Visual design breakdown

What is on the screen, what draws it, in what order, and which of it a designer can change on their own. Every picture is taken from the game itself; every number in a table is read out of the code when this document is built.



IN THIS DOCUMENT

1. How to read it: three kinds of knob
2. The screen and its layers
3. The board, layer by layer: ground, wires, buildings, gems, units
4. Aiming, placing, effects and motion
5. The rail, cards and sheets
6. Palette, art and type
7. Where to change what

How to read this

The game has almost no pictures in it. Apart from the logo, everything is drawn by code from flat colour, simple shapes and a set of single-colour SVG icons. So "changing the art" means one of three different things.

ART

Replace a file in `art/`. Every icon is a one-colour SVG: the game tints it at draw time, so only its **shape** counts and its own colours are thrown away. Keep the file name and a square canvas.

TOKEN

Change a number or a colour at the top of a script, or in `config.json`. Each is named, commented and listed in this document with its file and line. No logic changes: open the file, change the value, relaunch.

CODE

The shape or the behaviour itself: a chevron instead of a box, what is drawn first, when something animates. Needs a programmer, but the file and function are named here so the request can be exact.

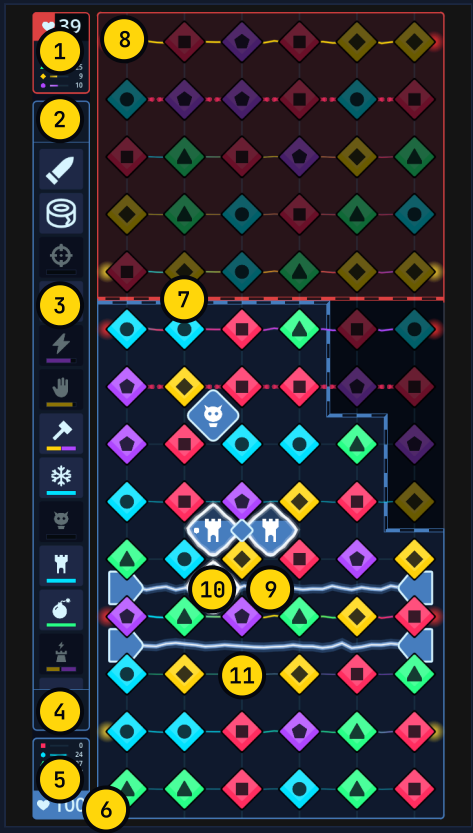
Sizes are in cells. Almost every length on the board is a fraction of one board cell, so the whole screen scales with the window. In the screenshots here the window is 720×1272 and a cell is about 89 px. Colours are written as hex; `Color(r, g, b, a)` in code is the same colour with each channel from 0 to 1.

The screen and its layers

The battle screen is two blocks side by side: the **board** on its dark slab, and the **rail** beside it with both players' keeps and the spellbook of the player to move.

