UNITY 6000.0.62F1 · COMPLETE OFFLINE DOCUMENTATION

Hearts Table Developer Guide

Install. Edit. Reskin. Build your release.

szekipapa77@gmail.com



CHAPTER 01

Start here: Hearts Table

Hearts Table is a standalone editable Unity game template. Play Hearts against three local computer opponents. Select legal cards while following suit, manage passing and avoid penalty points. Tables introduce rule variations and target scores; consult each table rules before attempting a moon strategy. Lowest-scoring play is the normal objective, and optional variants are stored in the table data.

Includes 12 curated tables, source-backed generation, a game-specific visual editor and validation, 4 guided helpers, local saving and portrait/landscape scenes. Keep the required private game Kit and publisher-owned Szekipapa77Ads layer. Provider SDK binaries are excluded. There is no bundled HLK, purchases, multiplayer server or remote LiveOps backend. Import only one game Kit into a project because the Kit classes share names.

CHAPTER 02

Requirements, installation and first play

Use Unity **6000.0.62f1**, URP **17.0.4** and Unity UI **2.0.0**. Install matching Android SDK/NDK/OpenJDK and WebGL modules in Unity Hub. Retain the supplied manifest and ProjectSettings.

1. Extract the standalone project to a short writable local path, then open it in the matching editor. Wait for package import and compilation.
2. For native unitypackage import, create a matching URP project and use Assets > Import Package > Custom Package. Import every game, Kit, Szekipapa77Ads and WebGL-template entry.
3. Run Szekipapa77 > Hearts Table > Setup. It configures rendering and player settings; review its changes when integrating with existing work.
4. Active Input Handling must support the classic Input Manager: choose Old or Both. New Input System only is insufficient.
5. Open `Assets/Games/HeartsTable/Scenes/HeartsTable_Mobile.unity` and press Play. Follow the level card and first-level instructions.
6. Stop Play mode before editing level assets. Keep all .meta GUIDs. The game uses classic Unity UI Text; do not assume an extra TextMesh Pro import is required.

CHAPTER 03

Project folders and exact source map

All game-relative paths below start at `Assets/Games/HeartsTable`. Runtime UI is built in code; editing an unrelated static Canvas does not reskin it. Scripts contains game rules and views; Editor contains authoring tools; Kit supplies UI/audio/save/platform support; Resources/heartstable contains level JSON, art, music and sound; Settings owns URP assets; Scenes contains both entry scenes.

- `Scripts/HeartsAI.cs`
- `Scripts/HeartsArt.cs`
- `Scripts/HeartsData.cs`
- `Scripts/HeartsHelpers.cs`
- `Scripts/HeartsLogic.cs`
- `Scripts/HeartsTableGame.cs`
- `Scripts/HeartsTables.cs`
- `Scripts/HeartsTableScreens.cs`

Main flow: HeartsTableGame.cs. UI and popup consumers: HeartsTableGame.cs. Model/generator/loading: HeartsData.cs, HeartsLogic.cs, HeartsTables.cs. Authoring: Editor/HeartsTableLevelEditor.cs. Shared ad settings live outside the game at Assets/Szekipapa77Ads/Resources/AdsSettings.asset.

CHAPTER 04

Gameplay, controls and progression

Play Hearts against three local computer opponents. Select legal cards while following suit, manage passing and avoid penalty points. Tables introduce rule variations and target scores; consult each table rules before attempting a moon strategy. Lowest-scoring play is the normal objective, and optional variants are stored in the table data.

The initial bank has 12 curated tables. Read the supplied loader/generator before changing continuation rules. Play the guided opening, ordinary progress, completion and retry paths; preserve saved indexes during a visual reskin. The Mobile scene is portrait and the WebGL-named scene is landscape, but the supplied Build menu selects portrait for both build targets. Select landscape explicitly in your Build Profile if desired. Test a near-startup scene replacement as well as a normal restart.

CHAPTER 05

Tutorials and every helper

Fresh local progress triggers the first-level guidance. Newly unlocked helpers have introduction and guided-use flows. Reset only a test save when reviewing tutorials. A free guided first use can differ from ordinary charge consumption. Exact names/unlocks/free charges: Scripts/HeartsTables.cs; effect/help strings: Scripts/HeartsTables.cs.

Helper	Unlock	Free charges	Action and effect
Card Hint	2	3	Highlights the safest card to play right now.
Pass Advisor	3	3	Picks the 3 best cards to pass for you.
Undo	5	3	Takes back your last card (and everything played after it).
Peek	7	3	Shows which suits each opponent has run out of, for the rest of the hand.

Activate the tool and perform its required target action: arming a tool alone does not demonstrate its effect. A successful use requires the stated preconditions from this game: show both activation and its actual effect. Exhausted charges may offer a rewarded ad; verify success/cancel/failure and grant a reward once only.

CHAPTER 06

Visual level editor: worked editing session

Open **Szekipapa77 > Hearts Table > Level Editor**. Exact implementation: `Editor/HeartsTableLevelEditor.cs`.

Open the Hearts Table editor and select an early table. Use Dup or Duplicate before editing. Change its visible name and one rule or board element with the exact controls below. Save explicitly and run the current entry through the editor validation and play controls. The per-game source-control table includes the current brush, geometry, target, rule and AI fields; do not treat hierarchy or resource IDs as display text.

1. Back up Resources/heartstable/Levels/levels.json.
2. Duplicate a simple early level and edit one small element.
3. Use the game-specific validation controls and resolve every reported issue.
4. Save explicitly; stop and restart Play mode, then use Play this level or navigate to the edited index.
5. Check portrait and landscape and rebuild demo outputs after accepting data changes.

The exact current authoring controls, including tooltips and generator switches, follow. These are game-specific source references, not a generic editor feature list.

Source	Exact current code
`HeartsTableLevelEditor.cs:38`	`[MenuItem("Szekipapa77/Hearts Table/Level Editor")]`
`HeartsTableLevelEditor.cs:41`	`[MenuItem("Szekipapa77/Hearts Table/Validate Tables")]`
`HeartsTableLevelEditor.cs:109`	`if (GUILayout.Button(_dirty ? "Save *" : "Save", EditorStyles.toolbarButton, GUILayout.Width(70))) Save();`
`HeartsTableLevelEditor.cs:110`	`if (GUILayout.Button("Reload", EditorStyles.toolbarButton, GUILayout.Width(60)) && (!_dirty || EditorUtility.DisplayDialog("Reload", "Discard unsaved changes?", "Discard", "Cancel"))) Load();`
`HeartsTableLevelEditor.cs:115`	`if (GUILayout.Button("Validate table", EditorStyles.toolbarButton, GUILayout.Width(95))) ValidateOne(_data.selected);`
`HeartsTableLevelEditor.cs:116`	`if (GUILayout.Button("Validate all", EditorStyles.toolbarButton, GUILayout.Width(85))) ValidateAll();`

