



# MALFEITOR: O Vilão que Sobreviveu

## SUPPLEMENT BIBLE v2.0 — Stack Web + Demo

Complemento ao MasterBible v4.0 | HTML + CSS + JavaScript | 2025-2026

◆ Este documento resolve lacunas específicas para a demo web (Ato I, Dias 1-7): código completo dos sistemas críticos, física dos personagens em Canvas, JSONs de diálogo prontos para uso, testes, e guia de implementação por ordem de sessão.

## ÍNDICE

PARTE A — Física dos Personagens no Canvas

PARTE B — Implementações de Código: Sistemas Críticos

- B.1 FlagSystem.js — Código Completo
- B.2 TrustSystem.js — Código Completo
- B.3 MascaraSystem.js — Código Completo
- B.4 DialogueSystem.js + DialogueBox.js
- B.5 TimeSystem.js + EconomySystem.js
- B.6 SceneManager.js
- B.7 Renderer.js + Player.js

PARTE C — Dados JSON Completos para a Demo

- C.1 act1\_week1.json — todos os nós de diálogo Dias 1-7
- C.2 gareth.json + lyra.json + aldric.json

PARTE D — Sessão de Implementação: Ordem e Checkpoints

PARTE E — Testes Manuais de Smoke Test (sem framework)

PARTE F — Perfil Expandido: Aldric e Stenn para Demo

PARTE G — Rotinas de NPC e Presença no Mundo

# PARTE A — Física dos Personagens no Canvas

◆ 'Física' aqui significa: como os personagens existem no espaço do Canvas — dimensões, hitboxes, velocidades, colisão, schedules e regras de presença. Godot tinha CharacterBody2D; na web, implementamos tudo manualmente.

## A.1 — Dimensões e Hitboxes

Personagem	Sprite px	Exibição 3x	Hitbox física (pés)	Hitbox interação	Velocidade (px/s world)
Malfeitor	24×32	72×96	12×10	20×28	56 (nunca corre — design narrativo)
Lyra	18×26	54×78	10×8	16×22	72 (mais rápida — compensação)
Gareth	22×32	66×96	12×10	20×28	44 (patrulha lenta)
Aldric	20×30	60×90	10×10	18×26	38 (deliberado, pausado)
Davi	20×30	60×90	10×10	18×26	48
NPC anônimo	16×24	48×72	8×8	14×20	52

● CRÍTICO: Malfeitor nunca corre. Pressionar shift não aumenta velocidade. Isso é design narrativo: ele está ferido, humilhado, sem razão para pressa. Máximo: +5% ao segurar direcional por > 2s.

## A.2 — Implementação de Colisão no Canvas

```
// actors/Player.js - movimento com colisão AABB
class Player {
    constructor() {
        this.x = 80; this.y = 90;
        this.vx = 0; this.vy = 0;
        this.speed = 56;
        this.hitbox = { w: 12, h: 10, ox: 6, oy: 22 }; // offset da base do sprite
    }

    update(dt, colliders) {
        const dx = InputSystem.axis('x') * this.speed * dt;
        const dy = InputSystem.axis('y') * this.speed * dt;
        this.x = this._resolveX(this.x + dx, colliders);
        this.y = this._resolveY(this.y + dy, colliders);
        this._updateAnimation(dx, dy);
    }

    _resolveX(nx, cols) {
```

```

        const hb = { x: nx + this.hitbox.ox, y: this.y + this.hitbox.oy,
                    w: this.hitbox.w,           h: this.hitbox.h };
        for (const c of cols)
            if (this._overlaps(hb, c)) return this.x; // rejeita movimento
        return nx;
    }
    _resolveY(ny, cols) {
        const hb = { x: this.x + this.hitbox.ox, y: ny + this.hitbox.oy,
                    w: this.hitbox.w,           h: this.hitbox.h };
        for (const c of cols)
            if (this._overlaps(hb, c)) return this.y;
        return ny;
    }
    _overlaps(a, b) {
        return a.x < b.x+b.w && a.x+a.w > b.x &&
               a.y < b.y+b.h && a.y+a.h > b.y;
    }
}

```

### A.3 — Z-Ordering (quem fica na frente)

```

// renderer.js - ordenação por posição Y
function drawScene(entities) {
    // Ordenar por Y - entidade mais ao sul aparece na frente
    const sorted = [...entities].sort((a, b) => {
        // Exceção: Berthilda sempre na frente (autoridade implícita)
        if (a.id === 'berthilda') return 1;
        if (b.id === 'berthilda') return -1;
        return a.y - b.y;
    });
    for (const e of sorted) e.draw(ctx);
}

```

### A.4 — Zonas de Interação e Diálogo

```

// NPC.js - zona de interação como círculo
class NPC {
    constructor(cfg) {
        this.id = cfg.id;
        this.x = cfg.x; this.y = cfg.y;
        this.interactRadius = 18; // pixels world-space
    }

    isPlayerNear(player) {
        const dx = player.x - this.x;
        const dy = player.y - this.y;
        return Math.sqrt(dx*dx + dy*dy) < this.interactRadius;
    }

    canInteract() {

```

```

    // Berthilda bloqueada no Ato I e II
    if (this.id === 'berthilda' && GameState.act < 3) return false;
    // NPC com arco quebrado: retorna fala de rejeição
    if (TrustSystem.getArcState(this.id) === 'broken') return 'rejected';
    return true;
}
}

