Glyph Console Manual — PsyRogue (printable)

Version 1.0 · for the in■game console (`)

This is a concise, printable guide to the programmable **Glyph Console** built into PsyRogue — Infinite Glyphlands. Every command below runs **while the game is live**. One command per line, or many inside a `(begin ...)` form.

---

0) Quick Start

• Open the console: press the backtick key **`** (above Tab).
• Run code: **Shift+Enter** or click **Run**.
• Comments start with `;` and run to end of line.

```
**Hello world**
(print "hello glyphlands")
**Starter buff** — heal 1 HP every 10 turns:
(set-hook 'onTurn (lambda () (if (= (% (turn) 10) 0) (heal 1) 0)))
```

---

1) The Language (tiny Lisp)

Core forms

• `(quote x)` – return `x` without evaluating it.
• `(if cond then else)` – truthy if not 0/false/empty.
• `(define name value)` – create a global binding.
• `(set! name value)` – mutate an existing binding.
• `(lambda (args...) body...)` – make a function.
• `(begin expr1 expr2 ...)` – run forms in sequence; return last value.

Literals

• Numbers: `1 3.14 -2`
• Strings: `"text"` (use `"` to escape quotes)
• Symbols: `foo bar + %` (unquoted names)
• Booleans: any non■zero is true; `0` is false

Math & misc

```
+ - * / % = < > <= >= sin cos floor ceil rand noise2
```

• `(rand)` → 0..1
• `(noise2 x y)` → 0..1 (hash■based smooth value, great for patterns)
• `(print ...)` → logs to the right panel and console output

---

2) Game Info Helpers

```
(turn) ; current turn number
(player-x) ; player X
(player-y) ; player Y
```

---

3) Player API

Read/Write stats

```
(player-get "hp") ; read any field
(player-set "hp" 20) ; hp, maxHp, def, scale, speed
(player-set "atkLo" 5) ; attack low bound
(player-set "atkHi" 9) ; attack high bound
```

Notes:

• `scale` is clamped **0.25–4.0** and only scales the `@` glyph (not the camera).
• `speed` adjusts monster frequency internally.

Actions

```
(heal 5)
(damage 3)
(teleport 0 0)
```

4) Tiles & Painting

```
Tile types: `floor wall water acid chasm terminal lorekey`
(tile-get x y) ; => string type
(tile-set x y "floor") ; set a single tile
(tile-disc cx cy r "wall") ; filled disc
(tile-ring cx cy r "acid") ; thin ring/border
(paint-near 6 "floor") ; disc centered on player
**Example – carve a safe bubble**
(tile-disc (player-x) (player-y) 6 "floor")
```

5) Enemies

```
**Spawn & query**
(spawn "slime" (+ (player-x) 3) (player-y)) ; returns enemy id
(enemy-at (player-x) (player-y)) ; id or 0
**Edit an enemy instance**
(enemy-set id "hp" 1)
(enemy-set id "aggro" 20)
(enemy-set id "dmgLo" 3)
(enemy-set id "dmgHi" 8)
**Edit a type**
(enemy-type-set "slime" "color" "#0f8")
(enemy-type-set "slime" "hpMin" 6)
(enemy-type-set "slime" "hpMax" 12)
**Define a new enemy type**
(enemy-type-define "orb" "■" 6 12 2 4 9 "#7ff")
(spawn "orb" (+ (player-x) 2) (player-y))
Fields: `(glyph hpMin hpMax dmgLo dmgHi aggro color)`
**Loop over nearby enemies**
(for-enemies 6 (lambda (id) (enemy-set id "hp" 1)))
```

6) Hooks (events)

Install with `(set-hook 'name fn)` and remove with `(remove-hook 'name)`.

- **`onTurn`** `() -> any`
- Runs after each full turn. Return value is ignored.
- **`onChunk`** `(cx cy) -> any`
- Called when chunks around the player are (re)generated.
- **`onMove`** `(dx dy nx ny) -> any`
- Called after the player steps; `nx,ny` are new coords.
- **`onAttack`** `(enemyId dmg) -> newDmg`
- You may return a **number** to replace outgoing damage.
- **`onDamaged`** `(dmg enemyId) -> newDmg`
- You may return a **number** to replace incoming damage.
- **`onKill`** `(enemyId type) -> any`
- Called when you kill something.

```
**Examples**
*Lifesteal on kill*
(set-hook 'onKill (lambda (id type) (heal 1)))
*Fiery stride (paint acid ring as you move)*
(set-hook 'onMove (lambda (dx dy nx ny)
(tile-ring nx ny 2 "acid")))
*Thorns (reduce incoming damage by 1, min 0)*
(set-hook 'onDamaged (lambda (dmg _)
```

```
(max 0 (- dmg 1))))
*Crits (double outgoing damage 25% of the time)*
(set-hook 'onAttack (lambda (id dmg)
(if (< (rand) 0.25) (* 2 dmg) dmg)))
Remove all hooks:
(begin (remove-hook 'onTurn) (remove-hook 'onMove)
(remove-hook 'onAttack) (remove-hook 'onDamaged)
(remove-hook 'onKill) (remove-hook 'onChunk))
```

