UNITY 6000.0.62F1 · COMPLETE OFFLINE DOCUMENTATION

Neon Waves Developer Guide

Install. Edit. Reskin. Build your release.

szekipapa77@gmail.com



CHAPTER 01

Start here: Neon Waves

Neon Waves is a standalone editable Unity game template. Hold anywhere or press Space to make the neon dart rise, and release to descend through the tunnel. Flat floors and ceilings permit sliding, but walls and oncoming slopes crash the dart. Weave around spikes, saws and moving blocks; speed gates alter the rhythm. Curated levels and local endless laps use the same track generation system.

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 > Neon Waves > 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/NeonWaves/Scenes/NeonWaves_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/NeonWaves`. 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/neonwaves contains level JSON, art, music and sound; Settings owns URP assets; Scenes contains both entry scenes.

- `Scripts/NeonWavesGame.cs`
- `Scripts/NeonWavesGame.UI.cs`
- `Scripts/NwArt.cs`
- `Scripts/NwBot.cs`
- `Scripts/NwData.cs`
- `Scripts/NwGen.cs`
- `Scripts/NwLogic.cs`
- `Scripts/NwWorld.cs`

Main flow: NeonWavesGame.cs. UI and popup consumers: NeonWavesGame.UI.cs. Model/generator/loading: NwData.cs, NwGen.cs, NwLogic.cs. Authoring: Editor/NeonWavesLevelEditor.cs. Shared ad settings live outside the game at Assets/Szekipapa77Ads/Resources/AdsSettings.asset.

CHAPTER 04

Gameplay, controls and progression

Hold anywhere or press Space to make the neon dart rise, and release to descend through the tunnel. Flat floors and ceilings permit sliding, but walls and oncoming slopes crash the dart. Weave around spikes, saws and moving blocks; speed gates alter the rhythm. Curated levels and local endless laps use the same track generation system.

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/NeonWavesGame.cs; effect/help strings: Scripts/NeonWavesGame.cs.

Helper	Unlock	Free charges	Action and effect
Shield	4	3	Absorbs one crash. You ghost through for a moment, then fly on.
Slow-mo	7	2	Slows time to 60% for 10 seconds.
Flag	10	3	Plants a checkpoint. Crashes respawn you at the flag until you leave the level.
Magnet	14	2	Pulls nearby orbs to you for 15 seconds.

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 > Neon Waves > Level Editor**. Exact implementation: `Editor/NeonWavesLevelEditor.cs`.

Open the Neon Waves 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/neonwaves/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
`NeonWavesLevelEditor.cs:52`	`[MenuItem("Szekipapa77/Neon Waves/Level Editor")]`
`NeonWavesLevelEditor.cs:55`	`[MenuItem("Szekipapa77/Neon Waves/Validate All Levels")]`
`NeonWavesLevelEditor.cs:136`	`if (GUILayout.Button(_dirty ? "Save *" : "Save", EditorStyles.toolbarButton, GUILayout.Width(70))) Save();`
`NeonWavesLevelEditor.cs:137`	`if (GUILayout.Button("Reload", EditorStyles.toolbarButton, GUILayout.Width(60)) && (!_dirty || EditorUtility.DisplayDialog("Reload", "Discard unsaved changes?", "Discard", "Cancel"))) Load();`
`NeonWavesLevelEditor.cs:140`	`if (GUILayout.Button("Validate", EditorStyles.toolbarButton, GUILayout.Width(70)) && Cur != null) ValidateOne(_sel);`
`NeonWavesLevelEditor.cs:141`	`if (GUILayout.Button("Validate All", EditorStyles.toolbarButton, GUILayout.Width(85))) ValidateAll();`

NeonWavesLevelEditor.cs:144	`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);`
NeonWavesLevelEditor.cs:170	`if (GUILayout.Button("Add")) { var l = NwGen.Generate(new NwGenParams { Level = _file.levels.Length + 1, Seed = Environment.TickCount & 0xffff, Difficulty = 0.4f }); Insert(_file.levels.Length, l); }`
NeonWavesLevelEditor.cs:171	`if (GUILayout.Button("Dup") && Cur != null) Insert(_sel + 1, Cur.Clone());`
NeonWavesLevelEditor.cs:172	`if (GUILayout.Button("Del") && Cur != null && EditorUtility.DisplayDialog("Delete level", $"Delete level {_sel + 1} ({Cur.name})?", "Delete", "Cancel"))`
NeonWavesLevelEditor.cs:176	`if (GUILayout.Button("Up") && _sel > 0) Swap(_sel, _sel - 1);`
NeonWavesLevelEditor.cs:177	`if (GUILayout.Button("Down") && _sel < _file.levels.Length - 1) Swap(_sel, _sel + 1);`
NeonWavesLevelEditor.cs:191	`l.name = EditorGUILayout.TextField("Name", l.name, GUILayout.Width(380));`
NeonWavesLevelEditor.cs:192	`l.theme = EditorGUILayout.Popup("Sector look", l.theme, NwArt.SectorNames, GUILayout.Width(300));`
NeonWavesLevelEditor.cs:195	`l.speed = EditorGUILayout.Slider("Speed", l.speed, 0.7f, 1.4f, GUILayout.Width(380));`
NeonWavesLevelEditor.cs:196	`l.startGap = EditorGUILayout.IntSlider("Start gap", l.startGap, 4, 12, GUILayout.Width(300));`
NeonWavesLevelEditor.cs:197	`l.breather = EditorGUILayout.ToggleLeft("Breather", l.breather, GUILayout.Width(80));`
NeonWavesLevelEditor.cs:199	`ii = EditorGUILayout.Popup("Intro hint", ii, Intros.Select(s => s == "" ? "(none)" : s).ToArray(), GUILayout.Width(260));`

