UNITY 6000.0.62F1 · COMPLETE OFFLINE DOCUMENTATION

Ledge Knockout Developer Guide

Install. Edit. Reskin. Build your release.

szekipapa77@gmail.com



CHAPTER 01

Start here: Ledge Knockout

Ledge Knockout is a standalone editable Unity game template. Choose one of the three weapon cards, drag toward a target and release to throw. Enemies take their turns in reply. Knock out all opposing ragdolls before they knock you out. Read weapon, room and enemy placement to choose an unobstructed trajectory. The duel is turn based against local enemies.

Includes 25 curated levels, 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 > Ledge Knockout > 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/LastSurvivor/Scenes/LastSurvivor_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/LastSurvivor`. 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/lastsurvivor contains level JSON, art, music and sound; Settings owns URP assets; Scenes contains both entry scenes.

- `Scripts/LastSurvivorGame.cs`
- `Scripts/LastSurvivorGame.UI.cs`
- `Scripts/LastSurvivorGame.View.cs`
- `Scripts/LsData.cs`
- `Scripts/LsGen.cs`
- `Scripts/LsLevels.cs`
- `Scripts/LsMeshes.cs`
- `Scripts/LsRagdoll.cs`
- `Scripts/LsSfx.cs`
- `Scripts/LsSim.cs`

Main flow: LastSurvivorGame.cs. UI and popup consumers: LastSurvivorGame.UI.cs. Model/generator/loading: LsData.cs, LsGen.cs, LsLevels.cs, LsSim.cs. Authoring: Editor/LastSurvivorLevelEditor.cs. Shared ad settings live outside the game at Assets/Szekipapa77Ads/Resources/AdsSettings.asset.

CHAPTER 04

Gameplay, controls and progression

Choose one of the three weapon cards, drag toward a target and release to throw. Enemies take their turns in reply. Knock out all opposing ragdolls before they knock you out. Read weapon, room and enemy placement to choose an unobstructed trajectory. The duel is turn based against local enemies.

The initial bank has 25 curated levels. 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/LastSurvivorGame.cs; effect/help strings: Scripts/LastSurvivorGame.cs.

Helper	Unlock	Free charges	Action and effect
Laser Sight	3	3	Shows the WHOLE flight of your next throw, bounces included. Line it up perfectly!
Bandage	6	2	Patch yourself up: heal 40% of your health right now. It doesn't use your turn.
Shield	9	2	A bubble that blocks every enemy throw of the next enemy turn.
Power Throw	13	2	Your next throw hits twice as hard and knocks twice as far.

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 > Ledge Knockout > Level Editor**. Exact implementation: `Editor/LastSurvivorLevelEditor.cs`.

Open the Ledge Knockout editor and select an early level. 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/lastsurvivor/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
`LastSurvivorLevelEditor.cs:56`	`[MenuItem("Szekipapa77/Ledge Knockout/Level Editor")]`
`LastSurvivorLevelEditor.cs:59`	`[MenuItem("Szekipapa77/Ledge Knockout/Validate All Levels")]`
`LastSurvivorLevelEditor.cs:123`	`if (GUILayout.Button(_dirty ? "Save *" : "Save", EditorStyles.toolbarButton, GUILayout.Width(70))) Save();`
`LastSurvivorLevelEditor.cs:124`	`if (GUILayout.Button("Reload", EditorStyles.toolbarButton, GUILayout.Width(60)) && (!_dirty || EditorUtility.DisplayDialog("Reload", "Discard unsaved changes?", "Discard", "Cancel"))) Load();`
`LastSurvivorLevelEditor.cs:127`	`if (GUILayout.Button("Validate", EditorStyles.toolbarButton, GUILayout.Width(70)) && Cur != null) ValidateOne(_sel);`
`LastSurvivorLevelEditor.cs:128`	`if (GUILayout.Button("Validate All", EditorStyles.toolbarButton, GUILayout.Width(85))) ValidateAll();`

LastSurvivorLevelEditor.cs:131	if (GUILayout.Button(new GUIContent("Play this level", "Optional quick test: saves, opens the WebGL scene at this level and enters Play mode. Editing never needs it."), EditorStyles.toolbarButton, GUILayout.Width(100)) && Cur != null) Play(_sel + 1);
LastSurvivorLevelEditor.cs:157	if (GUILayout.Button("Add"))
LastSurvivorLevelEditor.cs:162	if (GUILayout.Button("Dup") && Cur != null) { var c = Cur.Clone(); c.name += " copy"; Insert(_sel + 1, c); }
LastSurvivorLevelEditor.cs:163	if (GUILayout.Button("Del") && Cur != null && EditorUtility.DisplayDialog("Delete", $"Delete level {_sel + 1} '{Cur.name}'?", "Delete", "Cancel"))
LastSurvivorLevelEditor.cs:168	if (GUILayout.Button("Up") && _sel > 0) Move(-1);
LastSurvivorLevelEditor.cs:169	if (GUILayout.Button("Down") && _sel < _file.levels.Length - 1) Move(1);
LastSurvivorLevelEditor.cs:191	EditorGUILayout.LabelField($"Level {_sel + 1}", EditorStyles.boldLabel);
LastSurvivorLevelEditor.cs:192	l.name = EditorGUILayout.TextField("Name", l.name);
LastSurvivorLevelEditor.cs:193	l.hint = EditorGUILayout.TextField("First-time hint", l.hint);
LastSurvivorLevelEditor.cs:195	l.difficulty = EditorGUILayout.Slider("Design difficulty", l.difficulty, 0, 1);
LastSurvivorLevelEditor.cs:196	l.breather = EditorGUILayout.ToggleLeft("Breather", l.breather, GUILayout.Width(80));
LastSurvivorLevelEditor.cs:197	l.tut = EditorGUILayout.ToggleLeft("Tutorial", l.tut == 1, GUILayout.Width(80)) ? 1 : 0;
LastSurvivorLevelEditor.cs:200	l.skill = EditorGUILayout.Slider(new GUIContent("Enemy aim", "0 = wild throws, 1 = near perfect"), l.skill, 0, 1);

