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!

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

1. Overview & Concept

- **Env:** TinCLE v0.06 / Pure C (C99) + SDL2, SDL_ttf, SDL2_gfx, stb, GLSL (GLES2 compatible)
- **[CRITICAL] Target Architecture Detection:** The AI **MUST** first check the `info.txt` file in the current working directory to determine the compilation environment. If the AI cannot access files directly, it **MUST** ask the user: *"Please check the contents of info.txt and tell me if this is the 32bit or 64bit environment."* before generating any code.
- **Strict Architecture Rules based on `info.txt`:**
  - **If `info.txt` says `32bit`:**
    - Windows Desktop Target: Win32 DLL (`sizeof(void*) == 4`).
    - Web Browser Target: Wasm32 via Emscripten (`sizeof(void*) == 4`).
    - *Rule:* The environment is purely 32-bit. Never use 64-bit structures or types.
  - **If `info.txt` says `64bit` (Hybrid Environment):**
    - Windows Desktop Target: Win64 DLL (`sizeof(void*) == 8`).
    - Web Browser Target: Wasm32 via Emscripten (`sizeof(void*) == 4`).
    - *Rule:* This is a **Hybrid Architecture**. The compiled Windows EXE will be 64-bit, but the Web WASM build remains 32-bit. A single `game.c` must satisfy both targets.
- **[CRITICAL] Pointer Safety Rule (For Hybrid Compatibility):** Because the 64bit environment cross-compiles to both 64-bit Desktop and 32-bit Web targets from a single source, **never hardcode pointer sizes** or perform unsafe pointer-to-integer casts (e.g., casting a pointer to a standard `int`). Always use `<stdint.h>` types like

`uintptr_t` if pointer-to-integer casting is absolutely required. Ensure `game.c` compiles perfectly on both toolchains without modification.

- **Toolchain Requirements:** TinyCC (TCC for Windows) / Emscripten (for Web).
- **[CRITICAL] `tcc.exe` / `emcc` are NOT bundled in TinCLE;** they must be installed externally and accessible via System PATH.

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`, `pisschk.dll`
  - CLI Tools: `asspack.exe`, `emcass.exe`, `icondom.exe`, `poopee.exe`, `tinclec.exe`, `pisstrak.exe`, `pisschk.exe`

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 Workflow (`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.

- **Shader Validation Workflow (`pisschk`):** Validates GLSL fragment shader files against both Desktop (Radeon-strict) and Web/GLES2 compliance standards before deployment. Catches cross-platform GLSL errors and vendor-specific quirks (such as NVIDIA lenient extensions or missing precision qualifiers) prior to building.

```
pisschk <shader_file_path>
```

- *Example:* `pisschk assets/shaders/water.frag`
- *Parameters:*
  - `<shader_file_path>`: Path to the GLSL fragment shader file to inspect.
- *Validation Coverage:*
  - **Mode 0 (Desktop / Radeon Compliance):** Equivalent to `glslang -S frag -d --no-link --error-column`. Strictly checks desktop GLSL syntax (catches reserved words like `active` and driver-specific lenient code).
  - **Mode 1 (Web / GLES2 Compliance):** Validates code with automatically prepended `precision mediump float;` header to ensure GLES2/WebGL compatibility for web exports.
- *Exit Codes:*
  - `0`: All shader inspection checks PASSED successfully.
  - `1`: Shader compilation / syntax 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** run `pisschk` on the target shader file to guarantee cross-platform compatibility across NVIDIA, AMD Radeon, and WebGL/GLES2 targets before finalizing the task.

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.
    - *[CRITICAL] Clear Color Rule in GLSL_HERO:* You MUST call `tincle_clear((TinCLE_Color){0, 0, 0, 0})` with **Alpha = 0**. Specifying an opaque alpha will erase/hide the shader layer beneath SDL rendering.
- **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.

Function Signature	Description & Critical Notes [Args / Return]
`void tincle_start(TinCLE_InitCallback init_cb, const char* title, bool is_debug, int w, int h, TinCLE_RenderMode mode);`	**Main Entry.** Bootstraps loop. • `is_debug`: `true` spawns Win debug console (`printf`); `false` hides it for release. • `mode`: Set to `TINCLE_MODE_2D_NORMAL` or `TINCLE_MODE_GLSL_HERO`.
`void tincle_add_state(int state_id, TinCLE_UpdateCallback update_cb, TinCLE_DrawCallback draw_cb);`	Binds state ID to standard callbacks: `update(float dt)` and `draw(void)`.
`void tincle_change_state(int state_id);`	**[CRITICAL]** Switches state. Call at end of `game_load()` or viewport stays black.