NeonWavesLevelEditor.cs:608	`EditorGUILayout.BeginVertical("box", GUILayout.Width(430));`
NeonWavesLevelEditor.cs:616	`k = EditorGUILayout.Popup("Shape", k, Kinds); s.kind = Kinds[k];`
NeonWavesLevelEditor.cs:617	`s.len = EditorGUILayout.IntSlider("Length", s.len, 2, 120);`
NeonWavesLevelEditor.cs:618	`s.gap = EditorGUILayout.IntSlider("Gap", s.gap, 3, 14);`
NeonWavesLevelEditor.cs:619	`s.amp = EditorGUILayout.IntSlider(s.kind == "teeth" ? "Tooth height (+floor/-ceil)" : s.kind == "pinch" ? "Pinch depth" : "Zig height (+up/-down)", s.amp, -8, 8);`
NeonWavesLevelEditor.cs:620	`s.rail = EditorGUILayout.Popup("Plasma rails", s.rail, new[] { "none", "floor", "ceiling", "both" });`
NeonWavesLevelEditor.cs:624	`if (GUILayout.Button("Insert")) { InsertSeg(_selSeg >= 0 ? _selSeg + 1 : segs.Length, new NwSeg { kind = "flat", len = 12, gap = 7 }); }`
NeonWavesLevelEditor.cs:626	`if (GUILayout.Button("Duplicate")) InsertSeg(_selSeg + 1, segs[_selSeg].Clone());`
NeonWavesLevelEditor.cs:627	`if (GUILayout.Button("Delete")) { var l = segs.ToList(); l.RemoveAt(_selSeg); Cur.segs = l.ToArray(); _selSeg = Mathf.Min(_selSeg, Cur.segs.Length - 1); _selObj = -1; Changed(); }`
NeonWavesLevelEditor.cs:628	`if (GUILayout.Button("<") && _selSeg > 0) { (segs[_selSeg], segs[_selSeg - 1]) = (segs[_selSeg - 1], segs[_selSeg]); _selSeg--; Changed(); }`
NeonWavesLevelEditor.cs:629	`if (GUILayout.Button(">") && _selSeg < segs.Length - 1) { (segs[_selSeg], segs[_selSeg + 1]) = (segs[_selSeg + 1], segs[_selSeg]); _selSeg++; Changed(); }`
NeonWavesLevelEditor.cs:635	`EditorGUILayout.BeginVertical("box", GUILayout.Width(430));`
NeonWavesLevelEditor.cs:641	`o.x = EditorGUILayout.FloatField("x (in chunk)", o.x);`

NeonWavesLevelEditor.cs:644	`case "spike": o.side = EditorGUILayout.Popup("Side", o.side, new[] { "floor", "ceiling" }); o.w = EditorGUILayout.IntSlider("Count", Mathf.RoundToInt(o.w), 1, 6); break;`
NeonWavesLevelEditor.cs:645	`case "saw": o.y = EditorGUILayout.FloatField("y (from centre)", o.y); o.w = EditorGUILayout.Slider("Radius", o.w, 0.3f, 2f); break;`
NeonWavesLevelEditor.cs:646	`case "block": o.y = EditorGUILayout.FloatField("y (from centre)", o.y); o.w = EditorGUILayout.Slider("Width", o.w, 0.5f, 6f); o.h = EditorGUILayout.Slider("Height", o.h, 0.5f, 6f); break;`
NeonWavesLevelEditor.cs:648	`o.y = EditorGUILayout.FloatField("y (from centre)", o.y); o.w = EditorGUILayout.Slider("Width", o.w, 0.5f, 4f); o.h = EditorGUILayout.Slider("Height", o.h, 0.5f, 4f);`
NeonWavesLevelEditor.cs:649	`o.a = EditorGUILayout.Slider("Amplitude", o.a, 0f, 6f); o.p = EditorGUILayout.Slider("Period (s)", o.p, 0.4f, 5f); o.ph = EditorGUILayout.Slider("Phase", o.ph, 0f, 1f); break;`
NeonWavesLevelEditor.cs:651	`o.side = EditorGUILayout.Popup("Side", o.side, new[] { "floor", "ceiling" }); o.h = EditorGUILayout.Slider("Bar length", o.h, 0.5f, 10f);`
NeonWavesLevelEditor.cs:652	`o.p = EditorGUILayout.Slider("Period (s)", o.p, 0.4f, 4f); o.a = EditorGUILayout.Slider("Closed fraction", o.a, 0.1f, 0.9f); o.ph = EditorGUILayout.Slider("Phase", o.ph, 0f, 1f); break;`
NeonWavesLevelEditor.cs:653	`case "orb": o.y = EditorGUILayout.FloatField("y (from centre)", o.y); break;`
NeonWavesLevelEditor.cs:654	`case "speed": o.v = EditorGUILayout.Slider("Speed x", o.v, 0.5f, 2f); break;`
NeonWavesLevelEditor.cs:655	`case "zoom": o.v = EditorGUILayout.Slider("View scale", o.v, 0.6f, 1.8f); break;`

