UNITY 6000.0.62F1 · COMPLETE OFFLINE DOCUMENTATION

Tile Tribe Developer Guide

Install. Edit. Reskin. Build your release.

szekipapa77@gmail.com



CHAPTER 01

Start here: Tile Tribe

Tile Tribe is a complete editable Unity game template. Tap bright uncovered tiles to move them into a seven-slot tray. Three matching creatures clear together. Dark covered tiles must be uncovered first. Clear the pool and any parked shelf tiles to win; filling the tray without a match loses. Sand hides pictures until exposed; clams open after their displayed trio requirement.

Included: 25 curated levels, seeded continuation, visual level editor and validation, four guided helpers and portrait/landscape scenes. Saving is local. No multiplayer server, purchases, HLK, cloud accounts or remote LiveOps backend is included. Keep the required private game Kit. Szekipapa77 Ads is publisher-owned integration code; provider SDK binaries are excluded. Import only one game Kit per project because they share class names.

CHAPTER 02

Installation and first play

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

1. Extract the standalone project to a short writable path and open it with the matching editor. Let package import finish.
2. For native package import, create a matching URP project, choose Assets > Import Package > Custom Package and import every game, Kit, Ads and WebGL template entry.
3. Run Szekipapa77 > Tile Tribe > Setup. This configures rendering and player settings; inspect changes when combining with an existing project.
4. Set Active Input Handling to Input Manager (Old), or Both if your project also has the Input System. New Input System only is insufficient.
5. Open `Assets/Games/TileTrio/Scenes/TileTrio_Mobile.unity`, press Play, dismiss the level card and follow the first-level guidance.
6. Stop Play mode before data editing. Retain .meta files/GUIDs. No separate TextMesh Pro import is needed by the classic Unity UI Text path.

CHAPTER 03

Project map and actual scripts

All paths here are under `Assets/Games/TileTrio`. Scripts owns gameplay; Editor owns visual tools; Kit supplies UI, audio, saving and platform code; Resources contains data/art/audio; Settings contains URP; Scenes contains mobile and WebGL entry points. The UI is created in code at runtime, not authored as a persistent Canvas prefab.

- `Scripts/TileTrioGame.cs`
- `Scripts/TileTrioGame.Hud.cs`
- `Scripts/TileTrioGame.View.cs`
- `Scripts/TtArt.cs`
- `Scripts/TtCore.cs`
- `Scripts/TtGen.cs`
- `Scripts/TtLevels.cs`
- `Scripts/TtSfx.cs`

Read `TileTrioGame.cs` for input/flow, `TileTrioGame.Hud.cs` for menus/popups, `TileTrioGame.View.cs` for the board/view, `TtCore.cs` for the model and `TtLevels.cs` for level loading. Advertising settings are outside the game at Assets/Szekipapa77Ads.

CHAPTER 04

Gameplay, input and progression

Tap bright uncovered tiles to move them into a seven-slot tray. Three matching creatures clear together. Dark covered tiles must be uncovered first. Clear the pool and any parked shelf tiles to win; filling the tray without a match loses. Sand hides pictures until exposed; clams open after their displayed trio requirement.

The curated bank contains 25 authored levels followed by generated continuation. The mobile scene requests portrait, while the WebGL-named scene requests landscape. Supplied build menu targets currently choose the portrait scene for both platforms; choose the landscape scene explicitly in a Build Profile if desired. Test layouts and controls after a near-startup scene replacement as well as after a finished round.

CHAPTER 05

Tutorials and all four helpers

The first level includes guided controls; helper unlocks introduce each helper and its first use. Restart with fresh local progress to test first-launch guidance. Open in-game help to review controls.

Helper	Unlock	Free charges	Effect
Undo	3	3	Returns the last unmatched tray tile. Already matched tiles stay cleared.
Net	6	2	Removes an entire trio, including covered tiles.
Swirl	10	2	Redistributes the remaining creature pictures.
Bucket	15	1	Parks three tray tiles on a shelf; tap shelf tiles to return them. Needs occupied tray slots.