LastSurvivorLevelEditor.cs:201	l.enemyDmg = EditorGUILayout.Slider(new GUIContent("Enemy damage x", "Multiplier on enemy weapon damage"), l.enemyDmg, 0.1f, 1.2f);
LastSurvivorLevelEditor.cs:204	l.heroHp = EditorGUILayout.IntSlider("Hero HP", l.heroHp, 40, 250);
LastSurvivorLevelEditor.cs:205	l.wind = EditorGUILayout.Slider(new GUIContent("Wind", "Sideways push on every throw; introduced at level " + LsGen.IntroWind), l.wind, -2.5f, 2.5f);
LastSurvivorLevelEditor.cs:208	float w = EditorGUILayout.Slider("Room width", l.w, 6, 10);
LastSurvivorLevelEditor.cs:209	float h = EditorGUILayout.Slider("Room height", l.h, 8, 13);
LastSurvivorLevelEditor.cs:223	EditorGUILayout.HelpBox($"Bot: {v}{(v.Solvable ? "\nSolvable: the perfect-aim bot beats every enemy-aim seed." : "\nNOT solvable: the perfect-aim bot lost a run or cannot hit every enemy. Lower enemy damage/aim, add hero HP or open a line of fire.")}", v.Solvable ? MessageType.Info : MessageType.Warning);
LastSurvivorLevelEditor.cs:238	if (msgs.Count > 0) EditorGUILayout.HelpBox("Used before its introduction level: " + string.Join(", ", msgs) + ". Players won't get those first-time popups in order.", MessageType.Warning);
LastSurvivorLevelEditor.cs:239	if (l.enemies.Length == 0) EditorGUILayout.HelpBox("No enemies: add at least one.", MessageType.Error);
LastSurvivorLevelEditor.cs:434	l.hero.x = EditorGUILayout.FloatField("x", l.hero.x); l.hero.y = EditorGUILayout.FloatField("y (feet)", l.hero.y);
LastSurvivorLevelEditor.cs:440	u.x = EditorGUILayout.FloatField("x", u.x); u.y = EditorGUILayout.FloatField("y (feet)", u.y);
LastSurvivorLevelEditor.cs:441	u.hp = EditorGUILayout.IntSlider("HP", u.hp, 10, 300);
LastSurvivorLevelEditor.cs:448	o.x = EditorGUILayout.FloatField("x", o.x);

LastSurvivorLevelEditor.cs:451	`o.y = EditorGUILayout.FloatField("pivot y", o.y);`
LastSurvivorLevelEditor.cs:452	`o.w = EditorGUILayout.Slider("swing deg", o.w, 5, 70);`
LastSurvivorLevelEditor.cs:453	`o.h = EditorGUILayout.Slider("rope", o.h, 1.5f, l.h - 0.5f);`
LastSurvivorLevelEditor.cs:457	`o.y = EditorGUILayout.FloatField("bottom y", o.y);`
LastSurvivorLevelEditor.cs:458	`if (k == ObKind.Ledge) { o.w = EditorGUILayout.Slider("width", o.w, 0.4f, 4f); o.h = EditorGUILayout.Slider("height", o.h, 0.3f, l.h - 2f); }`
LastSurvivorLevelEditor.cs:459	`if (k == ObKind.Glass) o.h = EditorGUILayout.Slider("height", o.h, 0.6f, 4f);`
LastSurvivorLevelEditor.cs:460	`if (k == ObKind.Crate) o.hp = EditorGUILayout.IntSlider("hits", o.hp, 1, 3);`
LastSurvivorLevelEditor.cs:471	`_showGen = EditorGUILayout.Foldout(_showGen, "Auto Generate (seeded, bot-proven, same generator as the runtime)", true);`
LastSurvivorLevelEditor.cs:474	`_genCount = EditorGUILayout.IntField("Count", _genCount); _genStart = EditorGUILayout.IntField("Start at level #", _genStart);`
LastSurvivorLevelEditor.cs:475	`_genSeed = EditorGUILayout.IntField("Seed", _genSeed);`
LastSurvivorLevelEditor.cs:478	`_genFrom = EditorGUILayout.Slider("Start difficulty", _genFrom, 0, 1); _genTo = EditorGUILayout.Slider("End difficulty", _genTo, 0, 1);`
LastSurvivorLevelEditor.cs:481	`_genBreather = EditorGUILayout.IntSlider("Breather every", _genBreather, 0, 8);`
LastSurvivorLevelEditor.cs:482	`_genRuns = EditorGUILayout.IntSlider("Proof runs", _genRuns, 1, 6);`

`LastSurvivorLevelEditor.cs:483`	`_genReplace = EditorGUILayout.ToggleLeft("Replace that range (else insert)", _genReplace);`
`LastSurvivorLevelEditor.cs:494`	`EditorGUILayout.LabelField("Difficulty curve (sawtooth; blue bars are breathers)", EditorStyles.miniLabel);`
`LastSurvivorLevelEditor.cs:495`	`if (GUILayout.Button($"Generate {_genCount} level(s)", GUILayout.Height(26))) 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
`LastSurvivorLevelEditor.cs:53`	`bool _genReplace, _showGen = true;`
`LastSurvivorLevelEditor.cs:483`	`_genReplace = EditorGUILayout.ToggleLeft("Replace that range (else insert)", _genReplace);`
`LastSurvivorLevelEditor.cs:516`	`if (_genReplace) { int rm = Mathf.Min(_genCount, list.Count - at); if (rm > 0) list.RemoveRange(at, rm); }`
`LastSurvivorLevelEditor.cs:517`	`list.InsertRange(at, made);`