```

## A.5 — Schedules de NPC

NPC	Manhã (6-12h)	Tarde (12-18h)	Noite (18-23h)	Domingo especial
Gareth	Taverna — cozinha	Taverna — salão	Taverna (Sáb: até meia-noite)	Taverna — ritmo lento
Lyra	Casa / percurso	Praça central	Casa — nunca sai	Casa
Aldric	Igreja (Qua: missa)	Igreja ou caminhada	Igreja — não recebe	Igreja
Berthilda	Casa	Casa	Casa	Jardim 6-10h — único horário acessível
Davi	Jardim/campo	Jardim/estúdio	Casa	Jardim

```

// TimeSystem.js - getNPCLocation(npcId)
getNPCLocation(npcId) {
  const schedule = NPC_SCHEDULES[npcId];
  const h = this.hour;
  const dow = this.dayOfWeek;
  for (const slot of schedule) {
    if (h >= slot.start && h < slot.end) {
      if (slot.days && !slot.days.includes(dow)) continue;
      return slot.location;
    }
  }
  return NPC_HOME[npcId];
}

```

# PARTE B — Implementações de Código: Sistemas Críticos

---

## B.1 — FlagSystem.js — Código Completo

🌐 WEB: Singleton exportado. Todos os sistemas importam de FlagSystem.js. Zero dependência circular — FlagSystem não importa nada de outros sistemas.

```
// src/systems/FlagSystem.js
const _flags = { /* objeto com todas as flags do MasterBible $1.6 */ };
const _auditLog = [];
const PERMANENT = new Set([
  'FLAG_NPC_LYRA_ARC_BROKEN', 'FLAG_VILLAGE_HOSTILE',
  'FLAG_PLAYER_POISON_USED', 'FLAG_ENDING_RELAPSE_LOCKED_IN'
]);

export function setFlag(id, value, source = '') {
  if (!(id in _flags)) { console.error(`Flag desconhecida: ${id}`); return; }
  if (PERMANENT.has(id) && _flags[id] === true && value === false) return;
  const old = _flags[id];
  if (old === value) return;
  _flags[id] = value;
  _auditLog.push({ day: _getDay(), flag: id, old, value, source, ts: Date.now() });
  _handleCritical(id, value);
  _recalcEnding();
}

export const getFlag = id => _flags[id];

function _handleCritical(id, value) {
  if (!value) return;
  if (id === 'FLAG_NPC_LYRA_ARC_BROKEN') {
    _flags['FLAG_ENDING_SCORE'] = Math.min(_flags['FLAG_ENDING_SCORE'], 30);
    console.warn('🔴 LYRA_ARC_BROKEN ativada - Redenção impossível');
  }
  if (id === 'FLAG_VILLAGE_HOSTILE') {
    _flags['FLAG_ENDING_REDEMPTION_AVAILABLE'] = false;
  }
}

function _recalcEnding() {
  let score = 0;
  if (_flags.FLAG_NPC_LYRA_ARC_COMPLETE) score += 30;
  if (_flags.FLAG_NPC_BERTHILDA_APPROVED) score += 30;
  if (_flags.FLAG_NPC_ALDRIC_FULL_REVEAL) score += 15;
  if (!_flags.FLAG_PLAYER_MAGIC_USED_COUNT) score += 15;
  if (_flags.FLAG_WORLD_STENN_CONFRONTED_PUBLIC) score += 10;
  if (_flags.FLAG_NPC_LYRA_ARC_BROKEN) score -= 30;
}
```

```

        if (_flags.FLAG_PLAYER_MAGIC_USED_COUNT >= 3) score -= 20;
        if (_flags.FLAG_PLAYER_POISON_USED) score -= 15;
        if (_flags.FLAG_VILLAGE_HOSTILE) score -= 25;
        _flags.FLAG_ENDING_SCORE = score;
        _flags.FLAG_ENDING_REDEMPTION_AVAILABLE = score >= 85;
        _flags.FLAG_ENDING_IMPASSE_AVAILABLE = score >= 50 && score < 85;
    }

    export function serialize() { return { flags: {..._flags}, log: [..._auditLog] }; }
    export function deserialize(data) {
        Object.assign(_flags, data.flags);
        _auditLog.push(...(data.log || []));
    }
}

```

## B.2 — TrustSystem.js — Código Completo

```

// src/systems/TrustSystem.js
import { getFlag, setFlag } from './FlagSystem.js';

const ARC = { HOSTILE:'hostile', NEUTRAL:'neutral', WARMING:'warming',
             TRUST:'trust', BROKEN:'broken', LOCKED:'locked' };

// Perfilis carregados do JSON e mutados em runtime
let _profiles = {};

export async function init() {
    const npcs =
['gareth','lyra','aldric','davi','serena','nora','stenn','berthilda'];
    for (const id of npcs) {
        const r = await fetch(`data/npcs/${id}.json`);
        _profiles[id] = await r.json();
    }
    _profiles.berthilda.arc_state = ARC.LOCKED;
}

export function applyDelta(npcId, deltas, contextScore = 0) {
    const p = _profiles[npcId];
    if (!p) return;
    if (p.arc_state === ARC.LOCKED) return;
    if (npcId === 'lyra' && getFlag('FLAG_NPC_LYRA_ARC_BROKEN')) {
        // Após arco quebrado, só penalidades passam
        Object.keys(deltas).forEach(k => { if (deltas[k] > 0) delete deltas[k]; });
    }
    const clamp = (v, min, max) => Math.max(min, Math.min(max, v));
    const ranges = { familiaridade:[-10,20], lealdade:[-10,15], respeito:[-10,15],
                    simpatia:[-10,15], medo:[0,15], curiosidade:[0,10] };
    for (const [dim, delta] of Object.entries(deltas)) {
        // Medo >= 10 bloqueia ganhos de simpatia
    }
}

