Dokumentasi SDK Resmi

SysX Engine v1.0

Tanggal Rilis: 14 Juli 2025

Status: Release Candidate 1

Selamat Datang di SysX!

Ini adalah panduan resmi untuk **SysX Engine**, sebuah game engine 2D JavaScript yang diciptakan untuk menghidupkan kembali "rasa" konsol fantasi retro di dalam platform SysMOGA.

Bab 1: Filosofi Engine

- **Simpel & Terbatas:** API engine sengaja dibuat terbatas untuk mendorong kreativitas, bukan membanjiri dengan fitur.
- **Juicy by Default:** Dilengkapi fitur bawaan seperti getaran layar (shake) dan sistem partikel (emit) untuk membuat game terasa hidup.
- **Modular & Profesional:** Kode engine dan game dipisah secara logis untuk kemudahan pengelolaan dan pengembangan.

Bab 2: Cara Menjalankan Dev Kit (Paling Penting!)

Karena SysX Dev Kit menggunakan file JavaScript yang terpisah-pisah (modular), Anda **tidak bisa** membukanya dengan cara klik dua kali pada file index.html. Browser modern memiliki fitur keamanan ("Satpam Browser") yang akan memblokirnya dan menyebabkan error 404 Not Found.

Untuk menjalankannya, kita perlu "resepsionis" yang disebut **Server Lokal**. Ini adalah cara kerja semua developer web profesional. Cara paling mudah adalah menggunakan **Live Server** di **Visual Studio Code**.

Langkah-Langkah (Hanya 1 Kali Setup):

1. **Install VS Code:** Jika belum punya, download gratis dari code.visualstudio.com.
2. **Install Ekstensi "Live Server":**
    - Buka VS Code.
    - Klik ikon Extensions di sebelah kiri (kotak-kotak).
    - Cari Live Server (dari Ritwick Dey).
    - Klik **Install**.
3. **Buka Folder Dev Kit:** Di VS Code, klik File > Open Folder... dan pilih folder utama Dev Kit Anda (folder yang berisi index.html).
4. **Jalankan Server:** Klik tombol **"Go Live"** di pojok kanan bawah VS Code.

Browser akan otomatis terbuka dengan alamat seperti http://127.0.0.1:5500, dan Dev Kit Anda akan berjalan **tanpa error**.

Bab 3: Alur Kerja Developer

Setelah Dev Kit berjalan, fokus Anda hanya pada 2 file:

1. **manifest.js**: Daftarkan semua asset (Opsional) (gambar/suara) Anda di sini.

2. **game.js**: Tulis semua logika game Anda di sini.

Setiap kali Anda menyimpan perubahan di game.js atau manifest.js, Live Server akan otomatis me-refresh browser Anda.

Bab 4: Referensi API Lengkap

(Gunakan API yang sudah kita definisikan di v0.9 dan v1.0)

Core & State

- SysX.state(newState)
- SysX.debug(boolean)
- SysX.watch(key, value)

Graphics

- SysX.cls(colorIndex)
- SysX.rectfill(x, y, w, h, colorIndex)
- SysX.print(text, x, y, colorIndex)
- SysX.spr(spriteIndex, x, y)
- SysX.map(tilemap, x, y)

Audio

- SysX.sfx(assetKey)
- SysX.music(assetKey, loop?, stop?)

Input

- SysX.btn(key) & SysX.btnp(key)

Effects & Physics

- SysX.collide(obj1, obj2)
- SysX.shake(intensity, durationMs)
- SysX.emit(x, y, count?, color?, life?)

Bab 5: Modul Booting dan Implementasi File .sysboot pada SysX Engine

1. Pendahuluan

Modul booting pada SysX Engine berfungsi sebagai sistem tampilan intro ketika game pertama kali dijalankan. Mulai dari simulasi BIOS hingga tampilan logo ASCII dan slogan khas engine. Fitur ini memberikan sentuhan profesional, identitas kuat, dan peluang personalisasi terhadap setiap game yang dibangun dengan SysX.

Apa Itu File .sysboot?

File .sysboot adalah file konfigurasi berbasis JSON yang memungkinkan developer mengatur tampilan dan urutan cinematic intro pada saat game booting. Jika file ini ditemukan oleh engine, maka engine akan menggantikan intro default dengan tampilan cinematic sesuai .sysboot.