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/LastSurvivor/Resources/lastsurvivor/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": "First Throw",
  "w": 7.6,
  "h": 10.5,
  "wind": 0,
  "skill": 0.1,
  "enemyDmg": 0.25,
  "heroHp": 100,
  "difficulty": 0.05,
  "breather": false,
  "tut": 1,
  "hint": "Pick a weapon card, then drag and release to throw!",
  "hero": {
    "x": -2.6,
    "y": 0,
    "hp": 60,
    "color": 0
  },
  "enemies": [
    {
      "x": 2.2,
      "y": 0,
      "hp": 70,
      "color": 1
    }
  ],
  "obs": []
}
```


Root keys: levels. Inspect LsData.cs, LsGen.cs, LsLevels.cs, LsSim.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/LastSurvivorGame.UI.cs`. Find `Ink`, currently `#2A2350`. 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
`LastSurvivorGame.UI.cs:54 / void BuildHud()`	`vs.AddOutline(Ink, 4);`
`LastSurvivorGame.UI.cs:55 / void BuildHud()`	`var vst = HzUI.Label("T", vs.transform, "VS", 54, Ink); vst.rectTransform.Stretch();`
`LastSurvivorGame.UI.cs:100 / void BuildHud()`	`var bl = HzUI.Label("BL", Root, "BOOSTERS", 44, Ink);`
`LastSurvivorGame.UI.cs:118 / void BuildHud()`	`_handIcon.AddOutline(Ink, 5);`
`LastSurvivorGame.UI.cs:139 / void BuildHud()`	`var who = HzUI.Label("Who", bg.transform, left ? "YOU" : "ENEMIES", 34, Ink);`
`LastSurvivorGame.UI.cs:142 / void BuildHud()`	`who.color = Color.white; who.AddOutline(Ink, 3);`
`LastSurvivorGame.UI.cs:149 / void BuildHud()`	`var btn = HzUI.Button("Card" + slot, parent, null, Color.white, Ink, () => SelectCard(slot), 40, 30);`
`LastSurvivorGame.UI.cs:163 / void BuildHud()`	`cv.Name = HzUI.Label("Name", cv.Root, "", P ? 42 : 40, Ink);`

Source	Exact current code
`LastSurvivorGame.cs:28 / class/static declarations`	`static readonly string[] BoostColor = { "#FF4D6D", "#2EC4B6", "#4C9EEB", "#FF9F1C" };`
`LastSurvivorGame.cs:142 / IEnumerator IntroFlow()`	`ArtPopup("New Weapon!", $"{w.Name}: {w.Desc}", HzUI.Hex("#FF9F1C"), w.Art, ("Awesome!", Green, () => closed = true));`
`LastSurvivorGame.cs:170 / IEnumerator IntroFlow()`	`ArtPopup(it.title, it.msg, HzUI.Hex("#7B61FF"), null, ("Got it!", Green, () => closed = true));`

`LastSurvivorGame.cs:308 / IEnumerator EnemyTurn()`	`if (first) { TurnBanner("Enemy turn", HzUI.Hex("#FF5A5F")); HzAudio.Play(Sfx.EnemyTurn, 0.6f); first = false; if (_tutStep == 3) ShowTip("Now they throw back! Last one standing wins.", 2.6f); }`
`LastSurvivorGame.cs:323 / IEnumerator EnemyTurn()`	`TurnBanner("Your turn", HzUI.Hex("#3FD9A4"));`
`LastSurvivorGame.cs:381 / IEnumerator LoseFlow()`	`ArtPopup("Knocked Out!", msg + "\nGet back up with half your health?", HzUI.Hex("#FF5A5F"), "badge_lose",`
`LastSurvivorGame.cs:385 / IEnumerator LoseFlow()`	`ArtPopup("Knocked Out!", msg, HzUI.Hex("#FF5A5F"), "badge_lose", ("Retry", Orange, () => LoadLevel(Level, true)));`
`LastSurvivorGame.cs:397 / void Revive()`	`SpawnStars(new Vector3(h.X, h.Y + 1.2f, 0), 14, HzUI.Hex("#3FD9A4"));`
`LastSurvivorGame.cs:400 / void Revive()`	`TurnBanner("Your turn", HzUI.Hex("#3FD9A4"));`
`LastSurvivorGame.cs:424 / void OnBooster(int b)`	`("No thanks", HzUI.Hex("#A8A3C7"), null));`
`LastSurvivorGame.cs:442 / void OnBooster(int b)`	`FloatText(new Vector3(h.X, h.Y + 2.1f, 0), "+" + add, HzUI.Hex("#3FD9A4"), 64);`
`LastSurvivorGame.cs:443 / void OnBooster(int b)`	`SpawnStars(new Vector3(h.X, h.Y + 1.1f, 0), 12, HzUI.Hex("#3FD9A4"));`
`LastSurvivorGame.UI.cs:10 / class/static declarations`	`static readonly Color Green = HzUI.Hex("#3DBE6E"), Orange = HzUI.Hex("#F5A623");`
`LastSurvivorGame.UI.cs:11 / class/static declarations`	`static readonly Color Ink = HzUI.Hex("#2A2350");`
`LastSurvivorGame.UI.cs:40 / void BuildHud()`	`var lvl = HzUI.Panel("Level", _hud, HzUI.Hex("#2A2350"), 40);`
`LastSurvivorGame.UI.cs:45 / void BuildHud()`	`var set = HzIcons.IconButton(_hud, HzIcons.Settings, HzUI.Hex("#7B61FF"), () => { HzAudio.Play(Sfx.Click, 0.6f); ShowSettingsPopup(); }, 110);`
`LastSurvivorGame.UI.cs:50 / void BuildHud()`	`_heroBar = TeamBar("HeroBar", new Vector2(-barW * 0.5f - 70, barY), barW, HzUI.Hex("#3FD9A4"), out _heroFill, out _heroHpText, true);`

LastSurvivorGame.UI.cs:51 / void BuildHud()	_enemyBar = TeamBar("EnemyBar", new Vector2(barW * 0.5f + 70, barY), barW, HzUI.Hex("#FF5A5F"), out _enemyFill, out _enemyHpText, false);
LastSurvivorGame.UI.cs:52 / void BuildHud()	var vs = HzUI.Panel("VS", _hud, HzUI.Hex("#FFD23F"), 50);
LastSurvivorGame.UI.cs:62 / void BuildHud()	var wt = HzUI.Label("T", wp.transform, "WIND", 40, HzUI.Hex("#4C9EEB")); wt.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(160, 70), new Vector2(_data.wind > 0 ? -40 : 40, 0));