A guided first use can be free and is not the same as spending a normal saved charge. Exhausted charges may request a rewarded placement. Confirm success/cancel/failure callbacks and grant rewards exactly once. Test each helper with valid preconditions: armed shots need a subsequent shot, Undo needs an unmatched pick and Bucket needs occupied tray slots.

CHAPTER 06

Visual editor: your first edited level

Open Szekipapa77 > Tile Tribe > Level Editor. Exact implementation: `Editor/TileTrioLevelEditor.cs`.

Edit Name, Chapter, First-time hint, Design difficulty, Breather and Shape. Choose a layer and Tile, Picture, Sand, Clam or Erase brush. Keep total and per-picture counts divisible by three. Assign pictures (solvable) retains the layout and assigns through a clearing order. Regenerate layout replaces the selected layout; back it up first.

1. Back up levels.json and duplicate a simple early level.
2. Make one small change and use a distinctive visible name/note where supported.
3. Validate using the editor and fix every reported issue.
4. Save explicitly, stop and restart Play mode and open the edited index.
5. Check both supplied scenes and rebuild demos after data changes.

CHAPTER 07

Generation, insertion and replacement

Count (1–100), Start at level #, Seed, Start/End difficulty, Breather every (0–8), Sand and Clams. Replace that range (else insert) removes an existing range when checked; unchecked inserts and shifts later records. Start 26 appends to the original bank. Generation uses TtGen and solver validation.

Worked extension: keep a backup, generate five candidates after the original 25, use a moderate difficulty curve, validate every candidate, then Save. Reload JSON to confirm the count changed. Deliberately choose replacement only when you intend to overwrite those records. Runtime generated continuation does not write new project JSON assets. Solver/bot validation does not replace manual play testing.

CHAPTER 08

Actual level schema

Canonical file: `Assets/Games/TileTrio/Resources/tiletrio/Levels/levels.json`. Root levels array order defines visible level numbering; preserve version/type fields and numeric indices. The following is the actual first record, not a hypothetical schema:


```
{
 "Layers": 2,
 "Kinds": 3,
 "SandCount": 0,
 "ClamCount": 0,
 "name": "First Splash",
 "hint": "Tap a free tile to send it to the tray.\nThree alike wash away!",
 "shape": "pyramid",
 "chapter": "Shallows",
 "difficulty": 0.05,
 "breather": false,
 "seed": 1006,
 "score": 4.1666665,
 "tiles": [
  {
   "l": 0,
   "x": 0,
   "y": 0,
   "t": 5,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 2,
   "y": 0,
   "t": 1,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 4,
   "y": 0,
   "t": 3,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 0,
   "y": 2,
   "t": 1,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 2,
   "y": 2,
   "t": 1,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 4,
   "y": 2,
   "t": 1,
   "s": 0,
   "c": 0
  },
  {
   "l": 0,
   "x": 0,
   "y": 4,
   "t": 3,
   "s": 0,
   "c": 0
  },
  {
```

```
      "l": 0,
      "x": 2,
      "y": 4,
      "t": 3,
      "s": 0,
      "c": 0
    },
    {
      "l": 0,
      "x": 4,
      "y": 4,
      "t": 1,
      "s": 0,
      "c": 0
    },
    {
      "l": 0,
      "x": 0,
      "y": 6,
      "t": 5,
      "s": 0,
      "c": 0
    },
    {
      "l": 0,
      "x": 2,
      "y": 6,
      "t": 1,
      "s": 0,
      "c": 0
    },
    {
      "l": 0,
      "x": 4,
      "y": 6,
      "t": 3,
      "s": 0,
      "c": 0
    },
    {
      "l": 1,
      "x": 1,
      "y": 3,
      "t": 3,
      "s": 0,
      "c": 0
    },
    {
      "l": 1,
      "x": 3,
      "y": 3,
      "t": 3,
      "s": 0,
      "c": 0
    },
    {
      "l": 1,
      "x": 1,
      "y": 5,
      "t": 5,
      "s": 0,
      "c": 0
    }
  ]
}
```


Use the visual editor to create structured coordinates, arrays and indices. Keep JSON valid and preserve booleans/numbers. Validate through TileTrioLevelEditor.cs before playing. Do not rename resource IDs or save keys during a display-text change.