Lokasi: assets/boot/intro.sysboot

2. Struktur File .sysboot

Contoh struktur dasar:

```
{
  "version": "1.0.0",
  "title": "BOOTING GALAXY RUSH",
  "steps": [
    "[SYSX BIOS v1.0.0]",
    "> Initializing galaxy core...",
    "> Mounting cartridge.sysx",
    "> Loading hyperspace engine...",
    "> Registering core...",
    "> Registering modules...",
    "> OK",
    "> SYSTEM READY."
  ],
  "logo": [
    "   _____  __      __  _____ __   __  ",
    "  / ___| \\ \   / / |_   _|  \\/  |",
    " | |      \\ \\ / /    | | | |\\/| |",
    " | |___    \\ V /     | | | |  | |",
    "  \\____|    \\_/      |_| |_|  |_|"
  ],
  "slogan": "Rush Beyond the Stars.",
  "color": 10,
  "duration": 10
}
```

Penjelasan Properti:

Properti	Tipe	Keterangan
version	String	Versi sistem / game (opsional kosmetik)
title	String	Judul boot (tidak ditampilkan secara default)
steps	Array	Baris proses boot seperti terminal BIOS
logo	Array	Logo dalam bentuk ASCII
slogan	String	Kalimat atau kutipan di akhir boot
color	Number	Warna utama teks dan logo (default: 10)
duration	Number	Waktu tampil per baris step (default: 10f)

3. Cara Kerja dan Prioritas Boot

Saat SysX.init() dipanggil, engine secara otomatis:

1. Mengecek apakah manifest.showIntro diatur false
    - Jika iya → langsung skip intro
2. Jika tidak, mencoba memuat file assets/boot/intro.sysboot
    - Jika ditemukan → akan dijalankan via drawSysBootIntro(data)
    - Jika gagal (tidak ada/file rusak) → fallback ke drawIntro() default

4. Keuntungan Penggunaan .sysboot

- Personalisasi intro tiap game

- Konsistensi identitas visual (tema, warna, pesan)
- Bisa diganti tanpa menyentuh source engine
- Fallback otomatis jika file tidak tersedia
- Tidak mengganggu struktur engine asli

5. Tips Penggunaan

- Gunakan monospace saat membuat ASCII logo untuk menjaga proporsi
- Jangan terlalu panjang baris steps, karena keterbatasan layar 320x240
- Hindari penggunaan karakter non-ASCII untuk kompatibilitas print()
- Gunakan color dari 0–15 (mengikuti palet PICO-8 bawaan engine)

6. Rencana Ekstensi Fitur (Optional Development)

Disarankan untuk pengembang lanjut atau kolaborasi open-source

- progressBar: Menampilkan animasi progress bar palsu
- bootSFX: Nama efek suara khusus untuk boot
- autoSkipAfter: Durasi sebelum intro otomatis skip
- backgroundColor: Warna background boot
- loopIntro: Jika true, intro dapat diulang sebelum lanjut

7. Kesimpulan

Fitur .sysboot membuat SysX Engine tidak hanya sebagai game engine, tapi juga memberi ruang ekspresi terhadap nuansa pembukaan tiap game. Ini adalah pondasi penting untuk membuat engine terasa seperti *konsol pribadi* yang punya gaya tersendiri.

Gunakan fitur ini sebaik-baiknya untuk membangun **identitas yang tak terlupakan** dari setiap karya digital lo.

End of Chapter - SysX Boot Module SDK

Fighting Developer !!! Selamat Membuat Game dengan Sysx Engine.