LastSurvivorGame.UI.cs:66 / void BuildHud()	var ic = HzIcons.Icon(wp.transform, arrow, 44, HzUI.Hex("#4C9EEB"));
LastSurvivorGame.UI.cs:77 / void BuildHud()	var strip = HzUI.Panel("Strip", _deckUi, HzUI.Hex("#FF9F1C"), 30);
LastSurvivorGame.UI.cs:81 / void BuildHud()	var panel = HzUI.Panel("Panel", _deckUi, HzUI.Hex("#4C6FE0"), 40);
LastSurvivorGame.UI.cs:155 / void BuildHud()	var ring = HzUI.Panel("Ring", cv.Root, HzUI.Hex("#FFD23F"), 34);
LastSurvivorGame.UI.cs:166 / void BuildHud()	var dmgBg = HzUI.Panel("DmgBg", cv.Root, HzUI.Hex("#FF5A5F"), 22);
LastSurvivorGame.UI.cs:184 / void BuildHud()	var badge = HzUI.Panel("Count", bv.Root, HzUI.Hex("#2A2350"), 24);
LastSurvivorGame.UI.cs:221 / void RefreshDeck()	cv.Bg.color = sel ? HzUI.Hex("#FFF6D6") : Color.white;
LastSurvivorGame.UI.cs:241 / void RefreshBoosters()	bv.Bg.color = un ? (armed ? HzUI.Hex("#FFD23F") : HzUI.Hex(BoostColor[b])) : HzUI.Hex("#8E89A8");
LastSurvivorGame.UI.cs:285 / void LevelBanner()	var n = HzUI.Label("N", bg.transform, _data.name, 52, HzUI.Hex("#FFD23F")); n.rectTransform.Place(new Vector2(0.5f, 0), new Vector2(740, 80), new Vector2(0, 50));
LastSurvivorGame.UI.cs:352 / void WinPopup(int lv, int stars, string msg)	var p = ArtPopup("Victory!", "\n" + msg, HzUI.Hex("#3DBE6E"), "badge_win",
LastSurvivorGame.UI.cs:354 / void WinPopup(int lv, int stars, string msg)	("Replay", HzUI.Hex("#7B61FF"), () => { Level = lv; LoadLevel(lv, true); }));

`LastSurvivorGame.UI.cs:357 / void WinPopup(int lv, int stars, string msg)`	`var s = HzIcons.Icon(p.Card, HzIcons.Star, k == 1 ? 120 : 96, k < stars ? HzUI.Hex("#FFC93C") : new Color(0.75f, 0.72f, 0.8f));`
`LastSurvivorGame.UI.cs:372 / void ShowSettingsPopup()`	`var how = HzUI.Button("How", p.Card, "How to play", HzUI.Hex("#FF9F1C"), Color.white, () => { p.Close(); ShowHowTo(); }, 56);`
`LastSurvivorGame.UI.cs:384 / void ShowHowTo()`	`HzUI.Hex("#FF9F1C"), "hero", ("Got it!", Green, null));`
`LastSurvivorGame.View.cs:63 / void SetupLook()`	`_cam.backgroundColor = HzUI.Hex("#E6E8F2");`
`LastSurvivorGame.View.cs:75 / void SetupLook()`	`_sun.color = HzUI.Hex("#FFF4E6"); _sun.intensity = 1.25f;`
`LastSurvivorGame.View.cs:82 / void SetupLook()`	`fill.transform.rotation = Quaternion.Euler(20, -140, 0); fill.color = HzUI.Hex("#C9CCFF"); fill.intensity = 0.35f;`
`LastSurvivorGame.View.cs:85 / void SetupLook()`	`rim.transform.rotation = Quaternion.Euler(15, 180, 0); rim.color = HzUI.Hex("#FFE0F0"); rim.intensity = 0.3f;`
`LastSurvivorGame.View.cs:88 / void SetupLook()`	`RenderSettings.ambientSkyColor = HzUI.Hex("#D9DBFF");`
`LastSurvivorGame.View.cs:89 / void SetupLook()`	`RenderSettings.ambientEquatorColor = HzUI.Hex("#C8C2E8");`
`LastSurvivorGame.View.cs:90 / void SetupLook()`	`RenderSettings.ambientGroundColor = HzUI.Hex("#7B76A6");`
`LastSurvivorGame.View.cs:103 / void SetupLook()`	`_smokeMat = Mat("#8D8AA6", 0.1f);`
`LastSurvivorGame.View.cs:106 / void SetupLook()`	`_whiteMat = Mat("#FFFFFF", 0.6f);`
`LastSurvivorGame.View.cs:107 / void SetupLook()`	`_inkMat = Mat("#1E1B33", 0.7f);`
`LastSurvivorGame.View.cs:108 / void SetupLook()`	`string[] sp = { "#FFFFFF", "#FFD23F", "#FF5A5F", "#3FD9A4", "#9B5DE5", "#4C9EEB", "#FF9F1C" };`
`LastSurvivorGame.View.cs:171 / void BuildPools()`	`BoxAt("blade", root, new Vector3(0, 0.13f, 0), new Vector3(0.07f, 0.26f, 0.02f), Mat("#DDE3EA", 0.9f));`

`LastSurvivorGame.View.cs:172 / void BuildPools()`	`BoxAt("guard", root, new Vector3(0, 0.0f, 0), new Vector3(0.11f, 0.025f, 0.05f), Mat("#2B2B3A", 0.5f));`
`LastSurvivorGame.View.cs:173 / void BuildPools()`	`BoxAt("handle", root, new Vector3(0, -0.08f, 0), new Vector3(0.05f, 0.14f, 0.045f), Mat("#D9363E", 0.5f));`
`LastSurvivorGame.View.cs:176 / void BuildPools()`	`{ var t = MeshObj("rock", root, LsMeshes.SphereMesh(7, 5), Mat("#8E8A86", 0.15f)); t.localScale = new Vector3(0.42f, 0.34f, 0.38f); t.localRotation = Quaternion.Euler(20, 30, 10); break; }`
`LastSurvivorGame.View.cs:179 / void BuildPools()`	`var t = MeshObj("ball", root, LsMeshes.Sphere, Mat("#7B3FA0", 0.92f)); t.localScale = Vector3.one * 0.48f;`
`LastSurvivorGame.View.cs:185 / void BuildPools()`	`var t = MeshObj("body", root, LsMeshes.SphereMesh(10, 8), Mat("#4E8A3A", 0.5f)); t.localScale = new Vector3(0.28f, 0.33f, 0.28f);`
`LastSurvivorGame.View.cs:186 / void BuildPools()`	`var c = MeshObj("cap", root, LsMeshes.Cylinder, Mat("#4A4A55", 0.7f)); c.localScale = new Vector3(0.1f, 0.08f, 0.1f); c.localPosition = new Vector3(0, 0.18f, 0);`
`LastSurvivorGame.View.cs:187 / void BuildPools()`	`var lv = BoxAt("lever", root, new Vector3(0.06f, 0.12f, 0), new Vector3(0.03f, 0.16f, 0.04f), Mat("#B8BCC6", 0.8f)); lv.localRotation = Quaternion.Euler(0, 0, -15);`