```

```

        if (dim === 'simpatia' && p.trust.medo >=
p.thresholds.medo_bloqueia_simpatia) continue;
        const finalDelta = Math.round(delta * (1 + contextScore));
        const [mn, mx] = ranges[dim];
        p.trust[dim] = clamp(p.trust[dim] + finalDelta, mn, mx);
    }
    _updateArcState(npcId);
    // Aldric: familiaridade mínima garantida de 3
    if (npcId === 'aldric') p.trust.familiaridade = Math.max(3,
p.trust.familiaridade);
}

export function recordLie(npcId) {
    const p = _profiles[npcId];
    if (!p) return;
    p.lies_detected++;
    if (npcId === 'lyra') {
        setFlag('FLAG_NPC_LYRA_ARC_BROKEN', true, 'lie_detected');
        p.arc_state = ARC.BROKEN;
    } else {
        applyDelta(npcId, { lealdade: -3 });
    }
}

function _updateArcState(npcId) {
    const p = _profiles[npcId];
    if ([ARC.BROKEN, ARC.LOCKED].includes(p.arc_state)) return;
    const t = p.trust;
    const thresh = p.thresholds;
    let next = p.arc_state;
    if (t.lealdade <= thresh.ruptura_irreversivel) next = ARC.BROKEN;
    else if (t.familiaridade >= 8 && t.ssimpatia >= 6 && t.lealdade >= 6) next =
ARC.TRUST;
    else if (t.familiaridade >= 4 && t.ssimpatia >= 2) next = ARC.WARMING;
    else if (t.familiaridade >= 2 || t.medo < 3) next = ARC.NEUTRAL;
    if (next !== p.arc_state) {
        p.arc_state = next;
        EventBus.emit('arc_changed', { npcId, arc: next });
    }
}

export function tickDayEnd(currentDay) {
    for (const [id, p] of Object.entries(_profiles)) {
        if (p.arc_state === ARC.LOCKED) continue;
        const daysAbsent = currentDay - p.last_interaction_day;
        if (daysAbsent >= 2) applyDelta(id, { familiaridade: -1 });
    }
}

export function tickWeekEnd() {
    const magicUsed      = getFlag('FLAG_PLAYER_MAGIC_USED_THIS_WEEK');
    const authorityUsed = getFlag('FLAG_PLAYER_AUTHORITY_USED_THIS_WEEK');
}

```

```

    if (!magicUsed && !authorityUsed)
      for (const id of Object.keys(_profiles))
        applyDelta(id, { medo: -1 });
    setFlag('FLAG_PLAYER_MAGIC_USED_THIS_WEEK', false, 'week_reset');
    setFlag('FLAG_PLAYER_AUTHORITY_USED_THIS_WEEK', false, 'week_reset');
  }

export const getTrust = id => _profiles[id]?.trust;
export const getArcState = id => _profiles[id]?.arc_state ?? ARC.NEUTRAL;
export function serialize() { return JSON.parse(JSON.stringify(_profiles)); }
export function deserialize(data) { _profiles = data; }

```

## B.3 — MascaraSystem.js — Código Completo

```

// src/systems/MascaraSystem.js
import * as TrustSystem from './TrustSystem.js';
import { getFlag, setFlag } from './FlagSystem.js';
import { ASPECT_DELTA_TABLE } from '../data/AspectTable.js';

export function processChoice(npcId, aspect, nodeId, contextScore = 0) {
  let deltas = ASPECT_DELTA_TABLE[aspect]?.[npcId] ?? {};
  deltas = { ...deltas }; // nunca mutar a tabela

  // Regra CULPA: 2ª vez → delta ÷ 2
  if (aspect === 'CULPA') {
    const p = TrustSystem.getProfile(npcId);
    if (p?.culpa_used) {
      for (const k of Object.keys(deltas)) deltas[k] = Math.floor(deltas[k] / 2);
    } else if (p) {
      p.culpa_used = true;
    }
  }

  // Regra AUTORIDADE: semanalmente trackada
  if (aspect === 'AUTORIDADE') setFlag('FLAG_PLAYER_AUTHORITY_USED_THIS_WEEK', true, nodeId);

  // Regra MAGIA: contador
  if (aspect === 'MAGIA_LATENTE') {
    const count = getFlag('FLAG_PLAYER_MAGIC_USED_COUNT') + 1;
    setFlag('FLAG_PLAYER_MAGIC_USED_COUNT', count, nodeId);
    setFlag('FLAG_PLAYER_MAGIC_USED_THIS_WEEK', true, nodeId);
  }

  TrustSystem.applyDelta(npcId, deltas, contextScore);
}

export function isOptionVisible(option, npcId) {
  for (const f of option.hidden_when ?? []) if (getFlag(f)) return false;
  for (const f of option.visible_when ?? []) if (!getFlag(f)) return false;
}

```

```

        return true;
    }

export function isCriticalAspect(aspect) {
    return ['AUTORIDADE', 'MANIPULAÇÃO'].includes(aspect);
}

```

## B.4 — DialogueSystem.js + DialogueBox.js

```

// src/systems/DialogueSystem.js
// Carrega JSONs de diálogo e provê acesso por node_id

const _db = new Map();

export async function loadDialogues(week) {
    const r = await fetch(`data/dialogues/act1_week${week}.json`);
    const nodes = await r.json();
    for (const node of nodes) _db.set(node.node_id, node);
}

export const getNode = id => _db.get(id) ?? null;

export function findTriggerable(scene, day, hour) {
    const candidates = [];
    for (const [id, node] of _db) {
        const t = node.trigger;
        if (t.type !== 'auto') continue;
        if (t.scene && t.scene !== scene) continue;
        if (t.day_range && (day < t.day_range[0] || day > t.day_range[1])) continue;
        if (getFlag(`FLAG_DIALOGUE_${id}_SEEN`)) continue; // já viu
        candidates.push({ id, priority: t.priority ?? 0 });
    }
    candidates.sort((a,b) => b.priority - a.priority);
    return candidates[0]??.id ?? null;
}