`HeartsTableLevelEditor.cs:119`	`if (GUILayout.Button("Play this table", EditorStyles.toolbarButton, GUILayout.Width(100))) PlayTable();`
`HeartsTableLevelEditor.cs:148`	`if (GUILayout.Button("Add")) { Rec("Add table"); int at = Mathf.Min(_data.selected + 1, File.tables.Count); File.tables.Insert(at, HtGenerator.Make(at + 1, 0.4f, Environment.TickCount & 0xffff)); _data.selected = at; _results.Clear(); }`
`HeartsTableLevelEditor.cs:150`	`if (GUILayout.Button("Dup")) { Rec("Duplicate table"); var c = Cur.Clone(); c.name += " copy"; File.tables.Insert(_data.selected + 1, c); _data.selected++; _results.Clear(); }`
`HeartsTableLevelEditor.cs:151`	`if (GUILayout.Button("Del") && EditorUtility.DisplayDialog("Delete table", $"Delete table {_data.selected + 1} ({Cur.name})?", "Delete", "Cancel")) { Rec("Delete table"); File.tables.RemoveAt(_data.selected); _data.selected = Mathf.Clamp(_data.selected, 0, File.tables.Count - 1); _results.Clear(); }`
`HeartsTableLevelEditor.cs:152`	`if (GUILayout.Button("Up") && _data.selected > 0) { Rec("Move table"); int s = _data.selected; (File.tables[s], File.tables[s - 1]) = (File.tables[s - 1], File.tables[s]); _data.selected--; _results.Clear(); }`
`HeartsTableLevelEditor.cs:153`	`if (GUILayout.Button("Down") && _data.selected < File.tables.Count - 1) { Rec("Move table"); int s = _data.selected; (File.tables[s], File.tables[s + 1]) = (File.tables[s + 1], File.tables[s]); _data.selected++; _results.Clear(); }`
`HeartsTableLevelEditor.cs:157`	`if (_results.TryGetValue(_data.selected, out var line)) EditorGUILayout.HelpBox(line, line.StartsWith("ERR") ? MessageType.Error : MessageType.Info);`
`HeartsTableLevelEditor.cs:299`	`string name = EditorGUILayout.TextField("Name", t.name);`
`HeartsTableLevelEditor.cs:300`	`string venue = EditorGUILayout.TextField("Subtitle", t.venue);`
`HeartsTableLevelEditor.cs:301`	`int target = EditorGUILayout.IntSlider("Target score", t.target, 25, 200);`

HeartsTableLevelEditor.cs:303	`ui = EditorGUILayout.Popup("Unlocks helper", ui, UnlockNames);`
HeartsTableLevelEditor.cs:304	`Color felt = EditorGUILayout.ColorField("Felt", HeartsArt_Color(t.felt));`
HeartsTableLevelEditor.cs:305	`Color accent = EditorGUILayout.ColorField("Accent", HeartsArt_Color(t.accent));`
HeartsTableLevelEditor.cs:306	`string note = EditorGUILayout.TextField("Designer note", t.note);`
HeartsTableLevelEditor.cs:308	`bool np = EditorGUILayout.Toggle("No passing", rules.noPassing);`
HeartsTableLevelEditor.cs:309	`bool jd = EditorGUILayout.Toggle("Jack of diamonds -10", rules.jackDiamonds);`
HeartsTableLevelEditor.cs:310	`bool qb = EditorGUILayout.Toggle("Queen breaks hearts", rules.queenBreaks);`
HeartsTableLevelEditor.cs:311	`bool bf = EditorGUILayout.Toggle("Points on trick 1", rules.bloodFirst);`
HeartsTableLevelEditor.cs:312	`bool mm = EditorGUILayout.Toggle("Moon subtracts 26", rules.moonMinus);`
HeartsTableLevelEditor.cs:322	`EditorGUILayout.LabelField("AI strength (avg)", t.Strength.ToString("0.00"));`
HeartsTableLevelEditor.cs:339	`string name = EditorGUILayout.TextField("Name", o.name);`
HeartsTableLevelEditor.cs:340	`float skill = EditorGUILayout.Slider(new GUIContent("Skill", "How often it plays its best idea"), o.skill, 0, 1);`
HeartsTableLevelEditor.cs:341	`float search = EditorGUILayout.Slider(new GUIContent("Look-ahead", "Monte-Carlo search strength (0 = heuristics only)"), o.search, 0, 1);`
HeartsTableLevelEditor.cs:342	`float memory = EditorGUILayout.Slider(new GUIContent("Card memory", "Uses shown voids and passed cards"), o.memory, 0, 1);`

HeartsTableLevelEditor.cs:343	float moon = EditorGUILayout.Slider(new GUIContent("Moon appetite", "Tendency to shoot the moon"), o.moon, 0, 1);
HeartsTableLevelEditor.cs:344	float defense = EditorGUILayout.Slider(new GUIContent("Moon defence", "How reliably it stops a moon run"), o.defense, 0, 1);
HeartsTableLevelEditor.cs:345	int pass = EditorGUILayout.Popup("Pass strategy", Mathf.Clamp(o.pass, 0, 3), PassNames);
HeartsTableLevelEditor.cs:346	int avatar = EditorGUILayout.IntSlider("Avatar", o.avatar, 0, HtGenerator.AvatarCount - 1);
HeartsTableLevelEditor.cs:353	if (GUILayout.Button("Casual")) Preset(o, 0.15f);
HeartsTableLevelEditor.cs:354	if (GUILayout.Button("Steady")) Preset(o, 0.4f);
HeartsTableLevelEditor.cs:355	if (GUILayout.Button("Sharp")) Preset(o, 0.65f);
HeartsTableLevelEditor.cs:356	if (GUILayout.Button("Shark")) Preset(o, 0.95f);
HeartsTableLevelEditor.cs:358	EditorGUILayout.LabelField("Strength", o.Strength.ToString("0.00"));
HeartsTableLevelEditor.cs:427	_genCount = EditorGUILayout.IntSlider("Count", _genCount, 1, 40);
HeartsTableLevelEditor.cs:428	_genReplace = EditorGUILayout.Toggle("Replace range", _genReplace);
HeartsTableLevelEditor.cs:429	_genStart = EditorGUILayout.IntField(_genReplace ? "Start table (1-based)" : "Ladder number of first", _genStart);
HeartsTableLevelEditor.cs:430	_genSeed = EditorGUILayout.IntField("Seed", _genSeed);
HeartsTableLevelEditor.cs:431	_dStart = EditorGUILayout.Slider("Start difficulty", _dStart, 0, 1);
HeartsTableLevelEditor.cs:432	_dEnd = EditorGUILayout.Slider("End difficulty", _dEnd, 0, 1);