Anatomy

`scenes/battle.tscn · scripts/battle.gd · _layout()`

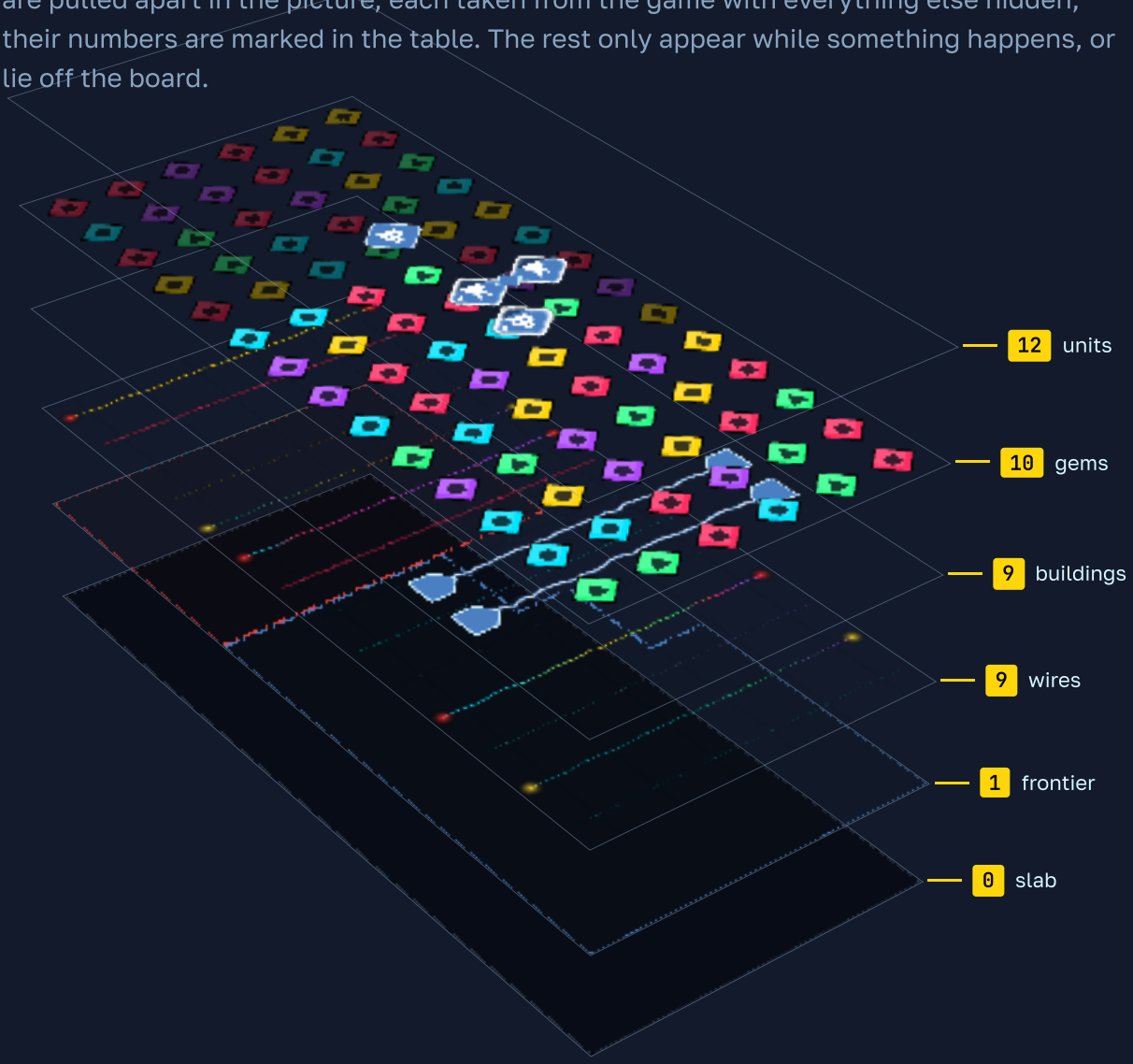


1. **Top keep**: the opponent's health and mana pools. `side_panel.gd · _draw_keep`
2. **Menu**: the game's logo as a button. `_draw_menu · paint_logo`
3. **Spellbook**: one card per spell, with a bar of how much of its price is banked. `paint_card`
4. **Turn**: the end-turn cross with four action diamonds round it. `_draw_turn · _draw_ap`
5. **Bottom keep**: the player at the bottom. `_draw_keep`
6. **Board slab**: the dark plate the board sits on. `battle.gd · SLAB`
7. **Frontier**: where each player's ground ends. `frontier.gd`
8. **Charged wire**: a row holding charge, close to its cap, so its ends glow. `wires_view.gd`
9. **E-Shield**: two pads and the discharge between them. `structures_view.gd`
10. **Unit**: a diamond in its owner's colour, filled as far as it has health. `unit_marker.gd`
11. **Gem**: one mana tile. `gem.gd`

The stack

`z_index in board_view.gd · setup() and battle.gd`

Each layer is a separate node. Higher draws over lower. The six layers of the board itself are pulled apart in the picture, each taken from the game with everything else hidden; their numbers are marked in the table. The rest only appear while something happens, or lie off the board.



Z	LAYER
0	Board slab `battle.gd`
1	Frontier and ground wash `frontier.gd`
2	Where a drag may go, washed `target_hint.gd`
9	Wires, charge, bolts `wires_view.gd`
9	Buildings: shields, towers `structures_view.gd`
10	Gems `gem.gd`
12	Units `unit_marker.gd`
15	A unit or gem under the finger
17	A building being aimed at, lifted in gold

Z	LAYER
18	Aim halos and info plates `target_hint.gd`
19	A unit being placed `place_ghost.gd`
20	Effects: sparks, shards, numbers, flying mana
50	Rail `side_panel.gd`
55	Screen dim
60	Card under the finger, card sheet
65	Banner
70–80	Menu, options, help, result

Each layer alone

`docs/design_capture.gd`

The same moment of the same game, one layer at a time, on a checkerboard where the layer is see-through. Wires and the frontier wash are faint on purpose: they sit under the gems and read only through the gaps.



Slab · z 0 | **Frontier** · z 1 | **Wires** · z 9 | **Buildings** · z 9 | **Gems** · z 10 | **Units** · z 12

Built up

The same layers added one at a time, bottom first. The last picture is the board as the player sees it.



slab | + **frontier** | + **wires** | + **structures** | + **gems** | + **units**

The board

Six columns by fourteen rows. Gems sit in the cells; units, wires and shield pads sit on the lines and nodes between them. Everything below is drawn in that order.

Board slab

battle.gd · _layout() · Kit.plate



The slab's top edge, with the opponent's ground washed red over it.

A flat plate, SLAB, with the grey frame every open sheet wears (Kit.SHEET_EDGE), rounded by Kit.ROUND. It is the board's size plus its frame (Kit.BOARD_INSET), so the outer gems run up to the frame. Nothing is drawn on it: the grid and the colour you see on the board come from the layers above.

TOKEN colour and frame. CODE a texture or pattern on the slab: the slab draws flat colour only.

CONSTANT	VALUE	WHAT IT SETS	WHERE
SLAB	#080c16	The board's backing panel: a shade off the window behind it, and inset by the one measure so the outermost gems are not cut flush with its frame.	scripts/battle.gd:82
SHADE	rgba(15,13,10,0.55)	What darkens the board under anything standing over it: enough to put the sheet in front, not so much that the board is gone.	scripts/battle.gd:85
RAIL	1.0	Cells of width the rail down the right of the board keeps for itself. It is narrow on purpose: whatever the board does not need goes to the board.	scripts/battle.gd:88
STRIP	3.0	Cells of width the rail takes while it is open over the board, where a card has room for its cost and a keep for its counts.	scripts/battle.gd:91

Frontier and ground

scripts/frontier.gd



Both frontiers meeting: each moved half its width onto its own side, so they lie side by side.

Each player's ground is **washed** in their colour at WASH strength, and its edge is drawn as a **barrier**: a 1 px rail down either side and solid pieces between, two to a cell, the second player's set half a piece behind the first's. Where the edge meets the board's rim it carries on to the frame, which takes the owner's colour beside that ground.

The colours are the players' own, Kit.ZONE, which a player picks before a campaign from Kit.ZONE_CHOICES.

CONSTANT	VALUE	WHAT IT SETS	WHERE
BAR	6.0	How thick the barrier runs and each rail of it, in pixels, rails set on whole pixels so they are never blurred; and how long a piece is, and each gap, in cells.	scripts/frontier.gd:22
RAIL	1.0	same as above	scripts/frontier.gd:23
PIECE	0.25	same as above	scripts/frontier.gd:24
WASH	0.15	How strongly the ground each player holds is washed in their colour: barely.	scripts/frontier.gd:26

TOKEN thickness, dash length, wash. CODE a different line style, such as a glow or a dotted edge.