```

```

// src/ui/DialogueBox.js
// Controla o HTML #dialogue-box
import * as DialogueSystem from '../systems/DialogueSystem.js';
import * as MascaraSystem from '../systems/MascaraSystem.js';
import { setFlag } from '../systems/FlagSystem.js';
import { applyDelta } from '../systems/TrustSystem.js';

const box      = document.getElementById('dialogue-box');
const nameEl   = document.getElementById('npc-name');
const textEl   = document.getElementById('dialogue-text');
const optsEl   = document.getElementById('options-container');
const NPC_THEMES = {
    gareth: { border: '#B45309', name: '#F59E0B' },

```

```

lyra:      { border: '#1D4ED8', name: '#60A5FA' },
aldric:    { border: '#374151', name: '#9CA3AF' },
davi:      { border: '#065F46', name: '#34D399' },
berthilda: { border: '#1F2937', name: '#6B7280' },
};

let _resolve = null;

export function startDialogue(nodeId) {
  return new Promise(res => {
    _resolve = res;
    _loadNode(nodeId);
  });
}

async function _loadNode(nodeId) {
  const node = DialogueSystem.getNode(nodeId);
  if (!node) { _endDialogue(); return; }
  setFlag(`FLAG_DIALOGUE_${nodeId}_SEEN`, true, 'dialogue');

  // Aplicar tema do NPC
  const theme = NPC_THEMES[node.speaker] ?? {};
  box.style.borderColor = theme.border ?? '#555';
  nameEl.style.color = theme.name ?? '#fff';
  nameEl.textContent = node.speaker.toUpperCase();

  box.classList.remove('hidden');
  optsEl.innerHTML = '';

  // Typewriter
  await _typewriter(node.text);

  // Mostrar opções ou fim
  const opts = (node.aspect_options ?? [])
    .filter(o => MascaraSystem.isOptionVisible(o, node.speaker));

  if (opts.length === 0) {
    // Aplicar no_choice_result (SILENCIO automático)
    const ncr = node.no_choice_result;
    if (ncr) {
      await new Promise(r => setTimeout(r, 800));
      _applyEffects(node.speaker, ncr.effects);
      if (ncr.effects?.next_node) { _loadNode(ncr.effects.next_node); return; }
    }
    _endDialogue();
  }
}

// Renderizar opções com delay escalonado
opts.forEach((opt, i) => {
  const btn = document.createElement('button');
  btn.className = `aspect-option aspect-${opt.aspect.toLowerCase()}`;
}

```

```

        btn.innerHTML =
          `<span class='aspect-tag'>${opt.aspect}</span>` +
          `<span class='option-text'>${opt.text}</span>` +
          (MascaraSystem.isCriticalAspect(opt.aspect) ? '<span class=warn>⚠</span>' :
        '');
        btn.style.animationDelay = `${i * 80}ms`;
        btn.addEventListener('click', () => _selectOption(node.speaker, opt));
        // Atalho numérico 1-4
        optsEl.appendChild(btn);
      });
    }

function _applyEffects(npcId, effects) {
  if (!effects) return;
  MascaraSystem.processChoice(npcId, effects.aspect ?? 'SILENCIO', '');
  for (const [npc, deltas] of Object.entries(effects.trust_deltas ?? {}))
    applyDelta(npc, deltas);
  for (const f of effects.flags_set ?? [])
    setFlag(f, true, 'dialogue_effect');
}

async function _selectOption(npcId, opt) {
  optsEl.innerHTML = '';
  MascaraSystem.processChoice(npcId, opt.aspect, opt.id);
  _applyEffects(npcId, opt.effects);
  await new Promise(r => setTimeout(r, 300));
  if (opt.effects?.next_node) { _loadNode(opt.effects.next_node); return; }
  _endDialogue();
}

function _endDialogue() {
  box.classList.add('hidden');
  if (_resolve) { _resolve(); _resolve = null; }
}

function _typewriter(text) {
  return new Promise(res => {
    textEl.textContent = '';
    let i = 0;
    const DELAYS = { '.':250, '!':250, '?':250, ',':100 };
    function next() {
      if (i >= text.length) { res(); return; }
      textEl.textContent += text[i];
      const delay = DELAYS[text[i]] ?? 30;
      i++;
      setTimeout(next, delay);
    }
    next();
  });
}