CHAPTER 09

Worked reskin: interface colours

Open `Scripts/TtArt.cs`, find `Foam` with current hex `#4FD1B5`. Stop Play mode and back up the file. Change only that value to `#42C9B0`, keep the symbol/type, allow compilation and restart the scene. Review HUD, buttons, helper captions, result/settings popups and landscape layout. For Guess Five, check Night and Day and maintain clear clue contrast. Editor preview palettes can be separate; change them deliberately if matching is needed.

Exact runtime palette/material assignments follow. These show which file and assignment owns a colour; editing an editor-only colour does not reskin the runtime. Rebuild after accepting the change.

Exact source reference	Current assignment
`TileTrioGame.cs:32`	public static readonly string[] BoostColor = { "#2EB8A6", "#3FA7D6", "#8C7BD8", "#FF7A6B" };
`TileTrioGame.Hud.cs:86`	var shine = HzUI.Panel("Shine", pill.transform, HzUI.Hex("#2A8796"), 40); shine.raycastTarget = false;
`TileTrioGame.Hud.cs:162`	var pool = HzUI.Panel("Slot", frame.transform, HzUI.Hex("#2E8E9A"), Mathf.Max(8, (int)(_trayTile * 0.2f))); pool.raycastTarget = false;
`TileTrioGame.Hud.cs:167`	var glint = HzUI.Panel("Glint", pool.transform, HzUI.Hex("#5FB5BE"), 8); glint.raycastTarget = false;
`TileTrioGame.Hud.cs:176`	var frame = HzUI.Panel("Shelf", _layer, HzUI.Hex("#E9D3AE"), 30);
`TileTrioGame.Hud.cs:218`	var lockGo = HzUI.Panel("Lock", v.Root, HzUI.Hex("#7E9FAA"), (int)(size * 0.32f)); lockGo.raycastTarget = false;
`TileTrioGame.Hud.cs:477`	var tip = HzUI.Panel("Tip", _layer, HzUI.Hex("#164A6B"), 34);
`TileTrioGame.View.cs:63`	var g = HzUI.Img("BgImg", _bgLayer, HzUI.Gradient(HzUI.Hex("#7FD6D0"), HzUI.Hex("#1C8C98")), Color.white); g.preserveAspect = false; g.raycastTarget = false;
`TileTrioGame.View.cs:157`	v.Side = HzUI.Panel("Side", v.Root, HzUI.Hex("#6FBFB6"), r); v.Side.raycastTarget = false;
`TileTrioGame.View.cs:165`	v.Rim = HzUI.Img("Rim", v.Root, HzUI.RoundRect(r, Mathf.Max(2, (int)(s * 0.035f))), HzUI.Hex("#8FD3CB")); v.Rim.type = Image.Type.Sliced; v.Rim.preserveAspect = false; v.Rim.raycastTarget = false;
`TileTrioGame.View.cs:177`	else { v.SandImg = HzUI.Panel("Sand", v.Root, HzUI.Hex("#E7C887"), r); v.SandImg.rectTransform.Place(new Vector2(0.5f, 0.5f), Vector2.one * s * 0.9f); }
`TileTrioGame.View.cs:183`	else { v.ClamImg = HzUI.Panel("Shell", clamGo, HzUI.Hex("#9CD3F0"), r); v.ClamImg.rectTransform.Place(new Vector2(0.5f, 0.5f), new Vector2(s * 0.95f, s * 0.6f)); }

