UNITY 6000.0.62F1 · COMPLETE OFFLINE DOCUMENTATION

Hex Tower Sort Developer Guide

Install. Edit. Reskin. Build your release.

szekipapa77@gmail.com



CHAPTER 01

Start here: your first sorting board

Hex Tower Sort is a 3D sorting-game template. Drag a candy tower from the three-slot tray to an empty hexagonal cell. Matching top colours pour together; groups of ten or more tiles pop. Make space while meeting the board's tile, crate and gem goals.

The project includes 25 authored boards, a deterministic continuation generator, four unlockable helpers, guided introductions, local saving, sound and music, a visual board painter and bot validation. It supplies portrait and landscape entry scenes. The towers and board are procedural 3D objects, rather than a collection of interchangeable screenshot sprites.

This handbook takes you from first play to authoring a board, changing the palette and wording, and building your own release. All paths are relative to the Unity project. Keep the game's private Kit and Szekipapa77 Ads folders; they supply rendering support, interface, saving, platform hooks and ad callbacks. Provider SDKs are not bundled. There are no supplied purchases, online accounts, multiplayer servers, HLK or remote LiveOps services.

Use one supplied game per clean project. The private Kits use shared class names and are not designed to be combined unchanged. Support: szekipapa77@gmail.com.

CHAPTER 02

Installation and first play

Requirements

Use Unity 6000.0.62f1, Universal Render Pipeline 17.0.4 and Unity UI 2.0.0. Keep the official dependencies from Packages/manifest.json. This game uses Unity UI Text and its supplied fonts; a separate TextMesh Pro import is unnecessary. For native builds install Android Build Support and Unity-matched SDK, NDK and OpenJDK through Hub. Browser builds require WebGL Build Support.

Standalone project

1. Extract the source to a short writable path such as C:/UnityWork/HexTower. Open the folder containing Assets, Packages and ProjectSettings through Hub.
2. Select Unity 6000.0.62f1, wait for packages and compilation, and fix the first Console error before continuing.
3. Run Szekipapa77 > Hex Tower Sort > Setup. Inspect the resulting Graphics, Quality and Player settings, particularly when importing into another project.
4. Set Active Input Handling to Input Manager (Old), or Both if you also install the new input system. The gameplay uses Input.GetMouseButton; New Input System only is insufficient.
5. Open Assets/Games/HexTower/Scenes/HexTower_Mobile.unity and press Play. Follow the hand: drag a tray tower onto an empty board cell, then place a matching colour beside it.
6. Keep playing until the goal bar fills. Stop Play mode before editing the authored level bank.

Native package import

1. Back up the destination and install the matching official URP and UI packages.
2. Choose Assets > Import Package > Custom Package. Import the game, its Kit, Szekipapa77Ads and the publisher WebGL template together.
3. Run Setup, check input and pipelines, and repeat first play. Preserve .meta files and their GUIDs.

A missing optional advertising provider leads to the supplied simulated fallback. It does not explain compile errors or missing game resources; diagnose those separately.

CHAPTER 03

The file map

Game root: `Assets/Games/HexTower`. Keep this structure and its Resources names.

File under game root	Responsibility
Scripts/HexTowerGame.cs	Lifecycle, drag input, placement, results, helpers and charge saving.
Scripts/HexTowerGame.View.cs	Procedural world, tile colours, materials, camera, lighting and effects.
Scripts/HexTowerGame.UI.cs	HUD, goal counters, popups, settings, tool captions and tutorial hands.
Scripts/HexLogic.cs	Pure engine, adjacency, pouring, pop rules, refill and bot moves.
Scripts/HexData.cs	JSON model, editable grid and conversion to engine definitions.
Scripts/HexLevels.cs	Authored resource cache and generated continuation.
Scripts/HexGen.cs	Shapes, seeded generator, difficulty curve and bot validation.
Scripts/HexMeshes.cs / HexSfx.cs	Procedural mesh building / game audio resource names.
Editor/HexTowerLevelEditor.cs	Board painter, refill bands, list editing, generation and validation.
Kit/Editor/HzTools.cs	Setup, export and platform build menus.
Kit/Runtime/HzGame.cs / HzCore.cs	Base scene lifecycle, local saving and platform service calls.
Kit/Runtime/HzUI.cs / HzAds.cs	Runtime interface and publisher advertising bridge.