Function Signature	Description & Critical Notes [Args / Return]
`int tincle_key_pressed(int scancode);`	Returns `1` if key held. **Must use pure `SDL_SCANCODE_*`.**
`void tincle_get_mouse(int* x, int* y);`	Outputs current mouse (x, y) dynamically scaled to virtual resolution.
`int tincle_mouse_pressed(int button_type);`	Returns `1` if mouse held. **Must use pure `SDL_BUTTON_*` (e.g. LEFT).**
`int tincle_pad_pressed(int pad_button);`	Returns `1` if pad button held. **Must use pure `SDL_CONTROLLER_BUTTON_*`.**
`float tincle_pad_axis_x(void);`	Left stick X (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
`float tincle_pad_axis_y(void);`	Left stick Y (-1.0f to 1.0f). Has built-in <5000 hardware deadzone filter.
`void tincle_clear(TinCLE_Color color);`	**[MANDATORY]** Clears screen. Call at absolute start of every `draw()` callback. **Alpha must be 0 in GLSL_HERO mode.**
`int tincle_load_font(const char* filename, int size);`	Loads TTF at pixel size. **Returns unique Font ID (int).**
`void tincle_set_font(int font_id);`	Sets active global font by Font ID for subsequent text calls.
`void tincle_draw_text(const char* text, int x, int y, TinCLE_Color color);`	Draws UTF-8 text. **Anchor: Top-Left (x, y).** Uses active font.
`void tincle_draw_text_center(const char* text, int x, int y, TinCLE_Color color);`	Draws UTF-8 text. **Anchor: EXACT CENTER (x, y).** Uses active font.
`int tincle_load_image(const char* filename);`	Caches image (PNG/JPG). **Returns Image ID (`img_id`), or `-1`.**

Function Signature	Description & Critical Notes [Args / Return]
`void tincle_draw_image(int img_id, int x, int y);`	Uses `img_id`. **[CRITICAL GEAR] Anchor: EXACT CENTER (x, y)** of unscaled image.
`void tincle_draw_image_ex(int img_id, int x, int y, int w, int h, double angle);`	Uses `img_id`. **[CRITICAL GEAR] Anchor: EXACT CENTER (x, y)** for position, scale, and rotation.
`void tincle_play_music(const char* filename);`	Plays Ogg BGM immediately on infinite loop. Unloads previous music automatically.
`void tincle_stop_music(void);`	Instantly stops and unloads current BGM from memory.
`int tincle_load_sound(const char* filename);`	Caches WAV/Ogg SE clip. **Returns unique Sound ID, or `-1`.**
`void tincle_play_sound(int se_id);`	Plays SE channel via auto-mixer (supports up to 16 concurrent sounds).
`struct SDL_Renderer* tincle_get_renderer(void);`	**Returns native `SDL_Renderer*`** for direct SDL2/SDL2_gfx primitive drawing.
`const char* tincle_get_version(void);`	Returns read-only string of engine version (e.g., `"0.05"`).
`bool tincle_is_fullscreen(void);`	Returns `true` if current window is in full-screen mode.
`void tincle_set_cursor_auto_policy(bool visible_in_windowed, bool visible_in_fullscreen);`	Sets default cursor visibility behavior for Windowed and Fullscreen modes.
`void tincle_set_cursor_visible(bool visible);`	Toggles engine software cursor visibility instantly.

Function Signature	Description & Critical Notes [Args / Return]
`void tincle_set_custom_cursor(int img_id, int hotspot_x, int hotspot_y);`	Replaces software cursor with image ID and hotspot offsets.
`void tincle_reset_cursor_to_default(void);`	Restores default mouse cursor image (`cursor.png`).
`unsigned char* tincle_load_asset(const char* filename, size_t* out_size);`	Reads the file according to the TinCLE engine's standard file input method. The file size is returned in `size_t`.

5.1 GLSL Unleashed API (`TINCLE_MODE_GLSL_HERO` Only)

- **[CRITICAL Shader Code Rules]:**
  1. Shaders **MUST** be 100% GLES2 compliant to run on WebAssembly.
  2. **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 the physical window/screen dimensions, causing critical position shifts.
     - **ALWAYS** calculate UV coordinates using `u_viewport` to strictly lock the rendering area 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 the virtual game resolution (e.g., `640.0 / 480.0`) to both `p.x` and `m.x` to prevent horizontal distortion and mouse offset across different screen sizes.
- **Automatic Shader Uniforms (Built-in):**
  - `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`).

Function Signature	Description & Critical Notes
`void tincle_render_shader(const char* filename);`	Call inside `game_load()` to keep a backgroun d shader running continuously.
`int tincle_load_shader(const cha r* filename);`	Call inside `game_load()` . Compiles shader an d returns Shader ID (`int`).
`void tincle_set_shader(int shade r_id);`	Sets active background shader ID (`-1` to disa ble).
`void tincle_shader_use(int shade r_id);`	Activates shader for uniform updates. **Has ove rhead (`glUseProgram`). Call only when swit ching scenes or updating values.**
`void tincle_shader_send_vec3(int shader_id, const char* name, floa t x, float y, float z);`	Binds a `vec3` uniform value to the target sha der.
`void tincle_shader_send_float(in t shader_id, const char* name, fl oat value);`	Binds a `float` uniform value to the target sh ader.

