

Dokumentasi SDK Resmi - SysMOGA™ Lite v2.0

Versi Dokumen : 2.0
Fokus : Mobile-First
Tanggal : 10 Juli 2025
Penulis : Oki Windu Tanturi

Selamat datang di dokumentasi resmi SDK SysMOGA™ Lite v2.0! Panduan ini dirancang untuk membantu pengembang menciptakan, mengemas, dan mendistribusikan game untuk platform konsol virtual SysMOGA™ Lite. Versi 2.0 menyempurnakan SDK v1.0 dengan integrasi firmware baru (SysmogaCore v4.0), panduan konversi .win ke .png, dan aturan pengembangan yang lebih jelas.

Bab 1: Visi, Filosofi, dan Pernyataan Legal

1.1. Visi dan Filosofi

SysMOGA™ Lite adalah platform konsol virtual mobile-first yang menghidupkan kembali pengalaman konsol game genggam klasik dengan estetika retro dan fleksibilitas modern. Filosofi kami meliputi:

- Pengalaman Otentik:** Menyediakan nuansa "kartrid" retro dengan rendering pixel art yang tajam, dioptimalkan untuk perangkat mobile.
- Inovasi Format:** Mengintegrasikan data game ke dalam file .png dengan metadata .win untuk pengalaman kartrid yang unik.
- Kekuatan Komunitas:** Memberdayakan pengembang indie melalui ekosistem terbuka.

1.2. Pernyataan Legal dan Lisensi



- Nama Produk:** SysMOGA™ Lite (System Mobile Gamestation)
- Pengembang Asli:** Oki Windu Tanturi

- **Lisensi Firmware:** MIT License. Kode firmware SysMOGA™ Lite dapat digunakan, dimodifikasi, dan dikomersialkan dengan syarat mencantumkan atribusi kepada pengembang asli.
 - **Lisensi Aset:** Aset visual (desain kartrid, UI), audio, dan elemen desain lainnya berada di bawah **Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)**. Pengguna harus mencantumkan atribusi dan mendistribusikan karya turunan di bawah lisensi yang sama.
 - **Pustaka Pihak Ketiga:** Menggunakan **Phaser.js** (game engine), **JSZip** (pengemasan kartrid), **TailwindCSS**, **DaisyUI** (UI), dan **Anime.js** (animasi), semuanya berlisensi MIT.
 - **Disclaimer:** SysMOGA™ Lite adalah platform independen dan tidak berafiliasi dengan Phaser, PICO-8, Nintendo, atau entitas lain.
-

Bab 2: Arsitektur Platform

SysMOGA™ Lite adalah konsol virtual berbasis web yang berjalan di browser modern, dirancang untuk mendukung game retro dengan kontrol mobile-friendly.

2.1. Komponen Firmware

Firmware SysMOGA™ Lite (didefinisikan dalam kelas SysmogaCore) terdiri dari:

- **Antarmuka Pengguna (UI):** Menyediakan tombol D-Pad, tombol A/B, menu eShop, dan jam real-time.
- **Pemuat ROM:** Memproses file .png dengan data .win tertanam untuk memuat kartrid game.
- **GameHostScene:** Objek Phaser.js yang mengelola eksekusi game, input pengguna, dan rendering.
- **Kontroler Virtual:** Mendukung input dari keyboard, sentuhan, dan gamepad fisik.

2.2. Komunikasi Firmware-ROM

Firmware berkomunikasi dengan ROM (file game.js) melalui:

1. **romData:** Objek konfigurasi yang berisi metadata dan pengaturan game (resolusi, fisika, kontrol).
2. **logicClasses:** Koleksi kelas logika game, dengan kelas utama (didefinisikan dalam scene.entry) yang menangani preload, create, dan update.

3. **AssetBlobs**: Objek URL untuk aset yang dimuat dari file .win.

2.3. Alur Eksekusi

1. Pengguna memilih file .png melalui tombol "Insert ROM" atau eShop.
 2. Firmware (SysmogaCore.unpackAndProcessRom) mengekstrak data .win dari metadata .png menggunakan JSZip.
 3. Firmware mem-parsing manifest.json dan game.js untuk memuat romData dan logicClasses.
 4. Firmware membuat instance kelas utama dari logicClasses dan menjalankan metode preload(), create(), dan update(finalInputState).
-

Bab 3: Alur Kerja Pengembangan

Setiap game untuk SysMOGA™ Lite harus mengikuti alur kerja berikut untuk memastikan kompatibilitas.