NeonWavesLevelEditor.cs:656	default: o.v = EditorGUILayout.Toggle("On", o.v > 0.5f) ? 1 : 0; break;
NeonWavesLevelEditor.cs:659	if (GUILayout.Button("Delete object")) { RemoveObj(_selSeg, _selObj); Changed(); }
NeonWavesLevelEditor.cs:662	if (GUILayout.Button("Re-place orbs on the bot path")) { RePlaceOrbs(); }
NeonWavesLevelEditor.cs:682	if (!_results.TryGetValue(_sel, out var v)) { EditorGUILayout.HelpBox("Not validated yet: press Validate.", MessageType.None); return; }
NeonWavesLevelEditor.cs:683	if (v.Passable) EditorGUILayout.HelpBox($"Bot: PASSABLE in {v.Seconds:0.0}s · difficulty {v.Score} · narrowest safe band {v.MinSafe:0.00} cells · mean tightness {v.MeanTight:0.00} · {v.SwitchesPerSec:0.0} taps/s · {v.TightX.Count} tight windows", MessageType.Info);
NeonWavesLevelEditor.cs:684	else EditorGUILayout.HelpBox($"Bot: IMPASSABLE. No hold/release timing gets past x = {v.FailX:0.0} (red line).", MessageType.Error);
NeonWavesLevelEditor.cs:735	_genCount = EditorGUILayout.IntField("Count", _genCount, GUILayout.Width(220));
NeonWavesLevelEditor.cs:736	_genStart = EditorGUILayout.IntField("Start level #", _genStart, GUILayout.Width(220));
NeonWavesLevelEditor.cs:737	_genSeed = EditorGUILayout.IntField("Seed", _genSeed, GUILayout.Width(220));
NeonWavesLevelEditor.cs:740	_genFrom = EditorGUILayout.Slider("Start difficulty", _genFrom, 0.05f, 1f, GUILayout.Width(330));
NeonWavesLevelEditor.cs:741	_genTo = EditorGUILayout.Slider("End difficulty", _genTo, 0.05f, 1f, GUILayout.Width(330));
NeonWavesLevelEditor.cs:742	_genBreather = EditorGUILayout.IntSlider("Breather every", _genBreather, 2, 8, GUILayout.Width(330));
NeonWavesLevelEditor.cs:745	_genReplace = EditorGUILayout.ToggleLeft("Replace the range (otherwise insert)", _genReplace, GUILayout.Width(260));

`NeonWavesLevelEditor.cs:758`	`if (GUILayout.Button("Generate", GUILayout.Width(120))) 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
`NeonWavesLevelEditor.cs:49`	`bool _genReplace;`
`NeonWavesLevelEditor.cs:745`	`_genReplace = EditorGUILayout.ToggleLeft("Replace the range (otherwise insert)", _genReplace, GUILayout.Width(260));`
`NeonWavesLevelEditor.cs:777`	`if (_genReplace) { for (int i = 0; i < made.Count; i++) { if (at + i < list.Count) list[at + i] = made[i]; else list.Add(made[i]); } }`
`NeonWavesLevelEditor.cs:778`	`else 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/NeonWaves/Resources/neonwaves/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 Light",
 "theme": 0,
 "speed": 0.9,
 "startGap": 8,
 "breather": false,
 "intro": "basics",
 "difficulty": 4,
 "segs": [
  {
   "kind": "flat",
   "len": 14,
   "gap": 8,
   "amp": 3,
   "rail": 0,
   "objs": []
  },
  {
   "kind": "up",
   "len": 6,
   "gap": 8,
   "amp": 3,
   "rail": 0,
   "objs": [
    {
     "type": "orb",
     "x": 3.69,
     "y": 3.67,
     "w": 1,
     "h": 1,
     "a": 0,
     "p": 2,
     "ph": 0,
     "side": 0,
     "v": 1
    }
   ]
  },
  {
   "kind": "flat",
   "len": 8,
   "gap": 7,
   "amp": 3,
   "rail": 0,
   "objs": []
  },
  {
   "kind": "down",
   "len": 6,
   "gap": 7,
   "amp": 3,
   "rail": 0,
   "objs": []
  },
  {
   "kind": "flat",
   "len": 8,
   "gap": 7,
   "amp": 3,
   "rail": 0,
   "objs": []
  },
  {
   "kind": "up",
   "len": 4,
   "gap": 7,
   "amp": 3,
   "rail": 0,
   "objs": [
    {
     "type": "orb",
```

```
        "x": 1.18,
        "y": -3.2,
        "w": 1,
        "h": 1,
        "a": 0,
        "p": 2,
        "ph": 0,
        "side": 0,
        "v": 1
      }
    ]
  },
  {
    "kind": "flat",
    "len": 6,
    "gap": 6,
    "amp": 3,
    "rail": 0,
    "objs": []
  },
  {
    "kind": "down",
    "len": 8,
    "gap": 6,
    "amp": 3,
    "rail": 0,
    "objs": []
  },
  {
    "kind": "flat",
    "len": 8,
    "gap": 6,
    "amp": 3,
    "rail": 0,
    "objs": [
      {
        "type": "orb",
        "x": 5.95,
        "y": -2.75,
        "w": 1,
        "h": 1,
        "a": 0,
        "p": 2,
        "ph": 0,
        "side": 0,
        "v": 1
      }
    ]
  },
  {
    "kind": "zig",
    "len": 16,
    "gap": 7,
    "amp": 3,
    "rail": 0,
    "objs": []
  },
  {
    "kind": "flat",
    "len": 8,
    "gap": 6,
    "amp": 3,
    "rail": 0,
    "objs": [
      {
        "type": "orb",
        "x": 5.58,
        "y": 2.75,
        "w": 1,
        "h": 1,
        "a": 0,
        "p": 2,
        "ph": 0,
        "side": 0,
        "v": 1
      }
    ]
  },
  {
    "kind": "up",
    "len": 5,
    "gap": 6,
    "amp": 3,
    "rail": 0,
    "objs": []
  },
  {
    "kind": "flat",
```