The bank is `Resources/hextower/Levels/levels.json`. Art, Materials, Sfx and Music are siblings under Resources/hextower. Fonts and shared UI resources are under Kit/Resources/hextower/kit. Entry scenes are Scenes/HexTower_Mobile.unity and Scenes/HexTower_WebGL.unity. The normal WebGL menu builds the portrait Mobile scene; landscape must be selected deliberately.

The separate integration lives in Assets/Szekipapa77Ads. Its settings are Resources/AdsSettings.asset. The browser shell is Assets/WebGLTemplates/Szekipapa77. Do not remove these as if they were another game. Most game UI is built at runtime, so scene hierarchy children are not the source of its visible wording.

CHAPTER 04

Controls, matching, goals and results

Place and pour

HandleInput in HexTowerGame.cs reads pointer-down, held movement and release. Start on a tray tower, drag above the board and release on an empty unblocked cell. A highlight previews the landing cell. Releasing away from a valid cell returns the tower to its tray slot without placing it. Input over interface controls is excluded by PointerOverUi.

DoPlace records an undo snapshot, calls HexEngine.Place and animates the resulting events. Matching adjacent top runs pour together and can reveal a second colour underneath. The pop threshold is ten tiles. Chain resolution is part of the engine, not an independently awarded cosmetic score. A new three-slot tray appears when the current tray is empty and play can continue.

Obstacles and goals

Sugar locks open after enough tiles have popped; their labels show the remaining requirement. Crates have one to three hits and are damaged by a neighbouring pop. Gems lie under cells and are collected by popping a tower on that cell. A board may require several goal types together: HexEngine.Won requires every positive tile, crate and gem target to be satisfied. A zero target disables that goal.

End of a board

A win advances hextower_level and offers Next Level. The result includes popped tiles and move count. If no empty playable cells remain and the goals are unfinished, the board is lost. The first loss offers a rewarded rescue that clears four towers; a declined or unsuccessful reward returns to a retry. Rescue is once per attempt. A retry changes the attempt-dependent random seed, so a new attempt need not refill identically.

This game tracks objective progress and placements, not a multiplayer ranking. The HUD goal counts and victory summary should not be described as an online leaderboard.

CHAPTER 05

First-play and helper tutorials

The first board has tut:1. IntroFlow starts a guided sequence, and RefreshTutorial in HexTowerGame.UI.cs displays: Drag a tower onto the board; Put the same colour right next to it; Same colours slide together. 10 on top pop! The animated hand points from a real tray slot to a suitable board cell. PlaceAndResolve advances the steps; the goal-bar explanation follows.

UnlockPopup and StartGuide teach helpers separately. The charge grant is recorded once using hextower_seenb plus the helper index. CompleteGuide writes hextower_guided plus the index. Smash asks for a real tower target, Lift has pick-up and destination steps, and Undo first requires a placement to undo. Reroll changes the tray directly.

Crates, locks and gems have separate one-time introductions using hextower_seen_crate, hextower_seen_lock and hextower_seen_gem. Authored hints use hextower_hint plus the level number. A returning player's flags can suppress introductions. Test with a controlled fresh save as well as an existing save.

When changing tutorial copy, edit RefreshTutorial, RefreshGuide, ModeText and the IntroFlow/Intro calls, not only a shared Kit label. Repeat both orientations and an orientation rebuild. Do not reset all PlayerPrefs in a customer release merely to repeat a lesson.

CHAPTER 06

Four tools, unlocks and charges

Tool	Unlock / starting charges	Actual effect
Smash	3 / 3	Choose an existing tower or crate and remove it.
Reroll	6 / 2	Replace all three tray towers with fresh ones.
Lift	10 / 3	Pick a board tower; move it to an empty cell or swap with another tower.
Undo	15 / 2	Restore the snapshot from the last placement.