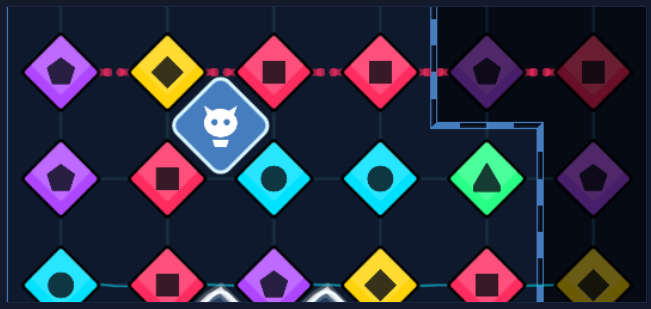
Wires and charge

scripts/wires_view.gd

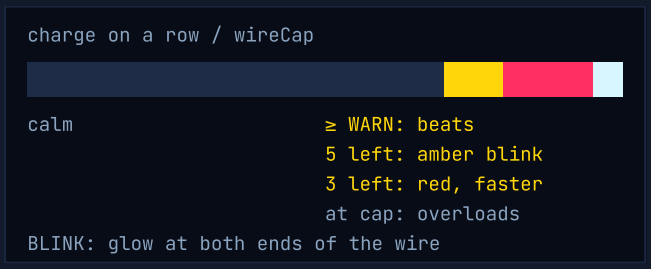
A wire runs through the middle of every row and every column, with a node where they cross. They are drawn under the gems, so they read only in the gaps between them. A row that holds charge carries a live discharge along its wire.



Charged. The colours on the row share its length; how much charge there is sets how far it throws, how bright and how thick it is.



Torn. A dead stretch under each gem, in TORN, cut where the gaps show.





CONSTANT	VALUE	WHAT IT SETS	WHERE
GRID	rgba(89,230,255,0.1)	The conductors themselves, and the thickness of every line, in cells.	scripts/wires_view.gd:17
NODE	rgba(89,230,255,0.18)	same as above	scripts/wires_view.gd:18
THICK	0.028	same as above	scripts/wires_view.gd:19
TORN	rgba(255,46,99,0.6)	A torn wire: the dead ends left behind, and how wide the gap between them runs, in cells. It is severed between every two gems of its row, where the gap shows, rather than once under the middle of the board.	scripts/wires_view.gd:23

CONSTANT	VALUE	WHAT IT SETS	WHERE
GAP	0.12	same as above	scripts/wires_view.gd:24
TEAR	0.55	The wire tearing: how long it takes, how far past its resting width each gap is thrown open before the ends settle, and the sparks thrown off every cut -- how many, and how far they fly, in cells.	scripts/wires_view.gd:28
RECOIL	2.4	same as above	scripts/wires_view.gd:29
SPARKS	5	same as above	scripts/wires_view.gd:30
SPARK_REACH	0.42	same as above	scripts/wires_view.gd:31
LIVE_THICK	0.015	The discharge: how thick, how far it dances off its wire, and how long one zigzag segment runs -- all in cells.	scripts/wires_view.gd:34
JITTER	0.032	same as above	scripts/wires_view.gd:35
STEP	0.17	same as above	scripts/wires_view.gd:36
FLICKER	18.0	How many times a second an arc re-rolls its shape.	scripts/wires_view.gd:38
BOLT_SPEED	26.0	How fast a released bolt travels, in cells per second.	scripts/wires_view.gd:40
HEM	0.16	How far a row stops short of the board's rim, in cells.	scripts/wires_view.gd:42
FLASH	0.22	How long the white flash of a release lasts.	scripts/wires_view.gd:44
WARN	0.7	How full a wire has to be before it beats.	scripts/wires_view.gd:46
BLINK	`[[3, 3.0, Color(1, 0.2, 0.2)], [5, 1.0, Color(1, 0.85, 0.2)]]`	The warning that a row is about to let go into whatever stands on it: a glow at either end of its wire, under the gems, blinking once the row is within so many units of its cap, at so many blinks a second, in this colour. The closer, the faster.	scripts/wires_view.gd:51
DOT	0.2	How far the glow reaches from the end of the wire, in cells: it is full at the middle and gone at this far.	scripts/wires_view.gd:54

TOKEN every colour, thickness, speed and blink step above. CODE the zigzag itself, or a sprite instead of drawn lines.

Buildings: E-Shield and towers

scripts/structures_view.gd



Pads are chevrons pointing in, flat on the outer side, drawn under the gems so only their corners show. A pad is filled from the bottom as far as the shield has health; the rest is

A charged E-Shield: a pad at each end, the discharge between them.

its owner's colour darkened (`Kit.unit_worn`). A hairline in the interface ink runs just inside its outline.

The **discharge** burns as hard as the two rows either side hold charge. With none, it is a white dashed hairline that flickers. A blow it puts out makes it flash white and throw a ring.



CONSTANT	VALUE	WHAT IT SETS	WHERE
RIM	0.45	How wide the rim is that a card let go on is aimed at a building, in cells.	scripts/structures_view.gd:10
PAD	0.65	How wide and how tall a shield's chevron is, in cells, how deep its point runs, and how close a finger has to come to it to aim at it.	scripts/structures_view.gd:13
PAD_TALL	0.6	same as above	scripts/structures_view.gd:14
ARM	0.25	same as above	scripts/structures_view.gd:15
REACH	0.4	same as above	scripts/structures_view.gd:16
SOFT	0.04	How far back from each of its corners a shield's chevron is rounded, in cells: its shoulders and the two corners of its flat back, and its point.	scripts/structures_view.gd:19
POINT	0.1	same as above	scripts/structures_view.gd:20
BEAM	0.09	A shield's discharge: how thick it runs with a full pair of rows behind it, how far it throws off its line, how long each stretch of it is, all in cells, and how many times a second it is re-rolled.	scripts/structures_view.gd:24
THROW	0.07	same as above	scripts/structures_view.gd:25
STEP	0.17	same as above	scripts/structures_view.gd:26
FLICKER	18.0	same as above	scripts/structures_view.gd:27
IDLE_LINE	1.5	How wide an uncharged shield's hairline is drawn, in pixels: white and plain enough to see on the board.	scripts/structures_view.gd:30
IDLE_JIG	0.2	How far its dashes shift each time it flickers, as a share of a dash.	scripts/structures_view.gd:32
DASH	0.08	How long each dash of that hairline is, in cells.	scripts/structures_view.gd:34
SOAK	0.38	A blow put out: how long the shield burns white for it, in seconds, and how wide the ring it throws up where the blow met it gets, in cells.	scripts/structures_view.gd:37