LAMPIRAN

Source Code **Sysx Engine.**

Sysx_audio.js

```
// --- /sysx_engine/sysx_audio.js ---
window.SysX_Modules = window.SysX_Modules || {};
window.SysX_Modules.audio = (() => {
    let _currentMusic = null;
    const getCoreContext = () => SysX_Modules.core.getContext();
    function sfx(key) { const { _assets } = getCoreContext(); const sound = _assets[key];
if (sound) { sound.currentTime = 0; sound.play().catch(e => {}); } }
    function music(key, loop = true, stop = false) { if (_currentMusic) {
_currentMusic.pause(); _currentMusic = null; } if (stop) return; const { _assets } =
getCoreContext(); const sound = _assets[key]; if (sound) { sound.loop = loop;
sound.currentTime = 0; sound.play().catch(e => {}); _currentMusic = sound; } }
    return { sfx, music };
})();
```

Sysx_core.js

```
// --- /sysx_engine/sysx_core.js ---
// Mengelola data inti, game loop, dan state.

// Pastikan wadah global sudah ada, lalu isi bagian 'core'
window.SysX_Modules = window.SysX_Modules || {};

window.SysX_Modules.core = (() => {
    let _ctx, _canvas, _assets;
    let _currentState = '', _debugMode = false;
    let _watchedValues = new Map();
    let _frameCount = 0, _lastFpsUpdate = 0, _currentFps = 0;
    const PICO8_PALETTE = ['#000000', '#1D2B53', '#7E2553', '#008751', '#AB5236',
'#5F574F', '#C2C3C7', '#FFF1E8', '#FF004D', '#FFA300', '#FFEC27', '#00E436', '#29ADFF',
'#83769C', '#FF77A8', '#FFCCAA'];

    function init(context) {
        _canvas = context.canvas; _ctx = context.ctx; _assets = context.assets;
        _ctx.imageSmoothingEnabled = false;
        _lastFpsUpdate = performance.now();
        if (typeof window.Game._create === 'function') window.Game._create();
```

```
        requestAnimationFrame(gameLoop);
    }

    function gameLoop(timestamp) {
        _frameCount++; if (timestamp > _lastFpsUpdate + 1000) { _currentFps =
_frameCount; _frameCount = 0; _lastFpsUpdate = timestamp; }

        // Panggil modul lain yang sudah pasti ada saat loop berjalan
        SysX_Modules.fx.updateParticles();
        SysX_Modules.fx.applyShake(_ctx);

        const updateFunc = window.Game[`_update_${_currentState}`]; if (typeof updateFunc
=== 'function') updateFunc.call(window.Game);
        const drawFunc = window.Game[`_draw_${_currentState}`]; if (typeof drawFunc ===
'function') drawFunc.call(window.Game);

        SysX_Modules.fx.drawParticles(_ctx, SysX_Modules.graphics.rectfill);
        drawDebugInfo();

        _ctx.setTransform(1, 0, 0, 1, 0, 0);
        SysX_Modules.input.updatePrevInput();
        requestAnimationFrame(gameLoop);
    }

    function drawDebugInfo() {
        if (_debugMode) {
            const info = [`FPS: ${_currentFps}`, `PART:
${SysX_Modules.fx.getParticleCount()}`];
            _watchedValues.forEach((v, k) => info.push(`${k.toUpperCase()}: ${v}`));
            let y = 0;
            for (const t of info) {
                _ctx.font = '14px "Roboto Mono"';
                SysX_Modules.graphics.rectfill(0, y, _ctx.measureText(t).width + 8, 16,
0);
                SysX_Modules.graphics.print(t, 4, y + 12, 7);
                y += 16;
            }
        }
    }

    const getContext = () => ({ _ctx, _canvas, _assets, PICO8_PALETTE });
    const debug = (isEnabled) => { _debugMode = !!isEnabled; };
    const watch = (key, value) => { if (_debugMode) _watchedValues.set(key, value); };
    const state = (newState) => {
        _currentState = newState;
        const initFunc = window.Game[`_init_${newState}`];
```

```
        if (typeof initFunc === 'function') initFunc.call(window.Game);
    };

    return { init, getContext, debug, watch, state };
})();
```