The parallel BoostUnlock, BoostStart, BoostName, BoostArt, BoostColor and BoostDesc arrays are in HexTowerGame.cs. Keep their four entries in the same order. RefreshBoosters in the UI partial chooses locked captions, charge badges and mode highlighting. Charges are hextower_b0 through hextower_b3.

OnBooster routes actions and rejects locked tools, busy animations or a conflicting active guide. Smash/Lift select a mode and consume a charge on a valid target. Reroll consumes one immediately. Undo requires _undo; Nothing to undo yet is a real precondition, not an error in the button. Reroll, Smash, Lift and rescue invalidate the previous undo snapshot.

The source guide says This one is free because initial charges are granted; the actions still decrement charge counts. Do not promise unlimited free guided use. Empty tools offer Get 1 (ad); only a successful earned callback adds one charge. Changing an unlock needs a fresh or migrated save to test grant and guide behaviour. A helper demonstration must show the subsequent board/tray effect, not only the button flash.

CHAPTER 07

Make one level in the editor

1. Exit Play mode and back up levels.json. Open Szekipapa77 > Hex Tower Sort > Level Editor.
2. Select First Bite and press Dup. The copy is inserted after the selected entry. Give it a new Name and a player-facing First-time hint.
3. Keep the simple tile goal while learning. Set Goal: hexes to 40, crates and gems to zero. Keep Tutorial checked only when you want the first-placement lesson.
4. Choose Cell and paint a small connected board. Use Hole to remove outer cells. Keep empty space for placing towers.
5. Choose Tower and enter 00000 as the bottom-to-top stack. Paint one tower. Put 00000,00000,11111 in Opening tray queue. Matching indices refer to palette entries, not text colours.
6. Choose an Easy refill preset or retain the original band. Validate the selected board with several seeds. Ease goals, blockers or refill bands when validation warns.
7. Save writes the JSON. Quick test saves, opens the selected level in the WebGL-named scene and enters Play mode. Test the Mobile scene separately.

Painter and list controls

The list offers Add, Dup, Del, Up and Down. Moving entries changes their runtime indices and can alter existing saved progress. Columns and Rows resize from three to nine; odd column counts help symmetry because odd columns sit half a cell lower. The editor keeps Unity Undo snapshots and an unsaved-changes marker.

Brushes are Cell, Hole, Lock, Crate, Gem, Tower and Clear. Lock uses a pop threshold; Crate uses hit points; Gem toggles a gem on a cell. Tower filters its stack to digits 0 through 7. Clear removes blockers, gem and tower content while retaining the cell. Choose deliberately between Clear and Hole.

Validate All checks the bank; Quick test is optional for editing. Save, stop the previous player and restart when evaluating data changes. Keep backups before large paint, reorder or generation operations.

CHAPTER 08

Generate a batch and tune refill

Worked safe append

1. Back up the 25-level bank and open Auto Generate. Set Count 5, Start at level # 26 and Seed 1234.
2. Set Start difficulty 0.4, End difficulty 0.8 and Breather every 5. Leave Replace that range unchecked.
3. Generate. Five levels are inserted at index 25, after the original bank. Validate the result and inspect actual play before saving.
4. Save deliberately. If appending to a longer bank, use its current count plus one; 26 is not a universal append command.

Generate computes Clamp(Start-1,0,current count), irrespective of the selected list entry. With Replace that range enabled it removes the available entries in that range and then inserts generated ones. Without it, later levels shift. Duplicating a selected row and generating a range therefore use different placement rules.

HexGen.Curve combines a rising base with a sawtooth challenge cycle and easier breather. Shape generation has 14 named patterns. Runtime continuation after the authored list calls ForLevel with a deterministic seed derived from the level number and bot validation. Validation uses sampled greedy-bot seeds; its reported success is evidence for tuning, not a guarantee of every human strategy.