---

7) World Parameters & Biomes

```
**Parameters**
(set-param "spawnRate" 0.09) ; enemy spawn chance near you
(get-param "spawnRate")
Common params: `wallThreshold`, `waterThreshold`, `acidThreshold`,
`chasmThreshold`, `paletteShift`, `spawnRate`.
**Biomes**
; define: name wall water acid chasm paletteShift spawnRate
(biome-define "marble" 0.40 0.60 0.76 0.90 300 0.05)
(biome-add "marble") ; include in world selection
(biome-set "marble" "spawnRate" 0.08)
```

---

8) Items, Abilities, Equipment

```
**Items** (consumables that run code)
(item-define "blink"
(lambda () (teleport (+ (player-x) 5) (player-y)))
"Short hop forward")
(item-give "blink" 2)
(item-use "blink")
**Abilities** (named functions you can trigger from code)
(ability-define "nova" (lambda ()
(for-enemies 6 (lambda (id) (enemy-set id "hp" 1)))))
(ability-trigger "nova")
**Equipment** (stats while equipped)
; name slot hp maxHp def atkLo atkHi scale speed desc
(equipment-define "boots_of_haste" "trinket" 0 0 0 0 0 0 0.6 "Move quicker")
(equip "boots_of_haste" "trinket")
; later
(unequip "trinket")
Slots available: `weapon`, `armor`, `trinket`.
```

---

9) Patterns & Recipes

```
**A) Safe meadow around you every turn**
(set-hook 'onTurn (lambda () (paint-near 5 "floor")))
**B) Define + spawn a new enemy swarm**
(begin
(enemy-type-define "wisp" "✧" 2 5 1 2 12 "#7ef")
(for-enemies 0 (lambda (_) 0)) ; no-op, just example
(spawn "wisp" (+ (player-x) 4) (+ (player-y) 0))
(spawn "wisp" (+ (player-x) 5) (+ (player-y) 1))
(spawn "wisp" (+ (player-x) 6) (+ (player-y) -1)))
**C) Throttle spawns**
(set-param "spawnRate" 0.02)
**D) Glass■cannon mode**
(begin (player-set "atkLo" 8) (player-set "atkHi" 16)
```

```
(player-set "hp" 8) (player-set "maxHp" 8))
**E) Crystal biome**
(biome-define "crystal2" 0.46 0.70 0.80 0.86 220 0.05)
(biome-add "crystal2")
**F) Emergency escape**
(begin
(tile-disc (player-x) (player-y) 4 "floor")
(heal 5))
```

---

10) Lore Keys & Codex

• Some tiles are `lorekey`. Stepping on one grants a **code snippet** to your Codex.
• Open Codex with **C** and click **Copy & Load** to paste a snippet into the console.

---

11) Troubleshooting

• **“unbound x”** – you used a name that wasn’t defined.
• **“not a function”** – the head of a list didn’t evaluate to a function.
• **Hook errors** are caught and shown as `{hook ... error: message}` in the log; the game keeps running.
• Player `scale` is clamped; camera never zooms.

```
**Reset common hooks**
(begin (remove-hook 'onTurn) (remove-hook 'onMove)
(remove-hook 'onAttack) (remove-hook 'onDamaged)
(remove-hook 'onKill) (remove-hook 'onChunk))
```

---

12) One■Page Cheat Sheet

```
**Forms:** `quote if define set! lambda begin`
**Math:** `+ - * / % = < > <= >= sin cos floor ceil rand noise2`
**Info:** `(turn) (player-x) (player-y)`
**Player:** `(player-get k) (player-set k v) (heal n) (damage n) (teleport x
y)`
**Tiles:** `(tile-get x y) (tile-set x y t) (tile-disc cx cy r t) (tile-ring
cx cy r t) (paint-near r t)`
**Enemies:** `(spawn type x y) (enemy-at x y) (enemy-set id k v)`
**Enemy Types:** `(enemy-type-define type glyph hpMin hpMax dmgLo dmgHi aggro
color)`
`(enemy-type-set type k v)`
**Loops:** `(for-enemies r (lambda (id) ...))`
**Hooks:** `(set-hook 'onTurn fn) (set-hook 'onMove fn) (set-hook 'onAttack
fn)`
`(set-hook 'onDamaged fn) (set-hook 'onKill fn) (set-hook 'onChunk fn)`
`(remove-hook 'name)`
**World:** `(set-param k v) (get-param k)`
**Biomes:** `(biome-define name wall water acid chasm pShift spawn)`
`(biome-add name) (biome-set name k v)`
**Items:** `(item-define name fn desc) (item-give name n) (item-use name)`
**Abilities:** `(ability-define name fn) (ability-trigger name)`
**Equipment:** `(equipment-define name slot hp maxHp def atkLo atkHi scale
speed desc)`
`(equip name slot) (unequip slot)`
```

End of manual.