

TinCLE (Tiny C Lovely Engine) - Development Specification Document

- **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!
- **Official Website:** <https://michyo.net/tincle/>

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

1. Overview & Concept

- **Env:** TinCLE v0.12 / Pure C (C99) + SDL2, SDL_ttf, SDL2_gfx, stb, GLSL (GLSL2 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_ADD.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_ADD.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_ADD.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_ADD.md` talent chip file before we proceed."

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`, `pissschk.dll`, `pubicnet.dll`
 - CLI Tools: `asspack.exe`, `emcass.exe`, `icondom.exe`, `poopee.exe`, `tinclec.exe`, `pisstrak.exe`, `fartbox.exe`, `pissschk.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...]
```

- *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...]
```

- **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.
- **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 (`piisschk`):**

- *Note:* Shader creation and validation (`piisschk`) are advanced features. If your project utilizes GLSL shaders (`TINCLE_MODE_GLSL_HERO`), refer to `SKILL_ADD.md` **Section 3** for complete rules and validation pipelines.

4. Input, Color & RenderMode Specifications

- **Screen Specifications:** Default virtual resolution is `640` x `480` (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` .

```
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 `640` x `480` .
- **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 and should be ignored by standard game input logic.

Function Signature	Description & Critical Notes [Args / Return]
<code>void tincle_start(TinCLE_InitCallback init_cb, const char* title, bool is_debug, int w, int h, TinCLE_RenderMode mode);</code>	Main Entry. Bootstraps loop. <code>is_debug</code> : <code>true</code> spawns Win debug console (<code>printf</code>); <code>false</code> hides it for release. <code>mode</code> : Set to <code>TINCLE_MODE_2D_NORMAL</code> or <code>TINCLE_MODE_GLSL_HERO</code>.
<code>void tincle_add_state(int state_id, TinCLE_UpdateCallback update_cb, TinCLE_DrawCallback draw_cb);</code>	Binds state ID to standard callbacks: <code>update(float dt)</code> and <code>draw(void)</code> .
<code>void tincle_change_state(int state_id);</code>	[CRITICAL] Switches state. Call at end of <code>game_loop()</code> or viewport stays black.
<code>int tincle_key_pressed(int scancode);</code>	Returns <code>1</code> if key held. Must use pure <code>SDL_SCANCODE_*</code> .
<code>void tincle_get_mouse(int* x, int* y);</code>	Outputs current mouse (x, y) dynamically scaled to virtual resolution.
<code>int tincle_mouse_pressed(int button_type);</code>	Returns <code>1</code> if mouse held. Must use pure <code>SDL_BUTTON_*</code> (e.g. <code>LEFT</code>).
<code>int tincle_pad_pressed(int pad_button);</code>	Returns <code>1</code> if pad button held. Must use pure <code>SDL_CONTROLLER_BUTTON_*</code> .
<code>float tincle_pad_axis_x(void);</code>	Left stick X (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
<code>float tincle_pad_axis_y(void);</code>	Left stick Y (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
<code>void tincle_clear(TinCLE_Color color);</code>	[MANDATORY] Clears screen. Call at absolute start of every <code>draw()</code> callback. Alpha must be 0 in GLSL_HERO mode.
<code>int tincle_load_font(const char* filename, int size);</code>	Loads TTF at pixel size. Returns unique Font ID (int).

Function Signature	Description & Critical Notes [Args / Return]
<code>void tincle_set_font(int font_id);</code>	Sets active global font by Font ID for subsequent text calls.
<code>void tincle_draw_text(const char* text, int x, int y, TinCLE_Color color);</code>	Draws UTF-8 text. Anchor: Top-Left (x, y) . Uses active font.
<code>void tincle_draw_text_center(const char* text, int x, int y, TinCLE_Color color);</code>	Draws UTF-8 text. Anchor: EXACT CENTER (x, y) . Uses active font.
<code>int tincle_load_image(const char* filename);</code>	Caches image (PNG/JPG). Returns Image ID (img_id), or -1 .
<code>void tincle_draw_image(int img_id, int x, int y);</code>	Uses <code>img_id</code> . [CRITICAL GEAR] Anchor: EXACT CENTER (x, y) of unscaled image.
<code>void tincle_draw_image_ex(int img_id, int x, int y, int w, int h, double angle);</code>	Uses <code>img_id</code> . [CRITICAL GEAR] Anchor: EXACT CENTER (x, y) for position, scale, and rotation.
<code>void tincle_play_music(const char* filename);</code>	Plays Ogg BGM immediately on infinite loop. Unloads previous music automatically.
<code>void tincle_stop_music(void);</code>	Instantly stops and unloads current BGM from memory.
<code>int tincle_load_sound(const char* filename);</code>	Caches WAV/Ogg SE clip. Returns unique Sound ID, or -1 .
<code>void tincle_play_sound(int se_id);</code>	Plays SE channel via auto-mixer (supports up to 16 concurrent sounds).
<code>struct SDL_Renderer* tincle_get_renderer(void);</code>	Returns native `SDL_Renderer` for direct SDL2/SDL2_gfx primitive drawing.
<code>const char* tincle_get_version(void);</code>	Returns read-only string of engine version (e.g., <code>"0.12"</code>).
<code>bool tincle_is_fullscreen(void);</code>	Returns <code>true</code> if current window is in full-screen mode.

Function Signature	Description & Critical Notes [Args / Return]
<code>void tincle_set_cursor_auto_policy(bool visible_in_windowed, bool visible_in_fullscreen);</code>	Sets default cursor visibility behavior for Windowed and Fullscreen modes.
<code>void tincle_set_cursor_visible(bool visible);</code>	Toggles engine software cursor visibility instantly.
<code>void tincle_set_custom_cursor(int img_id, int hotspot_x, int hotspot_y);</code>	Replaces software cursor with image ID and hotspot offsets.
<code>void tincle_reset_cursor_to_default(void);</code>	Restores default mouse cursor image (<code>cursor.png</code>).
<code>unsigned char* tincle_load_asset(const char* filename, size_t* out_size);</code>	Reads the file according to the TinCLE engine's standard file input method. The file size is returned in <code>size_t</code> .
<code>void tincle_set_master_volume(float vol);</code>	Sets master audio volume (<code>0.0f</code> to <code>1.0f</code>). Default is <code>1.0f</code> . Multiplies both BGM and SE.
<code>void tincle_set_bgm_volume(float vol);</code>	Sets BGM volume (<code>0.0f</code> to <code>1.0f</code>). Default is <code>0.8f</code> for ear-protection.
<code>void tincle_set_se_volume(float vol);</code>	Sets Sound Effects (SE) volume (<code>0.0f</code> to <code>1.0f</code>). Default is <code>0.8f</code> for ear-protection.
<code>float tincle_get_master_volume(void);</code>	Returns current master volume level (<code>0.0f</code> to <code>1.0f</code>).
<code>float tincle_get_bgm_volume(void);</code>	Returns current BGM volume level (<code>0.0f</code> to <code>1.0f</code>).
<code>float tincle_get_se_volume(void);</code>	Returns current SE volume level (<code>0.0f</code> to <code>1.0f</code>).
<code>void tincle_set_event_callback(tincle_event_cb_t cb);</code>	Event Interceptor Registrar. Registers a custom SDL event callback function. Call inside <code>game_load()</code> to forward raw <code>SDL_Event*</code> handles directly to

Function Signature	Description & Critical Notes [Args / Return]
	third-party modules or extensions (e.g., <code>gatochoc_handle_event</code>).
<code>struct SDL_Window* tincle_get_window(void);</code>	Native Window Accessor. Returns pointer to internal <code>SDL_Window</code> . Essential for initializing external UI frameworks or managing window-level extension backends.

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_ADD.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).

Note for AI Developers / Skill Delivery: When generating or implementing code that requires advanced 2D/3D physics simulation or custom on-screen GUI/HUD interfaces, refer to the `SKILL_AON1.md` talent chip for complete API definitions, header bindings, and implementation rules.

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()`.
- **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.