Refill bands

Each band activates from a normalized progress point. Colors is the number of active palette colours. MinC/MaxC control colour segments per tower; MinF/MaxF control tiles per segment. Match and Full are mercy probabilities used near a full board, for partial matching or a guaranteed-clear recolour. Keep bands sorted and ranges sensible. Easy/Mid/Hard buttons provide editable presets; they do not change the underlying pop threshold.

Opening tray queue supplies exact stack strings before normal generation. Colour order can override the seeded mapping. Keep zero-based indices 0..7. Revalidate after changing goal counts, map connectivity, blockers or mercy settings together.

CHAPTER 09

The real level-data schema

HexData.cs defines LevelFile version 2 and LevelData. This is the actual first board, wrapped in its file container:

```
{
  "version": 2,
  "levels": [
    {
      "name": "First Bite",
      "shape": "hexagon",
      "hint": "",
      "difficulty": 0.05,
      "breather": false,
      "tut": 1,
      "hex": 40,
      "crates": 0,
      "gems": 0,
      "map": [
        ".ooo.",
        "ooooo",
        "ooooo",
        "ooooo",
        "..o.."
      ],
      "pre": [],
      "locks": [],
      "queue": [
        "00000",
        "00000",
        "11111"
      ],
      "order": [],
      "bands": [
        {
          "From": 0,
          "Colors": 2,
          "MinC": 1,
          "MaxC": 1,
          "MinF": 3,
          "MaxF": 5,
          "Match": 1,
          "Full": 1
        }
      ]
    }
  ]
}
```

map stores rows, with one character per column. A dot is a hole, o an empty cell, L a sugar lock, digits 1..3 crates with that many hits, and g an empty gem cell. Lowercase a..j references pre entries; uppercase A..J adds a gem under that preset tower. pre and queue strings read bottom to top; each digit is a colour index 0..7. locks supplies thresholds in row-reading order.

name and hint are display text. shape is descriptive/generator data, not a Resources path. difficulty and breather describe intended pacing; tut:1 enables the opening guide. hex, crates and gems are goal counts, not automatically inferred from every painted object. order and bands control refill colour choices. Keep goals achievable with the supplied objects.

LevelGrid indexes cells as x*Rows+y internally; the JSON map is row-oriented. Do not transpose it when writing a custom importer. The painter serializes up to ten distinct preset stacks. HexLevels caches resource data; editor Save invalidates that cache. A standalone player must be rebuilt to contain changed JSON.

CHAPTER 10

Reskin recipe: world and interface colours

The actual colour sources and consumers are below. UI changes alone do not recolour the 3D tiles.

Source / symbol	Current value and consumer
Scripts/HexTowerGame.View.cs / TileHex	Eight tile materials: #FF5C86, #3FD39E, #5B7CFF, #FFC93C, #A66BFF, #FF8A4C, #3CC4F0, #9C6A5A.
Editor/HexTowerLevelEditor.cs / Palette	The same eight colours in the board painter; PaletteNames labels their indices.
Scripts/HexTowerGame.UI.cs / Ink	#5B3F7A: goal text, tutorial text, helper captions and goal chips.
Scripts/HexTowerGame.UI.cs / Purple	#7B5CD6: BuildHud level pill and settings button; ShowSettingsPopup accent.
Scripts/HexTowerGame.View.cs / SetupLook	#E6D6FF camera background, #FFF0E2 main light and #C8B6FF fill.
Scripts/HexTowerGame.cs / BoostColor	Per-tool popup/button accents; independent of TileHex.

Worked mint-interface edit

1. Back up the two UI/View files and exit Play mode.
2. In HexTowerGame.UI.cs change only Purple from HzUI.Hex("#7B5CD6") to HzUI.Hex("#398574"). Change Ink to #214B42 if you want compatible dark text.
3. Compile and run both scenes. Inspect the level pill, settings button, settings popup, goal counts and helper captions. The rebuild in OnLayout recreates the HUD, so check a layout refresh too.
4. If changing strawberry tiles, change TileHex[0] from #FF5C86 to your chosen value, and make the same index edit in the editor Palette. Keep indices stable in existing stack strings.