3.1. Langkah 1: Siapkan Struktur Proyek

Buat folder proyek dengan struktur berikut:

```
my-game/
└── manifest.json
└── game.js
└── assets/ (opsional)
    ├── sprites/
    ├── audio/
    └── data/
```

- **manifest.json**: File konfigurasi yang mencantumkan semua aset game.
- **game.js**: File utama yang berisi logika game dan konfigurasi ROM.
- **assets/**: Folder opsional untuk sprite, audio, atau data lainnya. Aset juga dapat dibuat secara programatik dalam game.js.

Contoh manifest.json:

```
{
  "version": "1",
  "main_script": "game.js",
  "assets": [
    { "key": "player", "path": "assets/sprites/player.png" },
```

```
        { "key": "shoot", "path": "assets/audio/shoot.wav" }
    ]
}
```

3.2. Langkah 2: Kembangkan Game

- Tulis logika game dalam game.js sesuai spesifikasi (lihat Bab 4).
- Gunakan Phaser.js untuk rendering, fisika, dan manajemen aset.
- Pastikan game mendukung input keyboard dan sentuhan (via gameConfig.mobileControls).

3.3. Langkah 3: Eksport ke Format .win

- Gunakan JSZip untuk mengemas folder proyek menjadi file .win (format kartrid SysMOGA).
- Contoh skrip untuk membuat .win:

```
import JSZip from 'jszip';
import { saveAs } from 'file-saver';
.

async function createWinFile(projectFolder) {
    const zip = new JSZip();
    zip.file('manifest.json', JSON.stringify(projectFolder.manifest));
    zip.file('game.js', projectFolder.gameJs);
    if (projectFolder.assets) {
        const assetsFolder = zip.folder('assets');
        assetsFolder.file('sprites/player.png',
            projectFolder.assets.player);
        // Tambahkan aset lain sesuai kebutuhan
    }
    const content = await zip.generateAsync({ type: 'blob' });
    saveAs(content, 'my-game.win');
}
```

3.4. Langkah 4: Konversi ke .png (Kartrid)

- File .win dikonversi menjadi file .png dengan menyisipkan data kartrid ke dalam metadata .png menggunakan alat konversi SysMOGA.
- File .png berfungsi sebagai kartrid visual di eShop, dengan data .win tertanam setelah delimiter ---SYSMOGA_ROM_DATA---.
- Contoh alur konversi (pseudocode):

```
async function convertWinToPng(winFile, thumbnailImage) {
    const png = new PNG({ width: 128, height: 128 });
    const winData = await winFile.arrayBuffer();
    const delimiter = new TextEncoder().encode('---SYSMOGA_ROM_DATA---');
    const combinedBuffer = new Blob([thumbnailImage, delimiter,
        winData], { type: 'image/png' });
}
```

- return saveAs(combinedBuffer, 'my-game.png');
- }
- Thumbnail visual harus beresolusi 128x128 piksel dan mencerminkan estetika game.

3.5. Langkah 5: Distribusi

- Unggah file .png (dengan data .win tertanam) ke eShop SysMOGA.
- Alternatif: Distribusikan file .win secara langsung melalui situs web atau platform lain.
- Pastikan metadata di romData (title, author, version) akurat untuk ditampilkan di eShop.

Bab 4: Spesifikasi Teknis game.js

File game.js adalah inti logika game dan harus mengekspor dua konstanta: romData dan logicClasses.

4.1. romData

Objek konfigurasi yang dibaca oleh firmware saat booting. Properti wajib:

Properti	Tipe	Deskripsi
metadata.title	string	Judul game, ditampilkan di menu eShop.
metadata.author	string	Nama pengembang atau studio.
metadata.version	string	Versi game (contoh: "2.0.0").
gameConfig.width	number	Lebar resolusi internal game (rekомендasi: 320).
gameConfig.height	number	Tinggi resolusi internal game (rekомендasi: 240).
gameConfig.zoom	number	Faktor zoom untuk rendering (default: 1).
gameConfig.pixelArt	boolean	true untuk rendering tajam khas pixel art.
gameConfig.backgroundColor	string	Warna latar belakang default (format hex, misalnya "#000000").
gameConfig.mobileControls	boolean	true untuk menampilkan kontroler virtual di layar.

