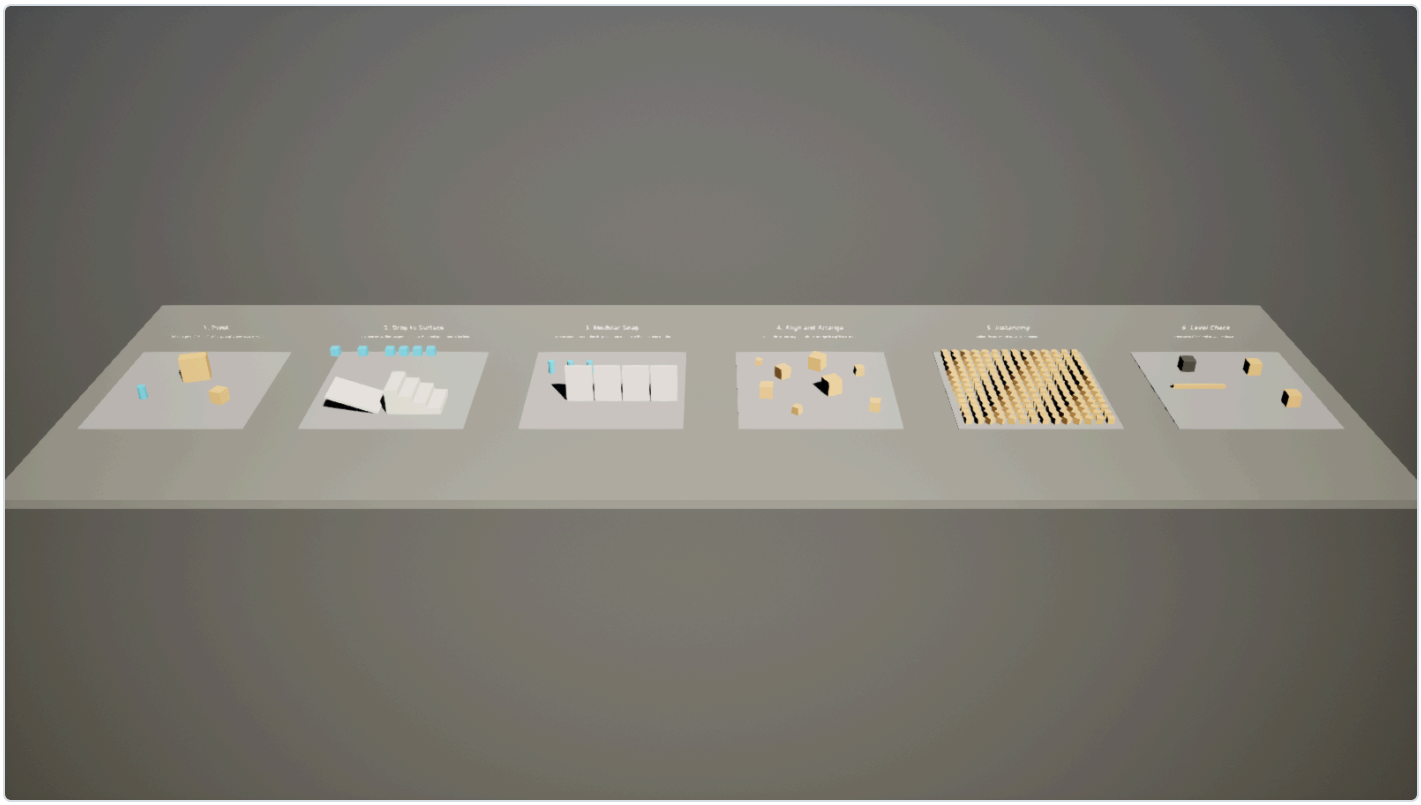
Rookspire Level Design Toolkit

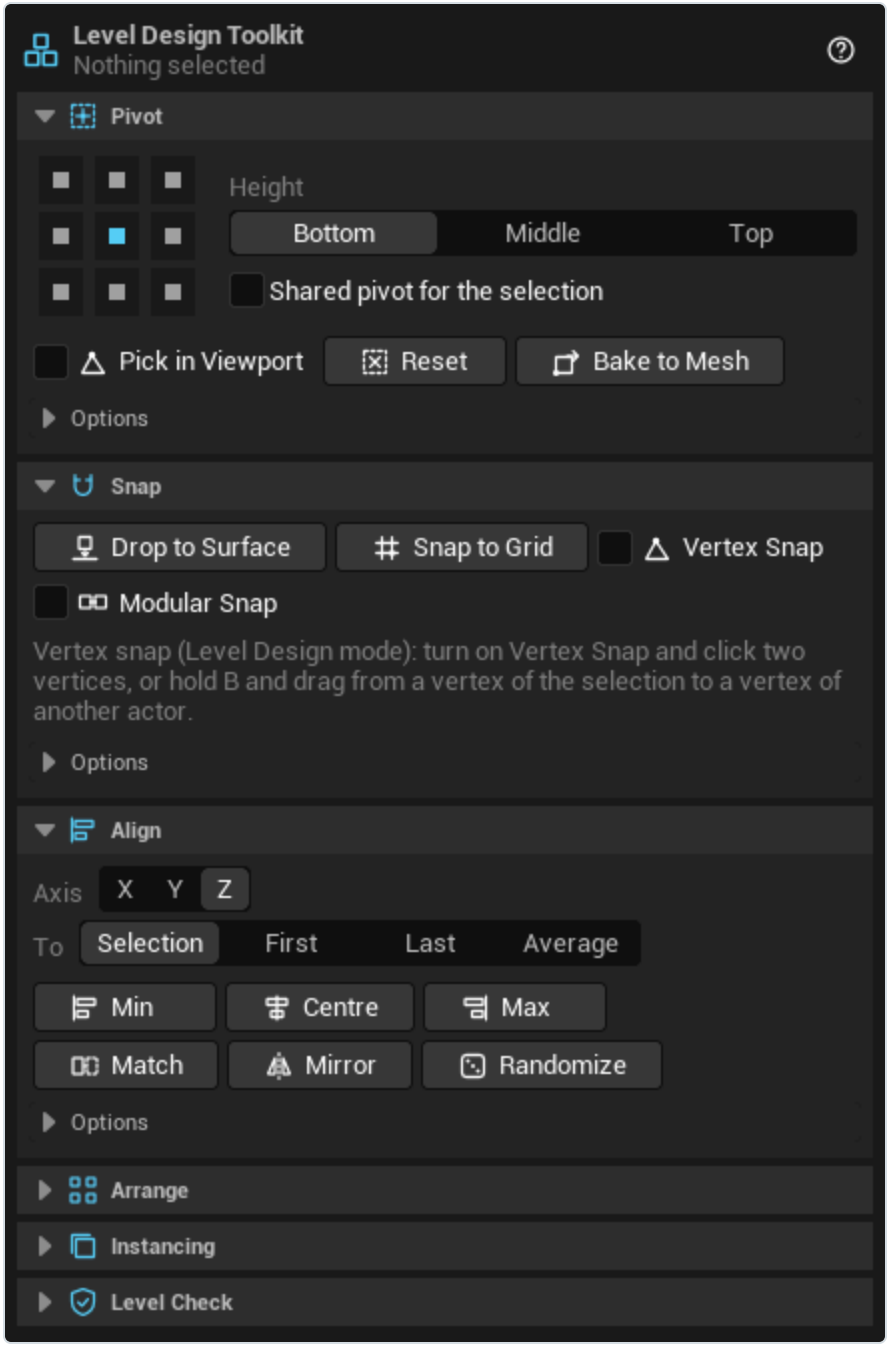
User manual · Version 1.0.0 · Unreal Engine 5.8



Rookspire Level Design Toolkit

Pivot editing, snapping, alignment, arrangement, instancing and a level check, in one editor mode for Unreal Engine 5.6, 5.7 and 5.8.

Everything is undoable, nothing is added to your packaged game, and the levels you edit keep working without the plugin: the toolkit only uses engine classes in your levels.



What is included

Area	What it does
Pivot	27 pivot presets (3×3×3 grid on the bounds), per actor or one shared pivot for the selection. Pick the pivot on a surface or a vertex in the viewport. **Bake to Mesh** makes the pivot the mesh origin: a copy of the mesh by default, or the mesh itself with every placed copy kept in place. Optional freeze of rotation and scale.
Snap	**Drop to Surface**: five rays per actor, so it rests on the highest point below; group mode; align to the surface normal. **Snap to Grid**: location and rotation. **Modular Snap**: an actor you drag snaps to its neighbour as it comes close, using `Snap` sockets, or the faces, edges and corners of its bounds when the mesh has none. **Vertex Snap**: hold a key and drag from a vertex of the selection to a vertex of another actor.
Align	Align min, centre or max on X, Y or Z, relative to the selection, the first or last selected, the average, a target or the grid. Match location, rotation or scale per axis. Mirror. Seeded randomize of rotation, scale and offset.