```

## B.5 — TimeSystem.js + EconomySystem.js

```
// src/systems/TimeSystem.js
import * as TrustSystem from './TrustSystem.js';
import * as EconomySystem from './EconomySystem.js';
import * as EventSystem from './EventSystem.js';

export const TimeSystem = {
    act: 1, day: 1, hour: 8, dayOfWeek: 1,
    // Avançar N horas in-game (1 hora ≈ 1 ação do jogador)
    advance(hours = 1) {
        this.hour += hours;
        if (this.hour >= 24) {
            this.hour = 6;
            this.day++;
            this.dayOfWeek = (this.dayOfWeek % 7) + 1;
            TrustSystem.tickDayEnd(this.day);
            EventSystem.checkTriggers(this.day, this.hour, this.dayOfWeek);
            HUD.update();
            SaveSystem.save(); // auto-save a cada novo dia
        }
        if (this.dayOfWeek === 1 && this.hour === 6) { // Seg manhã = inicio de semana
            TrustSystem.tickWeekEnd();
            EconomySystem.onWeekStart(this.day);
        }
    },
    label() {
        const t = this.hour < 12 ? 'MANHÃ' : this.hour < 18 ? 'TARDE'
            : this.hour < 22 ? 'ENTARDECER' : 'NOITE';
        return `DIA ${this.day} - ${t}`;
    }
};
```

```
// src/systems/EconomySystem.js
export const EconomySystem = {
    gold: 5,
    _dailyIncome: 0,
    RENT: 15,

    earn(amount, source = '') {
        this.gold += amount;
        HUD.updateGold(this.gold);
        AudioSystem.play('sfx_coin');
    },

    spend(amount) {
        if (this.gold < amount) return false;
        this.gold -= amount;
        HUD.updateGold(this.gold);
        return true;
    },
}
```

```

onWeekStart(day) {
    // Aluguel cobrado toda Sexta (dayOfWeek = 5)
    // Verificar na Sexta-feira de cada semana
    if (TimeSystem.dayOfWeek !== 5) return;
    if (!this.spend(this.RENT)) {
        setFlag('FLAG_PLAYER_EVICTED', true, 'rent_failure');
        HUD.notify('Gareth exigiu pagamento. Você não tinha.', 'danger');
    }
},
evictionRisk() {
    const daysLeft = (5 - TimeSystem.dayOfWeek + 7) % 7 || 7;
    const projected = this.gold + (this._dailyIncome * daysLeft);
    if (projected < 15) return 'high';
    if (projected < 22) return 'medium';
    return 'none';
}
};

```

## B.6 — SceneManager.js

```

// src/scenes/SceneManager.js
let _current = null;
const _scenes = {};

export function register(name, scene) { _scenes[name] = scene; }

export async function go(name, params = {}) {
    if (_current?.onExit) _current.onExit();
    await _fadeOut();
    _current = _scenes[name];
    if (!_current) throw new Error(`Cena não registrada: ${name}`);
    await _current.onEnter(params);
    await _fadeIn();
    TimeSystem.advance(1); // transição de cena = 1 hora in-game
}

export function update(dt) { _current?.update(dt); }
export function draw(ctx) { _current?.draw(ctx); }

function _fadeOut() {
    return new Promise(res => {
        const overlay = document.getElementById('fade-overlay');
        overlay.style.opacity = '1';
        setTimeout(res, 300);
    });
}
function _fadeIn() {
    return new Promise(res => {

```

```

        const overlay = document.getElementById('fade-overlay');
        overlay.style.opacity = '0';
        setTimeout(res, 300);
    });
}

```

## B.7 — Renderer.js: Pixel Art no Canvas

```

// src/engine/renderer.js
let _ctx, _sheets = {};

export function init(canvas) {
    _ctx = canvas.getContext('2d');
    _ctx.imageSmoothingEnabled = false; // NUNCA borrar sprites
}

export async function loadSheet(name, src) {
    const img = new Image();
    img.src = src;
    await img.decode();
    _sheets[name] = img;
}

// Desenhar frame de spritesheet
// frame: indice na linha (0-based). row: linha na sheet (0-based).
export function drawSprite(sheet, frame, row, dx, dy, w, h, scale = 1) {
    const img = _sheets[sheet];
    if (!img) return;
    _ctx.drawImage(img, frame * w, row * h, w, h,
                    Math.round(dx), Math.round(dy), w * scale, h * scale);
}

export function drawTile(sheet, tx, ty, dx, dy, size = 16) {
    const img = _sheets[sheet];
    if (!img) return;
    _ctx.drawImage(img, tx * size, ty * size, size, size,
                    Math.round(dx), Math.round(dy), size * 3, size * 3); // 3x
    scale
}

export function clear() {
    _ctx.fillStyle = '#0D0A1F';
    _ctx.fillRect(0, 0, 320, 180);
}

// Sombra ellipse sob personagem
export function drawShadow(x, y) {
    _ctx.globalAlpha = 0.4;
    _ctx.fillStyle = '#000';
    _ctx.beginPath();
    _ctx.ellipse(x + 12, y + 30, 6, 2, 0, 0, Math.PI * 2);
}

```

```
    _ctx.fill();
    _ctx.globalAlpha = 1;
}
```

## PARTE C — Dados JSON Completos para a Demo

◆ Estes são os JSONs prontos para usar. Colocar em data/dialogues/act1\_week1.json e data/npcs/. O DialogueSystem.js os carrega por fetch.

### C.1 — gareth.json

```
{  
    "npc_id": "gareth",  
    "display_name": "Gareth",  
    "age": 45,  
    "trust": {  
        "familiaridade": 0, "lealdade": 0, "respeito": 0,  
        "simpatia": 0, "medo": 2, "curiosidade": 3  
    },  
    "arc_state": "neutral",  
    "last_interaction_day": 0,  
    "lies_detected": 0,  
    "culpa_used": false,  
    "thresholds": {  
        "dialogo_desbloqueado": 5,  
        "ruptura_irreversivel": -5,  
        "medo_bloqueia_simpatia": 10  
    },  
    "schedule": [  
        { "start": 6, "end": 11, "location": "tavern_kitchen" },  
        { "start": 11, "end": 23, "location": "tavern_main",  
            "saturday_override": { "end": 24 } }  
    ],  
    "sprite": "gareth",  
    "portrait_states": ["neutral", "pleased", "suspicious", "angry"]  
}
```