Many popup ribbons, backgrounds, locked captions and effect colours use inline Hex values in UI.cs, View.cs and game.cs. For a full theme, inspect those actual consumers as well. Keep contrast against white labels. Do not globally replace numeric colour indices, save prefixes, shader resource IDs or all purple-looking asset names.

CHAPTER 11

Reskin recipe: visible text and localization

Worked Stage wording edit

1. Open Scripts/HexTowerGame.UI.cs. In BuildHud, find _levelText = HzUI.Label("Lvl", lp.transform, "Level " + Level, ...). Change only the displayed "Level " to "Stage ". Lvl is the hierarchy name and should remain.
2. In LevelBanner change its separate "Level " + Level assignment too. Otherwise the banner and persistent pill disagree.
3. In HexTowerGame.cs update WinFlow's interpolated Level {lv} cleared! and the Next Level button if desired. Preserve {lv}, {_e.Cleared}, {_e.Moves} and newline formatting.
4. Compile and play a fresh level, win and transition. Repeat in portrait, landscape and after OnLayout rebuild.

Text map

BuildHud displays _def.Name as the level subtitle; edit the relevant levels.json name through the visual editor. LevelBanner also uses _def.Name and the breather suffix. UpdateGoalUi refreshes goal counters and the literal done; preserve the min/current/goal interpolation when translating. RefreshBoosters displays BoostName or a locked Level + unlock caption.

BoostName and BoostDesc are in HexTowerGame.cs, alongside Need a..., Get 1 (ad), No thanks, Unlocks at level..., Nothing to undo yet and the win/loss/rescue popup copy. The game's Title override is also there. RefreshTutorial, RefreshGuide, ModeText and ShowSettingsPopup are in UI.cs. SoundLabel and MusicLabel return both ON and OFF states; change both branches, because clicking toggles replaces the current label.

Settings, Restart Level and Close belong to ShowSettingsPopup, not merely the base Kit's generic ShowSettings method. Hierarchy strings such as GoalT, Chip, Lvl and Card are object identifiers rather than displayed sentences. Keep resource IDs like booster_smash, hextower/Levels/levels and save keys unchanged unless intentionally implementing a migration.

No shared localization service automatically translates these hard-coded strings. For multiple languages, introduce your own lookup and route every creation and refresh path through it, including helper arrays and result interpolations. Use fonts containing your target glyphs and inspect longer translations on narrow screens.

CHAPTER 12

Artwork, meshes, fonts and sound

The 3D board and tiles are built by HexMeshes.cs and the View partial. TileR, TileH, TileStep and S control geometry/spacing. Altering these affects camera framing, hit targets, stacking and animation offsets together; do not rescale one constant and assume the rest follow. Keep the supplied materials and their shader GUIDs when changing the colour-only theme.

Resources/hextower/Art contains booster_smash, booster_reroll, booster_lift, booster_undo, goal_hex, goal_crate, goal_gem, lock and result/unlock artwork. Replace an image in place while preserving its .meta, dimensions/aspect purpose and transparent padding. Do not rename resource keys unless updating every loader consumer. Marketing illustrations are separate from these runtime assets.

The private Kit fonts include Lilita One, Nunito and Material Icons. Retain the included licence notices. UI wording, button sizes and font selection are separate changes; icon-font characters are not ordinary translatable words. Use an appropriate licensed fallback when adding scripts not covered by the supplied fonts.

HexSfx.cs holds resource names; Resources/hextower/Sfx contains the actual effects. Build plays the candy_loop music resource. Replace audio with suitable loop/one-shot settings and preserve keys. Test sound and music toggles independently, first browser interaction, pause/resume and a scene restart. Ensure any replacement media is licensed for your redistribution and release.

CHAPTER 13

Local saves and update safety

HzSave in Kit/Runtime/HzCore.cs wraps PlayerPrefs and coalesces saves; browser builds persist through browser storage. The Level property in Kit/Runtime/HzGame.cs uses GameId + "_level", so this game's progress key is hextower_level. Charge keys are hextower_b0..3; unlock-grant flags are hextower_seenb0..3; guide completion flags are hextower_guided0..3.