`HeartsTableLevelEditor.cs:433`	`_dSaw = EditorGUILayout.Slider("Sawtooth climb", _dSaw, 0, 0.2f);`
`HeartsTableLevelEditor.cs:434`	`_genBreather = EditorGUILayout.IntSlider("Breather every", _genBreather, 2, 10);`
`HeartsTableLevelEditor.cs:435`	`_dDrop = EditorGUILayout.Slider("Breather drop", _dDrop, 0, 0.5f);`
`HeartsTableLevelEditor.cs:436`	`_genValidate = EditorGUILayout.Toggle("Validate generated", _genValidate);`
`HeartsTableLevelEditor.cs:437`	`if (_genValidate) _genMatches = EditorGUILayout.IntSlider("Matches per table", _genMatches, 4, 60);`
`HeartsTableLevelEditor.cs:447`	`if (GUILayout.Button(_genReplace ? $"Generate and replace tables {_genStart}-{_genStart + _genCount - 1}" : $"Generate and append {_genCount} tables")) Generate();`

CHAPTER 07

Generator, insertion and replacement

Use the actual generation panel and its fields listed below. Set a reproducible seed and a small count, then review the start/index and replace option before clicking Generate. The exact generation implementation is reproduced separately with line references so append, insert and replacement behavior can be distinguished. Save and reopen the data file to verify the intended entry count; runtime generated continuation is not a project-asset save.

Source	Exact current code
`HeartsTableLevelEditor.cs:29`	`bool _genReplace, _genValidate = true;`
`HeartsTableLevelEditor.cs:428`	`_genReplace = EditorGUILayout.Toggle("Replace range", _genReplace);`
`HeartsTableLevelEditor.cs:429`	`_genStart = EditorGUILayout.IntField(_genReplace ? "Start table (1-based)" : "Ladder number of first", _genStart);`
`HeartsTableLevelEditor.cs:447`	`if (GUILayout.Button(_genReplace ? $"Generate and replace tables {_genStart}-{_genStart + _genCount - 1}" : $"Generate and append {_genCount} tables")) Generate();`
`HeartsTableLevelEditor.cs:459`	`int index = (_genReplace ? _genStart : File.tables.Count + 1) + i;`
`HeartsTableLevelEditor.cs:472`	`if (_genReplace)`
`HeartsTableLevelEditor.cs:476`	`File.tables.RemoveRange(start, remove);`
`HeartsTableLevelEditor.cs:477`	`File.tables.InsertRange(start, made);`
`HeartsTableLevelEditor.cs:480`	`else { File.tables.AddRange(made); _data.selected = File.tables.Count - made.Count; }`

Worked extension: back up the bank, generate a small set at moderate difficulty using the exact per-game controls, validate them, then Save. Reopen the JSON and confirm the count changed. Do not assume every editor has the same insertion behavior: the actual implementation below distinguishes insert, append and replace. Runtime continuation does not automatically append new records to project JSON. Validation and bot scores help find problems but do not certify a pleasant human difficulty curve.

CHAPTER 08

Actual level JSON and identifiers

Canonical asset: `Assets/Games/HeartsTable/Resources/heartstable/Levels/levels.json`. Array order controls visible numbering. Preserve numeric/boolean types, root version fields where present and resource/save identifiers. The actual first record is shown below; use the editor for structured coordinates and block/word data.


```
{
  "name": "The Front Porch",
  "venue": "Warm-up with the neighbours",
  "felt": "#1F5C3A",
  "accent": "#F4C15D",
  "target": 50,
  "rules": {
    "noPassing": false,
    "jackDiamonds": false,
    "queenBreaks": false,
    "bloodFirst": false,
    "moonMinus": false
  },
  "unlock": "",
  "note": "intro: classic rules, short match",
  "opponents": [
    {
      "name": "Mira",
      "avatar": 0,
      "skill": 0.35,
      "search": 0,
      "memory": 0.1,
      "moon": 0.05,
      "defense": 0.1,
      "pass": 0
    },
    {
      "name": "Felix",
      "avatar": 3,
      "skill": 0.4,
      "search": 0,
      "memory": 0.15,
      "moon": 0.05,
      "defense": 0.15,
      "pass": 0
    },
    {
      "name": "Theo",
      "avatar": 9,
      "skill": 0.38,
      "search": 0,
      "memory": 0.1,
      "moon": 0.05,
      "defense": 0.1,
      "pass": 1
    }
  ],
  "difficulty": 0.6,
  "winRate": 0.52
}
```


Root keys: version, tables. Inspect HeartsData.cs, HeartsLogic.cs, HeartsTables.cs for field semantics and validation. Keep visible names/hints distinct from technical keys. The game-specific board, wave, card, opponent or question records are gameplay data; UI translation is separate.

CHAPTER 09

Reskin recipe: interface colours

Stop Play mode and open `Scripts/HeartsArt.cs`. Find `Ink`, currently `#1E2933`. Change that hex literal to `#42C9B0`, preserve its symbol/type, compile and reopen the scene. Review HUD, menus, helper cards, settings/results and both orientations. Art sprites and procedural fallback shapes can have additional colours; edit them deliberately.

Actual consumers of this symbol are listed first, then all current palette/hex assignments. Editor-only palettes do not control the runtime. Preserve state/goal contrast and rebuild after accepting a change.

Source	Exact current code
`HeartsArt.cs:10 / class/static declarations`	`public static readonly Color Red = Hex("#E0413A"), Ink = Hex("#1E2933"), Cream = Hex("#FFF9EF"), Coral = Hex("#FF6B5B"),`
`HeartsArt.cs:40 / public static Color SuitColor(int suit) => suit == Cards.Hearts || suit == Cards.Diamonds ? Red : Ink;`	`public static Color SuitColor(int suit) => suit == Cards.Hearts || suit == Cards.Diamonds ? Red : Ink;`
`HeartsArt.cs:213 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`if ((u - 0.15f) * (u - 0.15f) + ey * ey < 0.0032f || (u + 0.15f) * (u + 0.15f) + ey * ey < 0.0032f) col = Ink;`
`HeartsArt.cs:218 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`if (v < -0.04f && Mathf.Abs(sm - 0.16f) < 0.022f && Mathf.Abs(u) < 0.12f) col = p.style == 3 ? Cream : Color.Lerp(Ink, Red, 0.5f);`
`HeartsTableGame.cs:27 / class/static declarations`	`static readonly Color Coral = HeartsArt.Coral, Gold = HeartsArt.Gold, Ink = HeartsArt.Ink, Cream = HeartsArt.Cream;`

Source	Exact current code
`HeartsArt.cs:10 / class/static declarations`	`public static readonly Color Red = Hex("#E0413A"), Ink = Hex("#1E2933"), Cream = Hex("#FFF9EF"), Coral = Hex("#FF6B5B"),`
`HeartsArt.cs:11 / class/static declarations`	`Gold = Hex("#F4C15D"), FeltTop = Hex("#11605F"), FeltBot = Hex("#06292D"), Teal = Hex("#0E4A50"), Night = Hex("#0A1F24");`
`HeartsArt.cs:168 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Mira", "#7E57C2", "#F2C6A0", "#3B2418", 1),`