`LastSurvivorGame.View.cs:191 / void BuildPools()`	`BoxAt("handle", root, new Vector3(0, 0, 0), new Vector3(0.055f, 0.46f, 0.055f), Mat("#B5651D", 0.35f));`
`LastSurvivorGame.View.cs:192 / void BuildPools()`	`BoxAt("head", root, new Vector3(0.03f, 0.22f, 0), new Vector3(0.24f, 0.09f, 0.09f), Mat("#8D97A8", 0.85f));`
`LastSurvivorGame.View.cs:196 / void BuildPools()`	`var t = MeshObj("ball", root, LsMeshes.SphereMesh(14, 10), Mat("#8B4A22", 0.35f)); t.localScale = new Vector3(0.44f, 0.27f, 0.27f);`
`LastSurvivorGame.View.cs:201 / void BuildPools()`	`BoxAt("handle", root, new Vector3(0, 0, 0), new Vector3(0.055f, 0.5f, 0.055f), Mat("#B5651D", 0.35f));`

`LastSurvivorGame.View.cs:202 / void BuildPools()`	`BoxAt("head", root, new Vector3(0.09f, 0.2f, 0), new Vector3(0.16f, 0.15f, 0.04f), Mat("#D9363E", 0.6f));`
`LastSurvivorGame.View.cs:203 / void BuildPools()`	`BoxAt("edge", root, new Vector3(0.18f, 0.2f, 0), new Vector3(0.03f, 0.17f, 0.035f), Mat("#E8ECF2", 0.9f));`
`LastSurvivorGame.View.cs:208 / void BuildPools()`	`var s = MeshObj("stick", root, LsMeshes.Cylinder, Mat("#E8412E", 0.4f)); s.localScale = new Vector3(0.1f, 0.36f, 0.1f); s.localPosition = new Vector3(-0.1f + k * 0.1f, 0, k == 1 ? -0.03f : 0);`
`LastSurvivorGame.View.cs:210 / void BuildPools()`	`BoxAt("band", root, Vector3.zero, new Vector3(0.33f, 0.06f, 0.13f), Mat("#2B2B3A", 0.4f));`
`LastSurvivorGame.View.cs:211 / void BuildPools()`	`BoxAt("fuse", root, new Vector3(0, 0.22f, 0), new Vector3(0.02f, 0.1f, 0.02f), Mat("#F4E1B0", 0.2f));`
`LastSurvivorGame.View.cs:246 / void BuildWorld()`	`var wallMat = Mat("#B9B4C9", 0.2f);`
`LastSurvivorGame.View.cs:247 / void BuildWorld()`	`var wallEdge = Mat("#A39DB8", 0.25f);`
`LastSurvivorGame.View.cs:248 / void BuildWorld()`	`var backMat = Mat("#8C93F0", 0.15f);`
`LastSurvivorGame.View.cs:249 / void BuildWorld()`	`var floorMat = Mat("#F4F1EA", 0.3f);`
`LastSurvivorGame.View.cs:252 / void BuildWorld()`	`BoxAt("Lip", _world, new Vector3(0, -0.06f, -1.25f), new Vector3(W + 4f, 0.14f, 0.12f), Mat("#E2DDD3", 0.3f));`
`LastSurvivorGame.View.cs:258 / void BuildWorld()`	`var c = Color.Lerp(HzUI.Hex("#A7AEFF"), HzUI.Hex("#7C83E6"), b / (float)(bands - 1));`
`LastSurvivorGame.View.cs:268 / void BuildWorld()`	`BoxAt("Skirting", _world, new Vector3(0, 0.12f, 1.55f), new Vector3(W, 0.24f, 0.1f), Mat("#6E74C9", 0.2f), false);`
`LastSurvivorGame.View.cs:280 / void BuildWorld()`	`BoxAt("body", t, new Vector3(o.X, o.Y + o.H * 0.5f, 0.2f), new Vector3(o.W, o.H, 2.4f), Mat("#C6BFD8", 0.25f));`

LastSurvivorGame.View.cs:281 / void BuildWorld()	BoxAt("top", t, new Vector3(o.X, o.T - 0.04f, 0.2f), new Vector3(o.W + 0.06f, 0.09f, 2.46f), Mat("#F4F1EA", 0.35f));
LastSurvivorGame.View.cs:288 / void BuildWorld()	var wood = Mat(o.Hp > 1 ? "#A86A3A" : "#D59A5B", 0.25f);
LastSurvivorGame.View.cs:289 / void BuildWorld()	var dark = Mat(o.Hp > 1 ? "#6E4022" : "#94603A", 0.3f);
LastSurvivorGame.View.cs:305 / void BuildWorld()	BoxAt("frameT", t, new Vector3(o.X, o.T + 0.03f, 0), new Vector3(o.W + 0.06f, 0.06f, 1.44f), Mat("#E8ECF8", 0.6f));
LastSurvivorGame.View.cs:313 / void BuildWorld()	var c = MeshObj("body", t, LsMeshes.Cylinder, Mat("#E8412E", 0.55f)); c.localScale = new Vector3(o.W, o.H, o.W);
LastSurvivorGame.View.cs:314 / void BuildWorld()	for (int k = -1; k <= 1; k += 2) { var band = MeshObj("band", t, LsMeshes.Cylinder, Mat("#FFD23F", 0.5f)); band.localScale = new Vector3(o.W * 1.04f, 0.07f, o.W * 1.04f); band.localPosition = new Vector3(0, k * o.H * 0.28f, 0); }
LastSurvivorGame.View.cs:315 / void BuildWorld()	var sign = BoxAt("sign", t, new Vector3(0, 0, -o.W * 0.5f - 0.01f), new Vector3(0.2f, 0.2f, 0.02f), Mat("#1E1B33", 0.4f), false); sign.localRotation = Quaternion.Euler(0, 0, 45);
LastSurvivorGame.View.cs:322 / void BuildWorld()	MeshObj("pivot", t, LsMeshes.SphereMesh(10, 8), Mat("#4A4A55", 0.7f)).localScale = Vector3.one * 0.22f;
LastSurvivorGame.View.cs:323 / void BuildWorld()	var rope = MeshObj("rope", t, LsMeshes.Cylinder, Mat("#5C5470", 0.4f)); rope.localScale = new Vector3(0.05f, o.H, 0.05f); rope.localPosition = new Vector3(0, -o.H * 0.5f, 0);
LastSurvivorGame.View.cs:324 / void BuildWorld()	var bob = MeshObj("bob", t, LsMeshes.Sphere, Mat("#3A3F55", 0.85f)); bob.localScale = Vector3.one * Match.BobR * 2; bob.localPosition = new Vector3(0, -o.H, 0);
LastSurvivorGame.View.cs:325 / void BuildWorld()	var ring = MeshObj("ring", bob, LsMeshes.Cylinder, Mat("#FFD23F", 0.6f)); ring.localScale = new Vector3(1.04f, 0.12f, 1.04f); ring.localRotation = Quaternion.Euler(90, 0, 0);