```
      "len": 4,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": []
    },
    {
      "kind": "down",
      "len": 5,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": []
    },
    {
      "kind": "flat",
      "len": 6,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": []
    },
    {
      "kind": "zig",
      "len": 20,
      "gap": 6,
      "amp": 4,
      "rail": 0,
      "objs": [
        {
          "type": "orb",
          "x": 2.99,
          "y": -2.23,
          "w": 1,
          "h": 1,
          "a": 0,
          "p": 2,
          "ph": 0,
          "side": 0,
          "v": 1
        }
      ]
    },
    {
      "kind": "flat",
      "len": 8,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": [
        {
          "type": "orb",
          "x": 7.9,
          "y": 1.32,
          "w": 1,
          "h": 1,
          "a": 0,
          "p": 2,
          "ph": 0,
          "side": 0,
          "v": 1
        }
      ]
    },
    {
      "kind": "pinch",
      "len": 12,
      "gap": 7,
      "amp": 2,
      "rail": 0,
      "objs": []
    },
    {
      "kind": "flat",
      "len": 6,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": []
    },
    {
      "kind": "up",
      "len": 6,
      "gap": 6,
      "amp": 3,
      "rail": 0,
      "objs": []
    },
```

```
      {
        "kind": "down",
        "len": 6,
        "gap": 6,
        "amp": 3,
        "rail": 0,
        "objs": [
          {
            "type": "orb",
            "x": 2.02,
            "y": -2.4,
            "w": 1,
            "h": 1,
            "a": 0,
            "p": 2,
            "ph": 0,
            "side": 0,
            "v": 1
          }
        ]
      },
      {
        "kind": "flat",
        "len": 10,
        "gap": 7,
        "amp": 3,
        "rail": 0,
        "objs": []
      }
    ]
  }
```


Root keys: levels. Inspect NwData.cs, NwGen.cs, NwLogic.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/NwArt.cs`. Find `Themes`, currently `#2EF2FF`. 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
`NwArt.cs:18 / class/static declarations`	`public static readonly NwTheme[] Themes =`
`NwArt.cs:32 / class/static declarations`	`public static NwTheme Theme(int i) => Themes[((i % Themes.Length) + Themes.Length) % Themes.Length];`

Source	Exact current code
`NeonWavesGame.cs:23 / class/static declarations`	`new HelperDef { Id = "shield", Name = "Shield", Sprite = "helper_shield", Col = "#2EF2FF", Unlock = 4, Free = 3, Cost = 30, Key = KeyCode.Alpha1,`
`NeonWavesGame.cs:24 / class/static declarations`	`Desc = "Absorbs one crash. You ghost through for a moment, then fly on.", Guide = "Tap <color=#2EF2FF>SHIELD</color> to arm it. It will absorb one crash!" },`
`NeonWavesGame.cs:25 / class/static declarations`	`new HelperDef { Id = "slowmo", Name = "Slow-mo", Sprite = "helper_slowmo", Col = "#B07CFF", Unlock = 7, Free = 2, Cost = 40, Key = KeyCode.Alpha2,`
`NeonWavesGame.cs:26 / class/static declarations`	`Desc = "Slows time to 60% for 10 seconds.", Guide = "Tap <color=#B07CFF>SLOW-MO</color> for 10 seconds of slower time." },`
`NeonWavesGame.cs:27 / class/static declarations`	`new HelperDef { Id = "flag", Name = "Flag", Sprite = "helper_flag", Col = "#A8FF3E", Unlock = 10, Free = 3, Cost = 30, Key = KeyCode.Alpha3,`

NeonWavesGame.cs:28 / class/static declarations	Desc = "Plants a checkpoint. Crashes respawn you at the flag until you leave the level.", Guide = "Tap <color=#A8FF3E>FLAG</color> to plant a checkpoint right here." },
NeonWavesGame.cs:29 / class/static declarations	new HelperDef { Id = "magnet", Name = "Magnet", Sprite = "helper_magnet", Col = "#FFB23E", Unlock = 14, Free = 2, Cost = 30, Key = KeyCode.Alpha4,
NeonWavesGame.cs:30 / class/static declarations	Desc = "Pulls nearby orbs to you for 15 seconds.", Guide = "Tap <color=#FFB23E>MAGNET</color>: nearby orbs fly to you for 15 seconds." },
NeonWavesGame.cs:557 / void OfferRefill(int h)	($"Buy 1 for {d.Cost} orbs", HzUI.Hex("#3DBE6E"), () =>