`HeartsArt.cs:169 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Otto", "#26A69A", "#E8B48E", "#7A4A24", 0),`
`HeartsArt.cs:170 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Juno", "#EF6C57", "#C98D68", "#1D1A1A", 2),`
`HeartsArt.cs:171 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Felix", "#5C8DD6", "#F5D2B5", "#D9A441", 0),`
`HeartsArt.cs:172 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Rosa", "#D84E8B", "#A86B4C", "#2B1810", 1),`
`HeartsArt.cs:173 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Hugo", "#8D9E3A", "#F0C4A4", "#9A9A9A", 3),`
`HeartsArt.cs:174 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Iris", "#3FA3C4", "#F6D8C2", "#C2562E", 2),`
`HeartsArt.cs:175 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Bruno", "#B5763A", "#7E5038", "#141010", 3),`
`HeartsArt.cs:176 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Nadia", "#9C5BD1", "#D9A27E", "#4A2A1A", 1),`
`HeartsArt.cs:177 / static float RoundRectDist(float px, float py, float w, float h, float r)`	`P("Theo", "#4CAF6E", "#EDC3A2", "#5A3A22", 0),`
`HeartsHelpers.cs:19 / class/static declarations`	`static readonly Color[] HelperColor = { HzUI.Hex("#F5A623"), HzUI.Hex("#4C9EEB"), HzUI.Hex("#7B61FF"), HzUI.Hex("#3CC3A0") };`
`HeartsHelpers.cs:64 / void BuildHelperBar()`	`var bd = HzUI.Panel("Badge", h.btn.transform, HzUI.Hex("#FF4D4D"), 24);`
`HeartsHelpers.cs:72 / void BuildHelperBar()`	`var li = HzIcons.Icon(lk.transform, HzIcons.Lock, (int)(sz * 0.4f), HzUI.Hex("#D8E3E6")); li.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(sz, sz * 0.5f), new Vector2(0, 12));`

HeartsHelpers.cs:118 / void RefreshHelpers()	h.badge.GetComponent<Image>().color = ch > 0 ? HzUI.Hex("#FF4D4D") : HzUI.Hex("#3CC3A0");
HeartsHelpers.cs:222 / void OfferCharge(string id)	var m = Modal("Out of " + HtHelpers.Title(id), new Vector2(820, 640), HzUI.Hex("#4C9EEB"));
HeartsHelpers.cs:274 / void ShowUnlock(string id, Action done)	var m = Modal("New helper!", new Vector2(860, 820), HzUI.Hex("#F5A623"));
HeartsHelpers.cs:284 / void ShowUnlock(string id, Action done)	CardText(m, "+" + HtHelpers.FreeCharges + " free uses", 575, 600, 50, 34, HzUI.Hex("#D99A1E"), true);
HeartsTableGame.cs:28 / class/static declarations	static readonly Color Mint = HzUI.Hex("#3CC3A0"), Plum = HzUI.Hex("#5B4B8A"), Slate = HzUI.Hex("#2E5560");
HeartsTableGame.cs:263 / void BuildStatic()	p.total = HzUI.Label("Total", p.rt, "", 26, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);
HeartsTableGame.cs:275 / void BuildStatic()	p.total = HzUI.Label("Total", p.rt, "", 26, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);
HeartsTableGame.cs:837 / void HeartsBrokenFx(bool byQueen)	Msg(byQueen ? "The queen broke hearts!" : "Hearts are broken!", 1.5f, HzUI.Hex("#FFB3AA"));
HeartsTableGame.cs:862 / void OnCardClick(int c)	Msg(why, 2f, HzUI.Hex("#FFD27A"));
HeartsTableGame.cs:875 / void OnPassClick()	if (_passSel.Count < 3) { HzAudio.Play(HzSfx.Wrong, 0.4f); Msg("Pick exactly 3 cards to pass", 1.6f, HzUI.Hex("#FFD27A")); HzTween.Shake(_passBtn.transform, 12, 0.3f); return; }
HeartsTableGame.cs:889 / void RefreshPassButton()	_passBtn.GetComponent<Image>().color = _passSel.Count == 3 ? Coral : HzUI.Hex("#7A5450");
HeartsTables.cs:12 / class/static declarations	public string felt = "#11605F"; // table theme (felt colour)
HeartsTables.cs:13 / class/static declarations	public string accent = "#F4C15D"; // rim / highlight colour

HeartsTables.cs:83 / public static string FirstUse(string id) => id == "hint" ? "Your turn: tap Card Hint to see the safest card"	static readonly string[] Felts = { "#11605F", "#1F5C3A", "#3A2F6B", "#6B2F3A", "#1D4E6E", "#4E3B1D", "#2F5A55", "#5A2B5E", "#23485A", "#3E5A2B", "#5E3A2B", "#2B3E5E" };
HeartsTables.cs:84 / public static string FirstUse(string id) => id == "hint" ? "Your turn: tap Card Hint to see the safest card"	static readonly string[] Accents = { "#F4C15D", "#FF8A6B", "#9AD7FF", "#C7A6FF", "#7FE0B0", "#FFD27A" };
HeartsTableScreens.cs:18 / class/static declarations	static readonly Color TextDark = HzUI.Hex("#24363D"), TextSub = HzUI.Hex("#5F7A80");
HeartsTableScreens.cs:28 / class/static declarations	var card = HzUI.Panel("Card", dim.transform, HzUI.Hex("#FFF8EE"), 48);
HeartsTableScreens.cs:109 / void ShowHandResult(int[] scores, int moon)	var st = HzIcons.Icon(m.card, HzIcons.Star, 120, i < _lastStars ? HzUI.Hex("#F5B82E") : HzUI.Hex("#D9CFC0"));
HeartsTableScreens.cs:129 / void ShowHandResult(int[] scores, int moon)	var row = HzUI.Panel("Row", m.card, s == 0 ? HzUI.Hex("#E7F4F1") : HzUI.Hex("#F3ECE2"), 26);
HeartsTableScreens.cs:132 / void ShowHandResult(int[] scores, int moon)	CardText(m, (i + 1).ToString(), y, 60, 60, 44, i == 0 ? HzUI.Hex("#D99A1E") : TextSub, true, -350);
HeartsTableScreens.cs:150 / void ShowHandResult(int[] scores, int moon)	CardText(m, reward, by + 15, 800, 50, 30, HzUI.Hex("#D99A1E"));
HeartsTableScreens.cs:190 / void ShowScoreSheet()	if (i % 2 == 0) { var band = HzUI.Panel("Band", m.card, HzUI.Hex("#F1E9DD"), 16); band.raycastTarget = false; CardAt(m, band.rectTransform, y, 820, 60); }
HeartsTableScreens.cs:196 / void ShowScoreSheet()	CardText(m, v.ToString(), y, 140, 56, 36, v <= 0 ? (moonRow && M.moons[hand] == s ? HzUI.Hex("#D99A1E") : Mint) : Coral, true, xs[s + 1]);
HeartsTableScreens.cs:201 / void ShowScoreSheet()	var line = HzUI.Panel("Line", m.card, HzUI.Hex("#D9CBB6"), 2); line.raycastTarget = false; CardAt(m, line.rectTransform, ty - 20, 820, 4);
HeartsTableScreens.cs:217 / void ShowRules()	"Avoid points. Every <color=#E0413A>heart</color> is 1 point, the queen of spades is 13.",