Sysx_fx.js

```
// --- /sysx_engine/sysx_fx.js ---
window.SysX_Modules = window.SysX_Modules || {};
window.SysX_Modules.fx = (() => {
    let _particles = []; let _shakeEndTime = 0, _shakeIntensity = 0;
    function collide(r1, r2) { if (!r1 || !r2) return false; return r1.x < r2.x + r2.w &&
r1.x + r1.w > r2.x && r1.y < r2.y + r2.h && r1.y + r1.h > r2.y; }
    function shake(intensity, duration) { _shakeIntensity = intensity; _shakeEndTime =
Date.now() + duration; }
    function emit(x, y, count = 10, color = 9, life = 30) { for (let i = 0; i < count;
i++) _particles.push({ x, y, vx: (Math.random() - 0.5) * 4, vy: (Math.random() - 0.5) * 4
- 1, size: Math.random() * 2 + 1, life: Math.random() * life, color }); }
    function updateParticles() { for (let i = _particles.length - 1; i >= 0; i--) { const
p = _particles[i]; p.x += p.vx; p.y += p.vy; p.vy += 0.1; p.life--; if (p.life <= 0)
_particles.splice(i, 1); } }
    function drawParticles(_ctx, rectfillFunc) { for (const p of _particles)
rectfillFunc(p.x, p.y, p.size, p.size, p.color); }
    function applyShake(_ctx) { if (Date.now() < _shakeEndTime) { const dx =
(Math.random() - 0.5) * _shakeIntensity * 2; const dy = (Math.random() - 0.5) *
_shakeIntensity * 2; _ctx.translate(dx, dy); } else { _shakeIntensity = 0; } }
    const getParticleCount = () => _particles.length;
    return { collide, shake, emit, updateParticles, drawParticles, applyShake,
getParticleCount };
})();
```

Sysx_graphics.js

```
// --- /sysx_engine/sysx_graphics.js ---
// Semua fungsi yang berhubungan dengan menggambar.

window.SysX_Modules = window.SysX_Modules || {};

window.SysX_Modules.graphics = (() => {
    const TILE_SIZE = 16;
    const SPRITESHEET_KEY = 'main_spritesheet';

    const getCoreContext = () => SysX_Modules.core.getContext();
```

```
    function cls(c) { const { _ctx, _canvas, PICO8_PALETTE } = getCoreContext();
_ctx.fillStyle = PICO8_PALETTE[c] || '#000'; _ctx.fillRect(0, 0, _canvas.width,
_canvas.height); }
    function rectfill(x, y, w, h, c) { const { _ctx, PICO8_PALETTE } = getCoreContext();
_ctx.fillStyle = PICO8_PALETTE[c] || '#FFF'; _ctx.fillRect(x, y, w, h); }
    function print(t, x, y, c) { const { _ctx, PICO8_PALETTE } = getCoreContext();
_ctx.font = '14px "Roboto Mono"'; _ctx.fillStyle = PICO8_PALETTE[c] || '#FFF';
_ctx.fillText(t, x, y); }
    function spr(idx, dx, dy) { const { _ctx, _assets } = getCoreContext(); const sheet =
_assets[SPRITESHEET_KEY]; if (!sheet || !sheet.complete) return; const cols =
Math.floor(sheet.width / TILE_SIZE); const sx = (idx % cols) * TILE_SIZE; const sy =
Math.floor(idx / cols) * TILE_SIZE; _ctx.drawImage(sheet, sx, sy, TILE_SIZE, TILE_SIZE,
dx, dy, TILE_SIZE, TILE_SIZE); }
    function map(tm, mx = 0, my = 0) { if (!tm) return; for (let r = 0; r < tm.length;
r++) for (let c = 0; c < tm[r].length; c++) if (tm[r][c] >= 0) spr(tm[r][c], mx + c *
TILE_SIZE, my + r * TILE_SIZE); }

    return { cls, rectfill, print, spr, map };
})();
```

Sysx_input.js

```
// --- /sysx_engine/sysx_input.js ---
window.SysX_Modules = window.SysX_Modules || {};
window.SysX_Modules.input = (() => {
    let _inputState = {}; let _prevInputState = {};
    function btn(k) { return _inputState[k] || false; }
    function btnp(k) { return _inputState[k] && !_prevInputState[k]; }
    function _updateInput(newState) { _inputState = newState; }
    function updatePrevInput() { _prevInputState = { ..._inputState }; }
    return { btn, btnp, _updateInput, updatePrevInput };
})();
```

Sysx_main.js

```
// --- /sysx_engine/sysx_main.js ---
// File ini merakit semua modul menjadi satu objek global `SysX`.
// Dia harus dimuat SETELAH semua modul SysX lainnya.

// Deklarasikan objek SysX global sekali saja.
var SysX = {};

// Gunakan Object.assign untuk menggabungkan semua fungsi dari setiap modul
// ke dalam satu objek SysX.
```

```
Object.assign(
    SysX,
    SysX_Modules.core,
    SysX_Modules.graphics,
    SysX_Modules.audio,
    SysX_Modules.fx,
    SysX_Modules.input
);
```