`LastSurvivorGame.View.cs:385 / void BuildWorld()`	`var fill = HzUI.Panel("Fill", bar.transform, HzUI.Hex(u.Team == 0 ? "#57E36A" : "#FF4D4D"), 8);`
`LastSurvivorGame.View.cs:388 / void BuildWorld()`	`var badge = HzUI.Panel("Badge", bar.transform, HzUI.Hex(u.Team == 0 ? "#2E9E5B" : "#C9302C"), 18);`
`LastSurvivorGame.View.cs:609 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`FloatText(new Vector3(e.X, e.Y + 0.5f, 0), "-" + e.V, e.Head ? HzUI.Hex("#FFD23F") : HzUI.Hex("#FF5A5F"), e.Head ? 78 : 62);`
`LastSurvivorGame.View.cs:612 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`FloatText(new Vector3(e.X, e.Y + 1.15f, 0), "HEADSHOT!", HzUI.Hex("#FFFFFF"), 56);`
`LastSurvivorGame.View.cs:614 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`SpawnStars(new Vector3(e.X, e.Y, -0.2f), 10, HzUI.Hex("#FFD23F"));`
`LastSurvivorGame.View.cs:645 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`FloatText(new Vector3(head.x, head.y + 0.9f, 0), "K.O.!", HzUI.Hex("#FFFFFF"), 70);`
`LastSurvivorGame.View.cs:646 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`SpawnStars(new Vector3(head.x, head.y, -0.2f), 14, HzUI.Hex("#FFD23F"));`
`LastSurvivorGame.View.cs:658 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`case ObKind.Crate: HzAudio.Play(Sfx.CrateBreak, 0.85f); SpawnDebris(at, Mat("#D59A5B", 0.25f), 10, 0.16f); break;`
`LastSurvivorGame.View.cs:660 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`case ObKind.Barrel: SpawnDebris(at, Mat("#E8412E", 0.5f), 8, 0.16f); break;`
`LastSurvivorGame.View.cs:686 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`SpawnStars(new Vector3(e.X, e.Y, -0.3f), 12, HzUI.Hex("#4C9EEB"));`
`LastSurvivorGame.View.cs:688 / void PlayEvent(Ev e, WeaponDef wd, int thrower, ref bool landed)`	`FloatText(new Vector3(e.X, e.Y + 0.6f, 0), "BLOCKED!", HzUI.Hex("#7FC8FF"), 60);`
`LastSurvivorGame.View.cs:748 / void Explosion(Vector3 at, float radius)`	`SpawnStars(at, 16, HzUI.Hex("#FFD23F"));`

`LsData.cs:73 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Knife, Name = "Knife", Art = "w_knife", Color = "#C9D3E0", Damage = 22, Push = 0.5f, Radius = 0.13f, Sticks = true, Unlock = 1, Desc = "Fast and light. Sticks where it lands." },`
`LsData.cs:74 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Rock, Name = "Rock", Art = "w_rock", Color = "#8E8A86", Damage = 26, Push = 1.1f, Radius = 0.2f, Bounce = 0.2f, MaxBounces = 1, Unlock = 1, Desc = "A solid rock. Smashes crates." },`
`LsData.cs:75 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Bowling, Name = "Bowling Ball", Art = "w_bowling", Color = "#7B3FA0", Damage = 28, Push = 2.0f, Radius = 0.24f, Bounce = 0.3f, MaxBounces = 3, Rolls = true, Unlock = 1, Desc = "Heavy! Big knockback and it keeps rolling." },`
`LsData.cs:76 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Grenade, Name = "Grenade", Art = "w_grenade", Color = "#4E8A3A", Damage = 34, Push = 2.0f, Blast = 1.7f, Radius = 0.16f, Explodes = true, Unlock = 2, Desc = "Explodes on impact. Hurts everyone in the blast, you too!" },`
`LsData.cs:77 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Hammer, Name = "Hammer", Art = "w_hammer", Color = "#B5651D", Damage = 36, Push = 1.5f, Radius = 0.2f, Unlock = 4, Desc = "A big bonk." },`
`LsData.cs:78 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Football, Name = "Football", Art = "w_football", Color = "#8B4A22", Damage = 20, Push = 1.4f, Radius = 0.2f, Bounce = 0.78f, MaxBounces = 5, Unlock = 7, Desc = "Bouncy! Bank it off walls and ceilings." },`
`LsData.cs:79 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Axe, Name = "Axe", Art = "w_axe", Color = "#D9363E", Damage = 44, Push = 1.0f, Radius = 0.2f, Sticks = true, Unlock = 11, Desc = "Huge damage, small knockback." },`
`LsData.cs:80 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Tnt, Name = "TNT", Art = "w_tnt", Color = "#E8412E", Damage = 50, Push = 3.0f, Blast = 2.4f, Radius = 0.18f, Explodes = true, Unlock = 16, Desc = "A massive blast. Keep your distance!" },`

`LsData.cs:104 / public static int PoolCount(int level) { int n = 0; foreach (var w in All) if (level >= w.Unlock) n++; return n; }`	`public const string Hero = "#3FD9A4";`
`LsData.cs:105 / public static int PoolCount(int level) { int n = 0; foreach (var w in All) if (level >= w.Unlock) n++; return n; }`	`public static readonly string[] EnemyColors = { "#FF5A5F", "#9B5DE5", "#F4B400", "#5C6B8A", "#FF8A3D" };`

CHAPTER 10

Reskin recipe: visible menu and HUD text