Properti	Tipe	Deskripsi
gameConfig.physics	object	Konfigurasi fisika Phaser (contoh: { default: 'arcade', arcade: { gravity: { y: 700 } } }).
scene.entry	string	Nama kelas scene utama dalam logicClasses.

Contoh romData:

```
const romData = {
  metadata: {
    title: "Jelly Shooter",
    author: "Windu Gamestudio",
    version: "2.0.0"
  },
  gameConfig: {
    width: 320,
    height: 240,
    zoom: 2,
    pixelArt: true,
    backgroundColor: "#000000",
    mobileControls: true,
    physics: {
      default: "arcade",
      arcade: { gravity: { y: 700 }, debug: false }
    }
  },
  scene: { entry: "JellyShooterGame" }
};
```

4.2. logicClasses

Objek yang berisi kelas logika game. Kelas utama (sesuai scene.entry) harus mengimplementasikan:

- **constructor(scene):** Menerima objek scene Phaser dari firmware.
- **preload():** Memuat aset (sprite, audio, dll.), baik dari file eksternal maupun dibuat secara programatik.
- **create():** Mengatur elemen game (sprite, fisika, UI).
- **update(finalInputState):** Memperbarui logika game setiap frame berdasarkan input finalInputState.

Contoh logicClasses:

```
const logicClasses = {
  JellyShooterGame: class JellyShooterGame {
    constructor(scene) {
      this.scene = scene;
      this.preload();
      this.create();
    }
  }
};
```

```

    }
    preload() { /* Muat aset */ }
    create() { /* Atur game */ }
    update(finalInputState) { /* Perbarui logika */ }
}
};


```

Bab 5: Sistem Input Terpadu

Sistem input SysMOGA™ Lite menggabungkan input dari keyboard, sentuhan, dan gamepad ke dalam objek finalInputState yang dikirim ke metode update(finalInputState).

5.1. Struktur Objek finalInputState

Properti Tipe Deskripsi

up	object Status tombol atas (contoh: { isDown: true }).
down	object Status tombol bawah.
left	object Status tombol kiri.
right	object Status tombol kanan.
space	object Status tombol aksi (A).
keyB	object Status tombol sekunder (B).

5.2. Game Loop

- **GameHostScene.update():**

1. Mengumpulkan input dari keyboard (keyboardCursors) dan kontroler virtual (virtualKeys).
2. Menggabungkan input ke dalam objek finalInputState.
3. Memanggil update(finalInputState) pada kelas game untuk memperbarui logika.

Contoh penggunaan finalInputState:

```

update(finalInputState) {
    if (finalInputState.left.isDown) {
        this.player.setVelocityX(-160);
    } else if (finalInputState.right.isDown) {
        this.player.setVelocityX(160);
    }
    if (finalInputState.up.isDown && this.player.body.touching.down) {
        this.player.setVelocityY(-420);
    }
    if (finalInputState.space.isDown) {
        this.shootBullet();
    }
}

```

Bab 6: Aturan Pengembangan

1. Struktur Proyek:

- o Sertakan manifest.json dan game.js di root folder.
- o Folder assets/ bersifat opsional; aset dapat dibuat secara programatik dalam game.js.

2. Kompatibilitas:

- o Gunakan Phaser.js versi terbaru yang kompatibel (lihat dokumentasi resmi Phaser).
- o Targetkan resolusi rendah (contoh: 320x240) untuk estetika retro.

3. Kontrol:

- o Aktifkan gameConfig.mobileControls untuk mendukung kontrol sentuh.
- o Gunakan objek finalInputState untuk menangani input secara konsisten.

4. Ekspor:

- o Hasilkan file .win menggunakan JSZip.
- o Konversi .win ke .png dengan menyisipkan data kartrid ke metadata menggunakan alat konversi SysMOGA.

5. Debugging:

- o Aktifkan gameConfig.physics.arcade.debug untuk visualisasi fisika.
- o Uji game di browser dengan mode mobile untuk memastikan kompatibilitas sentuh.

