Stickman Game – Referenz & Node.js Setup

Ziel

Webgame, das aus Textbefehlen 36 Frames @ 12 FPS generiert (Stickmen, Objekte, Effekte), mit Vanilla Canvas 2D, deterministischem Frame-by-Frame JSON, Editor-UI und Tests. Implementation in Node.js + TypeScript (Monorepo), entwickelt mit Cursor.

---

Tech-Stack

Node.js: v20 LTS (oder v22 LTS)

Package Manager: pnpm

TypeScript (strict)

Frontend: React + Vite (Editor)

Lint/Format: ESLint + Prettier

Tests: Vitest (unit) + Playwright (E2E Canvas Snapshots)

CI (optional): GitHub Actions

---

Monorepo-Struktur

/stickman

.editorconfig

.nvmrc

package.json # pnpm-workspace root

pnpm-workspace.yaml

tsconfig.base.json

/apps

/web # Vite + React (Editor & Preview)

/packages

/schema # Zod + TS-Types + parseScene()

/engine # Player & Canvas-Renderer

/patterns # buildThrowScene(), weitere Patterns

/parser # (optional) Text→Timeline-Regeln/LLM-Adaptionen

/utils # Mathe, easing, helpers

/tests

vitest.config.ts

playwright.config.ts

---

Node.js & Project Bootstrap

Empfohlene Versionen / Dateien

.nvmrc:

v20

.editorconfig:

root = true

[\*]

charset = utf-8

end\_of\_line = lf

insert\_final\_newline = true

indent\_style = space

indent\_size = 2

trim\_trailing\_whitespace = true

pnpm Workspace

pnpm-workspace.yaml

packages:

- "apps/\*"

- "packages/\*"

- "tests"

Root package.json (Scripts & Dev-Tools)

{

"name": "stickman",

"private": true,

"packageManager": "pnpm@9",

"scripts": {

"dev": "pnpm -C apps/web dev",

"build": "pnpm -r build",

"typecheck": "pnpm -r typecheck",

"lint": "pnpm -r lint",

"test": "pnpm -r test",

"test:e2e": "pnpm -C tests test:e2e",

"format": "prettier --write ."

},

"devDependencies": {

"typescript": "^5.5.0",

"eslint": "^9.0.0",

"prettier": "^3.3.0"

}

}

tsconfig.base.json

{

"compilerOptions": {

"target": "ES2022",

"lib": ["ES2022", "DOM"],

"module": "ESNext",

"moduleResolution": "Bundler",

"strict": true,

"noUncheckedIndexedAccess": true,

"exactOptionalPropertyTypes": true,

"declaration": true,

"sourceMap": true,

"skipLibCheck": true,

"types": ["vite/client"]

}

}

---

Package-Skeletons

/packages/schema/package.json

{

"name": "@stickman/schema",

"version": "0.1.0",

"type": "module",

"main": "./dist/index.js",

"types": "./dist/index.d.ts",

"scripts": {

"build": "tsc -p tsconfig.json",

"typecheck": "tsc -p tsconfig.json --noEmit",

"lint": "eslint src --ext .ts",

"test": "vitest run"

},

"dependencies": {

"zod": "^3.23.8"

}

}

/packages/schema/tsconfig.json

{

"extends": "../../tsconfig.base.json",

"include": ["src"]

}

/packages/schema/src/index.ts (Platzhalter; wird per Cursor-Prompt gebaut)

---

/packages/engine/package.json

{

"name": "@stickman/engine",

"version": "0.1.0",

"type": "module",

"main": "./dist/index.js",

"types": "./dist/index.d.ts",

"scripts": {

"build": "tsc -p tsconfig.json",

"typecheck": "tsc -p tsconfig.json --noEmit",

"lint": "eslint src --ext .ts",

"test": "vitest run"

},

"dependencies": {

"@stickman/schema": "workspace:\*"

}

}

---

/packages/patterns/package.json

{

"name": "@stickman/patterns",

"version": "0.1.0",

"type": "module",

"main": "./dist/index.js",

"types": "./dist/index.d.ts",

"scripts": {

"build": "tsc -p tsconfig.json",

"typecheck": "tsc -p tsconfig.json --noEmit",

"lint": "eslint src --ext .ts",

"test": "vitest run"