Open `Scripts/LastSurvivorGame.cs`, find the displayed literal `"Ledge Knockout"`, and replace its displayed value with `"Ledge Knockout 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/LastSurvivorGame.cs; effect/help strings: Scripts/LastSurvivorGame.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
`LastSurvivorGame.cs:25 / class/static declarations`	`static readonly string[] BoostName = { "Laser Sight", "Bandage", "Shield", "Power Throw" };`
`LastSurvivorGame.cs:142 / IEnumerator IntroFlow()`	`ArtPopup("New Weapon!", $"{w.Name}: {w.Desc}", HzUI.Hex("#FF9F1C"), w.Art, ("Awesome!", Green, () => closed = true));`
`LastSurvivorGame.cs:156 / IEnumerator IntroFlow()`	`ArtPopup(BoostName[b] + " unlocked!", BoostDesc[b] + $"\nYou get {BoostStart[b]} free.", HzUI.Hex(BoostColor[b]), BoostArt[b], ("Got it!", Green, () => closed = true));`
`LastSurvivorGame.cs:170 / IEnumerator IntroFlow()`	`ArtPopup(it.title, it.msg, HzUI.Hex("#7B61FF"), null, ("Got it!", Green, () => closed = true));`
`LastSurvivorGame.cs:367 / IEnumerator WinFlow()`	`WinPopup(lv, stars, $"Last one standing with {pct}% health!");`
`LastSurvivorGame.cs:381 / IEnumerator LoseFlow()`	`ArtPopup("Knocked Out!", msg + "\nGet back up with half your health?", HzUI.Hex("#FF5A5F"), "badge_lose",`
`LastSurvivorGame.cs:385 / IEnumerator LoseFlow()`	`ArtPopup("Knocked Out!", msg, HzUI.Hex("#FF5A5F"), "badge_lose", ("Retry", Orange, () => LoadLevel(Level, true)));`

`LastSurvivorGame.cs:413 / void OnBooster(int b)`	`if (!Unlocked(b)) { Toast("Unlocks at level " + BoostUnlock[b], null, 1.4f, 0.7f); HzAudio.Play(Sfx.Invalid, 0.5f); ShakeBooster(b); return; }`
`LastSurvivorGame.cs:418 / void OnBooster(int b)`	`if (b == BShield && _m.ShieldHero) { Toast("Shield is already up!", null, 1.2f, 0.7f); return; }`
`LastSurvivorGame.cs:419 / void OnBooster(int b)`	`if (b == BHeal && _m.U[0].Hp >= _m.U[0].MaxHp) { Toast("You're at full health!", null, 1.2f, 0.7f); HzAudio.Play(Sfx.Invalid, 0.4f); return; }`
`LastSurvivorGame.cs:422 / void OnBooster(int b)`	`ArtPopup("Need a " + BoostName[b] + "?", "Watch a short ad to get one more.", HzUI.Hex(BoostColor[b]), BoostArt[b],`
`LastSurvivorGame.cs:423 / void OnBooster(int b)`	`("Get 1 (ad)", Green, () => HzPlatform.Rewarded(GameId, "booster_" + b, ok => { if (ok) { SetCharges(b, Charges(b) + 1); Toast("+1 " + BoostName[b]); HzAudio.Play(Sfx.Coin, 0.7f); } })),`
`LastSurvivorGame.cs:517 / string DebugCore(string cmd)`	`case "settings": ShowSettingsPopup(); return "ok";`
`LastSurvivorGame.UI.cs:43 / void BuildHud()`	`_levelText = HzUI.Label("T", lvl.transform, "LEVEL " + Level, 60, Color.white); _levelText.rectTransform.Stretch();`
`LastSurvivorGame.UI.cs:45 / void BuildHud()`	`var set = HzIcons.IconButton(_hud, HzIcons.Settings, HzUI.Hex("#7B61FF"), () => { HzAudio.Play(Sfx.Click, 0.6f); ShowSettingsPopup(); }, 110);`
`LastSurvivorGame.UI.cs:55 / void BuildHud()`	`var vst = HzUI.Label("T", vs.transform, "VS", 54, Ink); vst.rectTransform.Stretch();`
`LastSurvivorGame.UI.cs:62 / void BuildHud()`	`var wt = HzUI.Label("T", wp.transform, "WIND", 40, HzUI.Hex("#4C9EEB")); wt.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(160, 70), new Vector2(_data.wind > 0 ? -40 : 40, 0));`
`LastSurvivorGame.UI.cs:80 / void BuildHud()`	`var st = HzUI.Label("T", strip.transform, "BATTLE DECK", 52, Color.white); st.rectTransform.Stretch(); st.AddOutline(new Color(0.6f, 0.3f, 0, 0.6f), 3);`

LastSurvivorGame.UI.cs:100 / void BuildHud()	`var bl = HzUI.Label("BL", Root, "BOOSTERS", 44, Ink);`
LastSurvivorGame.UI.cs:109 / void BuildHud()	`_tipText = HzUI.Label("T", _tip, "", 44, Color.white, TextAnchor.MiddleCenter, false);`
LastSurvivorGame.UI.cs:138 / void BuildHud()	`hp = HzUI.Label("Hp", bg.transform, "", 40, Color.white); hp.rectTransform.Stretch(); hp.AddOutline(new Color(0, 0, 0, 0.45f), 3);`
LastSurvivorGame.UI.cs:139 / void BuildHud()	`var who = HzUI.Label("Who", bg.transform, left ? "YOU" : "ENEMIES", 34, Ink);`
LastSurvivorGame.UI.cs:149 / void BuildHud()	`var btn = HzUI.Button("Card" + slot, parent, null, Color.white, Ink, () => SelectCard(slot), 40, 30);`
LastSurvivorGame.UI.cs:163 / void BuildHud()	`cv.Name = HzUI.Label("Name", cv.Root, "", P ? 42 : 40, Ink);`
LastSurvivorGame.UI.cs:169 / void BuildHud()	`cv.Dmg = HzUI.Label("Dmg", dmgBg.transform, "", 32, Color.white); cv.Dmg.rectTransform.Stretch();`
LastSurvivorGame.UI.cs:176 / void BuildHud()	`var btn = HzUI.Button("Boost" + b, Root, null, HzUI.Hex(BoostColor[b]), Color.white, () => OnBooster(b), 40, 40);`
LastSurvivorGame.UI.cs:188 / void BuildHud()	`bv.Count = HzUI.Label("T", badge.transform, "", 38, Color.white); bv.Count.rectTransform.Stretch();`