Bab 7: Contoh Implementasi (Jelly Shooter)

Berikut adalah contoh game.js berdasarkan game *Jelly Shooter*, disesuaikan dengan firmware SysmogaCore:

```
class JellyShooterGame {  
  constructor(scene) {  
    this.scene = scene;  
    this.player = null;  
    this.gameState = 'TITLE';  
    this.preload();  
    this.create();  
  }  
  
  preload() {  
    const graphics = this.scene.make.graphics({ x: 0, y: 0, add: false });  
    graphics.fillStyle(0x29ADFF, 1).fillRect(0, 0, 16, 16);  
    graphics.generateTexture('player-texture', 16, 16);  
  }  
}
```

```
create() {
    this.scene.cameras.main.setBackgroundColor('#1D2B53');
    this.scene.add.text(160, 80, 'JELLY SHOOTER', { fontFamily: 'monospace', fontSize: '38px', fill: '#FFF0E8' }).setOrigin(0.5);
    this.scene.input.on('pointerdown', () => {
        if (this.gameState === 'TITLE') {
            this.gameState = 'PLAYING';
            this.scene.cameras.main.setBackgroundColor('#5F574F');
            this.player = this.scene.physics.add.sprite(160, 200, 'player-texture');
            this.player.setCollideWorldBounds(true);
        }
    });
}

update(finalInputState) {
    if (this.gameState !== 'PLAYING') return;
    if (this.player && this.player.active) {
        if (finalInputState.left.isDown) {
            this.player.setVelocityX(-160);
        } else if (finalInputState.right.isDown) {
            this.player.setVelocityX(160);
        } else {
            this.player.setVelocityX(0);
        }
        if (finalInputState.up.isDown && this.player.body.touching.down) {
            this.player.setVelocityY(-420);
        }
        if (finalInputState.space.isDown) {
            this.shootBullet();
        }
    }
}

shootBullet() {
    // Implementasi tembak
}

const romData = {
    metadata: { title: "Jelly Shooter", author: "Windu Gamestudio", version: "2.0.0" },
    gameConfig: {
        width: 320,
        height: 240,
        zoom: 2,
        pixelArt: true,
        backgroundColor: '#000000',
        mobileControls: true,
        physics: { default: 'arcade', arcade: { gravity: { y: 700 } } }
    },
    scene: { entry: "JellyShooterGame" }
};

const logicClasses = { JellyShooterGame };
```

Bab 8: Debugging dan Pengujian

1. Konsol Browser:

- Gunakan konsol developer (F12) untuk memeriksa kesalahan.
- Tambahkan console.log dalam game.js untuk debugging.

2. Pengujian Mobile:

- Gunakan "Device Toolbar" di browser untuk mensimulasikan perangkat mobile.
- Uji kontrol sentuh dengan gameConfig.mobileControls: true.

3. Debug Fisika:

- Aktifkan gameConfig.physics.arcade.debug: true untuk visualisasi tabrakan dan lintasan.

4. Validasi ROM:

- Pastikan file .png berisi data .win yang valid dengan delimiter ---SYSMOGA_ROM_DATA---.

Bab 9: Distribusi dan eShop

• Unggah ke eShop:

- Kirim file .png (dengan data .win tertanam) ke eShop SysMOGA.
- Pastikan metadata di romData (title, author, version) dan gambar cover di .png akurat.

• Distribusi Langsung:

- Bagikan file .win atau .png melalui situs web atau platform lain.
- Sertakan dokumentasi singkat tentang cara memuat ROM di SysMOGA™ Lite.

Bab 10: Pembaruan di SDK v2.0

- Mengintegrasikan firmware baru SysmogaCore v4.0 untuk pemrosesan ROM .png dengan data .win tertanam.
- Menyempurnakan sistem input dengan finalInputState untuk mendukung kontrol virtual.
- Mengklarifikasi folder assets/ bersifat opsional.
- Menambahkan panduan konversi .win ke .png dengan metadata.
- Menyertakan lisensi CC BY-SA 4.0 untuk aset visual dan audio.