HeartsTableScreens.cs:230 / void ShowRules()	CardText(m, lines[i], y + 46, 800, 112, 32, i == 7 ? HzUI.Hex("#7B3FA0") : TextDark, false, 30, TextAnchor.MiddleLeft);
HeartsTableScreens.cs:243 / void ShowMenu()	sound = CardButton(m, SoundLabel(), HzUI.Hex("#4C9EEB"), 315, 262, 96, null, 36, -139);
HeartsTableScreens.cs:245 / void ShowMenu()	music = CardButton(m, MusicLabel(), HzUI.Hex("#4C9EEB"), 315, 262, 96, null, 36, 139);
HeartsTableScreens.cs:249 / void ShowMenu()	CardButton(m, "Restart table", HzUI.Hex("#F5A623"), 675, 540, 100, () => { CloseAllModals(); StopHand(false); HzPlatform.Interstitial(GameId, () => StartTable(M != null ? M.table : Level)); }, 44);
HeartsTableScreens.cs:250 / void ShowMenu()	CardButton(m, "Main menu", HzUI.Hex("#7B61FF"), 795, 540, 100, () => { CloseAllModals(); StopHand(); ShowTitle(); }, 44);
HeartsTableScreens.cs:324 / void ShowTitle()	var tv = HzUI.Label("TVenue", tc.transform, table.venue + " - to " + table.target, 28, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);
HeartsTableScreens.cs:381 / void ShowTitle()	var si = HzIcons.Icon(chip.transform, HzIcons.Star, 60, HzUI.Hex("#F5B82E")); si.rectTransform.Place(new Vector2(0, 0.5f), new Vector2(70, 70), new Vector2(50, 0));
HeartsTableScreens.cs:414 / void ShowLadder()	var row = HzUI.Button("Row" + n, content, null, locked ? HzUI.Hex("#E3DDD3") : HzUI.Hex("#F3ECE2"), Color.white, () =>
HeartsTableScreens.cs:421 / void ShowLadder()	var badge = HzUI.Panel("Num", row.transform, locked ? HzUI.Hex("#9AA3A6") : HeartsArt.FeltOf(t), 40); badge.raycastTarget = false;
HeartsTableScreens.cs:430 / void ShowLadder()	var il = HzUI.Label("Info", row.transform, info, 24, tw.Count > 0 ? HzUI.Hex("#7B3FA0") : TextSub, TextAnchor.MiddleLeft, false);
HeartsTableScreens.cs:437 / void ShowLadder()	var ul = HzUI.Label("Unlock", row.transform, unlock, 24, HzUI.Hex("#D98A1E"), TextAnchor.MiddleLeft, false);

`HeartsTableScreens.cs:455 / void ShowLadder()`	`var s = HzIcons.Icon(row.transform, HzIcons.Star, 44, i < stars ? HzUI.Hex("#F5B82E") : HzUI.Hex("#D3C8B8"));`
`HeartsTableScreens.cs:497 / void ShowStats()`	`if (i % 2 == 0) { var band = HzUI.Panel("Band", m.card, HzUI.Hex("#F1E9DD"), 18); band.raycastTarget = false; CardAt(m, band.rectTransform, y + 40, 800, 78); }`

CHAPTER 10

Reskin recipe: visible menu and HUD text

Open `Scripts/HeartsTableGame.cs`, find the displayed literal `"Hearts Table"`, and replace its displayed value with `"Hearts Table Edition"`. Keep hierarchy names and technical IDs unchanged. In HzUI.Label/Button calls the first string may be an object name; change the displayed text argument. Read the method references below to distinguish construction from later refresh assignments.

Preserve interpolation braces, invariant number formatting, ON/OFF updates and line breaks. Update both initial and refresh paths, portrait and landscape. Exact names/unlocks/free charges: Scripts/HeartsTables.cs; effect/help strings: Scripts/HeartsTables.cs. Level names/hints remain separate JSON content. If a title appears in art, Title property or another screen, update those visible uses deliberately without renaming GameId, namespace, resources or save keys.

Hard-coded game strings do not automatically use shared localization. To translate the interface, route every display assignment through your translation table. Dictionaries, word banks and theme pools are independent gameplay assets and need separate validation. Test help, settings, results, tutorial and helper introductions after a fresh launch.

Source	Exact current code
`HeartsArt.cs:34 / public static Color Hex(string h) { return ColorUtility.TryParseHtmlString(h != null && h.StartsWith("#") ? h : "#" + h, out var c) ? c : FeltTop; }`	`public static Sprite HelperIcon(string id) => Load("Icons/icon_" + id);`
`HeartsData.cs:19 / class/static declarations`	`try { _file = ta ? JsonUtility.FromJson<HtTableFile>(ta.text) : null; } catch (Exception e) { Debug.LogWarning("[HeartsTable] levels.json: " + e.Message); }`
`HeartsHelpers.cs:19 / class/static declarations`	`static readonly Color[] HelperColor = { HzUI.Hex("#F5A623"), HzUI.Hex("#4C9EEB"), HzUI.Hex("#7B61FF"), HzUI.Hex("#3CC3A0") };`
`HeartsHelpers.cs:44 / void BuildHelperBar()`	`h.btn = HzUI.Button("Helper_" + id, _top, null, HelperColor[i], Color.white, () => UseHelper(id), 40, (int)(sz * 0.3f));`
`HeartsHelpers.cs:45 / void BuildHelperBar()`	`h.btn.name = "Helper_" + id;`
`HeartsHelpers.cs:60 / void BuildHelperBar()`	`h.label = HzUI.Label("Name", h.btn.transform, HtHelpers.Title(id), 24, Color.white, TextAnchor.MiddleCenter, false);`
`HeartsHelpers.cs:67 / void BuildHelperBar()`	`h.badgeText = HzUI.Label("BadgeText", bd.transform, "", (int)(sz * 0.25f), Color.white); h.badgeText.rectTransform.Stretch();`