NeonWavesGame.cs:562 / void OfferRefill(int h)	("Watch ad: +1 free", HzUI.Hex("#7B61FF"), () => HzPlatform.Rewarded(GameId, "helper_" + d.Id, ok => { if (ok) { SetCharges(h, Charges(h) + 1); HzAudio.Play("buy", 0.7f); RefreshHud(true); } })),
NeonWavesGame.cs:563 / void OfferRefill(int h)	("Close", HzUI.Hex("#4C5A7A"), null));
NeonWavesGame.UI.cs:108 / void BuildHelperBar()	ui.badgeBg = HzUI.Panel("Badge", b.transform, HzUI.Hex("#FF3E7A"), 22);
NeonWavesGame.UI.cs:182 / void RefreshHelpers()	ui.badgeBg.color = c > 0 ? HzUI.Hex("#FF3E7A") : HzUI.Hex("#3DBE6E");
NeonWavesGame.UI.cs:287 / void ShowPause()	("Resume", HzUI.Hex("#3DBE6E"), null),
NeonWavesGame.UI.cs:288 / void ShowPause()	("Restart", HzUI.Hex("#F5A623"), () => { _flagSet = false; Retry(); }),
NeonWavesGame.UI.cs:289 / void ShowPause()	("Settings", HzUI.Hex("#4C9EEB"), () => OpenSettings()),
NeonWavesGame.UI.cs:290 / void ShowPause()	("Levels", HzUI.Hex("#7B61FF"), () => ShowMenu()));
NeonWavesGame.UI.cs:303 / void OpenSettings()	Restyle(p, HzUI.Hex("#4C9EEB"));
NeonWavesGame.UI.cs:332 / void BuildDeathCard()	_dcRetry = HzUI.Button("Retry", card.transform, "RETRY", HzUI.Hex("#3DBE6E"), Color.white, () => Retry(), 60);

NeonWavesGame.UI.cs:334 / void BuildDeathCard()	_dcRevive = HzUI.Button("Revive", card.transform, "REVIVE HERE", HzUI.Hex("#7B61FF"), Color.white, () => Revive(), 44);
NeonWavesGame.UI.cs:337 / void BuildDeathCard()	_dcLevels = HzUI.Button("Levels", card.transform, "LEVELS", HzUI.Hex("#4C5A7A"), Color.white, () => ShowMenu(), 40);
NeonWavesGame.UI.cs:388 / void ShowWin(int stars, int total, int gain)	("Next", HzUI.Hex("#3DBE6E"), () => NextLevel()),
NeonWavesGame.UI.cs:389 / void ShowWin(int stars, int total, int gain)	("Replay", HzUI.Hex("#F5A623"), () => PlayLevel(_playLevel)),
NeonWavesGame.UI.cs:390 / void ShowWin(int stars, int total, int gain)	("Levels", HzUI.Hex("#7B61FF"), () => ShowMenu()));
NeonWavesGame.UI.cs:425 / void ShowIntro(string id, Action next)	var p = NeonPopup(title, text, W.Theme.Edge, ("Got it!", HzUI.Hex("#3DBE6E"), null));
NeonWavesGame.UI.cs:434 / void ShowUnlock(int h, Action next)	var p = NeonPopup("New helper: " + d.Name, d.Desc + $"\n<color=#FFD23E>+{d.Free} free charges</color>", HzUI.Hex(d.Col), ("Awesome!", HzUI.Hex("#3DBE6E"), null));
NeonWavesGame.UI.cs:527 / void PopulateMenu()	title.AddOutline(NwArt.A(HzUI.Hex("#FF3EC8"), 0.9f), 5);
NeonWavesGame.UI.cs:528 / void PopulateMenu()	title.AddShadow(NwArt.A(HzUI.Hex("#2EF2FF"), 0.8f), 8);
NeonWavesGame.UI.cs:554 / void PopulateMenu()	var play = HzUI.Button("Play", _menuContent, playLabel, HzUI.Hex("#3DBE6E"), Color.white, () => PlayLevel(next), P ? 58 : 52);
NeonWavesGame.UI.cs:558 / void PopulateMenu()	var end = HzUI.Button("Endless", _menuContent, endlessOpen ? "ENDLESS RUN" : "ENDLESS (LV 6)", endlessOpen ? HzUI.Hex("#7B61FF") : new Color(0.25f, 0.25f, 0.35f), Color.white, () =>
NwArt.cs:20 / class/static declarations	T("Cyan Reach", "#2EF2FF", "#7A8CFF", "#0B1838", "#050A1C", "#FF3E7A", "#FFD23E", "#060818", "#1C3A7A"),
NwArt.cs:21 / class/static declarations	T("Magenta Drift", "#FF3EC8", "#FF8A3E", "#2A0B33", "#12041A", "#FFD23E", "#2EF2FF", "#0E0414", "#5A1A6A"),

`NwArt.cs:22 / class/static declarations`	`T("Verdant Rift", "#A8FF3E", "#2EFFB0", "#0A2622", "#03110E", "#FF5A3E", "#FFD23E", "#030E0C", "#1A5A3A"),`
`NwArt.cs:23 / class/static declarations`	`T("Amber Storm", "#FFB23E", "#FF5A3E", "#2E140B", "#160804", "#3EE8FF", "#FFF07A", "#120604", "#6A3A1A"),`
`NwArt.cs:24 / class/static declarations`	`T("Violet Core", "#B07CFF", "#FF3EC8", "#170B33", "#09041A", "#FF3E5A", "#2EF2FF", "#07040F", "#3E2A7A"),`
`NwWorld.cs:337 / void BuildDart(bool mini)`	`Color32 w = Color.white, e = Theme != null ? Theme.Edge : HzUI.Hex("#2EF2FF");`
`NwWorld.cs:578 / public static Color PortalColor(NwZoneKind k, float v)`	`case NwZoneKind.Speed: return v > 1.01f ? HzUI.Hex("#FFE23E") : v < 0.99f ? HzUI.Hex("#3EFF8A") : HzUI.Hex("#C8C8FF");`
`NwWorld.cs:579 / public static Color PortalColor(NwZoneKind k, float v)`	`case NwZoneKind.Mini: return HzUI.Hex("#FF3EC8");`