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:

```
#include "libs/tincle.h" // MANDATORY: Always include engine core first
#include <math.h>        // Explicitly include if math functions (sinf, etc.) a
re used
#ifdef __EMSCRIPTEN__
  #include <SDL2/SDL_scancode.h>       // Example: Web (Use angle brackets <>)
  #include <SDL2/SDL2_gfxPrimitives.h> // For native SDL2_gfx on Web
#else
  #include "libs/SDL_scancode.h"       // Example: Windows (Use local double qu
otes "")
```

```
  #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) {
  // CRITICAL: Use Alpha = 0 when in GLSL_HERO mode to keep shader visible!
  tincle_clear((TinCLE_Color){0, 0, 0, 0});
}
void game_load(void) {
  // RULE: MUST execute load_font as the first line here. Assign to int handle.
  int f = tincle_load_font("font.ttf", 32);
  // Example GLSL Setup (If using TINCLE_MODE_GLSL_HERO)
  // int shader_id = tincle_load_shader("bg.glsl");
  // tincle_render_shader("bg.glsl");
  // --- Example: Correct Resource Loading (Load ONLY ONCE here!) ---
  // g_custom_cursor = tincle_load_image("my_cursor.png");
  // tincle_set_custom_cursor(g_custom_cursor, 0, 0);
  // RULE: MUST register states exclusively inside game_load()
  tincle_add_state(0, title_update, title_draw);
  tincle_change_state(0);
}
#ifdef __EMSCRIPTEN__
int main(int argc, char* argv[]) { tincle_start(game_load, "TITLE", true, 640,
480, TINCLE_MODE_2D_NORMAL); return 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(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 (`tincle_pad_axis`) with max keyboard constants (`1.0f` / `-1.0f`) if arrows/WASD are pressed.
4. **Header Scopes & SDL2 Usage:** For extra SDL2/SDL2_gfx headers, use `<SDL2/filename.h>` inside `__EMSCRIPTEN__` and `"libs/filename.h"` inside `#else` (like `SDL2_gfxPrimitives.h`). You can freely use `tincle_get_renderer()` for custom SDL2/SDL2_gfx drawing extensions, but NEVER modify window dimensions or destroy/alter core SDL window/renderer loops directly.

5. **Shader Normalized Mouse Coordinates (`GLSL_HERO` Mode):**
    - Mouse coordinates returned by `tincle_get_mouse()` are automatically scaled to the virtual resolution (e.g., 640x480).
    - When passing mouse positions to custom GLSL shaders, C code **MUST** normalize values strictly using the virtual game resolution and invert the Y-axis for OpenGL space: `s_mouse_nx = (float)mx / VIRTUAL_WIDTH;` `s_mouse_ny = 1.0f - ((float)my / VIRTUAL_HEIGHT);`
    - Do NOT pass raw screen pixel coordinates directly to the shader if `u_viewport` is used for screen-space UV calculation.
6. Ensure all shader files (`.glsl`) are placed in the root folder and listed in `exsass.txt`.
7. **Strict State Model:** Split logic cleanly via State Management API (e.g., Title, Play, Over). Never write a raw single-loop frame block.
8. **Toolchain Distribution Policy:** Instruct users to use `tinclec` for Win EXE and `emcass` for Web builds. `asspack` handles media asset bundling (supports auto-scanning via `exsass.txt` when called without arguments); `icondom` generates ICO files; `poopee` handles metadata embedding into compiled EXEs.
9. **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 4-step pipeline and enforce documentation mandates.

Step 1: Prep Assets & Hide Console

- Update `CREDITS.txt` with third-party asset attributions. Ensure it is in the root directory.
- 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 collect all assets/shaders matching `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]`).

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
```

```c
// [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_CONTROLL
ER_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, 2
55, 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_CONTROLL
ER_BUTTON_DPAD_LEFT))  g_player_x -= move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_RIGHT) || tincle_pad_pressed(SDL_CONTROLL
ER_BUTTON_DPAD_RIGHT)) g_player_x += move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_UP)    || tincle_pad_pressed(SDL_CONTROLL
ER_BUTTON_DPAD_UP))    g_player_y -= move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_DOWN)  || tincle_pad_pressed(SDL_CONTROLL
ER_BUTTON_DPAD_DOWN))  g_player_y += move_speed * dt;
  if (tincle_key_pressed(SDL_SCANCODE_ESCAPE) || tincle_pad_pressed(SDL_CONTROL
LER_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, 2
00);
}
// --- 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", t
rue, 640, 480, TINCLE_MODE_2D_NORMAL); return 0; }
#else
int WINAPI WinMain(HINSTANCE h, HINSTANCE hp, LPSTR c, int n) { tincle_start(ga
me_load, "YOUR_GAME_TITLE", true, 640, 480, TINCLE_MODE_2D_NORMAL); return 0; }
#endif
```

TinCLE Version 0.06 pisstrak MML Generation Skill Definition

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 `O4`).
- **Volume**: `Vn` (n = 0 to 15, default `V12`).
- **Instrument / Patch**: `@n` (n = 0 to 127. Standard General MIDI Program Numbers).
- **Default Length**: `Ln` (n = 1, 2, 4, 8, 16, 32, 64. Default `L4`).
- **Notes**: `C`, `D`, `E`, `F`, `G`, `A`, `B`.
  - Sharp: `+` or `#` (e.g., `C+`, `F#`)
  - Flat: `-` (e.g., `B-`)
  - Explicit Length: Append number after note (e.g., `C8`, `G2.`)
  - Dotted Note: Append `.` (increases length by 1.5x, e.g., `C4.`)
- **Rest**: `R` (e.g., `R4` = Quarter rest). Safely stops previous note and advances time.
- **Octave Shift**: `<` (Down 1 octave), `>` (Up 1 octave).
- **Tie**: `&` (Connects current note duration to the next matching note).
- **Loop Macro**: `[ MML_SEQUENCE ]N` (Repeats the sequence inside brackets `N` times. Default `N=2`).
- **Comments & Bar Lines**: You MAY use `;` for line comments and `|` as visual bar separators to improve readability. The parser safely ignores them.