`HeartsHelpers.cs:73 / void BuildHelperBar()`	`h.lockText = HzUI.Label("LockText", lk.transform, "", 24, Gold, TextAnchor.MiddleCenter, false); h.lockText.fontStyle = FontStyle.Bold;`
`HeartsHelpers.cs:110 / void RefreshHelpers()`	`h.lockText.text = at > 0 ? "Table " + at : "Soon";`
`HeartsHelpers.cs:117 / void RefreshHelpers()`	`h.badgeText.text = ch > 0 ? ch.ToString() : "+";`
`HeartsHelpers.cs:133 / void UseHelper(string id)`	`Toast(HtHelpers.Title(id) + (at > 0 ? " unlocks at table " + at : " is locked"));`
`HeartsHelpers.cs:140 / void UseHelper(string id)`	`Toast(id == "hint" ? "Use Card Hint on your turn" : id == "pass" ? "Use Pass Advisor while passing" : id == "undo" ? "Nothing to undo yet" : _peek ? "Peek is already on" : "Use Peek during a hand");`
`HeartsHelpers.cs:222 / void OfferCharge(string id)`	`var m = Modal("Out of " + HtHelpers.Title(id), new Vector2(820, 640), HzUI.Hex("#4C9EEB"));`
`HeartsHelpers.cs:226 / void OfferCharge(string id)`	`CardText(m, "Watch a short ad for one more " + HtHelpers.Title(id) + ".\nWinning a table also refills every helper by 1.", 360, 720, 120, 32, TextDark);`
`HeartsHelpers.cs:227 / void OfferCharge(string id)`	`CardButton(m, "Watch ad +1", Mint, 470, 480, 104, () =>`
`HeartsHelpers.cs:235 / void OfferCharge(string id)`	`Toast("+1 " + HtHelpers.Title(id), Gold);`
`HeartsHelpers.cs:239 / void OfferCharge(string id)`	`CardButton(m, "No thanks", Slate, 585, 360, 80, () => CloseModal(m), 36);`
`HeartsHelpers.cs:284 / void ShowUnlock(string id, Action done)`	`CardText(m, "+" + HtHelpers.FreeCharges + " free uses", 575, 600, 50, 34, HzUI.Hex("#D99A1E"), true);`
`HeartsHelpers.cs:285 / void ShowUnlock(string id, Action done)`	`CardButton(m, "Great!", Mint, 690, 420, 110, () => { CloseModal(m); done(); }, 52);`
`HeartsHelpers.cs:297 / void ShowNewRule(string title, string text, Action done)`	`CardButton(m, "Got it", Mint, 520, 400, 110, () => { CloseModal(m); done(); }, 52);`
`HeartsHelpers.cs:315 / void ShowTableIntro(Action done)`	`CardButton(m, "Deal!", Mint, 662, 400, 100, () => { CloseModal(m); done(); }, 50);`

HeartsTableGame.cs:203 / void BuildStatic()	`_handLabel = HzUI.Label("Hand", chip.transform, "", 36, Color.white, TextAnchor.MiddleLeft);`
HeartsTableGame.cs:206 / void BuildStatic()	`_dirLabel = HzUI.Label("Dir", chip.transform, "", 26, Gold, TextAnchor.MiddleLeft, false); _dirLabel.fontStyle = FontStyle.Bold;`
HeartsTableGame.cs:224 / void BuildStatic()	`_msgText = HzUI.Label("T", _msgBg.transform, "", 34, Color.white, TextAnchor.MiddleCenter, false); _msgText.fontStyle = FontStyle.Bold;`
HeartsTableGame.cs:229 / void BuildStatic()	`_passBtn = HzUI.Button("Pass", _top, "", Coral, Color.white, OnPassClick, 40, 60);`
HeartsTableGame.cs:235 / void BuildStatic()	`_passLabel = HzUI.Label("L", _passBtn.transform, "PASS", 40, Color.white); _passLabel.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(220, 50), new Vector2(0, -60));`
HeartsTableGame.cs:257 / void BuildStatic()	`p.name = HzUI.Label("Name", p.rt, NameOf(seat), 34, Color.white);`
HeartsTableGame.cs:261 / void BuildStatic()	`p.hand = HzUI.Label("Hand", p.rt, "0", 32, Coral, TextAnchor.MiddleLeft);`
HeartsTableGame.cs:263 / void BuildStatic()	`p.total = HzUI.Label("Total", p.rt, "", 26, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);`
HeartsTableGame.cs:269 / void BuildStatic()	`p.name = HzUI.Label("Name", p.rt, NameOf(seat), 36, Color.white, TextAnchor.MiddleLeft);`
HeartsTableGame.cs:273 / void BuildStatic()	`p.hand = HzUI.Label("Hand", p.rt, "0", 32, Coral, TextAnchor.MiddleLeft);`
HeartsTableGame.cs:275 / void BuildStatic()	`p.total = HzUI.Label("Total", p.rt, "", 26, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);`
HeartsTableGame.cs:312 / void RefreshPlates()	`p.hand.text = S != null ? S.points[s].ToString() : "0";`
HeartsTableGame.cs:313 / void RefreshPlates()	`p.total.text = "Total " + (M != null ? M.totals[s] : 0);`
HeartsTableGame.cs:346 / void RefreshHud()	`_handLabel.text = (M != null ? "Table " + M.table + " " : "") + "Hand " + hand;`

`HeartsTableGame.cs:394 / void Msg(string text, float secs = 0, Color? color = null)`	`var rl = HzUI.Label("R", g, Cards.RankName(c), rank == 10 ? 36 : 44, col);`
`HeartsTableGame.cs:412 / void Msg(string text, float secs = 0, Color? color = null)`	`var letter = HzUI.Label("L", frame.transform, Cards.RankName(c), 70, col); letter.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(90, 80), new Vector2(0, -4));`
`HeartsTableGame.cs:436 / void Msg(string text, float secs = 0, Color? color = null)`	`HzUI.Label("t", tag.transform, c == Cards.QueenSpades ? "13" : "-10", 22, Color.white).rectTransform.Stretch();`
`HeartsTableGame.cs:888 / void RefreshPassButton()`	`_passLabel.text = "PASS " + _passSel.Count + "/3";`
`HeartsTableGame.cs:958 / public override string DebugCommand(string cmd)`	`case "helper": UseHelper(p.Length > 1 ? p[1] : "hint"); return $"helper charges=[{string.Join(",", Prog.charges)}] hint={(_hintCard >= 0 ? Cards.Code(_hintCard) : "-")} peek={_peek} sel={string.Join(",", _passSel.Select(Cards.Code))}";`
`HeartsTableGame.cs:959 / public override string DebugCommand(string cmd)`	`case "charges": foreach (var id in HtHelpers.Ids) if (Prog.Has(id)) Prog.AddCharges(id, 9); SaveProg(); return "charges=" + string.Join(",", Prog.charges);`
`HeartsTables.cs:57 / class/static declarations`	`/// <summary>Helpers (boosters) of this game. Unlock tables come from the table data ("unlock" field).</summary>`
`HeartsTables.cs:260 / public static string Check(HtTable t)`	`if (!string.IsNullOrEmpty(t.unlock) && Array.IndexOf(HtHelpers.Ids, t.unlock) < 0) return "unknown helper '" + t.unlock + "'";`
`HeartsTableScreens.cs:34 / class/static declarations`	`var tl = HzUI.Label("Title", ribbon.transform, title, 62, Color.white); tl.rectTransform.Stretch(6);`
`HeartsTableScreens.cs:77 / void CloseAllModals() { foreach (var m in _modals.ToList()) CloseModal(m); }`	`var t = HzUI.Label("Txt", m.card, text, size, color, align, display);`
`HeartsTableScreens.cs:151 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Next table", Mint, by + 110, 520, 112, () => { CloseModal(m); HzPlatform.Interstitial(GameId, () => StartTable(M.table + 1)); });`