`NwWorld.cs:580 / public static Color PortalColor(NwZoneKind k, float v)`	`case NwZoneKind.Gravity: return HzUI.Hex("#3E9BFF");`
`NwWorld.cs:581 / public static Color PortalColor(NwZoneKind k, float v)`	`case NwZoneKind.Mirror: return HzUI.Hex("#FF8A3E");`
`NwWorld.cs:582 / public static Color PortalColor(NwZoneKind k, float v)`	`default: return HzUI.Hex("#E8FFFF");`

CHAPTER 10

Reskin recipe: visible menu and HUD text

Open `Scripts/NeonWavesGame.cs`, find the displayed literal `"Neon Waves"`, and replace its displayed value with `"Neon Waves 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/NeonWavesGame.cs; effect/help strings: Scripts/NeonWavesGame.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
`NeonWavesGame.cs:23 / class/static declarations`	`new HelperDef { Id = "shield", Name = "Shield", Sprite = "helper_shield", Col = "#2EF2FF", Unlock = 4, Free = 3, Cost = 30, Key = KeyCode.Alpha1,`
`NeonWavesGame.cs:24 / class/static declarations`	`Desc = "Absorbs one crash. You ghost through for a moment, then fly on.", Guide = "Tap <color=#2EF2FF>SHIELD</color> to arm it. It will absorb one crash!" },`
`NeonWavesGame.cs:25 / class/static declarations`	`new HelperDef { Id = "slowmo", Name = "Slow-mo", Sprite = "helper_slowmo", Col = "#B07CFF", Unlock = 7, Free = 2, Cost = 40, Key = KeyCode.Alpha2,`
`NeonWavesGame.cs:26 / class/static declarations`	`Desc = "Slows time to 60% for 10 seconds.", Guide = "Tap <color=#B07CFF>SLOW-MO</color> for 10 seconds of slower time." },`
`NeonWavesGame.cs:27 / class/static declarations`	`new HelperDef { Id = "flag", Name = "Flag", Sprite = "helper_flag", Col = "#A8FF3E", Unlock = 10, Free = 3, Cost = 30, Key = KeyCode.Alpha3,`
`NeonWavesGame.cs:28 / class/static declarations`	`Desc = "Plants a checkpoint. Crashes respawn you at the flag until you leave the level.", Guide = "Tap <color=#A8FF3E>FLAG</color> to plant a checkpoint right here." },`

NeonWavesGame.cs:29 / class/static declarations	new HelperDef { Id = "magnet", Name = "Magnet", Sprite = "helper_magnet", Col = "#FFB23E", Unlock = 14, Free = 2, Cost = 30, Key = KeyCode.Alpha4,
NeonWavesGame.cs:30 / class/static declarations	Desc = "Pulls nearby orbs to you for 15 seconds.", Guide = "Tap <color=#FFB23E>MAGNET</color>: nearby orbs fly to you for 15 seconds." },
NeonWavesGame.cs:299 / protected override void Update()	if (_slowLeft > 0) { _slowLeft -= dt; if (_slowLeft <= 0) { HzAudio.Play("slowmo_end", 0.5f); W.SetFxSpeed(1f); RefreshHelpers(); } }
NeonWavesGame.cs:374 / void StepOnce()	Toast("Shield saved you!", HzUI.Hex(Defs[HShield].Col), 1.1f, 0.72f);
NeonWavesGame.cs:517 / void OnHelper(int h)	if (_endless && h == HFlag) { Toast("No flags in the endless run!", Color.white, 1.2f, 0.7f); HzAudio.Play("locked", 0.5f); return; }
NeonWavesGame.cs:518 / void OnHelper(int h)	if (!Unlocked(h)) { HzAudio.Play("locked", 0.5f); Toast($"{Defs[h].Name} unlocks at level {Defs[h].Unlock}", Color.white, 1.2f, 0.7f); return; }
NeonWavesGame.cs:542 / void Activate(int h)	Toast("Checkpoint planted!", col, 1.1f, 0.72f);
NeonWavesGame.cs:556 / void OfferRefill(int h)	var p = NeonPopup($"Out of {d.Name}", $"You have {Wallet} orbs.", HzUI.Hex(d.Col),
NeonWavesGame.cs:560 / void OfferRefill(int h)	else { HzAudio.Play("locked", 0.5f); Toast("Not enough orbs: collect more in levels!", Color.white, 1.4f, 0.6f); }
NeonWavesGame.cs:678 / public override string DebugCommand(string cmd)	case "helper": if (parts.Length > 1 && int.TryParse(parts[1], out var h) && h >= 0 && h < 4) { OnHelper(h); return "ok"; } return "usage";
NeonWavesGame.UI.cs:39 / void BuildUi()	_title = HzUI.Label("Title", _hud, "", P ? 40 : 36, Ink, P ? TextAnchor.MiddleCenter : TextAnchor.MiddleLeft);
NeonWavesGame.UI.cs:41 / void BuildUi()	_attemptTxt = HzUI.Label("Attempt", _hud, "", P ? 30 : 28, Dim, P ? TextAnchor.MiddleCenter : TextAnchor.MiddleLeft, false);
NeonWavesGame.UI.cs:52 / void BuildUi()	_pctTxt = HzUI.Label("Pct", _hud, "0%", P ? 46 : 42, Color.white);

NeonWavesGame.UI.cs:57 / void BuildUi()	_orbTxt = HzUI.Label("Orbs", _hud, "0/0", P ? 42 : 40, Color.white, TextAnchor.MiddleLeft);