### C.2 — lyra.json

```
{  
    "npc_id": "lyra",  
    "display_name": "Lyra",  
    "age": 13,  
    "trust": {  
        "familiaridade": 0, "lealdade": 0, "respeito": 0,  
        "simpatia": 0, "medo": 8, "curiosidade": 5  
    },  
    "arc_state": "hostile",  
    "last_interaction_day": 0,  
    "lies_detected": 0,  
    "culpa_used": false,  
    "thresholds": {  
        "dialogo_desbloqueado": 5,  
        "ruptura_irreversivel": -10,  
        "medo_bloqueia_simpatia": 15  
    },  
    "schedule": [  
        { "start": 12, "end": 18, "location": "tavern_main" },  
        { "start": 18, "end": 23, "location": "tavern_kitchen",  
            "saturday_override": { "end": 24 } }  
    ],  
    "sprite": "lyra",  
    "portrait_states": ["hostile", "neutral", "suspicious", "angry"]  
}
```

```

        "ruptura_irreversivel": -5,
        "medo_bloqueia_simpatia": 10
    },
    "schedule": [
        { "start": 7, "end": 12, "location": "home_area", "days": [1,2,3,4,5] },
        { "start": 12, "end": 18, "location": "town_square" },
        { "start": 18, "end": 23, "location": "home_area" }
    ],
    "sprite": "lyra",
    "portrait_states":
    ["hostile_shouting","hostile","confused","quiet","neutral"]
}

```

### C.3 — aldric.json

```

{
    "npc_id": "aldric",
    "display_name": "Frei Aldric",
    "age": 67,
    "trust": {
        "familiaridade": 3, "lealdade": 1, "respeito": 1,
        "simpatia": 2, "medo": 0, "curiosidade": 4
    },
    "arc_state": "warming",
    "last_interaction_day": 0,
    "lies_detected": 0,
    "culpa_used": false,
    "thresholds": {
        "dialogo_desbloqueado": 4,
        "segredo_parcial": 6,
        "segredo_total": 8,
        "ruptura_irreversivel": -3,
        "medo_bloqueia_simpatia": 99
    },
    "schedule": [
        { "start": 7, "end": 11, "location": "church", "days": [3], "label": "missa" },
        { "start": 7, "end": 12, "location": "church" },
        { "start": 12, "end": 18, "location": "church_or_walk" },
        { "start": 18, "end": 22, "location": "church" }
    ],
    "special": { "familiaridade_min": 3 },
    "sprite": "aldric",
    "portrait_states": ["calm", "thoughtful", "amused", "serious", "warm"]
}

```

### C.4 — act1\_week1.json (estrutura completa — excerto)

◆ O arquivo completo contém todos os nós das cenas D1-01, D1-02, D1-03, D2-01, D3-01, D5-01 e DREAM-01. Excerto mostrando 2 nós com estrutura completa para referência de implementação.

```
[
  [
    {
      "node_id": "D1-02-001",
      "scene": "town_square",
      "trigger": {
        "type": "auto",
        "day_range": [1, 1],
        "conditions": [],
        "priority": 100
      },
      "speaker": "lyra",
      "portrait_state": "hostile_shouting",
      "text": "Olha só quem apareceu. O Destruidor de Aldeias. O Terror do Norte...",,
      "aspect_options": [
        {
          "id": "D1-02-001-A",
          "aspect": "SARCASMO",
          "text": "Obrigado pelo resumo. Economizou minha apresentação.",,
          "visible_when": [],
          "hidden_when": [],
          "effects": {
            "trust_deltas": { "lyra": { "simpatia": -1, "familiaridade": 1, "medo": -1 } },
            "flags_set": ["FLAG_NPC_LYRA_D1_SARCASM"],
            "next_node": "D1-02-002-A"
          }
        },
        {
          "id": "D1-02-001-B",
          "aspect": "CULPA",
          "text": "Você tem razão em cada palavra.",,
          "visible_when": [],
          "hidden_when": ["FLAG_NPC_LYRA_CULPA_USED"],
          "effects": {
            "trust_deltas": { "lyra": { "simpatia": 1, "curiosidade": 2, "medo": -1 } },
            "flags_set": ["FLAG_NPC_LYRA_D1_CULPA", "FLAG_NPC_LYRA_CULPA_USED"],
            "next_node": "D1-02-002-B"
          }
        },
        {
          "id": "D1-02-001-C",
          "aspect": "AUTORIDADE",
          "text": "Criança. Você não sabe de nada.",,
          "visible_when": [],
          "hidden_when": [],
          "effects": {
            "trust_deltas": { "lyra": { "lealdade": -3, "medo": 2 } },
            "flags_set": ["FLAG_VILLAGE_TRUST_HIT_1", "FLAG_PLAYER_AUTHORITY_USED_THIS_WEEK"],
            "next_node": "D1-02-002-D"
          }
        }
      ]
    }
  ]
]
```

```
        }
    },
],
"no_choice_result": {
    "aspect": "SILENCIO",
    "text": "[Malfeitor olha para ela, acena com a cabeça, continua andando]",
    "effects": {
        "trust_deltas": { "lyra": { "curiosidade": 2, "familiaridade": 1 } },
        "flags_set": ["FLAG_NPC_LYRA_D1_SILENCE"],
        "next_node": "D1-02-002-C"
    },
    "music_override": null
}
]
```

# **PARTE D — Ordem de Implementação: Sessão a Sessão**

---

- ◆ Para um desenvolvedor solo com IA. Cada sessão tem um deliverable claro — código que roda, não que compila. No final de cada sessão, o jogo deve ser abrível.