`HeartsTableScreens.cs:152 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Replay", Slate, by + 235, 300, 90, () => { CloseModal(m); StartTable(M.table); }, 40, -165);`
`HeartsTableScreens.cs:153 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Tables", Slate, by + 235, 300, 90, () => { CloseModal(m); ShowTitle(); ShowLadder(); }, 40, 165);`
`HeartsTableScreens.cs:158 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Try again", Mint, by + 105, 520, 112, () => { CloseModal(m); HzPlatform.Interstitial(GameId, () => StartTable(M.table)); });`
`HeartsTableScreens.cs:159 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Tables", Slate, by + 230, 520, 100, () => { CloseModal(m); ShowTitle(); ShowLadder(); }, 44);`
`HeartsTableScreens.cs:164 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Next hand", Mint, by + 75, 520, 120, () => { CloseModal(m); HzPlatform.Interstitial(GameId, BeginHand); }, 54);`
`HeartsTableScreens.cs:165 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Scores", Slate, by + 205, 300, 90, () => ShowScoreSheet(), 40, -165);`
`HeartsTableScreens.cs:166 / void ShowHandResult(int[] scores, int moon)`	`CardButton(m, "Menu", Slate, by + 205, 300, 90, () => ShowMenu(), 40, 165);`
`HeartsTableScreens.cs:204 / void ShowScoreSheet()`	`CardButton(m, "Close", Mint, h - 72, 420, 100, () => CloseModal(m), 46);`
`HeartsTableScreens.cs:233 / void ShowRules()`	`CardButton(m, "Got it!", Mint, 1090 + extra, 440, 112, () => CloseModal(m), 52);`
`HeartsTableScreens.cs:241 / void ShowMenu()`	`CardButton(m, "Resume", Mint, 190, 540, 110, () => CloseModal(m), 52);`
`HeartsTableScreens.cs:243 / void ShowMenu()`	`sound = CardButton(m, SoundLabel(), HzUI.Hex("#4C9EEB"), 315, 262, 96, null, 36, -139);`
`HeartsTableScreens.cs:244 / void ShowMenu()`	`sound.onClick.AddListener(() => { HzAudio.Muted = !HzAudio.Muted; sound.SetText(SoundLabel()); HzAudio.Play("toggle", 0.6f); });`
`HeartsTableScreens.cs:245 / void ShowMenu()`	`music = CardButton(m, MusicLabel(), HzUI.Hex("#4C9EEB"), 315, 262, 96, null, 36, 139);`

HeartsTableScreens.cs:246 / void ShowMenu()	music.onClick.AddListener(() => { HzAudio.MusicMuted = !HzAudio.MusicMuted; music.SetText(MusicLabel()); HzAudio.Play("toggle", 0.6f); });
HeartsTableScreens.cs:247 / void ShowMenu()	CardButton(m, "Rules", Coral, 435, 540, 100, () => ShowRules(), 44);
HeartsTableScreens.cs:248 / void ShowMenu()	CardButton(m, "Score sheet", Plum, 555, 540, 100, () => ShowScoreSheet(), 44);
HeartsTableScreens.cs:249 / void ShowMenu()	CardButton(m, "Restart table", HzUI.Hex("#F5A623"), 675, 540, 100, () => { CloseAllModals(); StopHand(false); HzPlatform.Interstitial(GameId, () => StartTable(M != null ? M.table : Level)); }, 44);
HeartsTableScreens.cs:250 / void ShowMenu()	CardButton(m, "Main menu", HzUI.Hex("#7B61FF"), 795, 540, 100, () => { CloseAllModals(); StopHand(); ShowTitle(); }, 44);
HeartsTableScreens.cs:253 / static string SoundLabel() => HzAudio.Muted ? "Sound: OFF" : "Sound: ON";	static string SoundLabel() => HzAudio.Muted ? "Sound: OFF" : "Sound: ON";
HeartsTableScreens.cs:254 / static string MusicLabel() => HzAudio.MusicMuted ? "Music: OFF" : "Music: ON";	static string MusicLabel() => HzAudio.MusicMuted ? "Music: OFF" : "Music: ON";
HeartsTableScreens.cs:307 / void ShowTitle()	var r = HzUI.Label("R", card.transform, Cards.RankName(fan[i]), 40, HeartsArt.SuitColor(su)); r.rectTransform.Place(new Vector2(0, 1), new Vector2(46, 48), new Vector2(26, -28));
HeartsTableScreens.cs:311 / void ShowTitle()	var title = HzUI.Label("Name", _title, "Hearts Table", portrait ? 130 : 120, Color.white);
HeartsTableScreens.cs:314 / void ShowTitle()	var tag = HzUI.Label("Tag", _title, "Climb the ladder of card tables", 34, Gold, TextAnchor.MiddleCenter, false); tag.fontStyle = FontStyle.Bold;
HeartsTableScreens.cs:321 / void ShowTitle()	var tl = HzUI.Label("TName", tc.transform, "Table " + cur + " " + table.name, 46, Color.white, TextAnchor.MiddleLeft);

HeartsTableScreens.cs:324 / void ShowTitle()	`var tv = HzUI.Label("TVenue", tc.transform, table.venue + " - to " + table.target, 28, HzUI.Hex("#BFE3DF"), TextAnchor.MiddleLeft, false);`
HeartsTableScreens.cs:328 / void ShowTitle()	`var tt = HzUI.Label("TTwist", tc.transform, tw.Count > 0 ? string.Join(" | ", tw) : "Classic rules", 26, Gold, TextAnchor.MiddleLeft, false);`
HeartsTableScreens.cs:341 / void ShowTitle()	`var play = HzUI.Button("Play", _title, cont ? "Continue" : "Play", Coral, Color.white, () =>`
HeartsTableScreens.cs:349 / void ShowTitle()	`var lad = HzUI.Button("Tables", _title, "Tables", Slate, Color.white, ShowLadder, 44);`
HeartsTableScreens.cs:351 / void ShowTitle()	`var st = HzUI.Button("Stats", _title, "Stats", Plum, Color.white, ShowStats, 44);`
HeartsTableScreens.cs:374 / void ShowTitle()	`var nameL = HzUI.Label("MyName", _title, "Your seat", 28, Color.white, TextAnchor.MiddleCenter, false); nameL.fontStyle = FontStyle.Bold;`
HeartsTableScreens.cs:382 / void ShowTitle()	`var sl = HzUI.Label("Stars", chip.transform, Prog.TotalStars + " / " + (Prog.unlocked * 3), 36, Color.white, TextAnchor.MiddleLeft); sl.rectTransform.Place(new Vector2(0, 0.5f), new Vector2(160, 60), new Vector2(92, 0), new Vector2(0, 0.5f));`
HeartsTableScreens.cs:414 / void ShowLadder()	`var row = HzUI.Button("Row" + n, content, null, locked ? HzUI.Hex("#E3DDD3") : HzUI.Hex("#F3ECE2"), Color.white, () =>`
HeartsTableScreens.cs:416 / void ShowLadder()	`if (locked) { HzAudio.Play(HzSfx.Wrong, 0.4f); Toast("Win table " + (tn - 1) + " to unlock"); return; }`
HeartsTableScreens.cs:424 / void ShowLadder()	`HzUI.Label("N", badge.transform, n.ToString(), 46, Color.white).rectTransform.Stretch();`
HeartsTableScreens.cs:425 / void ShowLadder()	`var nm = HzUI.Label("Name", row.transform, t.name, 40, locked ? TextSub : TextDark, TextAnchor.MiddleLeft);`
HeartsTableScreens.cs:430 / void ShowLadder()	`var il = HzUI.Label("Info", row.transform, info, 24, tw.Count > 0 ? HzUI.Hex("#7B3FA0") : TextSub, TextAnchor.MiddleLeft, false);`