},

"dependencies": {

"@stickman/schema": "workspace:\*",

"@stickman/utils": "workspace:\*"

}

}

---

/apps/web (Vite + React + Tailwind)

/apps/web/package.json

{

"name": "@stickman/web",

"version": "0.1.0",

"private": true,

"type": "module",

"scripts": {

"dev": "vite",

"build": "tsc -p tsconfig.json && vite build",

"preview": "vite preview",

"typecheck": "tsc -p tsconfig.json --noEmit",

"lint": "eslint src --ext .ts,.tsx"

},

"dependencies": {

"react": "^18.3.0",

"react-dom": "^18.3.0",

"@stickman/schema": "workspace:\*",

"@stickman/engine": "workspace:\*",

"@stickman/patterns": "workspace:\*"

},

"devDependencies": {

"@types/react": "^18.3.3",

"@types/react-dom": "^18.3.0",

"vite": "^5.3.0",

"@vitejs/plugin-react": "^4.3.0",

"tailwindcss": "^3.4.7",

"autoprefixer": "^10.4.19",

"postcss": "^8.4.39"

}

}

Einfacher vite.config.ts:

import { defineConfig } from "vite";

import react from "@vitejs/plugin-react";

export default defineConfig({

plugins: [react()],

});

Tailwind minimal (optional): postcss.config.js, tailwind.config.js, src/index.css (mit @tailwind base; @tailwind components; @tailwind utilities;).

---

Tests

Vitest (Unit)

/tests/vitest.config.ts

import { defineConfig } from "vitest/config";

export default defineConfig({

test: {

environment: "node",

include: ["\*\*/\*.test.ts"]

}

});

Playwright (E2E)

/tests/playwright.config.ts

import { defineConfig } from "@playwright/test";

export default defineConfig({

timeout: 30\_000,

use: { headless: true },

webServer: {

command: "pnpm -C apps/web dev",

port: 5173,

reuseExistingServer: !process.env.CI

}

});

---

ESLint & Prettier (Root)

.eslintrc.json

{

"root": true,

"env": { "es2022": true, "browser": true, "node": true },

"extends": ["eslint:recommended"],

"parserOptions": { "ecmaVersion": "latest", "sourceType": "module" },

"ignorePatterns": ["dist", "node\_modules"],

"overrides": [

{

"files": ["\*\*/\*.ts", "\*\*/\*.tsx"],

"parser": "@typescript-eslint/parser",

"plugins": ["@typescript-eslint"],

"extends": ["plugin:@typescript-eslint/recommended"],

"rules": { "@typescript-eslint/consistent-type-imports": "warn" }

}

]

}

.prettierrc

{ "semi": true, "singleQuote": false, "printWidth": 100 }

---

Arbeitsplan mit Cursor-Prompts (copy & paste)

0) Bootstrap Monorepo

Erstelle im Root ein pnpm Monorepo:

- pnpm-workspace.yaml gemäß Struktur (apps/\*, packages/\*, tests).

- Root package.json mit Scripts (dev/build/typecheck/lint/test/test:e2e/format).

- tsconfig.base.json (strict, ES2022, DOM, declaration=true).

- .nvmrc mit v20.

- .editorconfig wie üblich.

- ESLint & Prettier Basiskonfig anlegen.

1) Schema definieren

In /packages/schema: Implementiere src/index.ts:

- Exportiere Typen: Vec2, JointName, Pose, ObjStatus, SceneObject, FrameObjectState, Frame, Scene.

- Zod-Schemas pro Typ (Scene: fps fix=12, frames exakt 36).

- parseScene(json): führt Zod-Validation aus und gibt { ok: true, data } oder wirft mit lesbarer Fehlermeldung.

- Optional: Soft-Warning Funktion checkSkeletonLengths(scene, tolerancePx=2), die Abweichungen je Frame sammelt (nur Warnung, keine Exception).

Erstelle tsconfig.json & package.json (build/typecheck/lint/test).

2) Engine / Renderer + Player

In /packages/engine: Implementiere

- drawStickman(ctx, pose): Linien für Arme/Beine/Torso, Kopf als Kreis (Radius 8).

- drawObject(ctx, frameObj, catalog): rect/circle/polygon/placeholder; rot wird respektiert.

- Player(ctx, scene):

- rAF-Loop, auf 12 FPS drosseln (83.33ms/frame); acc-basierter Ticker.