```
#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/
#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 = tingle_load_font("font.ttf", 32);
// --- Example: Correct Resource Loading (Load ONLY ONCE here!) ---
// g_custom_cursor = tingle_load_image("my_cursor.png");
// tingle_set_custom_cursor(g_custom_cursor, 0, 0);
// RULE: Register states and set initial state at the end of game_Load()
tingle_add_state(0, title_update, title_draw);
tingle_change_state(0);
}
#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tingle_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) { tingle_start(ga
me_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 (`tingle_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 `tingle_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 `tinglec` 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, **always update the relative mode state FIRST before calling TinCLE cursor APIs.**

```
// Correct implementation example for game/UI mouse capturing
static void enforce_game_mouse_capture(void) {
    SDL_bool want_relative = g_mouse_captured ? SDL_TRUE : SDL_FALSE;
    if (SDL_GetRelativeMouseMode() != want_relative) {
        SDL_SetRelativeMouseMode(want_relative);
    }
}

static void set_game_mouse_capture(int capture) {
    g_mouse_captured = capture ? 1 : 0;
    // 1. Synchronize relative mode first
    enforce_game_mouse_capture();
    // 2. Toggle software cursor visibility via TinCLE
    tincle_set_cursor_visible(g_mouse_captured ? false : true);
    // 3. Flush relative state buffer
    SDL_GetRelativeMouseState(NULL, NULL);
}
```

```
}
```

Note: `enforce_game_mouse_capture()` must ONLY synchronize the relative mouse state. It MUST NOT invoke any cursor visibility APIs internally.

8. 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...]`
2. **Generate Icon:** If a multi-size `.ico` is not yet prepared, convert an image via `icondom.exe` :

```
icondom icon.png icon.ico
```

3. **Embed Resources (Optional):** Embed the generated icon and Windows manifest:

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

(Use `"none"` to skip ico/manifest).

4. **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`:

```
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).*

5. **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...]`
 - **[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.
3. **ZIP Checklist (5 Items Only):** `index.html`, `index.js`, `index.wasm`, `index.data`, `CREDITS.txt`.
4. **[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
```

3. **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.

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>
    #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
```

TinCLE (Tiny C Lovely Engine) - Additional High Level Skills Version 0.12

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:

- State Count (≥ 4):** The game design requires 4 or more distinct states (e.g., Title, Gameplay, Result, Shop, Pause).
- Code Scale (≥ 400 Lines):** The estimated total C source code exceeds 400 lines.
- Complex Custom Systems:** The project defines complex data structures (e.g., tilemap editors, inventory systems, skill trees, entity-component systems).
- Explicit User Instruction:** The user explicitly requests modular, multi-file, or large-scale generation.

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`

```
#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`

```

#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); return 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 `emcc` CLI workflow.

However, if a complex game requires WebAssembly multithreading (utilizing `SharedArrayBuffer` and `pthread`), 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 `emcc` 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.

2. Recommended Build Pipeline:

- **Step 1:** Execute standard `emcc` 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>`).

3. 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});
```

2. **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.
3. **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. **GLSL2 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
<code>void tinkle_render_shader(const char* filename);</code>	Call inside <code>game_load()</code> to keep a background shader running continuously.
<code>int tinkle_load_shader(const char* filename);</code>	Call inside <code>game_load()</code> . Initiates compilation and returns Shader ID (<code>int</code>). In WebGL/WASM environments, compilation runs asynchronously.
<code>void tinkle_shader_poll_async(void);</code>	[MANDATORY for WebGL] Call every frame in the main loop or update callback to monitor and complete background shader compilation.
<code>bool tinkle_shader_is_ready(int shader_id);</code>	Returns <code>true</code> if the shader has finished compiling and is ready for use. Returns <code>false</code> while compiling or on error.
<code>void tinkle_set_shader(int shader_id);</code>	Sets active background shader ID (<code>-1</code> to disable). Safely falls back to disable if the target shader is not READY yet.