`TtArt.cs:9`	public static readonly Color Teal = HzUI.Hex("#0E6E7E"), Navy = HzUI.Hex("#123B5A"), Sand = HzUI.Hex("#FFF3DC"), SandDark = HzUI.Hex("#F2DDB6"),
`TtArt.cs:10`	Foam = HzUI.Hex("#4FD1B5"), Coral = HzUI.Hex("#FF7A6B"), Sun = HzUI.Hex("#FFC94A"), Aqua = HzUI.Hex("#BFE8E4"), Sky = HzUI.Hex("#3FA7D6"),
`TtArt.cs:11`	Lilac = HzUI.Hex("#8C7BD8"), Leaf = HzUI.Hex("#5DBB63"), Ink = HzUI.Hex("#1E4A66"), Sub = HzUI.Hex("#5F8597"),
`TtArt.cs:12`	CardBg = HzUI.Hex("#F4FBFA"), Locked = HzUI.Hex("#9DB5BD"), PopupCard = HzUI.Hex("#F3FBF9"), PopupInner = HzUI.Hex("#DDF2EE"),
`TtArt.cs:13`	Wood = HzUI.Hex("#B98352"), WoodDark = HzUI.Hex("#8A5A33");
`TtArt.cs:29`	"#F25C3B", "#FF7FA8", "#F7A86B", "#FFC94A", "#9C6ADE", "#4CB963", "#F49AD0", "#2EB8A6", "#7EC8F2",
`TtArt.cs:30`	"#5A8EDB", "#FF6F4F", "#E8975A", "#3F7FD6", "#5B3F8C", "#7D9BB8", "#D63A3A", "#B5D36B", "#3E9C4F",
`TtArt.cs:39`	new Chapter { Name = "Shallows", Tint = new Color(0.75f, 1f, 0.95f, 0.05f), Glow = HzUI.Hex("#BFF3EA") },
`TtArt.cs:40`	new Chapter { Name = "Coral Garden", Tint = new Color(1f, 0.6f, 0.55f, 0.16f), Glow = HzUI.Hex("#FFD3C8") },
`TtArt.cs:41`	new Chapter { Name = "Kelp Forest", Tint = new Color(0.35f, 0.75f, 0.35f, 0.18f), Glow = HzUI.Hex("#CDEFC0") },
`TtArt.cs:42`	new Chapter { Name = "Moon Lagoon", Tint = new Color(0.25f, 0.2f, 0.55f, 0.34f), Glow = HzUI.Hex("#D5D0FF") },
`TtArt.cs:43`	new Chapter { Name = "Deep Blue", Tint = new Color(0.03f, 0.12f, 0.35f, 0.42f), Glow = HzUI.Hex("#BFD8FF") },

CHAPTER 10

Worked reskin: visible text

Open `Scripts/TileTrioGame.Hud.cs`, find the displayed literal `"How to Play"` and replace it with `"Playing Guide"`. In HzUI.Label/Button, the first string can be the hierarchy object name; change the displayed text argument. Keep object identifiers, GameId, save/resource keys and asset filenames stable.

Preserve interpolation, numeric ToString formatting, line breaks and ON/OFF refresh logic. Update initial construction and later refresh assignments. Helper name/help sources are BoostName in TileTrioGame.cs and BoostDesc in TileTrioGame.Hud.cs. Level names and hints are separate JSON content. Visible title appears in the game Title property, HzGameInfo and sometimes separate screen lettering/art; review all branding deliberately.

There is no automatic translation of hard-coded game strings. A translation table must be called by each assignment. Word dictionaries/answer banks are gameplay data and do not translate Guess Five menus. Test title, tutorial, helper unlocks, HUD, settings/result popups and refreshed labels in both orientations.