- Menyederhanakan contoh game.js untuk kompatibilitas dengan firmware baru.
-

Lampiran: Kode Firmware Inti

Berikut adalah kode JavaScript inti dari firmware SysMOGA™ Lite (versi 4.0):

```
/**=
 * =====
 * SYSMOGA™ LITE - FIRMWARE CORE
 * =====
 * Versi: 4.0
 * File ini berisi kelas `SysmogaCore` yang menjadi otak dari
 * seluruh operasi konsol virtual, mulai dari membaca kartrid,
 * menjalankan game, hingga mengelola UI.
 * =====
 */
export default class SysmogaCore {
  constructor() {
    this.gameInstance = null;
    this.loadedROMs = new Map();
    this.DELIMITER = '---SYSMOGA_ROM_DATA---';
    this.activeScene = null;
    this.initializeUI();
  }

  initializeUI() {
    const romInput = document.getElementById('rom-input');
    const insertRomCard = document.getElementById('insert-rom-card');
    const exitGameBtn = document.getElementById('exit-game-btn');
    const clockEl = document.getElementById('clock');

    insertRomCard.addEventListener('click', () => romInput.click());

    romInput.addEventListener('change', async (event) => {
      const file = event.target.files[0];
      if (!file) return;
      try {
        const processedData = await this.unpackAndProcessRom(file);
        const romKey = file.name + '-' + file.lastModified;
        this.loadedROMs.set(romKey, processedData);
        this.addGameToMenu(romKey, processedData);
      } catch (e) {
        alert('Gagal memuat kartrid. Pastikan file .png valid. Error: ' +
          e.message);
        console.error(e);
      } finally {
        romInput.value = '';
      }
    });
  }
}
```

```

exitGameBtn.addEventListener('click', () => this.exitGame());

this.setupVirtualButton('.d-pad-up', 'up');
this.setupVirtualButton('.d-pad-down', 'down');
this.setupVirtualButton('.d-pad-left', 'left');
this.setupVirtualButton('.d-pad-right', 'right');
this.setupVirtualButton('.action-btn-a', 'space');
this.setupVirtualButton('.action-btn-b', 'keyB');

setInterval(() => {
  clockEl.textContent = new Date().toLocaleTimeString('en-US', {
  hour: 'numeric', minute: '2-digit', hour12: true });
}, 1000);
}

setupVirtualButton(selector, key) {
  const element = document.querySelector(selector);
  if (!element) return;
  const press = (e) => {
    e.preventDefault();
    if (this.activeScene) this.activeScene.virtualKeys[key] = true;
  };
  const release = (e) => {
    e.preventDefault();
    if (this.activeScene) this.activeScene.virtualKeys[key] = false;
  };
  element.addEventListener('mousedown', press);
  element.addEventListener('mouseup', release);
  element.addEventListener('mouseleave', release);
  element.addEventListener('touchstart', press, { passive: false });
  element.addEventListener('touchend', release);
}

async unpackAndProcessRom(pngBlob) {
  const buffer = await pngBlob.arrayBuffer();
  const delimiterBytes = new TextEncoder().encode(this.DELIMITER);
  let delimiterIndex = -1;
  for (let i = 0; i < buffer.byteLength - delimiterBytes.byteLength; i++) {
    let found = true;
    for (let j = 0; j < delimiterBytes.byteLength; j++) {
      if (new Uint8Array(buffer, i + j, 1)[0] !== delimiterBytes[j]) {
        found = false;
        break;
      }
    }
    if (found) { delimiterIndex = i; break; }
  }
  if (delimiterIndex === -1) throw new Error("Invalid cartridge: ROM data delimiter not found.");
}

const coverBuffer = buffer.slice(0, delimiterIndex);
const romBuffer = buffer.slice(delimiterIndex +
delimiterBytes.byteLength);