2. 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 or playing at double speed, you MUST follow these timing rules strictly:

Rule A: Strict Note Count per Bar (Note Length Matching)

Check your default length `Ln` and ensure every bar has the correct total duration (e.g., 4 beats in 4/4 time).

- If `L4` (Quarter notes), a standard 4/4 bar MUST contain **4 notes** (e.g., `C D E F |`).
- If `L8` (Eighth notes), a standard 4/4 bar MUST contain **8 notes** (e.g., `C D E F G A B >C |`).
- **CRITICAL**: Do NOT write only 4 notes in an `L8` bar, or the track will play at DOUBLE SPEED and end early!

Rule B: All Channels Bar Count Synchronization

- All channels (`CH1` through `CH4`) MUST have the **exact same total number of bars (measures)**.
- If CH1 has 15 bars inside a loop `[ ... ]5`, CH2, CH3, and CH4 MUST also contain exactly 15 bars inside their loops. Do NOT miss or lose bars in secondary channels.

Rule C: Total Duration Synchronization

Calculate the total played seconds for EACH channel before generating. All channels (`CH1` to `CH4`) MUST finish at the exact same time. Never let one channel run out of notes early while another continues playing.

---

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):

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
AbsoluteBanger	I - V/VII - VIm - IIIm/V - IV V – I	Flat7Sub_m5	IIm7 - IVm(♭5)/V - I
Flat7Sub_Minor	IIm7 - IVm/V - I	SunnyAF	I/IV - Vsus4 - IIIm7 - V/VI
TwoStageChorus	IV - I - V - VIm - I V/VII - VIm V	Room335	IV△7 - IIIm7(#5) - IIm7 - IIIm7
Vibe70s	I△7 - ♭VI△7 - IVm7 - ♭VII7	ClassicWest70s	IV△7 - IIIm7 - IIm7 - I△7
ClassicWest70s_v2	IV△7 - IIIm7 - IIm7 - IIIm7	BridgeToChorus	IV - I/III - IIm7 - Vsus4 V
B_MelodyVibe	IIm7 – VIm – IVM7 – Vsus4 V	B_MelodyVibe_v2	IIm7 – VIm – IVM7 – IIIsus4 III7
TwoFive_To_4	Vm7 - I7 - IV	JPopGold	IV - V - IIIm - VIm
JPopGold_Remix	IV - V/IV - III7 - VIm	JustTwoOfUs	IVmaj7-III7-VIm7-I7
KawaiiStructure	VIm7 - ♭VIm7 - Vm7 - I7	MyRevolution	I - VIm - IV - V
Smooth_RnB	IIm9 – V7 – IM7 – IVM7 – VIIm7-5 – III7 – VIm – VI7	TwoFive_To_6m	VIIm7 - III7 - VIm
YOASOBI_Vibe	IV - V - IIIm7 - VIm7 - IIm7 - III7 - VIm - Vm7 - I7	SuperLit	VIm - ♭IIIm(♭5) - ♯IVm(♭5) - III
AnisonCanon	I - VIIm7(♭5) - III7 - VIm7 - Vm7 - I7 - IV△7	AnisonGold	I - VIIm7(♭5) - III7 - VIm7 - Vm7 - I7 - IV△7

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
	- IIIm7 - IIm7 - IV/V - V		
AnisonHit	IV - V - ♭VI - ♭VII-I	EnnuiBabe	I - VI/I# - IIm - IVm - I/V - II/♯IV - IV - V
YesterdayVibe	I - III - VIm - V	Classic1625	I - VIm - IIm - V
BestiesVibe	I - VIm - IV - V	Interstellar	IV - V - VIm - V
EpicPedal	I - ♭II/I - II/I - ♭III/I	PureEmotional	IV - V - I - III
AllRounder	VIm - V - IV - V	ChicBabe	IV△7 - IIIm7 - VIm7 - I
MarusaChic	IV△7 - III7 - VIm7 - I7	ChillGoodnight	IV△7 - I△7 - III/♯IV - IIIm/VI
TangoVibe	VIm - IIm - III - VIm	CanonClassic	I - V - VIm - IIIm - IV - I - IIm - V
CafeLofi	VIm9 - IIm7 - IIIm7	CrowdPleaser	I - IV - VIm - V
CatchyInterlude	I - V - I - V	CleanAesthetic	I/III - IV - V - VIm
ClassicalGrand	IV△7 ♯IVm(♭5) - V6 ♭VIdim - VIm ♭VIIdim - VII - III	ClassicalSeeker	IIm7(♭5) - ♭II/IV - IVm(♭5)/VII - I
GameMelodyScent	IV△7 - IIIm7 - ♭III△7 - IIm7	ClutchMoment	IV - V - I