CONSTANT	VALUE	WHAT IT SETS	WHERE
RING	0.55	same as above	scripts/structures_view.gd:38

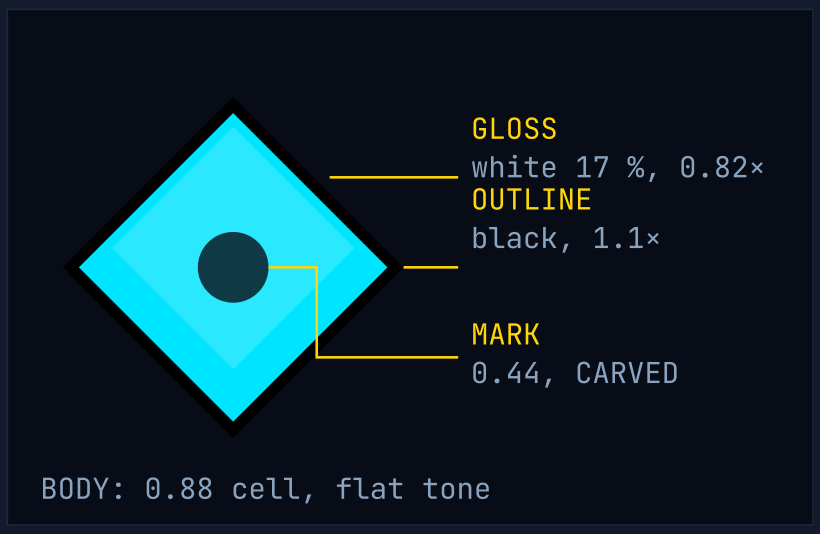
TOKEN sizes, rounding, beam. CODE the outline itself, which is a list of points in `_pad_outline()` and SHAPE.

Gems

scripts/gem.gd · config.json tileColors



One gem, close up.



Every gem is the same diamond (`art/i-diamond.svg`) in four stacked pieces: a black outline behind, the body in the colour's **flat** tone, a smaller white gloss nudged up, and the colour's **mark** cut into the middle so colour-blind players can tell them apart. Gems the player to move cannot swap are turned down to `OUT_OF_REACH`; while a spell picks its target all of them go to `DIMMED`. A gem in a match flashes white and grows a little.



The player to move's own ground: full colour.



The other side: `OUT_OF_REACH`, the frontier wash showing through.

CONSTANT	VALUE	WHAT IT SETS	WHERE
GLOSS	0.17	The highlight, as flat white over the gem's own colour.	scripts/gem.gd:10
OUTLINE	0.1	The outline, as a fraction of the diamond it sits behind.	scripts/gem.gd:12
BODY	0.88	How wide the diamond is, as a fraction of its cell, and how wide the mark inside it is, as a fraction of the diamond.	scripts/gem.gd:15

CONSTANT	VALUE	WHAT IT SETS	WHERE
MARK	0.44	same as above	scripts/gem.gd:16
CARVED	rgba(8,13,23,0.8)	The mark is cut into the gem rather than laid on it.	scripts/gem.gd:18
DIAMOND	preload("res://art/i-diamond.svg")	same as above	scripts/gem.gd:19

CONSTANT	VALUE	WHAT IT SETS	WHERE
DIMMED	rgba(128,128,128,1)	How far the gems darken while a placement or a spell is picking its cell. They are turned down, not made see-through: nothing under a gem shows.	scripts/board_view.gd:10
OUT_OF_REACH	rgba(107,107,107,1)	How far a gem darkens when it is not the player's to touch: swaps only happen on ground you hold, so ground you do not hold is turned down.	scripts/board_view.gd:13

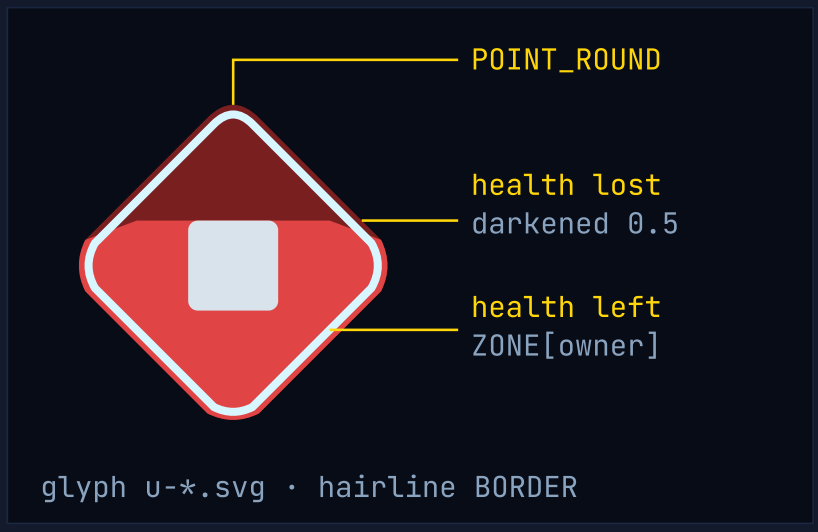
ART `i-diamond.svg` (the body) and each colour's mark, `i-*.svg` and `g-triangle.svg`. TOKEN the colours in `config.json` and the proportions above. CODE a gradient, a shadow or a full-colour gem bitmap.

Units

scripts/unit_marker.gd · Kit.diamond



A unit at full health.



A unit stands on the nodes between gems. Its body is a rounded diamond the size of a cell, in its **owner's** colour (not its race's): the share of health it has left fills it from the bottom. Its glyph and a hairline are in the interface ink, gold while a spell is aimed at it. The same body, lighter, is the ghost shown while a unit is being placed and the card carried under the finger.