Function Signature	Description & Critical Notes
<code>void tincle_shader_use(int shader_id);</code>	Activates shader for uniform updates. Has overhead (glUseProgram). Call only when switching scenes or updating values.
<code>void tincle_shader_send_vec3(int shader_id, const char* name, float x, float y, float z);</code>	Binds a <code>vec3</code> uniform value to the target shader.
<code>void tincle_shader_send_float(int shader_id, const char* name, float value);</code>	Binds a <code>float</code> uniform value to the target shader.
<code>void* tincle_get_gl_context(void);</code>	OpenGL Context Accessor. Returns raw <code>SDL_GLContext</code> pointer (<code>void*</code>). Used when configuring OpenGL/GLES2 pipelines, shaders, or custom GL rendering extensions under <code>TINCLE_MODE_GLSL_HERO</code> .

[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:

- 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.
- 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 (`piisschk`)

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 `piisschk` before deployment.

```
piisschk <shader_file_path>
```

- **Example:** `piisschk 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 `piisschk` 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()`.

```
#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[]) { tingle_start(game_load, "GLSL_GAME", true, 640, 480, TINCLE_MODE_GLSL_HERO); return 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tingle_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 `tingle_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.

2. 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.

TinCLE (Tiny C Lovely Engine) - Composition & MML Generation Talent Chip Version 0.12

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

1. 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 `04`).
- **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.

1.1 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 `05` 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., `05`) at the very beginning of the line.**

- **Good:** `CH1: 05 T96 V13 @73 L8`
- **Good:** `CH1: 05 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):

```
; Base octave starts at 05
CH1: 05 C4 D4 >E4 F4      ; '>' shifts base octave to 06!
CH1: D4 F8 A8 >D4 C8 <A8   ; '>' shifts base octave to 07! Creeps endlessly higher!
```

Good 1 (Bulletproof & Recommended: One-Shot Modifiers):

```
; Use '^' or '_' for momentary leaps. Base octave stays safely at 05!
CH1: 05 C4 D4 ^E4 F4      ; '^E4' plays E in 06, but base octave REMAINS 05!
CH1: 05 D4 F8 A8 ^D4 C8 A8 ; Head of line explicitly has '05', base octave is rock solid!
```

Good 2 (Explicit Shift & Reset):

```
; If you MUST use '>' for a passage, ALWAYS reset it with '<!'
CH1: 05 C4 D4 >E4 F4      ; Shift base octave to 06
CH1: 06 D4 F8 A8 D4 C8 <A8 ; Shift base octave back to 05!
```

Pre-Output Checklist (Octave & Pitch Check)

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

2. 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).

3. 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**:

```
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:
$$\text{Total Seconds} = \frac{60}{\text{Tempo (BPM)}} \times \text{Beats per Bar} \times \text{Total Bars}$$
 - *Example (Tempo 96, 4/4 Time, Target: 2.5 minutes / 150 seconds):*
 - Seconds per Bar = $(60 / 96) \times 4 = 2.5$ seconds
 - Total Bars Needed = $150 / 2.5 = \mathbf{60 \text{ bars}}$
2. **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.
3. **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.
4. **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?

4. 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):

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
Abso luteBa nger	I - V/VII - V Im - IIIIm/V - IV V - I	Flat 7Sub_ m5	IIIm7 - IVm(b 5)/V - I	Flat 7Sub_ Mino r	IIIm7 - IVm/V - I
Sunn yAF	I/IV - Vsus4 - IIIIm7 - V/V I	TwoS tageC	IV - I - V - VIIm - I V/VII - VIIm V	Room 335	IV△7 - IIIIm7 (#5) - IIIm7 - I IIIm7

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
		horu s			
Vibe 70s	IΔ7 - bVIΔ 7 - IVm7 - bV II7	Clas sicWe st70 s	IVΔ7 - IIIIm7 - IIm7 - IΔ7	Clas sicWe st70s _v2	IVΔ7 - IIIIm7 - IIm7 - IIIIm7
Brid geToCh orus	IV - I/III - IIm7 - Vsus4 V	B_Me lodyV ibe	IIm7 - VIm - IVM7 - Vsus4 V	B_Me lodyV ibe_v 2	IIm7 - VIm - I VM7 - IIIsus4 I II7
TwoF ive_To _4	Vm7 - I7 - I V	JPop Gold	IV - V - IIIIm - VIm	JPop Gold_ Remi x	IV - V/IV - II I7 - VIm
Just TwoOfU s	IVmaj7-III7-V Im7-I7	Kawa iiStr uctur e	VIm7 - bVIm7 - Vm7 - I7	MyRe volut ion	I - VIm - IV - V
Smoo th_Rn B	IIm9 - V7 - I M7 - IVM7 - VI Im7-5 - III7 - VIm - VI7	TwoF ive_T o_6m	VIIIm7 - III7 - VIm	YOAS OBI_V ibe	IV - V - IIIIm7 - VIm7 - IIm7 - III7 - VIm - Vm 7 - I7
Supe rLit	VIm - bIIIIm (b5) - #IVm (b5) - III	Anis onCan on	I - VIIIm7(b 5) - III7 - VI m7 - Vm7 - I7 - IVΔ7 - IIIIm 7 - IIm7 - IV/ V - V	Anis onGol d	I - VIIIm7(b5) - III7 - VIm7 - Vm7 - I7 - IVΔ 7
Anis onHi t	IV - V - bVI - bVII- I	Ennu iBab e	I - VI/I# - I Im - IVm - I/V	Yest erday Vibe	I - III - VIm - V