## **Sessão 1 — Esqueleto que abre (2-3h)**

- Criar index.html com canvas 320x180 + div#hud + div#dialogue-box + fade-overlay.
- style.css: body sem margens, game-wrapper centralizado, canvas com image-rendering: pixelated.
- main.js: game loop (requestAnimationFrame), input.js básico (WASD/Arrows).
- renderer.js: clear() + drawSprite() com placeholder rect colorido.
- CHECKPOINT: Abre no browser. Um quadrado se move com WASD.

## **Sessão 2 — Sistemas Core (3-4h)**

- FlagSystem.js completo (código da Parte B.1).
- TrustSystem.js + carregar gareth.json, lyra.json, aldric.json via fetch.
- MascaraSystem.js com ASPECT\_DELTA\_TABLE.
- save.js: saveGame() / loadGame() com localStorage.
- CHECKPOINT: console.log(TrustSystem.getTrust('lyra')) mostra objeto correto.

## **Sessão 3 — Diálogo funcionando (3-4h)**

- DialogueSystem.js: carrega act1\_week1.json, indexa por node\_id.
- DialogueBox.js completo (código da Parte B.4).
- Cena de teste: pressionar E aciona D1-01-004 (fala de Gareth sobre a vassoura).
- Todas as 4 opções de D1-01 funcionando com efeito visual de aspecto.
- CHECKPOINT: Diálogo completo da cena D1-01 jogável do início ao fim.

## **Sessão 4 — Cena da Taverna (3-4h)**

- TavernScene.js: tilemap estático (pode ser retângulos coloridos por enquanto).
- Player.js com colisão básica contra paredes da taverna.
- Gareth.js como NPC com zona de interação de raio 18px.
- SceneManager.js: fade e troca para town\_square ao sair pela porta.
- TimeSystem.js + HUD mostrando dia e hora.
- CHECKPOINT: Andar pela taverna, falar com Gareth (D1-01), sair para a praça.

## **Sessão 5 — Ato I Dias 1-7 completo (4-5h)**

- TownSquareScene.js com Lyra (D1-02) e trigger automático no Dia 1.
- ChurchScene.js com Aldric (D1-03, D3-01).
- DaviScene.js com jardim (D5-01).
- EconomySystem.js: renda de varrer, controle de aluguel na Sexta.
- DREAM-01: tela escurecida + narração typewriter + retorno à taverna.
- CHECKPOINT: Demo jogável completa. Dias 1-7. Todas as cenas. Save automático.

## **Sessão 6 — Polimento da Demo (2-3h)**

- Sprites reais (mesmo que simples — formas geométricas com cores corretas por personagem).
- Áudio: Howler.js + mus\_village\_day.ogg + sfx\_text\_blip.wav.
- TrustPanel.js: painel sem números, barras de 10 quadradinhos.
- Notificações: HUD.notify() para alertas de renda e eventos.
- CHECKPOINT: Demo pronta para playtest. URL abrível. Zero instalação.

## PARTE E — Testes Manuais de Smoke Test

- ◆ Sem framework de teste. Smoke tests via `console.assert()` em `test_smoke.js`, executado no browser. Abrir DevTools e checar por erros.

```
// tests/test_smoke.js - importar no index.html antes de main.js
// Executar: abrir index.html, F12, ver Console

import { setFlag, getFlag } from '../src/systems/FlagSystem.js';
import * as TrustSystem from '../src/systems/TrustSystem.js';
import * as MascaraSystem from '../src/systems/MascaraSystem.js';

async function runTests() {
  await TrustSystem.init();

  // T1: Lyra ARC_BROKEN jamais reverte
  setFlag('FLAG_NPC_LYRA_ARC_BROKEN', true, 'test');
  setFlag('FLAG_NPC_LYRA_ARC_BROKEN', false, 'test_revert');
  console.assert(getFlag('FLAG_NPC_LYRA_ARC_BROKEN') === true,
    '✗ T1 FAIL: ARC_BROKEN reverteu');
  console.log('✓ T1: ARC_BROKEN é permanente');

  // T2: Familiaridade decai após 2 dias sem contato
  TrustSystem.applyDelta('gareth', { familiaridade: 5 });
  const before = TrustSystem.getTrust('gareth').familiaridade;
  TrustSystem.tickDayEnd(1);
  TrustSystem.tickDayEnd(3);
  const after = TrustSystem.getTrust('gareth').familiaridade;
  console.assert(after === before - 1, `✗ T2 FAIL: FAM não decaiu. Era ${before}, é ${after}`);
  console.log('✓ T2: Familiaridade decai corretamente');

  // T3: Aldric FAM nunca cai abaixo de 3
  for (let i = 0; i < 20; i++) TrustSystem.tickDayEnd(i * 3);
  console.assert(TrustSystem.getTrust('aldric').familiaridade >= 3,
    '✗ T3 FAIL: Aldric FAM caiu abaixo de 3');
  console.log('✓ T3: Aldric FAM mínima garantida');

  // T4: Medo >= 10 bloqueia ganhos de simpatia em Lyra
  TrustSystem.applyDelta('lyra', { medo: 10 });
  const simBefore = TrustSystem.getTrust('lyra').simpatia;
  TrustSystem.applyDelta('lyra', { simpatia: 5 });
  console.assert(TrustSystem.getTrust('lyra').simpatia === simBefore,
    '✗ T4 FAIL: Simpatia aumentou com medo >= 10');
  console.log('✓ T4: Medo bloqueia simpatia');

  // T5: Save e Load preserva estado
  setFlag('FLAG_NPC_GARETH_FIRST_JOB', true, 'test');
  const saved = FlagSystem.serialize();
```

```
setFlag('FLAG_NPC_GARETH_FIRST_JOB', false, 'test_overwrite');
FlagSystem.deserialize(saved);
console.assert(getFlag('FLAG_NPC_GARETH_FIRST_JOB') === true,
  '✗ T5 FAIL: Save/Load não preservou flag');
console.log('✓ T5: Save/Load funciona');

// T6: CULPA segunda vez tem delta reduzido
TrustSystem.applyDelta('lyra', { simpatia: -10 }); // zerar
MascaraSystem.processChoice('lyra', 'CULPA', 'test1');
const simAfter1 = TrustSystem.getTrust('lyra').simpatia;
MascaraSystem.processChoice('lyra', 'CULPA', 'test2');
const simAfter2 = TrustSystem.getTrust('lyra').simpatia;
console.assert(simAfter2 - simAfter1 < simAfter1,
  '✗ T6 FAIL: Segunda CULPA não reduziu delta');
console.log('✓ T6: CULPA segunda vez reduzida');

console.log('--- Smoke tests concluídos ---');
}
runTests().catch(console.error);
```