Mechanic-introduction flags use hextower_seen_lock, hextower_seen_crate and hextower_seen_gem. Per-level authored hint flags use hextower_hint plus the index. Sound/music preferences belong to the Kit audio layer. Test save behavior after a completed move sequence, a result, restart and browser reload. Storage can be cleared by the player or restricted by the browser.

A level number identifies a bank position. Appending is safer for existing players than reordering or replacing earlier levels. A title/display-text reskin must not rename hextower prefixes by a global replace. A deliberate app identity/save schema change requires a migration plan.

The active board/undo state is runtime memory; local progression persistence does not imply cloud synchronization or cross-device restoration. Do not advertise secure server currency or accounts. Clear only controlled test data during QA; never expose broad PlayerPrefs.DeleteAll as normal customer flow.

CHAPTER 14

Advertising: included integration and external provider

The buyer source contains Szekipapa77 Ads integration, settings and an optional adapter. It does not contain Google's Mobile Ads SDK or External Dependency Manager. This distinct source edition does not include HyperCasual LiveOpsKit, purchases, LiveOps stores or a remote backend. Do not follow older templates' HLK configuration instructions.

The call path is game → HzPlatform → game Kit HzAds → Szekipapa77.Ads.AdsService. The optional adapter detects a separately installed Google plugin. Real rewards depend on an earned callback; an installed provider failing to show an ad does not silently become a successful simulated reward.

AdsSettings.asset field	Supplied meaning
backend: 2	Requests AdMob. This enum is None=0, Dummy=1, AdMob=2; it is not the HLK enum.
missingSdkFallback: 1	Dummy simulation when the provider is absent.
useGoogleTestIds: 1	Test application and ad unit IDs if an optional SDK is installed.
dummyDelaySeconds: 0.4	Development simulation delay.
pauseGameDuringFullscreenAds: 1	Pauses active game flow during full-screen ads.
loadTimeoutSeconds: 25	Ad load waiting budget.

Development

Open `Tools > Szekipapa77 Ads > Setup & Diagnostics`. With no Google plugin, fallback is simulated. WebGL always simulates because this AdMob adapter does not support browser ads. Simulation tests UI/callback flow; it is not revenue. A separate test-ad Android demo can be compiled with an externally imported SDK without redistributing that SDK in the buyer source.

Configure an optional Google provider

1. Back up the project. Obtain the official Google Mobile Ads Unity plugin and its dependency manager from Google; import them separately and follow their current platform instructions.
2. Check detection in the publisher diagnostics window. Keep test IDs enabled while testing. Exercise success, cancellation/failure, earned reward exactly once, banners and interstitial continuation.
3. Enter your own app ID and format-specific unit IDs in `Assets/Szekipapa77Ads/Resources/AdsSettings.asset`. Use your account identifiers for your release.

4. Before a production build, set Missing SDK Fallback to None and disable Use Google Test IDs only after entering valid production IDs. The supplied build tooling refuses a missing-provider release or a release still configured for simulated fallback.
5. Provide your own privacy policy, consent flow, Privacy Options and required disclosures. Check them on physical devices in the relevant regions. Template configuration alone does not establish compliance or revenue.

Interstitials are throttled to at most one per 60 seconds. Failure/skipping continues game flow once. Phone banners reserve bottom space; inspect camera/layout clearance and device safe areas after reskinning. The game has no supplied production IAP configuration.

CHAPTER 15

Android and browser releases

Android

1. Install the Unity-matched Android toolchain. Run Setup and inspect the Mobile scene, Graphics/Quality pipelines and input handling.
2. Replace com.szekipapa77.hextower with your own unique application identifier. Set product/company labels, icon, version and version code.
3. Configure your optional provider separately as described above. Keep test IDs during testing; use your valid production IDs and missing-SDK fallback None for release.
4. Run Szekipapa77 > Hex Tower Sort > Build Android. The menu outputs Builds/Android/hextower.apk. Review the actual build profile, architecture and scripting backend for your destination.
5. Use your own production keystore. Keep private keys/passwords outside distributable source. Create an App Bundle if your marketplace needs one.
6. Install on physical devices and test drag/release edges, invalid drops, chain pours, helpers, ads, pause/resume and progress saving.