CONSTANT	VALUE	WHAT IT SETS	WHERE
DISC	1.0	How wide the disc is, as a fraction of a cell.	scripts/unit_marker.gd:11
BORDER	0.035	How thick the hairline runs, in cells.	scripts/unit_marker.gd:13
DOT	0.05	A dot on a corner: how big, as a fraction of a cell, and how far out towards the corner it stands, as a fraction of the way.	scripts/unit_marker.gd:16
DOT_OUT	0.6	same as above	scripts/unit_marker.gd:17

CONSTANT	VALUE	WHAT IT SETS	WHERE
STEP	0.12	The burning border: how long one zigzag runs, how far it dances off its line and how thick it burns, in cells, how many times a second it re-rolls, and how many times a second it beats with little charge and with a lot.	scripts/unit_marker.gd:21
THROW	0.035	same as above	scripts/unit_marker.gd:22
BEAM	0.065	same as above	scripts/unit_marker.gd:23
FLICKER	18.0	same as above	scripts/unit_marker.gd:24
BEAT_SLOW	0.7	same as above	scripts/unit_marker.gd:25
BEAT_FAST	4.0	same as above	scripts/unit_marker.gd:26

CONSTANT	VALUE	WHAT IT SETS	WHERE
POINT_ROUND	0.22	How far a unit's diamond is rounded at each point, as a fraction of its radius, and how many stretches each rounded point is drawn in.	scripts/ui_kit.gd:135
POINT_STEPS	6	same as above	scripts/ui_kit.gd:136

ART `u-imp.svg` (Ranger), `u-tower.svg` (Tower), `u-bomb.svg` (Bombarder). The Tower is two diamonds with a small diamond between them (`Kit.unit_blocks`, `Kit.PLATE`). TOKEN size, hairline, rounding. CODE another body shape, or bars instead of the fill.

Aiming, placing and motion

What the board shows while a card is in the player's hand, and how it moves when something happens.

Aiming a spell

scripts/target_hint.gd



Aiming. A Strike carried over the enemy: every gem dims, what it may hit is washed in gold, what it is over now is ringed.



Placing. The unit it will be, lighter, on the node under the finger. Red dots are the nodes it would strike from there.

CONSTANT	VALUE	WHAT IT SETS	WHERE
EDGE	#ffd60a		scripts/target_hint.gd:9
FAINT_FILL	0.12	How far back the ground a drag may go to is washed. It is a hint, and there is a lot of it at once.	scripts/target_hint.gd:41
AIM_FILL	0.85	How strongly the rows a spell is aimed at are washed inside their frame.	scripts/target_hint.gd:43

CONSTANT	VALUE	WHAT IT SETS	WHERE
PLATE_UP	0.78	A plate over something picked out: how far over its middle it stands, how tall its writing is, and how much room is kept round the writing, in cells.	scripts/target_hint.gd:46
PLATE_TEXT	0.3	same as above	scripts/target_hint.gd:47
PLATE_PAD	0.1	same as above	scripts/target_hint.gd:48
MARK	0.34	How big the spell's mark is on each cell it is aimed at, drawn in the carved ink a filled card's mark is, as a share of a cell.	scripts/target_hint.gd:51
HALO_POINT	0.04	The edge of a block of cells aimed at runs parallel to the edges of its gems: a point at the middle of every cell's side, this far out of the block, and a notch between every two, this far in, both in cells, the two adding up to half a cell. And how far back from a point and from a notch the edge starts to round.	scripts/target_hint.gd:56
HALO_NOTCH	0.46	same as above	scripts/target_hint.gd:57
HALO_POINT_ROUND	0.12	same as above	scripts/target_hint.gd:58
HALO_NOTCH_ROUND	0.2	same as above	scripts/target_hint.gd:59
HALO_STEPS	8	How many stretches a rounded corner of it is drawn in.	scripts/target_hint.gd:61
REACH	0.22	The wash over ground a picked unit lays its influence on.	scripts/target_hint.gd:63

CONSTANT	VALUE	WHAT IT SETS	WHERE
DISC	1.0	The same disc the unit marker draws, and the same hairline of health round it. The numbers are repeated rather than read off that script: everything on the board is loaded by path, so a fresh checkout has no class to read them from.	scripts/place_ghost.gd:11
BORDER	0.035	same as above	scripts/place_ghost.gd:12

> The ghost repeats the unit's `DISC` and `BORDER` rather than reading them from `unit_marker.gd`. Change both together.

Effects and motion

`board_view.gd · apply() · config.json timing`

Every animation replays an event the rules already decided, so none of them can be skipped or reordered by art alone. The durations live in `config.json`, in milliseconds:

`timing.swap`	190 ms	
`timing.matchHighlight`	150 ms	
`timing.remove`	220 ms	
`timing.mana`	220 ms	

timing.gravity	240 ms	
timing.projectile	340 ms	
timing.impact	220 ms	
timing.unitMove	240 ms	
timing.step	90 ms	

What each effect is

- **spark**: a disc that grows 1.9× and fades, on every hit and heal.
- **burst**: 8 triangle shards thrown out, spinning, 0.34–0.5 s, when something is destroyed.
- **float text**: the number, 0.4 cell high, rising 0.9 cell over 0.6 s.
- **mana mote**: the colour's mark flying to the keep, shrinking to 45 %.
- **bolt**: lightning running down a column at `BOLT_SPEED`.
- **bomb**: `u-bomb.svg` lobbed, swelling `LOB_SWELL`× at the top of its arc.

Effect colours

damage	heal	shield
#ff2e63	#2bff88	#9df3ff

> These are written into `board_view.gd · apply()` where each event is handled, not as named constants. Changing them is a small code change.