LAMPIRAN

Source Code **Contoh *Template* Penerapan Game.**

Judul : Testing Game

Author : Internal Sysx team

Game.js

```
// ================================================================
//
//          SysX Game Template
//
// Ini adalah file utama untuk game Anda.
// Tulis semua logika game Anda di dalam objek `Game` ini.
//
// ================================================================

const Game = {
    // Variabel-variabel game Anda bisa disimpan di sini
    player: {},
    score: 0,

    //-----------------------------------------------------------
--
    // _create()
    // Fungsi ini dipanggil sekali oleh engine saat game pertama
kali dimulai.
    // Gunakan untuk setup awal dan memulai state pertama.
    //-----------------------------------------------------------
--
    _create: function() {
```

```
        // Memulai game dari state 'title'
        SysX.state('title');
    },

    //============================================================
==
    // STATE: title
    // State untuk layar judul.
    //============================================================
==

    // Dipanggil sekali saat state 'title' dimulai
    _init_title: function() {
        // Hentikan musik apa pun yang mungkin berjalan
        SysX.music(null, false, true);
    },

    // Dipanggil setiap frame selama state adalah 'title'
    _update_title: function() {
        // Jika tombol A (spasi) ditekan, pindah ke state 'playing'
        if (SysX.btnp('space')) {
            SysX.state('playing');
        }
    },

    // Dipanggil setiap frame untuk menggambar selama state adalah
'title'
    _draw_title: function() {
        SysX.cls(1); // Latar biru gelap
        SysX.print("NAMA GAME ANDA", 80, 100, 7);
        SysX.print("Tekan Tombol A untuk Mulai", 40, 140, 6);
    },

    //============================================================
==
    // STATE: playing
    // State untuk permainan utama.
    //============================================================
==

    // Dipanggil sekali saat state 'playing' dimulai
    _init_playing: function() {
        // Reset semua variabel permainan
        this.player = { x: 152, y: 180, w: 16, h: 16, speed: 2,
spriteIndex: 3 };
        this.score = 0;
```

```
        // Mainkan musik latar untuk level ini
        SysX.music('level_music');
    },

    // Dipanggil setiap frame selama state adalah 'playing'
    _update_playing: function() {
        // Logika pergerakan player
        if (SysX.btn('left')) this.player.x -= this.player.speed;
        if (SysX.btn('right')) this.player.x += this.player.speed;

        // Logika menembak
        if (SysX.btnp('space')) {
            SysX.sfx('shoot_sfx');
        }

        // Toggle debug mode dengan tombol B (x)
        if (SysX.btnp('keyB')) {
            this.isDebug = !this.isDebug;
            SysX.debug(this.isDebug);
        }

        // Tampilkan info di debug overlay
        SysX.watch('score', this.score);
        SysX.watch('player_x', Math.floor(this.player.x));
    },

    // Dipanggil setiap frame untuk menggambar selama state adalah
'playing'
    _draw_playing: function() {
        SysX.cls(12); // Latar biru langit

        // Gambar player
        SysX.spr(this.player.spriteIndex, this.player.x,
this.player.y);

        // Gambar UI
        SysX.print(`SKOR: ${this.score}`, 8, 20, 7);
    }

    //============================================================
==
    // Tambahkan state lain di sini (misal: game_over, win, dll.)
    //============================================================
==
};
```

Manifest.js

```
// manifest.js
// Konfigurasi game tanpa aset eksternal.

window.manifest = {
  "title": "...",
  "version": "...",
  "main_script": "game.js",
  "assets": [] // Array aset dikosongkan
};
```

```
{
  "version": "1",
  "main_script": "game.js",
  "assets": [
    {
      "key": "main_spritesheet",
      "path": "assets/spritesheet.png"
    },
    {
      "key": "shoot_sfx",
      "path": "assets/audio/shoot.wav"
    },
    {
      "key": "level_music",
      "path": "assets/audio/music.mp3"
    }
  ]
}
```