`HeartsTableScreens.cs:434 / void ShowLadder()`	`var unlock = string.IsNullOrEmpty(t.unlock) ? "" : "Unlocks " + HtHelpers.Title(t.unlock);`
`HeartsTableScreens.cs:437 / void ShowLadder()`	`var ul = HzUI.Label("Unlock", row.transform, unlock, 24, HzUI.Hex("#D98A1E"), TextAnchor.MiddleLeft, false);`
`HeartsTableScreens.cs:462 / void ShowLadder()`	`HzUI.Label("T", nw.transform, "NEW", 26, Color.white).rectTransform.Stretch();`
`HeartsTableScreens.cs:470 / void ShowLadder()`	`CardButton(m, "Close", Mint, h - 75, 400, 100, () => CloseModal(m), 46);`
`HeartsTableScreens.cs:492 / void ShowStats()`	`("Helpers used", Stats.helpersUsed.ToString()),`
`HeartsTableScreens.cs:502 / void ShowStats()`	`CardButton(m, "Close", Mint, 1145, 400, 100, () => CloseModal(m), 46);`

CHAPTER 11

Artwork, fonts, audio and licences

Replace game images under Resources/heartstable through Unity while retaining resource keys, .meta GUIDs, Sprite/UI import settings, alpha, pivots and slicing. Procedural graphics in the art/view sources may need separate colour changes. Replacing one PNG does not change every fallback. Keep third-party font and other licence notices with the corresponding shipped assets.

Effects are under Resources/heartstable/Sfx. Preserve each actual load key and importer settings when replacing sounds. Supplied music: Resources/heartstable/Music/lounge.ogg. Preserve that load key and verify loop/compression and the independent Music toggle. Kit UI uses supplied fonts; preserve actual licence notices and verify new glyphs.

CHAPTER 12

Local saves, reset and migration

HzSave wraps PlayerPrefs and flushes local state on focus loss/quit. WebGL persistence uses browser storage tied to the hosting origin. Clearing site data or changing origin starts a new local save; this is not cloud synchronization. Inspect HzSave/PlayerPrefs calls in the actual game source before altering progression, stars, helper charges or tutorial flags.

Verify fresh launch, a completed arena, helper use, sound options, focus loss and restart. Use a dedicated test profile for resets. Keep technical save keys stable during colour/text reskins. If changing schema or keys, implement and test a migration rather than silently discarding existing users' progress.

CHAPTER 13

Advertising integration and configuration

Asset: `Assets/Szekipapa77Ads/Resources/AdsSettings.asset`. Inspect backend, useGoogleTestIds and missingSdkFallback in the publisher-owned Ads configuration. Buyer projects include adapter/integration code and exclude external provider SDK binaries. WebGL and a missing-provider fallback simulate ads; these are not live commercial impressions.

For real Android ads install the supported provider SDK from its official source into your working project, enter your own app/placement IDs and use test ads first. Verify banner/interstitial/rewarded callbacks on supported hardware. For production use your own IDs and disable simulated missing-SDK fallback. Build tooling rejects unsafe production fallback settings. No IAP, remote backend or HLK is included in this publisher edition.

CHAPTER 14

Android build and release

Install Unity 6000.0.62f1 Android modules through Hub. Select Android, run Setup and review application ID `com.szekipapa77.heartstable`, version, scene orientation and signing. The build command writes `Builds/Android/heartstable.apk`. Confirm BuildResult.Succeeded and the actual output file; intermediate report totals are not the final APK size.

Use your own keystore/signing and distribution settings. A test-signed demo is not a signed store release. A provider-free build simulates ads; a provider test-demo does not prove production ad-account configuration. Test real hardware touch, safe areas, focus/audio, save/relaunch and reward callbacks. Keep passwords, keystores and private service credentials out of buyer packages.

CHAPTER 15

WebGL build and hosting

Install matching WebGL support. Run Setup and Build WebGL; output is `Builds/WebGL/heartstable/`. Serve index.html with all loader/framework/data/WASM/StreamingAssets files over HTTP(S), not file://. Match compression, Content-Encoding and MIME types to the host. Outer page sizing and actual scene orientation are different settings.

The publisher's playable demo is hosted on HyperCodeStore. Upload the actual runtime files rather than merely changing a URL. Test a fresh anonymous load, real pointer controls, audio after interaction, saved progress/reload and phone/desktop sizing. Rebuild and rehost when code/assets change. A successful HTTP page alone is not gameplay verification.

CHAPTER 16

Troubleshooting

Compile errors: restore matching packages and the full private Kit/Ads layer; only one game Kit belongs in a project. **Pink/invisible rendering:** run Setup and inspect Graphics and every Quality URP assignment. **No input:** classic input handling must be enabled; check canvas focus and blocking modals. **Missing resources:** restore exact paths, .meta GUIDs and importer settings.

Edits do not appear: stop Play, Save in the level editor, reload the canonical bank and check current saved level. **Text reverts:** update construction and refresh assignments from the exact table. **No live ads:** provider binaries are deliberately excluded; inspect your installed provider and fallback mode. **Scene reload exception:** retain owner-lifetime and generation guards on delayed callbacks. **WebGL fails:** inspect browser Console, compression headers and missing payloads.

CHAPTER 17

Support and final release checklist

Publisher: Szekipapa77. Support: szekipapa77@gmail.com. Include game title, exact Unity version, platform, first Console error and reproduction steps. Never send passwords/tokens.

1. Clean-import the standalone project and native package.
2. Play both scenes and replace a scene near startup.
3. Review tutorial, normal completion/retry and every helper activation plus effect.
4. Validate editor changes and test reskin text/colours in both layouts.
5. Rebuild affected Android/WebGL outputs.
6. Verify complete public APK bytes, actual browser demo, documentation and video links.
7. Keep licences and exclude all internal prompts, planning/capture/build/QA tooling from buyers.