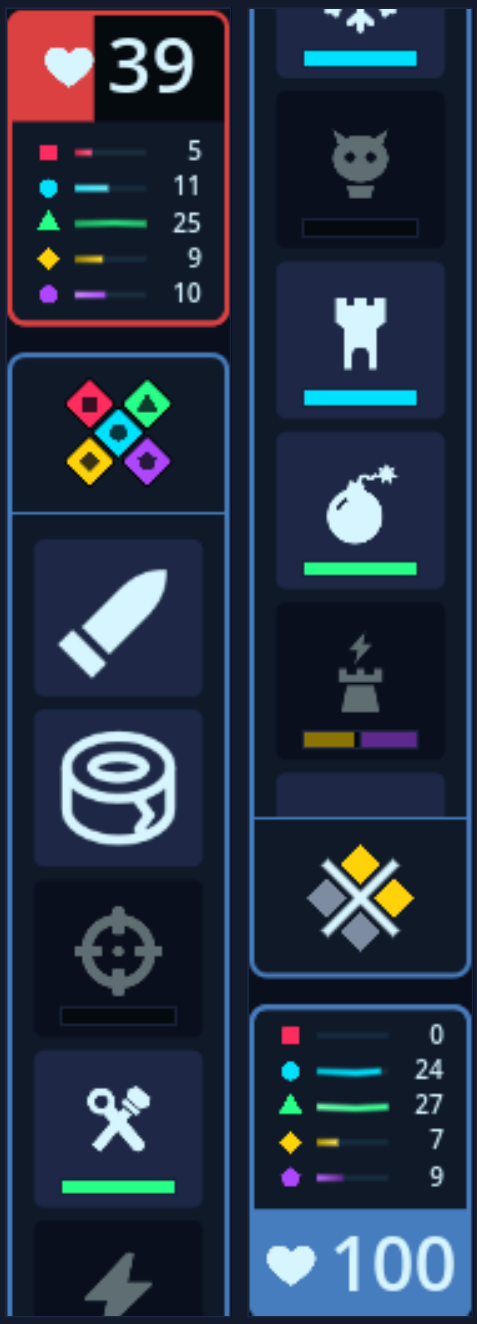
CONSTANT	VALUE	WHAT IT SETS	WHERE
LOB_SIZE	0.4	How big a lobbed bomb is drawn, in cells, and how many times that it swells to at the top of its flight.	scripts/board_view.gd:16
LOB_SWELL	1.8	same as above	scripts/board_view.gd:17
SHUFFLE	420	How long the gems take to fly to where a shuffle left them, in milliseconds.	scripts/board_view.gd:863

The rail, cards and sheets

Everything off the board is drawn by `side_panel.gd`, `detail_panel.gd` and the sheets in `battle.gd`, from the plates and buttons in `ui_kit.gd`.

The rail

`scripts/side_panel.gd · _draw()`



The rail, narrow: its top and its foot. On a wide window it is pinned open at three cells, with names on the cards.

Drawn in this order

1. The backing, `Kit.BACK`.
2. The book's recess, `BOOK`, round the menu, the cards and the turn button.
3. The cards. Then the tick and cross over a card that waits to be confirmed.
4. Both keeps: a health wall at the outer end, then a mana pool per colour, each with a little live wire as long as that pool.
5. The menu button (the logo) and the turn button.
6. A frame round menu, book and turn in the colour of the player to move: the one place the rail says whose turn it is.

A card's faces

- `CARD` normally; `TINT_LIT` and a little smaller while pressed.
- `Kit.TINT_DIM` with `DIM_INK` when it cannot be cast.
- Framed in `GOLD` while armed; filled with `GOLD` and written dark when chosen.
- The bar along its foot: one segment per colour of its price, filled as far as the pool covers it.

CONSTANT	VALUE	WHAT IT SETS	WHERE
HOLD	0.28	How long a press has to last before it picks the card up: framed in gold, and a pull any way, along the column too, carries it out rather than scrolling the book. Let go where it is, it opens as a tap would.	scripts/side_panel.gd:32
SLOP	0.12	Movement that turns a press into a scroll or a drag, in cells.	scripts/side_panel.gd:34
SWIPE	0.3	Movement across the rail that opens it or shuts it, in cells.	scripts/side_panel.gd:36
SLIDE	0.14	How long the rail takes to slide the whole way open or shut.	scripts/side_panel.gd:38
BLINK_AT	0.25	The clock blinks once it is down to this much of the turn.	scripts/side_panel.gd:40
MANA_FULL	20.0	What runs a pool's discharge the whole length of its wire. Mana is not capped: once a pool is past it, the fullest pool in the keep is what the others are drawn against instead, so none of them reads full while another holds more.	scripts/side_panel.gd:45
WIRE	rgba(89,230,255,0.1)	A pool is drawn the way a row's charge is on the board: a wire, and a discharge dancing along it. The measures are in cells -- how thick the wire and the discharge run, how far it dances off the wire, how long one zigzag runs -- and how many times a second it re-rolls its shape. A pool's line is a fraction of a row, so its discharge runs thinner than the board's and is thrown less far.	scripts/side_panel.gd:52
WIRE_THICK	0.028	same as above	scripts/side_panel.gd:53
ARC_THICK	0.02	same as above	scripts/side_panel.gd:54
ARC_JITTER	0.014	same as above	scripts/side_panel.gd:55
ARC_STEP	0.17	same as above	scripts/side_panel.gd:56
ARC_FLICKER	18.0	same as above	scripts/side_panel.gd:57
ARC_SHADE	0.4	How far a point of the discharge may be lit or darkened off its own colour.	scripts/side_panel.gd:59
INK	#d8f6ff		scripts/side_panel.gd:61
DIM_INK	#4d6478		scripts/side_panel.gd:62
GOLD	#ffd60a		scripts/side_panel.gd:63
BADGE_BG	rgba(5,8,15,0.85)		scripts/side_panel.gd:64