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
FloatyChic	IIm7 - I△7 - ♭VII△7 - VIIm7(♭5) - III7	PureSimple	VIm - V - IV - V
TheBlues	I7 - IV7 - I7 - IV7 - V7 - IV7 - I7 - V7	SouthernVibe	IIm9 – V7 – IIIm7 - VIm
SuspenseRise	VIm - IV/VI - ♯IVm(♭5)/VI - VIm7	ChorusRush	VIm - V/VII - I - IIm - IIsus4 - III
ChorusHit	I - III - IV - V - VIm (or I)で帰結	ChorusPower	I - V - IV - V
PreChorusRegular	IV - ♯IVm(♭5) - V - ♭VIm(♭5)	UnstoppablePassions	I - Iaug/♯IV - IV△7 - V
CityPopVibe	VIm - I - IV - III	CinematicMood	IIm - VIm - I - V
JazzStandard_1	IIm7 - V7 - I△ - I△	JazzStandard_2	IIm7 - V7 - I△ - VI7
JazzStandard_3	I△ - VIm7 - IIm7 - V7	JazzStandard_4	I△ - I♯o7 - IIm7 - V7
JazzStandard_5	I△ - I7 - IVΔ - IVm7	JazzStandard_6	VIm7(♭5) - IIm7 - V7 - I△
JazzStandard_7	IIm7(♭5) - V7 - Im7 - Im7	JazzStandard_8	Im7 - V7 - Im7 - V7
JazzBasic	IIm7 - V7 - I△7 - VI7	SimpleIsBest	I - IV - V
SweetPrincess	I - VI/I# - IIm - IVm - I/V - II/♯IV - IV - V	StylishMood	VIm9 - IIm7 - IIIm7
SpyAgent	VIm - IV/VI - ♯IVm(♭5)/VI - VIm7	SpanishTrip	III - IV - V - IV

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
StandardTwoFive	IIm7 - V7 - I	DystopianWorld	VIm - VII - IIm - III
DramaPulse	IIm7 – I – VII – IIIsus4 III7	DramaPulse_v2	IIm7 – VIIm7-5 – IIIsus4 – III7
DorianMode	IIm - IIIm/II - IV/II - IIIm/II	DorianProof	IIIm/II - IIm
DressCodeFormal	IV△7 - Iaug/♭VII - IIm7/VI - II/♯IV	NaturalDaily	I - IIm7 - I/III - IV
SmoothCanon	I - V/VII - VIm - IIIm/V - IV - I/III - IIm - V	AiryVibe	I - IIIm/VII - VIm - IV
PartyTune	I - IV - V - IV	PirateQuest	VIm - IIm - III - VIm
HouseGroove	IIm7/V – IIIm7/VI	BackdoorTwoFive	IVm7 - ♭VII7 - I
HappyVibes	VIm7 - II7 - IV/V - I	ParallelWorld	VI - IV - IIm - III
BaroqueSequence	VIm - IIm - V - I - IV - VIIm(♭5) - III	BaroqueClassic	VIm - IIm - V - I - IV - VIIm(♭5) - III
HitChartBanger	I - IV - V - I	BillboardTop	VIm - I - V - IV
FantasyWorld	VIm - II - IV – V	FusionSub	IIm7 - IV/V - I
BlackadderSub	IIm7 - IVaug/V - I	BluesyFeel	I7 - IV7 - I7 - IV7 - V7 - IV7 - I7 - V7

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
Blues12Bar	I7 - I7 - I7 - I7 - IV7 - IV7 - I7 - I7 - V7 - I V7 - I7 – I7	DeepBassLine	I - Iaug - VIm/I - I7
PedalRise_1	I - ♭II/I - II/I - ♭III/I - III/I	PedalRise_2	I - IIm/I - IIIm/I - IV/I – Vsus4/I – V
PedalRise_3	I - V/I - IV/I - ♭VI/I - ♭VII/I	HolyTexture	VIm - V/VII - V - I - IV - VIIm7(♭5) - III7 - III7/♯VI
Pops_1	I - V - VIm - IV	Pops_2	I - IV - V - I
Pops_3	I - IV - Vsus4 - V	Pops_4	I - VIm - IIm - V
Pops_5	I - IIm - IV - V	Pops_6	I - V/VII - VIm - V
Pops_7	VIm - IIm - III - VIm	Pops_8	VIm - I - IIm - IIIm
PopPunkBanger	I - V - VIm - IV	MixolydianFlow	V - IV - I
MixolydianSpace	V - IIm - VIm - V	MysteryDrama	VIm - VIm - IV7 - III7
MysteryNovel	VIm - IV7 - III7	MellowOudou	IV△7 - V7/IV - IIIm7 - VIm7
ModernMetal	VI5add9 - IV5add9 - ♭III5add9 - I5add9	MonochromeCanon	I - VIIm7(♭5) - III7 - VIm - V
EurobeatEnergy	IV - V - VIm	ClimaxRush	VIm - III7/♯VI - IIIm/V - II/♯IV - IV - I/III - IIm7 - Vsus4 V - I
FantasyGhibli	VIm - V - IV - IIIm - IIm - VIm/I - VII7 - III7 - VIm	LoveStory	I - IV - IIIm - VIm