- drawFrame(i): clear canvas; actors, objects, effects (in dieser Reihenfolge).

- goto(i): setzt Index und rendert sofort.

- Onion-Skin-Option (prev/next Frame globalAlpha=0.2).

Exportiere alle Funktionen/Klassen.

3) Pattern „Throw“

In /packages/patterns: Implementiere buildThrowScene(text: string, seed=0): Scene

Konstanten:

- CANVAS: 640x400, FLOOR\_Y = 360

- A.x=120, B.x=320, beide pelvis.y=FLOOR\_Y

- GRAVITY = 0.9 px/frame^2

Ablauf (Frames 0..35):

- 0–5: A 'attached' vase01 in handR, leichte Beugepose.

- 6–10: Wind-up (Arm zurück), B idle.

- 11: Release → vase01.status="flying", vx=~5, vy=~-10 (seed für Varianz).

- 12–16: Parabel: x+=vx, y+=vy; vy+=GRAVITY.

- 17: Impact bei B head → vase01.status="destroyed"; spawn shards[6] am Impact mit zufälligen vx/vy; kurzer effect "bang".

- 18–20: shards fliegen auseinander.

- 21–35: shards fallen; am Ende status="fallen".

- Posen: B reagiert (head back, stumble), A follow-through; Recovery bis 35.

Rückgabe: vollständige Scene inkl. katalog objects (vase01 placeholder).

4) Web-App (Editor)

In /apps/web: Erstelle React + Vite App:

Komponenten:

- <CanvasView />: bekommt scene & currentFrame, zeichnet mit Engine.Player oder direktem drawFrame; Canvas 640x400.

- <Timeline />: 36 Buttons (0..35), zeigt aktuellen Frame; Klick = setCurrentFrame.

- <Controls />: Play/Pause, Step ±1, Loop, Onion-Skin Toggle.

- <PromptPanel />: Textarea + "Generate" → buildThrowScene(text) → parseScene → setScene & goto(0).

- Export/Import: JSON Download/Upload.

Zustand:

- currentScene | null, currentFrame (0), playing (bool), onionSkin (bool).

- useEffect: wenn playing true → rAF gesteuert oder setInterval 83ms (einfacher), aber bitte Engine-Player verwenden.

Styling: Tailwind minimal (optional).

5) Unit-Tests (Vitest)

In /tests: Lege Tests an

- schema.spec.ts:

- parseScene(validScene) → ok

- frames Länge != 36 → Fehler

- unbekannte object.id im Frame → Fehler

- patterns.spec.ts:

- buildThrowScene("A wirft eine Vase auf B") → parseScene ok

- Frame 17 hat mindestens einen effect "bang" und vase01.status="destroyed"

- shards existieren ab Frame 17 und erreichen status "fallen" bis Frame 35

Konfig: vitest.config.ts wie im GAMEPLAN.

6) E2E (Playwright)

In /tests: E2E Test

- Startet dev server (vite).

- Öffnet / (Editor).

- Tippt "A wirft eine Vase auf B" ins Textfeld, klickt "Generate".

- Scrub zu Frame 17; macht Screenshot des Canvas und vergleicht (Snapshot-Test).

- Prüft, dass "Impact" visuell vorhanden ist (z.B. Pixel-Änderung vs Frame 16).

Konfig: playwright.config.ts mit webServer Command pnpm -C apps/web dev.

7) ESLint/Prettier

Richte im Root ESLint & Prettier ein; für TS-Files plugin:@typescript-eslint/recommended aktivieren.

Skripte: "lint": "pnpm -r lint", "format": "prettier --write ."

8) CI (optional)

Erstelle .github/workflows/ci.yml:

- Node v20

- pnpm i

- pnpm build

- pnpm typecheck

- pnpm lint

- pnpm test

---

Developer-UX & Tipps

Determinismus: Frame-by-Frame Output ist debug-sicher; keine Hidden Tweens.

Seeds für Varianz (Wurf-Höhe, Shard-Winkel).

Onion Skin für Editing-Lesbarkeit.

Fallback: unbekannte Objekte als placeholder rendern (nie „kaputt“ gehen).

Canvas Schärfe: ganzzahlige Koordinaten → crisp lines.

Screen Shake: beim Impact 1–2 Frames ±2px Offsets (subtil, effektiv).