```

```
    const coverImageURL = URL.createObjectURL(new Blob([coverBuffer], {  
type: 'image/png' }));  
  
    const zip = await JSZip.loadAsync(romBuffer);  
    const manifestStr = await  
zip.file("manifest.json").async("string");  
    if (!manifestStr) throw new Error("manifest.json not found in  
ROM.");  
  
    const manifest = JSON.parse(manifestStr);  
    const gameScriptStr = await zip.file(manifest.main_script ||  
"game.js").async("string");  
    if (!gameScriptStr) throw new Error("Game script not found in  
ROM.");  
  
    const gameCode = new Function(` ${gameScriptStr}; return { romData,  
logicClasses }`);  
    const { romData, logicClasses } = gameCode();  
    if (!romData || !logicClasses) throw new Error("romData or  
logicClasses not found in game script.");  
  
    const assetBlobs = {};  
    if (manifest.assets && manifest.assets.length > 0) {  
        for (const asset of manifest.assets) {  
            const file = zip.file(asset.path);  
            if (file) assetBlobs[asset.key] = URL.createObjectURL(await  
file.async("blob"));  
        }  
    }  
    return { romData, logicClasses, assetBlobs, coverImageURL };  
}  
  
bootGame(romKey) {  
    const { romData, logicClasses, assetBlobs } =  
this.loadedROMs.get(romKey);  
    const self = this;  
  
    const virtualControls = document.getElementById('virtual-  
controls');  
    if (romData.gameConfig.mobileControls) {  
        virtualControls.style.display = 'flex';  
    } else {  
        virtualControls.style.display = 'none';  
    }  
  
    class GameHostScene extends Phaser.Scene {  
        constructor() {  
            super('GameHostScene');  
            this.gameLogic = null;  
            this.virtualKeys = { up: false, down: false, left: false,  
right: false, space: false, keyB: false };  
            self.activeScene = this;  
        }  
        preload() { for (const key in assetBlobs) this.load.image(key,  
assetBlobs[key]); }  
        create() {
```

```
        this.keyboardCursors = this.input.keyboard.createCursorKeys();
        this.keyboardCursors.keyB =
this.input.keyboard.addKey(Phaser.Input.Keyboard.KeyCodes.B);
        this.gameLogic = new logicClasses[romData.scene.entry](this);
    }
    update() {
        const finalInputState = {
            up: { isDown: this.keyboardCursors.up.isDown || this.virtualKeys.up },
            down: { isDown: this.keyboardCursors.down.isDown || this.virtualKeys.down },
            left: { isDown: this.keyboardCursors.left.isDown || this.virtualKeys.left },
            right: { isDown: this.keyboardCursors.right.isDown || this.virtualKeys.right },
            space: { isDown: this.keyboardCursors.space.isDown || this.virtualKeys.space },
            keyB: { isDown: this.keyboardCursors.keyB.isDown || this.virtualKeys.keyB }
        };
        if (this.gameLogic?.update) {
            this.gameLogic.update(finalInputState);
        }
    }
}

if (this.gameInstance) this.gameInstance.destroy(true);
const phaserConfig = {
    ...romData.gameConfig,
    type: Phaser.AUTO,
    parent: 'sysmoga-screen-container',
    scale: { mode: Phaser.Scale.FIT, autoCenter: Phaser.Scale.CENTER_BOTH },
    scene: [GameHostScene]
};
this.gameInstance = new Phaser.Game(phaserConfig);
}

addGameToMenu(romKey, processedData) {
    const { romData, coverImageURL } = processedData;
    const card = document.createElement('div');
    card.className = 'game-card card w-40 h-56 bg-base-200 shadow-xl cursor-pointer overflow-hidden';
    card.dataset.romKey = romKey;
    card.innerHTML = `<figure></figure>`;
    document.getElementById('game-list').prepend(card);

    anime({ targets: card, scale: [0.5, 1], opacity: [0, 1], duration: 500, easing: 'easeOutExpo' });

    card.addEventListener('click', () => {
        document.querySelectorAll('.game-card').forEach(c => c.classList.remove('active', 'border-4', 'border-primary'));
        card.classList.add('active', 'border-4', 'border-primary');
    });
}
```

```
        this.showGameInfo(romKey);
    });
}

showGameInfo(romKey) {
    const { romData, coverImageURL } = this.loadedROMs.get(romKey);
    const safeId = romKey.replace(/[^a-zA-Z0-9_-]/g, '_');
    const buttonId = `play-game-btn-${safeId}`;

    const infoContentHTML = `
        
        <h3 class="font-pixel text-xl text-white truncate">${romData.metadata.title}</h3>
        <p class="text-sm text-gray-400">oleh ${romData.metadata.author}</p>
        <button id="${buttonId}" class="btn btn-primary btn-md mt-4 font-pixel text-sm">Play</button>
    `;

    const mobileDrawer = document.getElementById('game-info-bar-mobile');
    mobileDrawer.innerHTML = infoContentHTML;
    mobileDrawer.querySelector(`#${buttonId}`).onclick = () =>
        this.launchGame(romKey);
    document.getElementById('info-drawer-toggle').checked = true;
}

launchGame(romKey) {
    document.getElementById('info-drawer-toggle').checked = false;
    const homeScreen = document.getElementById('home-screen');
    const bootScreen = document.getElementById('boot-screen');
    const gameScreen = document.getElementById('game-screen');
    const bootLogo = document.getElementById('boot-logo');
    const bootText = document.getElementById('boot-text');

    const logoText = "SysMOGA™ Lite";
    bootLogo.innerHTML = logoText.split('').map(letter => `<span class="letter">${letter}</span>`).join('');
    bootText.textContent = '';

    const self = this;

    anime.timeline({
        easing: 'easeOutExpo',
    })
    .add({
        targets: homeScreen,
        opacity: 0,
        duration: 400,
        complete: () => homeScreen.classList.add('hidden')
    })
    .add({
        targets: bootScreen,
        opacity: [0, 1],
        begin: () => bootScreen.classList.remove('hidden'),
    })
}
```

```
        duration: 100
    })
    .add({
        targets: '#boot-logo .letter',
        opacity: [0, 1],
        translateX: [40, 0],
        translateZ: 0,
        scaleX: [0.3, 1],
        delay: (el, i) => 70 * i,
        duration: 800,
    }, '--=200')
    .add({
        targets: bootText,
        begin: () => {
            let text = "BOOTING CARTRIDGE...";
            let i = 0;
            bootText.textContent = "";
            const typingInterval = setInterval(() => {
                if (i < text.length) {
                    bootText.textContent += text.charAt(i);
                    i++;
                } else {
                    clearInterval(typingInterval);
                }
            }, 100);
        }
    }, '--=400')
    .add({
        targets: bootScreen,
        opacity: 0,
        duration: 500,
        delay: 1500,
        complete: () => bootScreen.classList.add('hidden')
    })
    .add({
        targets: gameScreen,
        opacity: [0, 1],
        begin: () => {
            gameScreen.classList.remove('hidden');
            gameScreen.classList.add('flex');
            self.bootGame(romKey);
        },
        duration: 500
    }, '--=500');
}

exitGame() {
    const homeScreen = document.getElementById('home-screen');
    const gameScreen = document.getElementById('game-screen');
    if (this.gameInstance) { this.gameInstance.destroy(true);
this.gameInstance = null; }
    anime.timeline()
        .add({ targets: gameScreen, opacity: 0, duration: 400, easing:
'easeInExpo', complete: () => gameScreen.classList.add('hidden') })
}
```

```
    .add({ targets: homeScreen, opacity: [0, 1], begin: () =>
  homeScreen.classList.remove('hidden'), duration: 400, easing:
  'easeOutExpo' });
}
}
```