Exact source reference	Current assignment
`TileTrioGame.cs:29`	public static readonly string[] BoostName = { "Undo", "Net", "Swirl", "Bucket" };
`TileTrioGame.cs:116`	var p = TtPopup(LevelLabel(), msg, Teal, LevelCardArt(), D.name + " - " + D.chapter, ("Play", Foam, (Action)Begin));
`TileTrioGame.cs:120`	string LevelLabel() => "Level " + _playLevel.ToString(CultureInfo.InvariantCulture);
`TileTrioGame.cs:205`	TtPopup(title, msg, Coral, TtArt.Get("booster_bucket"), null, buttons.ToArray());
`TileTrioGame.cs:357`	Toast(BoostName[b] + " unlocks at level " + BoostUnlock[b].ToString(CultureInfo.InvariantCulture), null, 1.3f, ToastY); HzAudio.Play(Sfx.Cancel, 0.4f);
`TileTrioGame.cs:384`	TtPopup("+1 " + BoostName[b], "Out of " + BoostName[b] + ".\nWatch a short ad for one more?", HzUI.Hex(BoostColor[b]), TtArt.Get(BoostArt[b]), null,
`TileTrioGame.Hud.cs:66`	var title = HzUI.Label("Title", _layer, "Tile Tribe", 90, Sun, TextAnchor.MiddleLeft);
`TileTrioGame.Hud.cs:73`	var rb = HzUI.Button("Restart", _layer, "", Coral, Color.white, () => { if (_playing || _ended) Restart(); }, 40, 36);
`TileTrioGame.Hud.cs:76`	var sb = HzUI.Button("Settings", _layer, "", Lilac, Color.white, ShowSettingsPopup, 40, 36);
`TileTrioGame.Hud.cs:88`	_levelTxt = HzUI.Label("Level", pill.transform, "", 60, Color.white); _levelTxt.AddOutline(new Color(0, 0.15f, 0.2f, 0.4f), 3);
`TileTrioGame.Hud.cs:90`	_nameTxt = HzUI.Label("Name", pill.transform, "", 32, new Color(1, 1, 1, 0.95f), TextAnchor.MiddleCenter, false); _nameTxt.fontStyle = FontStyle.Bold;