WebGL

1. Install WebGL support and run Setup. The standard menu selects the portrait Mobile scene, even though a landscape WebGL-named scene is included.
2. Run Szekipapa77 > Hex Tower Sort > Build WebGL. Output is Builds/WebGL/hextower.
3. Deploy index.html together with Build and other generated content. Serve using HTTP(S). A file:// launch cannot load the full player correctly.
4. Keep the referenced loader/data/framework/wasm names and compression/decompression-fallback arrangement. Check server content and compression headers if loading stalls.
5. Build HexTower_WebGL.unity explicitly for a landscape edition, then test mouse and touch, browser storage, audio unlock and the hosted player.

WebGL ad callbacks are simulated, not native AdMob revenue. Hash the final APK and browser archive, and verify anonymous complete downloads after hosting. A copied demo is a useful example; it does not validate your new source edits, signing or production provider configuration.

CHAPTER 16

Troubleshooting by symptom

Symptom	Concrete check
A tower does not drop	Start on a non-empty tray slot, release on an empty unblocked cell and check classic Input handling. UI elements intentionally intercept pointer-down.
The board looks unchanged after editing	Stop Play, Save the editor, restart and rebuild standalone players. HexLevels caches loaded data.
Goal never completes	All positive tile/crate/gem targets must be achievable; zero disables a target. Inspect painted crates/gems and refill bands.
No matching pour	Only adjacent matching top colours connect. Covered colours are not exposed until the upper run moves/pops.
The tutorial is absent	Check tut:1 and the current board index; guide/introduction flags can already be saved.
Undo says nothing to undo	Make a placement first. Other helpers and rescue can invalidate the snapshot.
Palette differs in editor	Change runtime TileHex and editor Palette at the same colour index.
HUD text reverts	Update both creation and refresh assignments, both Sound/Music branches and LevelBanner separately.
Pink or missing meshes	Check official URP version, the selected Graphics/Quality pipeline and material/shader GUIDs; run Setup.
Duplicate classes after import	Use a clean one-game project; private game Kits intentionally share types.
Native ads simulate	No provider SDK is bundled. Inspect diagnostics, backend and missing-SDK fallback.
WebGL has no real AdMob	Expected for this adapter. A browser provider requires a separate implementation.
Production build refuses to proceed	Configure the actual provider and own IDs, disable test IDs when ready, and use fallback None; do not remove the release guard.
Player stalls on hosting	Serve every build file with correct compression headers, keep filenames and inspect the first browser Console error.
Exception after changing scene	Capture the first stack and exact sequence; delayed work must not access destroyed UI/world owners.

Report Unity version, platform, scene, board number, changes made and the first relevant error stack. Never include passwords, production signing keys or private advertising account secrets in a support message.

CHAPTER 17

Before shipping and getting help

1. Import the final native package into a clean project with only the documented official dependencies. Play both included scenes and reload/switch scenes.
2. Test real drag/release, a chain pop, each goal, win/loss/retry, rescue success/failure, helper preconditions and actual effects.
3. Exercise fresh and existing saves, first-play guidance, every unlock introduction and a layout rebuild.
4. Validate edited/generated boards across multiple seeds, then play them yourself. Bot results do not replace player testing.
5. Inspect contrast, long translated text, font licences, new media rights, privacy/consent setup and provider release settings.
6. Replace application identity and signing, rebuild after source changes, hash final outputs and verify public download/demo/manual/video links.
7. Keep the complete offline handbook accessible before importing and inside the project. Remove preparation tools and internal instructions from buyer deliveries.

Publisher: Szekipapa77. Support: szekipapa77@gmail.com. Your release accounts, provider SDK installation, consent configuration, new content and store submission remain separate from this editable game template.