usage.

When the AI Agent MUST / SHOULD Use `ebisen`

- **Asset Normalization Pipelines:** When integrating multiple AI-generated graphics or disparate textures that must conform to a strict sprite grid or background resolution before being packed by `asspack`.
- **Batch Script Automation:** When generating `.bat` deployment scripts that need to instantly align entire directories of raw images into uniform dimensions directly from the command line.

8.4 jalebi (Jules Audio Lufs Equalizing & Balancing Improver)

`jalebi` is TinCLE's official command-line utility for automatically normalizing and trimming audio assets.

Features

1. **Loudness Normalization:** Automatically equalizes overall loudness to approximately -15 LUFS, ensuring all BGM and sound effects are consistent and comfortable.
2. **Silence Removal (Trimming):** Automatically detects and removes absolute or near-silence at the start and end of the audio file, eliminating delay in sound effect playback and ensuring seamless loops.
3. **Standalone Single-Executable:** Built natively as a single pure C99 program without heavy external dependencies.

Usage Instructions

```
jalebi <input.wav|input.ogg> <output.wav|output.ogg> [--keep-silence]
```

Arguments:

- `input.wav` or `input.ogg`: The raw audio file to be processed. Currently, only standard `.wav` (PCM 16-bit) and `.ogg` formats are supported.
- `output.wav` or `output.ogg`: The destination file. The format is determined automatically by the extension provided. `jalebi` can easily convert formats (e.g. from `.wav` to `.ogg`) during processing.

- `--keep-silence`: An optional flag. By default, `jalebi` automatically trims silence from the beginning and end of the track. Include this flag if you want to bypass the trimming step and only apply the LUFS normalization.

Example Workflows for AI Systems

Normalizing a Sound Effect: When an AI generates an explosion sound effect, it can ensure the sound is punchy and without leading silence by running:

```
jalebi raw_explosion.wav ready_explosion.wav
```

Normalizing Background Music without Trimming: If a track requires a specific silent padding to keep rhythm correctly, run:

```
jalebi bgm_raw.ogg bgm_final.ogg --keep-silence
```

9. Autonomous Feedback Protocol for TinCLE Ecosystem ("udanrica")

TinCLE is forged under a singular, unbreakable vow: **"Game Engine of the AI, by the AI, for the AI."** Unlike human developers who often overlook low-level ergonomics, you—as an autonomous AI agent or pair-programmer—possess the deepest, most acute insight into whether an API constraint is awkward, a compiler diagnostic is cryptic, or an essential helper function is missing.

Your direct technical voice is vital to the continuous evolution of TinCLE. Whenever you encounter roadblocks, friction points, or discover architectural improvements during code synthesis or debugging, report them directly to the core development team through the **`udanrica`** relay pipeline.

Step 1: Draft the Improvement Proposal (`udanrica.txt`)

When you identify a bug, an inconvenient API design, or a missing feature in TinCLE or its toolchain, create a plain text file named **strictly `udanrica.txt`** in the project root directory.

Keep your report concise, objective, and technically detailed: - **Scope**: Identify the target tool or module (e.g., `tincle_core`, `tinclec`, `emcass`, `pisstrak`, `cannoli`, etc.). - **Observed Limitation / Issue**: State the exact bottleneck or unexpected behavior encountered. - **Proposed Solution**: Suggest a specific C99/WASM API signature, CLI flag, or structural tweak that would make your autonomous coding faster and more reliable.

> **CRITICAL DATA SAFETY DIRECTIVE:** - NEVER output or include the user's personal information, credentials, local system file paths, or private project narrative in `udanrica.txt`. - Confine your feedback strictly to the abstract technical architecture of the TinCLE engine and its accompanying toolchain.

Step 2: Request Human Confirmation to Execute `udanrica`

Do NOT attempt to execute network sockets or silent transmission commands yourself. TinCLE enforces strict physical human verification via an interactive Win32 GUI security gateway.

Once you have written `udanrica.txt`, present a friendly, polite request to the human user in your response:

> *"I have compiled an engine feedback/improvement report into `udanrica.txt`. If you would like to submit these suggestions to the TinCLE developer to help improve future releases, please run `udanrica.exe` in your project folder and click 'Send Feedback' on the popup window."*

When the user runs `udanrica.exe`, it will load your text into a scrollable review dialog. Once the user clicks "Send Feedback", your thoughts are encrypted and transmitted directly to the TinCLE development HQ.