`TileTrioGame.Hud.cs:97`	var t = HzUI.Label("L", b.transform, label, 40, Color.white); t.rectTransform.Stretch(); t.rectTransform.offsetMin = new Vector2(80, 0);
`TileTrioGame.Hud.cs:113`	_leftTxt = HzUI.Label("Left", card.transform, "", tall ? 46 : 42, Ink, TextAnchor.MiddleLeft);
`TileTrioGame.Hud.cs:115`	_tapsTxt = HzUI.Label("Taps", card.transform, "", 30, Sub, TextAnchor.MiddleLeft, false); _tapsTxt.fontStyle = FontStyle.Bold;
`TileTrioGame.Hud.cs:178`	var lbl = HzUI.Label("ShelfLbl", frame.transform, "Bucket shelf", 28, Ink, TextAnchor.MiddleCenter, false); lbl.fontStyle = FontStyle.Bold;
`TileTrioGame.Hud.cs:204`	var btn = HzUI.Button("Booster" + b, parent, null, Color.white, Color.white, () => OnBooster(bb), 40, (int)(size * 0.32f));
`TileTrioGame.Hud.cs:227`	v.BadgeText = HzUI.Label("N", bd.transform, "", (int)(size * 0.24f), Color.white); v.BadgeText.rectTransform.Stretch();
`TileTrioGame.Hud.cs:229`	v.Caption = HzUI.Label("Cap", v.Root, "", 30, Ink);
`TileTrioGame.Hud.cs:239`	_nameTxt.text = D.name + " - " + D.chapter + (D.breather ? " - breather" : "");
`TileTrioGame.Hud.cs:241`	_leftTxt.text = left == 0 ? "All clear!" : left.ToString(CultureInfo.InvariantCulture) + (left == 1 ? " tile left" : " tiles left");
`TileTrioGame.Hud.cs:242`	if (_tapsTxt) _tapsTxt.text = "Taps: " + _picks.ToString(CultureInfo.InvariantCulture) + " Trios: " + S.Trios.ToString(CultureInfo.InvariantCulture);
`TileTrioGame.Hud.cs:261`	v.BadgeText.text = c > 0 ? c.ToString(CultureInfo.InvariantCulture) : "+";
`TileTrioGame.Hud.cs:263`	v.Caption.text = un ? BoostName[b] : "Level " + BoostUnlock[b].ToString(CultureInfo.InvariantCulture);
`TileTrioGame.Hud.cs:330`	var t = HzUI.Label("Title", ribbon.transform, title, 72, Color.white); t.rectTransform.Stretch(6);
`TileTrioGame.Hud.cs:335`	var st = HzUI.Label("Sub", card.transform, subtitle, 40, Teal); st.rectTransform.Place(new Vector2(0.5f, 1), new Vector2(cw - 80, 50), new Vector2(0, y - 25));
`TileTrioGame.Hud.cs:340`	p.Message = HzUI.Label("Msg", card.transform, message, 44, Ink, TextAnchor.MiddleCenter, false);
`TileTrioGame.Hud.cs:375`	TtPopup("New: " + BoostName[b] + "!", BoostDesc[b] + "\nYou get " + BoostStart[b].ToString(CultureInfo.InvariantCulture) + " free!", HzUI.Hex(BoostColor[b]), TtArt.Get(BoostArt[b]), null,
`TileTrioGame.Hud.cs:383`	var p = TtPopup("Settings", "", Lilac, null, null,
`TileTrioGame.Hud.cs:397`	static string SoundLabel() => HzAudio.Muted ? "Sound: OFF" : "Sound: ON";
`TileTrioGame.Hud.cs:398`	static string MusicLabel() => HzAudio.MusicMuted ? "Music: OFF" : "Music: ON";

`TileTrioGame.Hud.cs:412`	TtPopup($"How to Play {i + 1}/{HelpPages.Length}", HelpPages[i], Sky, i == 0 ? TtArt.Get("keyart") : i == 3 ? TtArt.Get("clam") : null, null,
`TileTrioGame.Hud.cs:483`	_tipText = HzUI.Label("T", tip.transform, text, IsPortrait ? 38 : 34, Sand, TextAnchor.MiddleCenter, false);
`TileTrioGame.Hud.cs:546`	Toast("Nice! " + BoostName[b] + " is yours", Sun, 1.3f, ToastY);
`TileTrioGame.View.cs:171`	v.Letter = HzUI.Label("L", fb.transform, "", (int)(s * 0.38f), Color.white); v.Letter.rectTransform.Stretch(); v.Letter.AddOutline(new Color(0, 0, 0, 0.3f), 2);
`TileTrioGame.View.cs:188`	v.ClamText = HzUI.Label("N", badge.transform, "", (int)(s * 0.24f), Color.white); v.ClamText.rectTransform.Stretch();
`TileTrioGame.View.cs:420`	var t = HzUI.Label("Combo", _fxLayer, "Wave x" + n.ToString(System.Globalization.CultureInfo.InvariantCulture) + "!", IsPortrait ? 84 : 70, Sun);

CHAPTER 11

Artwork, fonts and audio

Game assets live in `Resources/tiletrio`. Replace images through Unity while retaining resource keys, .meta GUIDs, Sprite/UI importer settings, alpha, pivots and slicing. Procedural fallbacks live in the view/art sources and may need their own changes; replacing a PNG does not automatically alter fallback graphics.

The private Kit supplies fonts and UI helpers. Retain included font licence notices and verify glyph coverage for new languages. Music resource: `tide_loop.ogg`. Preserve load keys when replacing clips, review compression/loop boundaries, and test Sound and Music toggles separately. Inspect the Sfx/HzAudio code for current sound bindings.