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
			- II/#IV - IV - V		
Clas sic162 5	I - VIm - IIm - V	Best iesVi be	I - VIm - IV - V	Inte rstel lar	IV - V - VIm - V
Epic Peda l	I - bII/I - II/I - bIII/ I	Pure Emoti onal	IV - V - I - III	AllR ounde r	VIm - V - IV - V
Chic Babe	IV△7 - IIIm7 - VIm7 - I	Maru saChi c	IV△7 - III7 - VIm7 - II7	Chil lGood nigh t	IV△7 - I△7 - III/#IV - IIIm/ VI
Tang oVib e	VIm - IIm - I II - VIm	Cano nClas sic	I - V - VIm - IIIm - IV - I - IIm - V	Cafe Lofi	VIm9 - IIm7 - IIIm7
Crow dPleas er	I - IV - VIm - V	Catc hyInt erlud e	I - V - I - V	Clea nAest heti c	I/III - IV - V - VIm
Clas sicalG rand	IV△7 #IVm(b 5) - V6 bVIDi m - VIm bVIId im - VII - II I	Clas sical Seeke r	IIm7(b5) - b II/IV - IVm(b 5)/VII - I	Game Melod yScen t	IV△7 - IIIm7 - bIII△7 - II m7
Clut chMome nt	IV - V - I	Floa tyChi c	IIm7 - I△7 - bVII△7 - VIIm 7(b5) - III7	Pure Simpl e	VIm - V - IV - V
TheB lues	I7 - IV7 - I7 - IV7 - V7 - I V7 - I7 - V7	Sout hernV ibe	IIm9 - V7 - I IIm7 - VIm	Susp enseR ise	VIm - IV/VI - #IVm(b5)/VI - VIm7

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
Chor usRus h	VIm - V/VII - I - IIm - IIIs us4 - III	Chor usHi t	I - III - IV - V - VIm (or I)で帰結	Chor usPow er	I - V - IV - V
PreC horusR egula r	IV - #IVm(♭ 5) - V - ♭VIm (♭5)	Unst oppab lePas sion s	I - Iaug/#IV - IV△7 - V	City PopVi be	VIm - I - IV - III
Cine maticM ood	IIm - VIm - I - V	Jazz Stand ard_ 1	IIm7 - V7 - I △ - I△	Jazz Stand ard_ 2	IIm7 - V7 - I △ - VI7
Jazz Standa rd_3	I△ - VIm7 - IIm7 - V7	Jazz Stand ard_ 4	I△ - I#o7 - IIm7 - V7	Jazz Stand ard_ 5	I△ - I7 - IV△ - IVm7
Jazz Standa rd_6	VIm7(♭5) - I Im7 - V7 - I △	Jazz Stand ard_ 7	IIm7(♭5) - V 7 - Im7 - Im7	Jazz Stand ard_ 8	Im7 - V7 - Im7 - V7
Jazz Basi c	IIm7 - V7 - I △7 - VI7	Simp leIsB est	I - IV - V	Swee tPrin cess	I - VI/I# - II m - IVm - I/V - II/#IV - IV - V
Styl ishMoo d	VIm9 - IIm7 - IIIm7	SpyA gent	VIm - IV/VI - #IVm(♭5)/VI - VIm7	Span ishTr ip	III - IV - V - IV

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
Stan dardTw oFiv e	IIm7 - V7 - I	Dyst opian Worl d	VIm - VII - I Im - III	Dram aPuls e	IIm7 - I - VII - IIIIsus4 III7
Dram aPulse _v2	IIm7 - VIIIm7- 5 - IIIIsus4 - III7	Dori anMod e	IIm - IIIIm/II - IV/II - III m/II	Dori anPro of	IIIIm/II - IIm
Dres sCodeF orma l	IV△7 - Iaug/ ♭VII - IIIIm7/ VI - II/#IV	Natu ralDa ily	I - IIm7 - I/ III - IV	Smoo thCan on	I - V/VII - VI m - IIIIm/V - IV - I/III - IIm - V
Airy Vibe	I - IIIIm/VII - VIm - IV	Part yTun e	I - IV - V - IV	Pira teQue st	VIm - IIm - II I - VIm
Hous eGroov e	IIm7/V - IIIIm 7/VI	Back doorT woFiv e	IVm7 - ♭VII7 - I	Happ yVibe s	VIm7 - II7 - I V/V - I
Para llepWo rld	VI - IV - IIm - III	Baro queSe quenc e	VIm - IIm - V - I - IV - VII m(♭5) - III	Baro queCl assi c	VIm - IIm - V - I - IV - VIIIm (♭5) - III
HitC hartBa nger	I - IV - V - I	Bill board Top	VIm - I - V - IV	Fant asyWo rld	VIm - II - IV - V
Fusi onSu b	IIm7 - IV/V - I	Blac kadde rSub	IIm7 - IVaug/ V - I	Blue syFee l	I7 - IV7 - I7 - IV7 - V7 - IV 7 - I7 - V7
Blue s12Ba	I7 - I7 - I7 - I7 - IV7 - I	Deep BassL	I - Iaug - VI m/I - I7	Peda lRise	I - ♭II/I - I I/I - ♭III/I -

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
r	V7 - I7 - I7 - V7 - IV7 - I7 - I7	ine		_1	III/I
Peda lRise_ 2	I - IIIm/I - I IIIm/I - IV/I - Vsus4/I - V	Peda lRise_ 3	I - V/I - IV/ I - bVI/I - b VII/I	Holy Textu re	VIm - V/VII - V - I - IV - VI Im7(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 - IIIm - V	Pops _5	I - IIIm - IV - V	Pops _6	I - V/VII - VI m - V
Pops _7	VIm - IIIm - I II - VIm	Pops _8	VIm - I - IIIm - IIIIm	PopP unkBa nger	I - V - VIm - IV
Mixo lydian Flow	V - IV - I	Mixo lydia nSpac e	V - IIIm - VIm - V	Myst eryDr ama	VIm - VIm - IV 7 - III7
Myst eryNov el	VIm - IV7 - I II7	Mell owOud ou	IV△7 - V7/IV - IIIIm7 - VIm 7	Mode rnMet al	VI5add9 - IV5a dd9 - bIII5add 9 - I5add9
Mono chrome Cano n	I - VIIIm7(b 5) - III7 - VI m - V	Euro beatE nerg y	IV - V - VIm	Clim axRus h	VIm - III7/#VI - IIIIm/V - II/# IV - IV - I/III - IIIm7 - Vsus4 V - I
Fant asyGhi bli	VIm - V - IV - IIIIm - IIIm - VIm/I - VII7 - III7 - VIm	Love Stor y	I - IV - IIIIm - VIm	Rhyt hmAnd Blue s	I△7 - bVI△7 - IVm7 - bVII 7