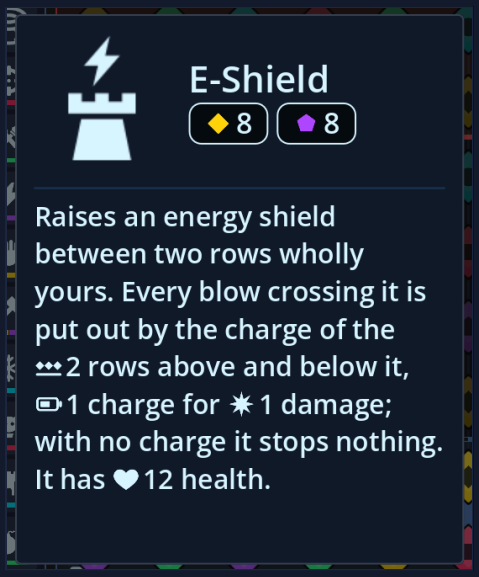
CONSTANT	VALUE	WHAT IT SETS	WHERE
CARVED	#060a12	Marks cut into the wood rather than laid on it: the cross and the menu bars.	scripts/side_panel.gd:66
BACK	Kit.BACK	The column stands on its own backing, and that is what shows between the panels and between the column and the board.	scripts/side_panel.gd:69
BOOK	#121a2b	The recess a book of cards sits in, and the card standing in it.	scripts/side_panel.gd:71
CARD	#1e2c48	same as above	scripts/side_panel.gd:72
TINT	Kit.TINT	same as above	scripts/side_panel.gd:73
TINT_LIT	Kit.TINT_LIT	same as above	scripts/side_panel.gd:74
GLYPH	0.99	How much of a button's height its mark takes: the picture on a card, the cross on the turn panel.	scripts/side_panel.gd:275
BANK	0.26	The strip at the foot of a card that shows what is banked towards it, as a fraction of the card's height.	scripts/side_panel.gd:278
HEAD	0.78	The part of an open card that holds its mark with its name and its price beside it, as a fraction of the card's height. What is banked towards it takes the rest, under it.	scripts/side_panel.gd:282
OPEN_TALL	1.35	How much taller every button stands on the open rail: a card over what its mark alone asks for, the menu and the turn button over their narrow selves.	scripts/side_panel.gd:285
TEXT_GAP	0.16	The air between a button's mark and what is written beside it, in cells.	scripts/side_panel.gd:287
SPENT	#7d8fa3	A place for an action the turn does not have: seen, and plainly not gold.	scripts/side_panel.gd:289
CROSS	0.6	How far one arm of the end-turn cross reaches from its middle, as a share of half a card's mark.	scripts/side_panel.gd:292
AP_FAR	0.96	The action diamonds round the end-turn cross, as fractions of one arm of the cross: how far each stands from its middle, and how big it is from its centre to a point. What is left between the two is the air off the cross.	scripts/side_panel.gd:296
AP_SIZE	0.58	same as above	scripts/side_panel.gd:297
SLOT	0.72	The panels that hold one thing -- the turn, the menu -- are the same height, so the rail reads as one stack of equal plates.	scripts/side_panel.gd:344

CONSTANT	VALUE	WHAT IT SETS	WHERE
WALL	0.5	A keep is cut to exactly what it holds, one thing under the other: the wall its health is drawn as, and a line for every pool. Whatever it does not need goes to the books, which is where a rail that is mostly cards wants its height.	scripts/side_panel.gd:349
POOL	0.16	same as above	scripts/side_panel.gd:350
COST_STEP	2.1	What a card costs: the mark of the mana it is paid in and the price beside it, on a badge `h` tall, dark unless `back` says otherwise. How far one price stands from the next on a card paid in more than one colour, in badge heights.	scripts/side_panel.gd:1028
BADGE_W	1.9	How wide a price badge is, in badge heights.	scripts/side_panel.gd:1030

The rail repeats several interface colours instead of reading them from `Kit`: `INK`, `GOLD`, `CARVED`, and `BOOK` / `CARD`, which equal `Kit.TINT` / `Kit.TINT_LIT`. The same is true of `detail_panel.gd`, `help_panel.gd`, `drag_card.gd` and `lobby.gd`. A palette change has to be made in each, or the copies merged into `Kit` first, which is a small code task.

The card sheet

scripts/detail_panel.gd



A card held open: picture, name, price, then the rules text with its own icons inline.

A plate with the sheet frame, over a dimmed screen. The picture is the spell's or unit's icon, `ART` cells high; a keep shows its race's crest instead, a hexagon in the race's colour. The rules text comes from `config.json`: a word in braces such as `{t-damage}` is drawn as that icon, at the size of the text.

`ART` icons. `TOKEN` sizes and colours below, and the text in `config.json`. `CODE` the layout of the sheet.

CONSTANT	VALUE	WHAT IT SETS	WHERE
PAD	0.26	inside the sheet, in cells	scripts/detail_panel.gd:8
ART	1.8	the picture, in cells	scripts/detail_panel.gd:9
LINE	0.5	a line of body text, in cells	scripts/detail_panel.gd:10
INK	#d8f6ff		scripts/detail_panel.gd:12

CONSTANT	VALUE	WHAT IT SETS	WHERE
DIM	#6f8aa8		scripts/detail_panel.gd:13
GOLD	#ffd60a		scripts/detail_panel.gd:14
RULE	#1b3450		scripts/detail_panel.gd:15
BADGE	0.56	the cost under a card's name, in cells	scripts/detail_panel.gd:16
WALL	0.5	a player's health, in cells	scripts/detail_panel.gd:17
POOL	0.4	a line of a player's mana, in cells	scripts/detail_panel.gd:18
CARVED	#060a12		scripts/detail_panel.gd:19

Sheets, buttons and the banner

ui_kit.gd · sheet_button · battle.gd · _popup



The menu: one sheet, sheet buttons.

Every question the game asks is the same **sheet**: a `Kit.TINT` plate framed in `Kit.SHEET_EDGE`, over the screen dimmed by `SHADE`. Every button on it is cut from the window, not in pixels: about 72 % of its width and 7 % of its height.

BUTTON STATE	PLATE	FRAME	TEXT
normal	TINT	SHEET_EDGE	INK
hover	CARVED	GOLD	GOLD
pressed	GOLD	GOLD	CARVED
disabled	TINT_DIM	SHEET_EDGE	INK_DIM

A pressed button dips to 95 % in 0.09 s. The banner ("Red to move") is a sheet without buttons; it stays 0.8 s for a turn and 1.6 s otherwise, then goes without a fade.

CONSTANT	VALUE	WHAT IT SETS	WHERE
ROUND	6	Everything in the interface is rounded a little, and by the same measure: a panel and the frame round it by one, a button inside it by a tighter one.	scripts/ui_kit.gd:12
ROUND_TIGHT	4	same as above	scripts/ui_kit.gd:13
SPACE	0.12	One measure spaces the whole interface -- the gap between two panels and the inset inside one are the same fraction of a board cell, everywhere.	scripts/ui_kit.gd:17
TEXT_MIN	0.36	The smallest writing the interface sets, as a fraction of a cell. A board of fourteen rows on a phone leaves a cell of about forty pixels, and a quarter of that is not writing anybody reads: what is written is sized off this floor as well as off the cell it stands in.	scripts/ui_kit.gd:28
FRAME	2.0	The frame round every panel: the same thickness wherever it is drawn, in the player's own colour when the panel is theirs to use and in `EDGE` when it is not.	scripts/ui_kit.gd:102
SHEET_TEXT	0.34	How tall a sheet's button is written, as a fraction of its height.	scripts/ui_kit.gd:267

Palette, art and type

Neon on near-black: plates are unlit circuitry, the ink is a pale cyan glow, and the one accent is an electric amber.