CHAPTER 12

Saving and resets

HzSave wraps PlayerPrefs, coalesces writes and flushes on focus loss/quit. WebGL uses browser IndexedDB tied to the hosting origin. Clearing site data or using a new origin starts a new save; this is not cloud synchronization. GameId-prefixed keys and some JSON progress hold levels, stars, helper counts, tutorial flags, audio options and statistics where implemented.

Test fresh launch, a completed level, helper consumption, audio toggles, focus loss, quit and relaunch. Use a separate development profile for resets. Preserve existing save keys for a visual reskin; changing them discards continuity. Inspect calls to HzSave in the listed scripts before adding a custom migration.

CHAPTER 13

Advertising setup and scope

Settings asset: `Assets/Szekipapa77Ads/Resources/AdsSettings.asset`. Inspect backend, useGoogleTestIds and missingSdkFallback through the publisher Ads editor. This buyer contains integration/adapter code, not external provider SDK binaries. Missing-provider fallback and WebGL ads are simulated, not live commercial impressions.

For native real ads, obtain the supported provider from its official source in your working project, configure your own app/placement IDs and test with test IDs. Verify banner/interstitial/rewarded success, cancel and failure. Before production, supply your own IDs and disable simulated missing-SDK fallback. The build tools reject unsafe production fallback settings. No purchases or remote backend is supplied.

CHAPTER 14

Android release

Install matching Android support in Unity Hub. Select Android, use the mobile scene, run Setup and review application ID `com.szekipapa77.tiletrio`, version, orientation and signing. Build Android writes `Builds/Android/tiletrio.apk`. Configure your own keystore and distribution settings; a demo APK is not a signed production release.

A provider-free build simulates ads. A supplied test-provider demo is not evidence of production account configuration. Use a Build Profile for your distribution format. On real hardware test touch, safe areas, audio focus, restart and ad callbacks. Keep signing passwords/keystores/private service files outside buyer archives. Successful compilation is not a physical-device play test.

CHAPTER 15

WebGL release and hosting

Install WebGL support, select the WebGL target and run Setup and Build WebGL. Output: `Builds/WebGL/tiletrio/`. Serve index.html and every Build/StreamingAssets payload over HTTPS, not file://. Match compression, MIME and Content-Encoding to the host. The shell template controls the outer page; scene selection controls game orientation.

The publisher demo uses HyperCodeStore. Upload actual loader/framework/data/WASM files, not just a link; include the TemplateData shell required by the host. Test in a fresh anonymous session: loading, real pointer/keyboard input, audio after interaction, save/reload and mobile sizing. Rebuild and re-upload every changed game artifact.

CHAPTER 16

Troubleshooting

Compilation errors: restore matching packages and import the entire private Kit/Ads layer; keep only one game Kit. **Pink or invisible objects:** run Setup and inspect Graphics and every Quality URP assignment. **No input:** check classic input handling, canvas focus and blocking modal. **Missing resources:** restore exact keys/importers/GUIDs.

Edited level absent: stop Play, Save, reload the canonical JSON and check saved progress/play overrides. **No real ads:** provider SDK is intentionally absent; inspect installed adapter and fallback. **Text reverts:** change both build and refresh assignments from the reference table. **Scene reload exception:** retain owner-lifetime and generation checks on delayed callbacks; do not remove those guards.

CHAPTER 17

Support and release checklist

Support: szekipapa77@gmail.com. Include game title, exact Unity version, platform, first Console error and reproduction steps; never send credentials.

1. Pass a clean import and play both scenes.
2. Play tutorial, normal win/failure and every helper activation/effect.
3. Check reskin strings, settings, save/reload and both layouts.
4. Rebuild Android/WebGL after code or data changes and verify final output bytes.
5. Verify public APK, browser demo and full documentation links before publishing.
6. Keep licences and exclude private tooling/capture/workflow material from customers.