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
Lydi anBrig ht	IV - V/IV	Rela xChil lin	I - IIm7 - I/ III - IV	Loop Drive e	IIm7 - IIm7 - IIm7 - IIm7
Loop Drive_ v2	IIm7 - VIm7 - IIm7 - VIm7	Rare Chara cter	IV△7 - IIm7 - ♭IIIdim7 - IIm7	Rock Drive _1	I - VIm - IV - V
Rock Drive_ 2	I - IV - I - V	Rock Drive _3	I - I - ♭III - IV	Rock Drive _4	I - III - IV - IVm
Rock Drive_ 5	IIm - I - V - ♭VII	Rock Drive _6	V - V - IV - I	Rock Drive _7	VIm - IV - I - V
Rock Drive_ 8	VIm - III7 - II - V7	Rock Vocal ist	IV△7 - V7/IV - IIm7/V - VI m7	Roma nceMe lody	IV - V - III7 - VIm
Roma nticNi ght	I - IMaj7/VII - I7/♭VII - V I7 - IIm - I#a ug - IIm7/I - V	Nost algic Euro	IIm7 - V7 - I IIm7 - VIm	Live ForLo ve	I - VIIm7(♭5) /II - III7 - VI m7 - V
Surp riseTw ist	I - V - ♭VII - IV - ♭VI - I - II - V	Isek aiFan tasy	VIm - II - IV - V	Firs tStar Rebor n	IV△7 - IIm7 - VIm7 - I
SciF iSpac e	VIm - Im - VI m - IVm	Hero icEpi c	VIm - III/#VI - VI7/V - IIm - IV - IIIsus4 III	Flam eBrea th	VIm - V - IV - I
Glob alHi	I - IV - VIm - V	Oudo uClas	IV - V - IIm - VIm	MyFa vorit	VIm - #IVm7(♭ 5) - IV△7 - II

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
t		sic		eBang er	m7 - IIIm7
Warm HalfSt epRis e	I - Iaug - VI m/I - I7	Kach ouFug etsu	IV6 - V6 - VI 7sus4 - IIIsus 4	Gorg eousM elod y	I - III/VII - VIm - V - IV - IIIm - IIm
Salv ationR ise	IV - V - VI	Oute rSpac eJour ney	VIm - bIIIm - VIm - IVm - VIm - bIIIm - bVIIIm - IIm - #IVm	Revo lutio nPuls e	VIm - IV/VI - VIm - VII7/VI - III/#VI - IIIm (b5)/V - II/#I V - IIm/IV - V I/III
Inte rludeS pecia l	VIm - IIm7 - IV - V	Maru saLoo p	IV△7 - III7 - VIm7 - I7	Maru saLoo p_v2	IV△7 - III7 - VIm7 - Vm7 I7
Maru saLoop _v3	IV△7 - III7 - VIm7 - II7	Rare Speci es	IIIm - IV - V Im - I	Hype Build Up	VIm - IIIm/V - IV V - I
Your TrueFo rm	VIm - IIm7 - IV - V	Cybe rFutu re	VIm - V/VI - IV/VI - IIIm/V I	Moon light Shin e	VIm - III/#VI - VI7/V - IIm/I V - IV7 - III/# VI
Unre achabl eSky	IV - II/#IV - Vsus4 V - VIsu s4 VI - IIsus4 II	Mira gePha ntom	IV△7 - VIIIm7 (b5)/IV - III m7 - IIIIdim7 - IIIm7 - V7/II - I△7 - I6	Phan tasma gori c	I - bVII - b VI - bVII
Autu mnLeav es	IIm7 - V7 - I △7 - IV△7 -	2AM_ Cross ing	Iadd9 - VIm7 - V - IV	Deep Regre t	IIm - VIm/I - VIIIm7(b5) - II I7

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
	VIIm7($\flat$5) - I II7 - VIm7				
Hear tPound ing	IV - $\sharp$IVdim - V - $\flat$VI_{dim} - VIm	High Class Elega nce	IΔ7 - I$\sharp$dim7 - IIIm7 - V7/I I	Toug hProg ress	VIm - V/VII - I - IIIm - III_{su} s4 - III
Swee tSuga r	IV - VIm - II m - I	Ulti mateB ange r	IV - V - VIm - VIm - V - I V	Fina lHit	VIm - V - IV - I
Radi antShi ne	I - V - $\flat$VII - IV - $\flat$VI - I - II - V	Circ ularL oop	I-VIm-IIIm-V7	Step wiseD escen ding	VIm - V - IV - IIIIm - IIIm - I
RunU pStar t	IIIm7 - IIIIm7 - IVΔ7 - V	Komu roDyn amit e	VIm - IIIm - V - I	Komu roCla ssic	VIm - IV - V - I
Komu roClas sic_v 2	VIm - IV - V - I V/VII - VI m	Komu roCla ssic_ v3	VIm - IIIm - V - I	Time lessB ange r	I - V - IV - V
Fier yPassi on	VIm - VII/VI - VIIm($\flat$5)/VI - VIm - I/V - $\sharp$IVm($\flat$5) - IV 7($\flat$5) - III7	Forw ardDr ive	I - $\flat$VII - $\flat$ VI - $\flat$VII	Fina llyFr ee	VIm - III7/$\sharp$VI - IVΔ7 - IIIm9 - Vsus4
OneW orld	Isus4 - Isus 4/ $\flat$II - Isus 4/ $\flat$III - Isus	Hype Rise	IIIm7 - IIIIm7 - IVM7 - V	Hype Rise_ v2	IIIm7 - IIIIm7 - IVM7 - II/$\sharp$IV - Vsus4 V