NeonWavesGame.UI.cs:60 / void BuildUi()	_prompt = HzUI.Label("Prompt", Root, "", P ? 62 : 56, Color.white);
NeonWavesGame.UI.cs:70 / void BuildUi()	_loadingTxt = HzUI.Label("T", _loading, "", 52, Ink); _loadingTxt.rectTransform.Stretch();
NeonWavesGame.UI.cs:78 / void BuildUi()	_guideTxt = HzUI.Label("T", gcard.transform, "", 44, Color.white, TextAnchor.MiddleCenter, false);
NeonWavesGame.UI.cs:91 / void BuildHelperBar()	_helperBar = HzUI.Rect("Helpers", Root);
NeonWavesGame.UI.cs:98 / void BuildHelperBar()	var b = HzUI.Button("Helper" + i, _helperBar, null, new Color(0.08f, 0.07f, 0.2f, 0.85f), Color.white, () => OnHelper(h), 40, (int)(size * 0.28f));
NeonWavesGame.UI.cs:111 / void BuildHelperBar()	ui.badge = HzUI.Label("N", ui.badgeBg.transform, "0", (int)(size * 0.24f), Color.white); ui.badge.rectTransform.Stretch();
NeonWavesGame.UI.cs:116 / void BuildHelperBar()	ui.lockTxt = HzUI.Label("Lv", ui.lockImg.transform, "LV " + Defs[i].Unlock, (int)(size * 0.2f), Dim);
NeonWavesGame.UI.cs:118 / void BuildHelperBar()	var key = HzUI.Label("Key", b.transform, P ? "" : (i + 1).ToString(), 24, Dim);
NeonWavesGame.UI.cs:145 / void RefreshHud(bool force)	_pctTxt.text = m + " m";
NeonWavesGame.UI.cs:150 / void RefreshHud(bool force)	if (force) _title.text = $"ENDLESS RUN · best {EndlessBest} m";
NeonWavesGame.UI.cs:158 / void RefreshHud(bool force)	_pctTxt.text = pct + "%";
NeonWavesGame.UI.cs:163 / void RefreshHud(bool force)	if (force) _title.text = _playLevel <= NwData.Curated ? $"{(_playLevel - 1) / 5 + 1}-{(_playLevel - 1) % 5 + 1} {_level.name.ToUpperInvariant()}" : $"DEEP SPACE {_playLevel} {_level.name.ToUpperInvariant()}";
NeonWavesGame.UI.cs:165 / void RefreshHud(bool force)	if (force || _attempt != _shownAttempt) { _shownAttempt = _attempt; _attemptTxt.text = "Attempt " + _attempt; }

NeonWavesGame.UI.cs:166 / void RefreshHud(bool force)	if (force || _collected != _shownOrbs) { _shownOrbs = _collected; _orbTxt.text = _endless ? _collected.ToString() : _collected + "/" + _track.Orbs.Count; }
NeonWavesGame.UI.cs:177 / void RefreshHelpers()	if (!usable) ui.lockTxt.text = "OFF";
NeonWavesGame.UI.cs:178 / void RefreshHelpers()	else ui.lockTxt.text = "LV " + Defs[i].Unlock;
NeonWavesGame.UI.cs:181 / void RefreshHelpers()	ui.badge.text = c > 0 ? c.ToString() : "+";
NeonWavesGame.UI.cs:286 / void ShowPause()	var p = NeonPopup("Paused", _endless ? $"{Mathf.FloorToInt(_s.x)} m" : $"{Mathf.FloorToInt(Progress * 100)}% · best {BestPct(_playLevel)}%", W.Theme.Edge,
NeonWavesGame.UI.cs:324 / void BuildDeathCard()	_dcHead = HzUI.Label("Head", card.transform, "CRASHED", 46, Dim);
NeonWavesGame.UI.cs:326 / void BuildDeathCard()	_dcBig = HzUI.Label("Big", card.transform, "0%", 150, Color.white);
NeonWavesGame.UI.cs:329 / void BuildDeathCard()	_dcSub = HzUI.Label("Sub", card.transform, "", 42, Ink, TextAnchor.MiddleCenter, false);
NeonWavesGame.UI.cs:332 / void BuildDeathCard()	_dcRetry = HzUI.Button("Retry", card.transform, "RETRY", HzUI.Hex("#3DBE6E"), Color.white, () => Retry(), 60);
NeonWavesGame.UI.cs:334 / void BuildDeathCard()	_dcRevive = HzUI.Button("Revive", card.transform, "REVIVE HERE", HzUI.Hex("#7B61FF"), Color.white, () => Revive(), 44);
NeonWavesGame.UI.cs:337 / void BuildDeathCard()	_dcLevels = HzUI.Button("Levels", card.transform, "LEVELS", HzUI.Hex("#4C5A7A"), Color.white, () => ShowMenu(), 40);
NeonWavesGame.UI.cs:339 / void BuildDeathCard()	var tip = HzUI.Label("Tip", card.transform, P ? "tap anywhere to retry" : "Space / click anywhere to retry", 28, Dim, TextAnchor.MiddleCenter, false);
NeonWavesGame.UI.cs:351 / void ShowDeathCard()	_dcHead.text = "RUN OVER";
NeonWavesGame.UI.cs:352 / void ShowDeathCard()	_dcBig.text = Mathf.FloorToInt(_s.x) + " m";
NeonWavesGame.UI.cs:353 / void ShowDeathCard()	_dcSub.text = _newBest ? "NEW BEST DISTANCE!" : $"Best {EndlessBest} m";

NeonWavesGame.UI.cs:357 / void ShowDeathCard()	_dcHead.text = "CRASHED";
NeonWavesGame.UI.cs:358 / void ShowDeathCard()	_dcBig.text = Mathf.FloorToInt(Progress * 100) + "%";