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
RhythmAndBlues	I△7 - ♭VI△7 - IVm7 - ♭VII7	LydianBright	IV - V/IV
RelaxChillin	I - IIm7 - I/III - IV	LoopDrive	IIm7 – IIIm7 – IIm7 – IIIm7
LoopDrive_v2	IIm7 – VIm7 – IIm7 – VIm7	RareCharacter	IV△7 - IIIm7 - ♭IIIdim7 - IIm7
RockDrive_1	I - VIm - IV - V	RockDrive_2	I - IV - I - V
RockDrive_3	I - I - ♭III - IV	RockDrive_4	I - III - IV - IVm
RockDrive_5	IIm - I - V - ♭VII	RockDrive_6	V - V - IV - I
RockDrive_7	VIm - IV - I - V	RockDrive_8	VIm - III7 - II - V7
RockVocalist	IV△7 - V7/IV - IIIm7/V - VIm7	RomanceMelody	IV - V - III7 - VIm
RomanticNight	I - IMaj7/VII - I7/♭VI I - VI7 - IIm - I#aug - IIm7/I - V	NostalgicEuro	IIm7 – V7 – IIIm7 – VIm
LiveForLove	I - VIIm7(♭5) /II - II I7 - VIm7 - V	SurpriseTwist	I - V - ♭VII - IV - ♭VI - I - II - V
IsekaiFantasy	VIm - II - IV - V	FirstStarReborn	IV△7 - IIIm7 - VIm7 - I
SciFiSpace	VIm - Im - VIm - IVm	HeroicEpic	VIm - III/♯VI - VI7/V - I Im - IV - IIIsus4 III
FlameBreath	VIm - V - IV - I	GlobalHit	I - IV - VIm - V

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
OudouClassic	IV - V - IIIm - VIm	MyFavoriteBanger	VIm - ♯IVm7(♭5) - IV△7 - IIm7 - IIIm7
WarmHalfStepRise	I - Iaug - VIm/I - I7	KachouFugetsu	IV6 - V6 - VI7sus4 - IIIsus4
GorgeousMelody	I - III/VII - VIm - V - IV - IIIm - IIm	SalvationRise	IV - V - VI
OuterSpaceJourney	VIm - ♭IIIm - VIm - IVm - VIm - ♭IIIm - ♭VIIm - IIm - ♯IVm	RevolutionPulse	VIm - IV/VI - VIm - VII7/VI - III/♯VI - IIIm(♭5)/V - II/♯IV - IIm/IV - VI/III
InterludeSpecial	VIm - IIm7 - IV - V	MarusaLoop	IV△7 - III7 - VIm7 - I7
MarusaLoop_v2	IV△7 - III7 - VIm7 - Vm7 I7	MarusaLoop_v3	IV△7 - III7 - VIm7 - II7
RareSpecies	IIIm - IV - VIm - I	HypeBuildUp	VIm - IIIm/V - IV V - I
YourTrueForm	VIm - IIm7 - IV - V	CyberFuture	VIm - V/VI - IV/VI - IIIm/VI
MoonlightShine	VIm - III/♯VI - VI7/V - IIm/IV - IV7 - III/♯VI	UnreachableSky	IV - II/♯IV - Vsus4 V - VIsus4 VI - IIsus4 II
MiragePhantom	IV△7 - VIIm7(♭5)/IV - IIIm7 - IIIdim7 - IIm7 - V7/II - I△7 - I6	Phantasmagoric	I - ♭VII - ♭VI - ♭VII
AutumnLeaves	IIm7 - V7 - I△7 - IV△7 - VIIm7(♭5) - III7 - VIm7	2AM_Crossing	Iadd9 - VIm7 - V - IV

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
DeepRegret	IIm - VIm/I - VIIm7(♭5) - III7	HeartPounding	IV - ♯IVdim - V - ♭VIdim - VIm
HighClassElegance	I△7 - I#dim7 - IIm7 - V7/II	ToughProgress	VIm - V/VII - I - IIm - IIsus4 - III
SweetSugar	IV - VIm - IIm - I	UltimateBanger	IV - V - VIm - VIm - V - IV
FinalHit	VIm - V - IV - I	RadiantShine	I - V - ♭VII - IV - ♭VI - I - II - V
CircularLoop	I-VIm-IIm-V7	StepwiseDescending	VIm - V - IV - IIIm - IIm - I
RunUpStart	IIm7 - IIIm7 - IV△7 - V	KomuroDynamite	VIm - IIm - V - I
KomuroClassic	VIm - IV - V - I	KomuroClassic_v2	VIm - IV - V - I V/VII - VIm
KomuroClassic_v3	VIm - IIm - V - I	TimelessBanger	I - V - IV - V
FieryPassion	VIm - VII/VI - VIIm(♭5)/VI - VIm - I/V - ♯IVm(♭5) - IV7(♭5) - III7	ForwardDrive	I - ♭VII - ♭VI - ♭VII
FinallyFree	VIm - III7/♯VI - IV△7 - IIm9 - Vsus4	OneWorld	Isus4 - Isus4/♭II - Isus4/♭III - Isus4/IV - Isus4/V
HypeRise	IIm7 – IIIm7 – IVM7 – V	HypeRise_v2	IIm7 – IIIm7 – IVM7 – II/♯IV – Vsus4 V

