Levelsmith Labs Grid-Based Platformer Construction Kit

System Specification One-Sheet · Godot Engine

School of Interactive Games and Media · Faculty-led entrepreneurial co-op · 05/2026

A reference guide for co-op students actively designing and building the Levelsmith Labs tool — covering every system component, its constraints, and its expected behavior.

System Rules (Non-Negotiable)

- ☐ Stability over features
- ☐ No user action can break the system
- ☐ All systems must be grid-aligned
- ☐ All assets must be swappable without code

System Components

Visual Language *Greybox · Grayscale*

Neutral greybox aesthetic. Color = function, not decoration.

- ☐ Grayscale or desaturated palette only
- ☐ Color codes behavior: ice (blue), sticky (red), bounce (green)
- ☐ No textures, decorative art, or complex geometry
- ☐ Clarity and readability over aesthetics

Movement Controls *Parameterized · No Code*

Clamped sliders only. Presets: Default, Heavy, Floaty, Fast.

- ☐ Run speed — clamped to tile-based range
- ☐ Jump height — mapped to grid units (1–4 tile clear range max)
- ☐ Air control — clamped slider
- ☐ Fall speed — gravity multiplier, clamped
- ☐ Coyote time — short forgiveness window
- ☐ Movement must remain predictable and consistent across levels

Template Tiles *Fixed Set · Grid-Aligned*

All tiles are 1 grid unit blocks. No textures. Special surfaces color-coded.

- ☐ Tile set is fixed for v1.0
- ☐ Solid — standard impassable block
- ☐ Air — empty, passable space
- ☐ One-way — platform passable from below
- ☐ Hazard — resets player on contact

Character Set *Player · Enemies · Boss*

All characters share one base asset pipeline. No unique hierarchies.

- ☐ Player — simple, neutral, minimally animated
- ☐ Enemies — recolored player variants, behavior-specific animations
- ☐ Boss — enlarged player variant (3×3 grid footprint)
- ☐ Props — arrow, stop sign, goal, spawn, warning markers
- ☐ All animations use consistent bounding box and origin

Customization Pipeline *Folder-Based · No Code Edits*

All assets swap via folder. Changes are independent of game logic.

- ☐ All graphics swap via named folder — no code edits
- ☐ Validates file name, type, dimensions, frame consistency
- ☐ Invalid assets trigger clear user-facing error messages
- ☐ Invalid assets fall back safely to default
- ☐ Asset changes remain independent of all game logic

Animation Tester *Preview · Validate · Confirm*

Built-in tool to preview and validate swapped animations before use.

- ☐ Preview animations before they go live
- ☐ Supports multiple states (idle, run, jump, etc.)
- ☐ Auto-detect frame count and loop type
- ☐ Controls: Play · Pause · Restart · Speed adjust · Reset
- ☐ Validates bounding box and alignment consistency
- ☐ Detects missing or invalid frames

Audio *SFX · Music · Neutral*

Generic, replaceable audio. Supports feedback — does not define experience.

- ☐ Generic neutral SFX for all core interactions
- ☐ Simple background music, swappable via pipeline
- ☐ Audio communicates feedback only — not experience
- ☐ Audio system remains minimal and non-intrusive

Grid & Play Loop *Build · Play · Reset · Adjust*

Everything aligns to the grid. The play loop must be instant and non-destructive.

- ☐ All elements align strictly to grid (no off-grid placement)
- ☐ Default footprint = 1 grid unit (boss = 3×3 exception)
- ☐ Build -> Play -> Reset: instant, non-destructive, fully restores build state
- ☐ No setup required between runs
- ☐ Supports both small test scenarios and full platform levels

Success Conditions

- ☐ System is stable under normal use
- ☐ Behavior is predictable and readable
- ☐ User cannot easily break the tool
- ☐ All constraints are consistently enforced