Bab 11: Pembaruan di SDK v2.0

10.1 Lisensi Kode Firmware

Kode sumber dari firmware SysMOGA™ Lite ini dirilis di bawah lisensi terbuka **MIT License**.

Lisensi ini memungkinkan siapa pun untuk menggunakan, menyalin, mengubah, dan mendistribusikan ulang kode, baik untuk penggunaan pribadi maupun komersial – dengan syarat mencantumkan kredit pembuat asli.

© 2025 Oki Windu Tanturi

Diberikan izin untuk menggunakan perangkat lunak ini berdasarkan ketentuan berikut:

MIT License

MIT License

Copyright (c) 2025 Oki Windu Tanturi

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



10.2 Hak Cipta Dokumen, Lisensi Aset Visual & Branding

You are free to:

1. **Share** – copy and redistribute the material in any medium or format for any purpose, even commercially.
2. **Adapt** – remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:



Attribution – You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.



ShareAlike – If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.

No additional restrictions – You may not apply legal terms or [technological measures](#) that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable [exception or limitation](#).

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as [publicity, privacy, or moral rights](#) may limit how you use the material.



Seluruh konten dokumentasi SDK, desain visual, layout, dan elemen UI yang bukan bagian dari kode program dilisensikan dengan lisensi **Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)**.

Materi branding seperti logo, tampilan UI, ikon, dan desain katalog SysMOGA™ (kecuali konten pihak ketiga) dilisensikan di bawah **Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)**.

Anda bebas untuk:

Menyalin dan menyebarluaskan ulang desain dalam media atau format apa pun.

Mengadaptasi, memodifikasi, bahkan menggabungkan dengan karya lain.

Asalkan Anda memberikan kredit kepada pembuat asli dan mendistribusikan turunan dengan lisensi serupa.

Selengkapnya Baca CC BY-SA 4.0 →

10.3 Catatan Tambahan

- Semua file game .win, .png, atau cartridge dari pihak ketiga **bukan bagian dari lisensi ini**, kecuali disebutkan lain.
- SysMOGA™ bukan emulator ROM dari konsol tertentu, tetapi platform distribusi **game orisinal buatan komunitas**.
- SDK ini bersifat open development – kontribusi dan forking dipersilakan dengan tetap menjaga integritas dan atribusi lisensi.