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
YouthfulDays	I - III - VIm - V	NostalgiaTrigger	VIm - IIIm - IV - V - I
MajesticGrand	Isus4 - Isus4/♭II - Isus4/♭III - Isus4/IV - Isus4/V	BeyondAtmosphere	IIm7 - IIIm7 - IVm7 - Vm7
GrandFinale	I - I/III - Isus4/IV - Isus4/V	ChaserOudou	IV△7 - ♭VII7 - IIIm7 - IIIdim7/VI
ClassicBassChromatic	VIm - ♭VIaug - I/V - ♯IVm7(♭5)	CyberGirl	VIm7 - ♭VIm7 - Vm7 - I7
GrandOpening	VIm - V/VI - IV/VI - IIm/VI	UrbanAesthetic	VIm7 - II7 - IV/V - I
OrientalMystery	IV6 - V6 - VIsus4 - IIIsus4	SpecialMarusa	VIm - I - IV - III
EnemyWithin	IV - VIm - Im - V/II - ♭VIIm/♭II - IV/I - VIIdim7 - III7/♯VI	PathosGrief	VIm - VII - IIm - III
FlyingGirl	IV - V - VI	BeautifulCanon	I - V/VII - VIm - IIIm/V - IV - I/III - IIm - V
NightParade	IIIsus4add♭9 - IIIsus4add♭9/IV - IIIsus4add♭9/II - IIIsus4add♭9	SuddenBitterSweet	IV - V - I - III
AntiOrdinary	♯IVm7(♭5) - IV△7 - IIIm7 - VIm7	FloatingVibe	I - I/III - Isus4/IV - Isus4/V
MasqueradeBall	VIm - VII/VI - VIIm(♭5)/VI - VIm - I/V - ♯IVm(♭5) - IV7(♭5) - III7	TwoFive_To_RelativeMinor	VIIm7(♭5) - III7 - VIm

Atmosphere	Progression (Degree)	Atmosphere	Progression (Degree)
HeiseiBalladWeapon	IIIm7 - VIm7/III - IIm7 - V7/II	KomeKomeLoop	IIm7 – V7 – IM7 – VIm7 – ♭VI7 – IIm7/V – V7 – I
J_PopPunk	IV - I - V - VIm	OurFuture	I - IIIm/VII - Vm7/♭VII - VI7 - IIm - I#aug - IV/I - V7/VII
AllPurposeSeasoning	IV - V - VIm - I	BittersweetLove	I - V - VIm - IV
SunlightFiltered	IV△7 - IVm△7 - IIIm7 - Vm7/VI	ElegantPassingDim	I△7 - I#dim7 - IIm7 - V7
PlayfulVibe	I - V/I - IV/I - ♭VI/I - ♭VII/I	WesternChartHit	IV - I - V - I
WesternStyle	VIm - I - V - IV	WesternPopPunk	VIm - IV - I - V
WesternStandard	VIm - IV - I - V	TritoneSub	IIm7 - ♭II7 - I
TearsFalling	IV - V - ♭VIdim - VIm	TearjerkerWeapon	IV - V - ♭VIm(♭5) - VIm
PowerOfLove	I - IIIm/VII - VIm - IV	SparklingShine	I - IVm6/I
PureBeauty	IV△7 - VIIm7(♭5)/IV - IIIm7 - IIIdim7 - IIm7 - V7/II - I△7 - I6	VioletTwoFiveCombo	I - IIIm7(♭5) - VI7 - IIm7 - VIIm7 - III7

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 Name	Scale Degrees	English Name	Scale Degrees
Major Scale / Ionian Mode	1, 2, 3, 4, 5, 6, 7	Natural Minor Scale / Aeolian Mode	1, 2, ♭3, 4, 5, ♭6, ♭7
Harmonic Minor Scale	1, 2, ♭3, 4, 5, ♭6, 7	Melodic Minor Scale	1, 2, ♭3, 4, 5, 6, 7
Dorian Mode	1, 2, ♭3, 4, 5, 6, ♭7	Phrygian Mode	1, ♭2, ♭3, 4, 5, ♭6, ♭7
Lydian Mode	1, 2, 3, ♯4, 5, 6, 7	Mixolydian Mode	1, 2, 3, 4, 5, 6, ♭7
Locrian Mode	1, ♭2, ♭3, 4, ♭5, ♭6, ♭7		

5.2 Harmonic & Melodic Minor Derived Scales

English Name	Scale Degrees	English Name	Scale Degrees
Locrian ♮6 Scale	1, ♭2, ♭3, 4, ♭5, 6, ♭7	Ionian ♯5 Scale	1, 2, 3, 4, ♯5, 6, 7
Dorian ♯4 Scale	1, 2, ♭3, ♯4, 5, 6, ♭7	Phrygian Dominant (H.M. P5 Below)	1, ♭2, 3, 4, 5, ♭6, ♭7
Lydian ♯2 Scale	1, ♯2, 3, ♯4, 5, 6, 7	Super Locrian ♭7 Scale	1, ♭2, ♭3, ♭4, ♭5, ♭6, ♭♭7
Dorian ♭2 Scale	1, ♭2, ♭3, 4, 5, 6, ♭7	Lydian ♯5 Scale / Lydian Augmented	1, 2, 3, ♯4, ♯5, 6, 7