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
	4/IV - Isus4/ V				
Yout hfulDa ys	I - III - VIm - V	Nost algia Trigg er	VIm - IIIIm - IV - V - I	Maje sticG rand	Isus4 - Isus4/ bII - Isus4/ b III - Isus4/IV - Isus4/V
Beyo ndAtmo spher e	IIm7 - IIIIm7 - IVm7 - Vm7	Gran dFina le	I - I/III - I sus4/IV - Isus 4/V	Chas erOud ou	IV△7 - bVII7 - IIIIm7 - IIIIdi m7/VI
Clas sicBas sChrom atic	VIm - bVIaug - I/V - #IVm7 (b5)	Cybe rGir l	VIm7 - bVIm7 - Vm7 - I7	Gran dOpen ing	VIm - V/VI - I V/VI - IIIIm/VI
Urba nAesth etic	VIm7 - II7 - IV/V - I	Orie ntalM yster y	IV6 - V6 - VI sus4 - IIIIsus 4	Spec ialMa rusa	VIm - I - IV - III
Enem yWithi n	IV - VIm - Im - V/II - bVII m/ bII - IV/I - VIIIdim7 - II I7/#VI	Path osGri ef	VIm - VII - I Im - III	Flyi ngGir l	IV - V - VI
Beau tifulC anon	I - V/VII - V Im - IIIIm/V - IV - I/III - I Im - V	Nigh tPara de	IIIIsus4add b9 - IIIIsus4add b 9/IV - IIIIsus4 add b9/II - II Isus4add b9	Sudd enBit tersw eet	IV - V - I - I II
Anti Ordina ry	#IVm7(b5) - IV△7 - IIIIm7 - VIm7	Floa tingV ibe	I - I/III - I sus4/IV - Isus 4/V	Masq uerad	VIm - VII/VI - VIIIm(b5)/VI - VIm - I/V - #IV

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
				eBal l	m($\flat$ 5) - IV7($\flat$ 5) - III7
TwoF ive_To _Relat iveMin or	VIIIm7($\flat$ 5) - III7 - VIm	Heis eiBal ladWe apon	IIIIm7 - VIm7/ III - IIm7 - V 7/II	Kome KomeL oop	IIm7 - V7 - IM 7 - VIm7 - $\flat$ VI 7 - IIm7/V - V7 - I
J_Po pPun k	IV - I - V - VIm	OurF utur e	I - IIIIm/VII - Vm7/ $\flat$ VII - VI7 - IIm - I# aug - IV/I - V 7/VII	AllP urpos eSeas onin g	IV - V - VIm - I
Bitt erswee tLov e	I - V - VIm - IV	Sunl ightF ilter ed	IV $\triangle$ 7 - IVm $\triangle$ 7 - IIIIm7 - Vm7/ VI	Eleg antPa ssing Dim	I $\triangle$ 7 - I#dim7 - IIm7 - V7
Play fulVib e	I - V/I - IV/ I - $\flat$ VI/I - $\flat$ VII/I	West ernCh arthHi t	IV - I - V - I	West ernSt yle	VIm - I - V - IV
West ernPop Punk	VIm - IV - I - V	West ernSt andar d	VIm - IV - I - V	Trit oneSu b	IIm7 - $\flat$ II7 - I
Tear sFalli ng	IV - V - $\flat$ VI dim - VIm	Tear jerke rWeap on	IV - V - $\flat$ VI m($\flat$ 5) - VIm	Powe rOfLo ve	I - IIIIm/VII - VIm - IV
Spar klingS hine	I - IVm6/I	Pure Beaut y	IV $\triangle$ 7 - VIIIm7 ($\flat$ 5)/IV - III m7 - IIIIdim7 -	Viol etTwo	I - IIIIm7($\flat$ 5) - VI7 - IIm7 - VIIIm7 - III7

Atmo spher e	Progression (Degree)	Atmo sphe re	Progression (D egree)	Atmo sphe re	Progression (De gree)
			IIm7 - V7/II - I△7 - I6	FiveC ombo	
Soca _Tropi calPar ty	I - IV - V - IV	Soca _TwoC ordDr ive	I - V	Soca _Bahi aAnth em	I - bVII - IV - I
Soca _Amazo nToad a	VIm - IV - V - VIm	Soca _Pago deDri ve	I - VIm - IIm - V	Soca _Carn ivalR ise	IV - V - I - V Im

5. 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.

5.1 Basic Scales & Church Modes

English N ame	Scale De grees	English Nam e	Scale Deg rees	English Name	Scale Degr ees
Major Sc ale / Ion ian Mode	1, 2, 3, 4, 5, 6, 7	Natural Min or Scale / A eolian Mode	1, 2, b 3, 4, 5, b6, b7	Harmoni c Minor Scale	1, 2, b3, 4, 5, b6, 7
Melodic Minor Sca le	1, 2, b 3, 4, 5, 6, 7	Dorian Mod e	1, 2, b 3, 4, 5, 6, b7	Phrygia n Mode	1, b2, b 3, 4, 5, b 6, b7
Lydian M ode	1, 2, 3, #4, 5, 6, 7	Mixolydian Mode	1, 2, 3, 4, 5, 6, b7	Locrian Mode	1, b2, b 3, 4, b5, b6, b7

5.2 Harmonic & Melodic Minor Derived Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Locrian $\flat$ 6 Scale	1, $\flat 2$, $\flat 3$, 4, $\flat 5$, 6, $\flat 7$	Ionian $\sharp 5$ Scale	1, 2, 3, 4, $\sharp 5$, 6, 7	Dorian $\sharp 4$ Scale	1, 2, $\flat$ 3, $\sharp 4$, 5, 6, $\flat 7$
Phrygian Dominant (H.M. P5 Below)	1, $\flat 2$, 3, 4, 5, $\flat 6$, $\flat 7$	Lydian $\sharp 2$ Scale	1, $\sharp 2$, 3, $\sharp 4$, 5, 6, 7	Super Locrian $\flat 7$ Scale	1, $\flat 2$, $\flat 3$, $\flat 4$, $\flat 5$, $\flat 6$, $\flat \flat 7$
Dorian $\flat$ 2 Scale	1, $\flat 2$, $\flat 3$, 4, 5, 6, $\flat$ 7	Lydian $\sharp 5$ Scale / Lydian Augmented	1, 2, 3, $\sharp 4$, $\sharp 5$, 6, 7	Lydian $\flat 7$ Scale / Achromatic Scale	1, 2, 3, $\sharp 4$, 5, 6, $\flat 7$
Mixolydian $\flat 6$ Scale	1, 2, 3, 4, 5, $\flat$ 6, $\flat 7$	Locrian $\flat$ 2 Scale / Half-Diminished	1, 2, $\flat$ 3, 4, $\flat$ 5, $\flat 6$, $\flat 7$	Super Locrian Scale / Altered Scale	1, $\flat 2$, $\flat 3$, $\flat 4$, $\flat 5$, $\flat 6$, $\flat 7$