NeonWavesGame.UI.cs:359 / void ShowDeathCard()	_dcSub.text = _newBest ? "NEW BEST!" : $"Best {BestPct(_playLevel)}%";
NeonWavesGame.UI.cs:375 / void ShowDeathCard()	if (Unlocked(i) && Charges(i) > 0) { Toast($"Tip: try {Defs[i].Name} ({i + 1})", HzUI.Hex(Defs[i].Col), 1.8f, 0.86f); break; }
NeonWavesGame.UI.cs:387 / void ShowWin(int stars, int total, int gain)	var p = NeonPopup(title, $"Orbs {_collected}/{total} · +{gain} orbs\nAttempts: {_attempt}", W.Theme.Accent,
NeonWavesGame.UI.cs:425 / void ShowIntro(string id, Action next)	var p = NeonPopup(title, text, W.Theme.Edge, ("Got it!", HzUI.Hex("#3DBE6E"), null));
NeonWavesGame.UI.cs:434 / void ShowUnlock(int h, Action next)	var p = NeonPopup("New helper: " + d.Name, d.Desc + $"\n<color=#FFD23E>+{d.Free} free charges</color>", HzUI.Hex(d.Col), ("Awesome!", HzUI.Hex("#3DBE6E"), null));
NeonWavesGame.UI.cs:452 / void ShowGuide(int h)	_guideTxt.text = Defs[h].Guide + "\n<size=32>It's free this time!</size>";
NeonWavesGame.UI.cs:525 / void PopulateMenu()	var title = HzUI.Label("Title", _menuContent, "NEON WAVES", P ? 132 : 128, Color.white);
NeonWavesGame.UI.cs:529 / void PopulateMenu()	var sub = HzUI.Label("Sub", _menuContent, "Hold. Release. Ride the beat.", P ? 40 : 36, Ink, TextAnchor.MiddleCenter, false);
NeonWavesGame.UI.cs:537 / void PopulateMenu()	_walletTxt = HzUI.Label("N", wal.transform, Wallet.ToString(), 40, Color.white, TextAnchor.MiddleLeft); _walletTxt.rectTransform.Place(new Vector2(0, 0.5f), new Vector2(140, 60), new Vector2(150, 0));
NeonWavesGame.UI.cs:554 / void PopulateMenu()	var play = HzUI.Button("Play", _menuContent, playLabel, HzUI.Hex("#3DBE6E"), Color.white, () => PlayLevel(next), P ? 58 : 52);
NeonWavesGame.UI.cs:558 / void PopulateMenu()	var end = HzUI.Button("Endless", _menuContent, endlessOpen ? "ENDLESS RUN" : "ENDLESS (LV 6)", endlessOpen ? HzUI.Hex("#7B61FF") : new Color(0.25f, 0.25f, 0.35f), Color.white, () =>

`NeonWavesGame.UI.cs:560 / void PopulateMenu()`	`if (unlocked > 5) StartEndless(); else { HzAudio.Play("locked", 0.5f); Toast("Clear sector 1 to unlock the endless run", Color.white, 1.4f, 0.4f); }`
`NeonWavesGame.UI.cs:566 / void PopulateMenu()`	`var eb = HzUI.Label("EB", _menuContent, $"best {EndlessBest} m", 30, Dim, TextAnchor.MiddleCenter, false);`
`NeonWavesGame.UI.cs:579 / void SectorColumn(int s, Vector2 pos, int unlocked)`	`var nm = HzUI.Label("Name", col, NwArt.SectorNames[s].ToUpperInvariant(), 30, open ? th.Edge : Dim);`
`NeonWavesGame.UI.cs:592 / void SectorRow(int s, Vector2 pos, int unlocked)`	`var nm = HzUI.Label("Name", row, NwArt.SectorNames[s].ToUpperInvariant(), 32, open ? th.Edge : Dim, TextAnchor.MiddleLeft);`
`NeonWavesGame.UI.cs:603 / void LevelButton(RectTransform parent, int lv, Vector2 pos, float size, int unlocked, NwTheme th, bool row = false)`	`var b = HzUI.Button("L" + lv, parent, open ? lv.ToString() : null, bg, Color.white, () =>`
`NeonWavesGame.UI.cs:606 / void LevelButton(RectTransform parent, int lv, Vector2 pos, float size, int unlocked, NwTheme th, bool row = false)`	`else { HzAudio.Play("locked", 0.5f); Toast("Clear the previous level first", Color.white, 1.2f, 0.4f); }`
`NeonWavesGame.UI.cs:627 / void LevelButton(RectTransform parent, int lv, Vector2 pos, float size, int unlocked, NwTheme th, bool row = false)`	`var t = HzUI.Label("pct", b.transform, BestPct(lv) + "%", (int)(size * 0.22f), th.Accent);`

CHAPTER 11

Artwork, fonts, audio and licences

Replace game images under Resources/neonwaves 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/neonwaves/Sfx. Preserve each actual load key and importer settings when replacing sounds. This game has no supplied music loop under its Resources Music folder; do not assume that a Music toggle is proof of a bundled track. 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.neonwaves`, version, scene orientation and signing. The build command writes `Builds/Android/neonwaves.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/neonwaves/`. 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.