Tiled Level Design Guide

This guide shows exactly what to place in Tiled, on which layer, with what names and custom properties. It covers entities, sensors, and transitions, plus how saving works.

1 pixel = 1 meter. Tiles are 16×16. The spawner places objects by their center point.

Layers to use

- ground (tilelayer)
  - Your visual tiles.
- solid (objectgroup)
  - Rectangles/polylines you want to be solid collision. Built by STI.
- entity (objectgroup)
  - Place spawnable game entities here (boxes, balls, bells, buttons, doors, statue).
- sensor or sensors (objectgroup)
  - Triggers. Named sensors and interactable sensors live here.
  - All fixtures in these layers are forced to be sensors (non-collidable).
- transitions (objectgroup)
  - Rectangles named transition1, transition2, …
  - Used for level switches between maps.

Optional:

- nm_map (tilelayer)
  - For normal-map visuals if you use the lighting shader later.

Naming and variant rules (entities)

The spawner supports two ways to create entities:

1. Name-based variants (recommended for simplicity)
   - Name the object with the variant, e.g. box1, ball2, bell1, button2, door1, statue.
   - The variant must exist exactly or the object is skipped.
2. Type-based with optional variant property (advanced)
   - Set object.type to a known type (box, ball, bell, button, door, statue).
   - Optionally set custom property variant to a known variant key (e.g. "box1").

Merging order for properties: type defaults ← variant preset ← object custom properties.

Entities (place on layer: entity)

Common notes:

- Size values are in pixels. Objects are placed at their centers.
- You may override dimensions via custom properties when supported.

Box

- Name: box1, box2 (or type=box and optional variant)
- Defaults: dynamic body
- Custom properties (optional):

- w, h: override size
- type: 'static' | 'dynamic'
- restitution: number
- friction: number
- density: number
- linearDamping, angularDamping: number

Ball

- Name: ball1, ball2 (or type=ball)
- Custom properties (optional):
  - r: radius (pixels)
  - restitution, friction: number

Bell

- Name: bell1 (or type=bell)
- Custom properties:
  - code: string pattern of L/S (e.g., "LSLS") to solve the bell puzzle
  - isSolved: boolean (applied from save and can be persisted)

Statue (singleton visual)

- Name: statue (or type=statue)
- Notes: reflects bell solve state; one statue instance is kept if multiple are placed.

Button

- Name: button1 (toggle on), button2 (momentary)
- Custom properties:

- sensor: number or string (1 → "interactableSensor1", or "interactableSensor2")
- key: string (e.g., "e") required to press; if omitted any key will work
- toggle: boolean (button1=true, button2=false)

Door

- Name: door1 (unlocked, closed), door2 (locked, closed)
- Behavior: closed = solid; open = non-collidable (sensor fixture).
- Custom properties:
  - sensor: number or string (1 → "interactableSensor1", or "interactableSensor2")
  - key: string (e.g., "e") to toggle open/close when unlocked
  - unlockKey: string (optional) to toggle locked/unlocked while inside the sensor
  - locked: boolean (initial lock state)
  - isOpen: boolean (initial open state)

Sensors (place on layer: sensor or sensors)

There are two kinds you can author.

Plain named sensors (read/check + callbacks)

- Name the object: sensor1, sensor2, …
  - OR add a boolean custom property sensorN=true on the object.
- Code can read `Sensors.sensor1` (boolean inside/outside) and hook:
  - `Sensors.onEnter.sensor1 = function(name) ... end`
  - `Sensors.onExit.sensor1 = function(name) ... end`

Interactable sensors (press a key while inside)

- Name the object: interactableSensor1, interactableSensor2, ...
- Optional custom property:
  - key: string (e.g., "e"). If set, only that key will trigger onPress.
- Usage in code:
  - `Interact.isInside('interactableSensor1')` → boolean
  - `Interact.onPress(1, key)` or `Interact.onPress('interactableSensor1', key)` → true when pressed inside
  - Optional callbacks:
    - `Interact.onEnter['interactableSensor1'] = function(name) ... end`
    - `Interact.onExit ['interactableSensor1'] = function(name) ... end`

Notes:

- All fixtures in sensor layers are forced to be Box2D sensors (non-solid).

Transitions (place on layer: transitions)

- Create rectangle objects named transition1, transition2, ... in both source and destination maps.
- The system scans all maps under `tiled/map` to pair transitions with the same name.
- When the player enters transitionX in the current map, they teleport to the center/edge of transitionX in the destination map.
- A short cooldown prevents immediate re-trigger.

SaveState (what persists between loads)

We do not modify exported Tiled files at runtime. Instead, we write a per-map overlay (in memory by default) and apply it after each map loads.

Examples of persisted properties:

- Bells: `bell1.isSolved`
- Doors: `doorX.locked`, `doorX.isOpen`
- You can add more by setting properties via code:

- `SaveState.setEntityPropCurrent('door1', 'locked', true)`

To enable disk persistence later:

```
SaveState.setPersistent(true)
SaveState.init('save/slot1.lua')
```

Quick cookbook

- Add a momentary button that opens a door:
  i. In sensors layer: add `interactableSensor2`, property `key="e"` (optional).
  ii. In entity layer: add `button2`, property `sensor=2`.
  iii. In entity layer: add `door1`, property `sensor=2`, `key="e"`.
- Add a locked door that can be unlocked in-place:
  i. In entity layer: add `door2`, properties `sensor=1`, `key="e"`, `unlockKey="f"`.
  ii. In sensors layer: add `interactableSensor1`.
- Trigger-only region with script hooks:
  i. In sensors layer: add `sensor3`.
  ii. Hook in code: `Sensors.onEnter.sensor3 = function() ... end`.

Reference: known types and variants

- box: box1, box2
- ball: ball1, ball2
- bell: bell1

- button: button1 (toggle), button2 (momentary)
- door: door1 (unlocked), door2 (locked)
- statue: statue (singleton)

If you need more variants, add them in `data/spawn_registry.lua` and use them by name in Tiled.