## PARTE F — Perfis para Demo: Aldric e Stenn

### F.1 — Aldric: O que o jogador precisa saber na Demo

◆ Aldric na demo (Dias 1-7) é o NPC mais misterioso. Ele sabe quem Malfeitor é. O jogador não sabe que ele sabe. Esta tensão é o motor da curiosidade do jogador.

#### Comportamento de Aldric na Demo:

- Nunca menciona o nome Kael. Nunca menciona o passado diretamente.
- Frases com duplo sentido: 'algumas dívidas se pagam de formas inesperadas' é sobre a dívida DELE com Malfeitor, não o contrário.
- Quando Malfeitor usa CURIOSIDADE com Aldric, ele responde — mas sempre com uma verdade que cria mais perguntas.
- FAM inicial de 3 é visível ao jogador no TrustPanel como 'Rosto familiar' — sem explicação de por quê.

#### Linha narrativa de Aldric nos Dias 1-7 da demo:

Dia	Cena	Aldric diz (tom)	Subtexto real
1	D1-03	Não fiz por bondade. (firme, direto)	Fez por culpa. Sabe que deve algo.
3-5	D3-01	Tem documentos para copiar. (prático)	Quer Malfeitor próximo onde pode observá-lo.
7	DREAM-01	Aparece no sonho, de costas, jovem.	O jogador vê sem entender. Conexão futura.

### F.2 — Stenn: Mencionado mas não encontrado na Demo

◆ Stenn não aparece diretamente na demo (Dias 1-7). Mas é mencionado por Gareth e nos rumores. Definir seu estado inicial para quando aparecer no Ato I, Semana 2.

Atributo	Valor	Notas
trust inicial	FAM:0 LEA:0 RES:0 SIM:0 MED:3 CUR:2	Ele conhece a reputação de Malfeitor
arc_state	neutral	Ainda avaliando utilidade
schedule	Sede Gov: 8-18h / Taverna Sáb noite	Primeira possibilidade de encontro: Sáb noite, Sem 2
ASPECT_DELTA especial	MANIPULAÇÃO: +2 RES +1 LEA	Único NPC que respeita manipulação
VULNERABILIDADE	Vê como fraqueza explorável: +1 MED	Inverso da maioria dos NPCs



## PARTE G — Rotinas de NPC e Presença no Mundo

◆ Na demo, as cenas são pequenas e os NPCs têm posições semi-fixas. Esta parte define onde cada NPC fica em cada cena e como o jogador os encontra.

### G.1 — Posições Default por Cena (Demo)

Cena	NPC presente	Posição (world px)	Condição de presença
TavernScene	Gareth	x:200 y:120	Sempre, 6h-23h
TownSquare	Lyra	x:160 y:100	Dia 1: sempre. Dia 2+: 12h-18h
TownSquare	Aldeões anônimos (3)	x:80,120,200 y:90-120	Sempre durante o dia
ChurchScene	Aldric	x:160 y:90	Sempre, exceto Qua manhã (missa = não interativo)
HerbGardenScene	Davi	x:140 y:110	Dias 4-7, 6h-18h

### G.2 — Regras de Presença e Ausência

- Se Malfeitor vai a uma cena e o NPC não está lá (fora do schedule), a cena está vazia naquele local. Sem aviso, sem cutscene.
- NPCs anônimos têm frases contextuais (pool de 5-8 por cena) que mudam por semana e por estado de rumores ativos.
- Ícone de interação [E] aparece apenas quando: (1) NPC está presente, (2) dentro do raio de 18px, (3) arc\_state != locked e != broken.
- Se arc\_state == broken: ícone aparece mas diálogo retorna uma fala curta de rejeição — sem opções de aspecto.

### G.3 — Frases de Aldeões Anônimos (Demo)

#### Pool para TownSquare — Semana 1:

- 'Não acredito que o Frei deixou ele ficar.'
- 'Gareth vai se arrepender disso.'
- 'A menina da Lyra jogou uma pedra nele hoje. Ele nem reagiu.'
- 'Dizem que ele é um mago. Não parece.'
- 'Boa sorte pagar o aluguel varrendo o chão de taverna.' (aparece se FLAG\_NPC\_GARETH\_FIRST\_JOB = true)

- ◆ Frases de aldeões são ouvidas como narração em itálico quando Malfeitor passa dentro de raio de 30px. Não abrem caixa de diálogo completa — apenas uma bolinha de pensamento que aparece e desaparece em 2.5 segundos.

---

## ❀ FIM DO SUPPLEMENT BIBLE v2.0 ❀

*MALFEITOR | Stack Web | Demo: Ato I, Dias 1-7 | Complemento ao MasterBible v4.0*