Palette

read from scripts/ui_kit.gd and config.json

Interface tokens

scripts/ui_kit.gd

Kit.TINT
#121a2b

Kit.TINT_LIT
#1e2c48

Kit.TINT_DIM
#0b1120

Kit.INK
#d8f6ff

Kit.INK_DIM
#6f8aa8

Kit.GOLD
#ffd60a

Kit.ICE
#6fcbff
The colour of what Frost lays over a row it holds.

Kit.CARVED
#060a12
The ink drawn over gold: near black.

Kit.EDGE
#1b2740
The frame round a panel that belongs to nobody in particular: a shade above the backing the whole interface stands on.

Kit.BACK
#0b1220
The frame round a panel that belongs to nobody in particular: a shade above the backing the whole interface stands on.

Kit.SHEET_EDGE
#3a4354
The frame round whatever stands open over the board -- a sheet, a banner, a card held open -- and round every button: one dark grey, whoever is to move.

Player colours

Kit.ZONE_CHOICES

#4a7fc2

#e04444

Mana colours

config.json · tileColors[] · light / flat / mid / dark

Plasma	red mana		#ff2e63
Ion	blue mana		#00e5ff
Flux	green mana		#2bff88
Photon	gold mana		#ffd60a
Hex	violet mana		#b14aff

TOKEN all of them. A mana colour has four tones: **flat** is the gem and everything on the board, **light**, **mid** and **dark** are kept for the interface.

Art

`art/*.svg`

Every icon the game has, as the game draws it: tinted one colour. The names say where they go: `i-` a mana colour's mark, `g-` a mark redrawn to sit in a gem, `s-` a spell, `u-` a unit, `t-` a term in the rules text.



`g-triangle.svg`
Flux mark inside a gem (board, logo, help)



`i-circle.svg`
Ion mark



`i-diamond.svg`
Photon mark, the body of every gem



`i-pentagon.svg`
Hex mark



`i-square.svg`
Plasma mark



`i-triangle.svg`
Flux mark



`s-bolt.svg`
spell Charge



`s-bullet.svg`
spell Killer



`s-crosshair.svg`
spell Strike



`s-fence.svg`
not drawn in the game



`s-frost.svg`
spell Frost



`s-hammer.svg`
spell Hammer



`s-hand.svg`
spell Collect



`s-ion.svg`
spell Storm



`s-node-shield.svg`
spell Load



`s-shield.svg`
spell Armor



`s-splice.svg`
spell Heal



`s-tape.svg`
spell Rewire



`s-tower-bolt.svg`
spell E-Shield



`t-ap.svg`
inline in rule text {t-ap}



`t-board.svg`
inline in rule text {t-board}



`t-charge.svg`
inline in rule text {t-charge}



`t-column.svg`
not drawn in the game



`t-damage.svg`
inline in rule text {t-damage}



`t-hp.svg`
inline in rule text {t-hp}



`t-influence.svg`
interface



`t-melee.svg`
not drawn in the game



`t-move.svg`
inline in rule text {t-move}



`t-node.svg`
not drawn in the game



`t-range.svg`
not drawn in the game



`t-ranged.svg`
not drawn in the game



`t-row.svg`
inline in rule text {t-row}



`t-time.svg`
not drawn in the game



`u-bomb.svg`
unit Bombarder



`u-imp.svg`
unit Ranger



`u-mage.svg`
not drawn in the game



`u-tower.svg`
unit Tower



`u-treant.svg`
not drawn in the game

ART replace any of them in place. They are drawn at many sizes, from 0.3 of a cell in the text to 1.8 cells on a card, so they should read at 16 px. Fine lines and inner detail will be lost; a solid silhouette is what works.

Type

The game ships no typeface. Everything in it is set in Godot's built-in default font, at sizes taken from the cell (never under `TEXT_MIN` of a cell). The faces in this document (Unbounded, Golos Text, JetBrains Mono) are the documents' own and are not in the game.

`CODE` a typeface for the game: a font file, loaded once into `Kit.theme()` and used wherever the code now asks for the fallback font.

Where to change what

Redraw an icon, a unit's glyph or a spell's picture Replace the SVG of the same name in `art/`. One colour, square, a solid shape.	ART
Recolour a mana type `config.json · tileColors[]`: flat, light, mid, dark.	TOKEN
Recolour the interface `ui_kit.gd`, and the copies in `side_panel.gd`, `detail_panel.gd`, `help_panel.gd`, `drag_card.gd`, `lobby.gd`.	TOKEN
Change the players' colours `Kit.ZONE` (the defaults) and `Kit.ZONE_CHOICES`.	TOKEN
Make gems bigger, glossier, more outlined `gem.gd`: BODY, GLOSS, OUTLINE, MARK.	TOKEN
Change how loud wires and charge are `wires_view.gd`: GRID, NODE, THICK, LIVE_THICK, JITTER, FLICKER, BLINK.	TOKEN
Change the frontier's look `frontier.gd`: BAR, RAIL, PIECE, WASH.	TOKEN
Speed up or slow down animations `config.json · timing`, and the effect lengths in `board_view.gd`.	TOKEN
A new shape for units, pads or towers `Kit.diamond`, `structures_view.gd · _pad_outline` and `SHAPE`.	CODE
Bitmaps, gradients, textures, shadows Nothing in the game draws one today; each layer would need it added.	CODE
A typeface See Type above.	CODE
A different layout of the rail or the screen `battle.gd · _layout()`, `side_panel.gd`.	CODE

Built from build 0.3.6 on 02 November 2026 by `docs/design_build.py`. Pictures: `docs/design_capture.gd`, one hot-seat battle on a fixed seed, Plasma Syndicate at the bottom, Flux Commune at the top.