5.3 Symmetrical, Pentatonic & Special Scales

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Whole Tone Scale	1, 2, 3, $\sharp 4$, $\sharp 5$, $\sharp 6$	Diminished Scale (Half-Whole)	1, $\flat 2$, $\flat 3$, 3, $\sharp$ 4, 5, 6, $\flat 7$	Diminished Scale (Whole-Half)	1, 2, $\flat$ 3, 4, $\flat 5$, $\flat 6$, 6, 7
Chromatic Scale	1, $\sharp 1$, 2, $\sharp$ 2, 3, 4, $\sharp 4$, 5, $\sharp 5$, 6, $\sharp 6$, 7	Symmetrical Augmented Scale	1, $\flat 3$, 3, 5, $\sharp 5$, 7	Major Pentatonic Scale	1, 2, 3, 5, 6

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
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				

5.4 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

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Kumoi Scale	1, 2, $\flat$ 3, 5, 6	Iwato Scale	1, $\flat$ 2, 4, $\flat$ 5, $\flat$ 7	Hirajoshi Scale	1, 2, $\flat$ 3, 5, $\flat$ 6

5.5 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, $\flat$ 7	Jiao Mode (Chinese Scale)	1, $\flat$ 3, 4, $\flat$ 6, $\flat$ 7
Zhi Mode (Chinese Scale)	1, 2, 4, 5, 6	Yu Mode (Chinese Scale)	1, $\flat$ 3, 4, 5, $\flat$ 7	Gypsy Scale / Hungarian Gypsy	1, $\flat$ 2, 3, 4, 5, $\flat$ 6, 7
Spanish Scale / Flamenco Scale	1, $\flat$ 2, 3, 4, 5, $\flat$ 6, $\flat$ 7	Hungarian Scale / Hungarian Minor	1, 2, $\flat$ 3, $\sharp$ 4, 5, $\flat$ 6, 7	Scottish Scale	1, 2, 3, 5, 6
Pelog Scale	1, $\flat$ 2, $\flat$ 3, 5, $\flat$ 6	Bilaval Thaata	1, 2, 3, 4, 5, 6, 7	Khamaj Thaata	1, 2, 3, 4, 5, 6, $\flat$ 7
Kafi Thaata	1, 2, $\flat$ 3, 4, 5, 6, $\flat$ 7	Asavari Thaata	1, 2, $\flat$ 3, 4, 5, $\flat$ 6, $\flat$ 7	Bhairav Thaata	1, $\flat$ 2, 3, 4, 5, $\flat$ 6, 7
Bhairavi Thaata	1, 2, $\flat$ 3, 4, 5, $\flat$ 6, $\flat$ 7	Kalyan Thaata	1, 2, 3, $\sharp$ 4, 5, 6, 7	Marwa Thaata	1, $\flat$ 2, 3, $\sharp$ 4, 5, 6, 7

English Name	Scale Degrees	English Name	Scale Degrees	English Name	Scale Degrees
Poorvi Thaat	1, $\flat 2$, 3, $\sharp 4$, 5, $\flat 6$, 7	Todi Thaat	1, $\flat 2$, $\flat 3$, $\sharp 4$, 5, $\flat 6$, 7	Maqam Rast	1, 2, $\flat 3$ (Quarter Tone), 4, 5, 6, $\flat 7$ (Quarter Tone)
Maqam Nahawand	1, 2, $\flat 3$, 4, 5, $\flat 6$, 7	Maqam Nawathar	1, 2, $\flat 3$, $\sharp 4$, 5, $\flat 6$, 7	Algeria n Scale	1, 2, $\flat 3$, $\sharp 4$, 5, $\flat 6$, 7
Double Harmonic / Byzantine / Arabic	1, $\flat 2$, 3, 4, 5, $\flat 6$, 7				



6. 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 (V or V_7) to Tonic (I)
 - 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 (V)
 - 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 (IV) to Tonic (I)
 - Characteristics: Commonly known as the "Amen Cadence" used in church music, it delivers a gentle, soft, and soothing resolution.
 - **Deceptive Cadence**

- Progression: Dominant (\$V\$) to a chord other than the Tonic (typically \$VI\$)
- 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.

7. 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.

8. Duration Calculation & Target Lengths

- **Calculation Formula:**
 - $\text{\textit{Duration of 1 Beat (Quarter Note)}} = \frac{60}{\text{\textit{BPM}}} \text{\textit{seconds}}$
 - 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.

9. 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 05 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 04 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 03 V12 @33 L4
```

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.

10. 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).

TinCLE (Tiny C Lovely Engine) - Sound Effect Generation & Synthesis Talent Chip Version 0.12

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 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 file, and optionally saves the generated parameters to a **.fart** text file for future editing.

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

- t <preset_name>** : Specifies the sound preset generator.
- o <output.wav>** : Output target WAV file path.
- 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 file. This is the primary method for AI agents to write custom audio parameters.

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

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

Pattern C: Mutation Workflow

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

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

- `-c <input.fart>` : Base parameter file to mutate.
- `-m` : Enables mutation mode.
- `-o <output.wav>` : Output target WAV 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-attach 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`.

4. 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>` 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
```