LastSurvivorGame.UI.cs:192 / void BuildHud()	`bv.LockText = HzUI.Label("Lv", lk.transform, "LV " + BoostUnlock[b], 34, Color.white); bv.LockText.rectTransform.Place(new Vector2(0.5f, 0), new Vector2(140, 44), new Vector2(0, 30));`
LastSurvivorGame.UI.cs:215 / void RefreshDeck()	`cv.Dmg.text = "DMG " + wd.Damage;`
LastSurvivorGame.UI.cs:239 / void RefreshBoosters()	`bv.Count.text = c > 0 ? c.ToString() : "+";`
LastSurvivorGame.UI.cs:284 / void LevelBanner()	`var t = HzUI.Label("T", bg.transform, "LEVEL " + Level, 96, Color.white); t.rectTransform.Place(new Vector2(0.5f, 1), new Vector2(740, 120), new Vector2(0, -70));`

LastSurvivorGame.UI.cs:285 / void LevelBanner()	var n = HzUI.Label("N", bg.transform, _data.name, 52, HzUI.Hex("#FFD23F")); n.rectTransform.Place(new Vector2(0.5f, 0), new Vector2(740, 80), new Vector2(0, 50));
LastSurvivorGame.UI.cs:298 / void TurnBanner(string text, Color c)	var t = HzUI.Label("T", bg.transform, text.ToUpperInvariant(), 66, Color.white); t.rectTransform.Stretch(); t.AddOutline(new Color(0, 0, 0, 0.3f), 3);
LastSurvivorGame.UI.cs:352 / void WinPopup(int lv, int stars, string msg)	var p = ArtPopup("Victory!", "\n" + msg, HzUI.Hex("#3DBE6E"), "badge_win",
LastSurvivorGame.UI.cs:372 / void ShowSettingsPopup()	var how = HzUI.Button("How", p.Card, "How to play", HzUI.Hex("#FF9F1C"), Color.white, () => { p.Close(); ShowHowTo(); }, 56);
LastSurvivorGame.UI.cs:382 / void ShowHowTo()	ArtPopup("How to play",
LastSurvivorGame.UI.cs:418 / void StartGuide(int b)	ShowTip("Try your new " + BoostName[b] + "! Tap it.", 0);
LastSurvivorGame.View.cs:338 / void BuildWorld()	var t = HzUI.Label("Num", _worldUi, "", 60, Color.white);
LastSurvivorGame.View.cs:391 / void BuildWorld()	var lab = HzUI.Label("Lv", badge.transform, u.Team == 0 ? Mathf.Max(1, (Level + 4) / 5).ToString() : Mathf.Max(1, u.MaxHp / 25).ToString(), 26, Color.white);
LsData.cs:73 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }	new WeaponDef { Id = Knife, Name = "Knife", Art = "w_knife", Color = "#C9D3E0", Damage = 22, Push = 0.5f, Radius = 0.13f, Sticks = true, Unlock = 1, Desc = "Fast and light. Sticks where it lands." },
LsData.cs:74 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }	new WeaponDef { Id = Rock, Name = "Rock", Art = "w_rock", Color = "#8E8A86", Damage = 26, Push = 1.1f, Radius = 0.2f, Bounce = 0.2f, MaxBounces = 1, Unlock = 1, Desc = "A solid rock. Smashes crates." },
LsData.cs:75 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }	new WeaponDef { Id = Bowling, Name = "Bowling Ball", Art = "w_bowling", Color = "#7B3FA0", Damage = 28, Push = 2.0f, Radius = 0.24f, Bounce = 0.3f, MaxBounces = 3, Rolls = true, Unlock = 1, Desc = "Heavy! Big knockback and it keeps rolling." },

`LsData.cs:76 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Grenade, Name = "Grenade", Art = "w_grenade", Color = "#4E8A3A", Damage = 34, Push = 2.0f, Blast = 1.7f, Radius = 0.16f, Explodes = true, Unlock = 2, Desc = "Explodes on impact. Hurts everyone in the blast, you too!" },`
`LsData.cs:77 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Hammer, Name = "Hammer", Art = "w_hammer", Color = "#B5651D", Damage = 36, Push = 1.5f, Radius = 0.2f, Unlock = 4, Desc = "A big bonk." },`
`LsData.cs:78 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Football, Name = "Football", Art = "w_football", Color = "#8B4A22", Damage = 20, Push = 1.4f, Radius = 0.2f, Bounce = 0.78f, MaxBounces = 5, Unlock = 7, Desc = "Bouncy! Bank it off walls and ceilings." },`
`LsData.cs:79 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Axe, Name = "Axe", Art = "w_axe", Color = "#D9363E", Damage = 44, Push = 1.0f, Radius = 0.2f, Sticks = true, Unlock = 11, Desc = "Huge damage, small knockback." },`
`LsData.cs:80 / public int Count(ObKind k) { int n = 0; if (obs != null) foreach (var o in obs) if (o.kind == (int)k) n++; return n; }`	`new WeaponDef { Id = Tnt, Name = "TNT", Art = "w_tnt", Color = "#E8412E", Damage = 50, Push = 3.0f, Blast = 2.4f, Radius = 0.18f, Explodes = true, Unlock = 16, Desc = "A massive blast. Keep your distance!" },`

CHAPTER 11

Artwork, fonts, audio and licences

Replace game images under Resources/lastsurvivor 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/lastsurvivor/Sfx. Preserve each actual load key and importer settings when replacing sounds. Supplied music: Resources/lastsurvivor/Music/arena_loop.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.lastsurvivor`, version, scene orientation and signing. The build command writes `Builds/Android/lastsurvivor.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/lastsurvivor/`. 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.