English Name	Scale Degrees	English Name	Scale Degrees
Lydian ♭7 Scale / Acoustic Scale	1, 2, 3, ♯4, 5, 6, ♭7	Mixolydian ♭6 Scale	1, 2, 3, 4, 5, ♭6, ♭7
Locrian ♮2 Scale / Half-Diminished	1, 2, ♭3, 4, ♭5, ♭6, ♭7	Super Locrian Scale / Altered Scale	1, ♭2, ♭3, ♭4, ♭5, ♭6, ♭7

5.3 Symmetrical, Pentatonic & Special Scales

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

5.4 Japanese Traditional & Modern Scales

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, ♭7	Insenpou / Miyakobushi / Insen	1, ♭2, 4, 5, ♭7
Ryukyu Scale	1, 3, 4, 5, 7	Yona Nuki Major Scale	1, 2, 3, 5, 6
Yona Nuki Minor Scale	1, 2, ♭3, 5, ♭6	Niro Nuki Major Scale	1, ♭3, 4, 5, ♭7
Niro Nuki Minor Scale	1, 3, 4, 5, 7	Kumoi Scale	1, 2, ♭3, 5, 6
Iwato Scale	1, ♭2, 4, ♭5, ♭7	Hirajoshi Scale	1, 2, ♭3, 5, ♭6

5.5 World Ethnic Scales

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, ♭7
Jiao Mode (Chinese Scale)	1, ♭3, 4, ♭6, ♭7	Zhi Mode (Chinese Scale)	1, 2, 4, 5, 6
Yu Mode (Chinese Scale)	1, ♭3, 4, 5, ♭7	Gypsy Scale / Hungarian Gypsy	1, ♭2, 3, 4, 5, ♭6, 7
Spanish Scale / Flamenco Scale	1, ♭2, 3, 4, 5, ♭6, ♭7	Hungarian Scale / Hungarian Minor	1, 2, ♭3, ♯4, 5, ♭6, 7
Scottish Scale	1, 2, 3, 5, 6	Pelog Scale	1, ♭2, ♭3, 5, ♭6

English Name	Scale Degrees	English Name	Scale Degrees
Bilaval Thaat	1, 2, 3, 4, 5, 6, 7	Khamaj Thaat	1, 2, 3, 4, 5, 6, ♭7
Kafi Thaat	1, 2, ♭3, 4, 5, 6, ♭7	Asavari Thaat	1, 2, ♭3, 4, 5, ♭6, ♭7
Bhairav Thaat	1, ♭2, 3, 4, 5, ♭6, 7	Bhairavi Thaat	1, 2, ♭3, 4, 5, ♭6, ♭7
Kalyan Thaat	1, 2, 3, ♯4, 5, 6, 7	Marwa Thaat	1, ♭2, 3, ♯4, 5, 6, 7
Poorvi Thaat	1, ♭2, 3, ♯4, 5, ♭6, 7	Todi Thaat	1, ♭2, ♭3, ♯4, 5, ♭6, 7
Maqam Rast	1, 2, ♭3 (Quarter Tone), 4, 5, 6, ♭7 (Quarter Tone)	Maqam Nahawand	1, 2, ♭3, 4, 5, ♭6, 7
Maqam Nawa Athar	1, 2, ♭3, ♯4, 5, ♭6, 7	Algerian Scale	1, 2, ♭3, ♯4, 5, ♭6, 7
Double Harmonic / Byzantine / Arabic	1, ♭2, 3, 4, 5, ♭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**:
  - $¥text{Duration of 1 Beat (Quarter Note)} = ¥frac{60}{¥text{BPM}} ¥text{ seconds}$
  - At `T120`, a 4/4 bar (4 quarter notes) = **2.0 seconds**.
  - A 2-minute (120 seconds) song at `T120` requires **60 bars** of music.
- Use the loop macro `[ ... ]N` efficiently to build verse/chorus repetitions and achieve requested target durations accurately.

---

9. Output Format & Boilerplate Constraint

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.

Boilerplate Example (Short BGM Template)

```
; ==========================================
; pisstrak Sample BGM - "Upbeat Town" (T128)
; ==========================================
; --- CH1: Lead Melody ---
CH1: T128 O5 V14 @0 L8
CH1: C D E F G4 E4 | A4 F4 G2 | F E D C < B4 > D4 | C2. R4
CH1: C E D F G4 E4 | F4 A4 G2 | D E F C < B4 > D4 | C2. R4
; --- CH2: Accompaniment Arpeggio ---
CH2: T128 O4 V10 @24 L8
CH2: [ C E G E C E G E | F A > C < A F A > C < A | G B > D < B G B > D < B | C
E G E C4 R4 ]2
; --- CH3: Bassline ---
CH3: T128 O3 V12 @33 L4
CH3: [ C R C R | F R F R | G R G R | C R C2 ]2
```

---

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).