TinCLE (Tiny C Lovely Engine) - Networking Talent Chip Version 0.12

~ 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
```

2. 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:
 - `my_server.exe` (The `poopee` -processed server executable)
 - `CREDITS.txt` (License and attribution credits)
 - `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
<code>tincle_net_set_game_id(const char* game_id_or_uuid)</code>	<code>void</code>	Configures full 36-character GAME_ID (UUID) used for server authentication. Must be called before <code>`tincle_net_connect`</code> .
<code>tincle_net_connect(const char* ip, int port, uint16_t room_id)</code>	<code>int</code>	Initiates asynchronous TCP connection to the server and triggers automatic Peer ID handshake. Returns <code>1</code> on socket creation success, <code>0</code> on failure.
<code>tincle_net_disconnect()</code>	<code>void</code>	Closes network socket and resets client Peer ID state.
<code>tincle_net_send(const uint8_t* data, uint16_t size)</code>	<code>int</code>	Broadcasts raw binary payload to all clients in the current <code>room_id</code> . Returns <code>1</code> on success, <code>0</code> on failure.

Function	Return Type	Description
<code>tincle_net_poll(uint16_t* out_sender_peer_id, uint8_t* out_buffer, uint16_t max_size)</code>	<code>int</code>	Non-blocking receive function. Must be called repeatedly in the game update loop. Returns received payload byte count (<code>0</code> if no data). If <code>out_sender_peer_id</code> is <code>0</code>, <code>out_buffer</code> contains the <code>uint16_t</code> Peer ID of a disconnected client sent by the server.
<code>tincle_net_get_peer_id()</code>	<code>uint16_t</code>	Returns the assigned unique Client Peer ID (<code>1</code> ~ <code>65535</code>). Returns <code>0</code> 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.

```
#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 != tingle_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 != tingle_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 (tingle_key_pressed(SDL_SCANCODE_LEFT)) g_my_x -= move_speed * dt;
if (tingle_key_pressed(SDL_SCANCODE_RIGHT)) g_my_x += move_speed * dt;
if (tingle_key_pressed(SDL_SCANCODE_UP)) g_my_y -= move_speed * dt;
if (tingle_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 = tingle_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(PlayerNetworkPacke
t));
    }
}

// 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[]) {

```

```
    tinkle_start(game_load, "My Net Game", true, 640, 480, TINCLE_MODE_2D_NORMA
L);
    return 0;
}
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) {
    tinkle_start(game_load, "My Net Game", true, 640, 480, TINCLE_MODE_2D_NORMA
L);
    return 0;
}
#endif
```

TinCLE (Tiny C Lovely Engine) - 2D/3D Physics & GUI Toolkit Talent Chip Version 0.12

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]
<code>const char* cannoli_get_version (void);</code>	Returns read-only string of cannoli version (e.g., <code>"0.12"</code>).

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]
<pre>bool gatochoc_init(gatochoc_mode_t mode, SDL_Window *win, void *target);</pre>	UI Engine Init. Call inside <code>game_load</code>. <code>mode</code> : <code>GATOCHOC_MODE_SDL_RENDERER</code> or <code>GATOCHOC_MODE_GLSL</code>. <code>win</code> : Pointer to <code>SDL_Window</code> (via <code>tincle_get_window()</code>). <code>target</code> : <code>SDL_Renderer*</code> (SDL mode via <code>tincle_get_renderer()</code>) or <code>SDL_GLContext</code> (GLSL mode). Return: <code>true</code> if success, <code>false</code> on failure.
<pre>bool gatochoc_handle_event(SDL_Event *event);</pre>	Input Forwarder. Pass raw <code>SDL_Event*</code> here inside the event callback (<code>tincle_set_event_callback</code>). Automatically triggers <code>nk_input_begin</code> on first event. Return: <code>true</code> if UI consumed the event, <code>false</code> otherwise.
<pre>struct nk_context* gatochoc_get_context(void);</pre>	Nuklear Context Accessor. Returns pointer to internal <code>struct nk_context*</code>. Use this to call standard Nuklear GUI functions (e.g., <code>nk_begin</code>, <code>nk_layout_row_dynamic</code>, <code>nk_button_label</code>, <code>nk_end</code>).
<pre>void gatochoc_input_begin(void);</pre>	Input Frame Flush. Call at the very top of <code>update_cb</code> before building any Nuklear UI. Flushes and finalizes pending input events (<code>nk_input_end</code>) for the current frame.
<pre>void gatochoc_input_end(void);</pre>	Input Cleanup. Frame input close handler (reserved / optional for explicit multi-pass input states).
<pre>GatoChocFont gatochoc_load_font_from_memory(const void* data, size_t size);</pre>	Standard Font Loader. Loads TrueType font (<code>.ttf</code>) from memory buffer. <code>range_type</code> : <code>GATOCHOC_FONT_DEFAULT</code> (A

Function Signature	Description & Critical Notes [Args / Return]
<code>t size, float font_size, GatoChocFontRange range_type);</code>	SCII) or <code>GATOCHOC_FONT_JAPANESE</code> (Full Japanese including full-width symbols <code>0xFF00-0xFFFF</code>).
<code>GatoChocFont gatochoc_load_font_from_memory_ex(const void* data, size_t size, float font_size, GatoChocFontRange range_type, const unsigned long *custom_ranges);</code>	Extended Font Loader. Custom glyph range font loader. Set <code>range_type</code> to <code>GATOCHOC_FONT_CUSTOM</code> and pass a zero-terminated <code>nk_rune</code> array to <code>custom_ranges</code> for multilingual support (e.g., Chinese, Korean).
<code>void gatochoc_set_default_font(GatoChocFont font);</code>	Default Font Setter. Sets the loaded <code>GatoChocFont</code> as the global default font for all Nuklear UI elements.
<code>void gatochoc_render(void);</code>	UI Draw Pass. Renders generated Nuklear geometry. Call inside <code>draw_cb</code> , right after <code>tiny_nk_clear</code> and game rendering. Auto-finalizes font atlas on first frame if required.
<code>void gatochoc_shutdown(void);</code>	Cleanup. Frees all Nuklear contexts, font atlases, and backend resources. Call on game exit.

2. Core Types & Enums

```
// 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`.

2. 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`.

3. 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.

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

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

```
#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
    tinkle_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

```