Arrange	Distribute by centres or by gaps, line up, arrange the selection in a grid, circle or arc, on a sphere or along a spline. Create an array. Place patterns (grid, circle, sphere, line, scatter) as separate actors or one instance holder, with a live preview.
Instancing	Convert Static Mesh Actors to ISM/HISM, grouped by mesh, materials, collision, mobility, shadows, tags and level, with a preview of the draw calls saved. Convert back to actors. Merge and split holders.
Level Check	Finds duplicates stacked on top of each other, missing collision, heavy meshes without Nanite or LODs, missing lightmap UVs, zero, negative or stretched scale, default materials and off-grid actors. Selects the actors involved, repairs the safe ones in one undo step and exports a CSV report.
Scripting	Every operation is a Blueprint and Python function (Editor Utility Blueprints, Python). An event fires after each operation.
Demo	A demo map with one station per tool: `/RookspireLevelDesign/Demo/Maps/L_RLD_Demo`.

Requirements

- Unreal Engine 5.6, 5.7 or 5.8, Windows 64-bit (the platforms tested for this version).
- The plugin is editor-only.

Installation

1. Install the plugin from Fab to your engine (Fab installs code plugins into the engine's plugin folder).

2. Open your project and enable **Rookspire Level Design Toolkit** in *Edit* > *Plugins*, then restart the editor.
3. Open the mode: *Modes* toolbar (top left of the level viewport) > **Rookspire Level Design**.

To try it first, open the demo map `L_RLD_Demo` from the plugin content (enable *Show Plugin Content* in the Content Browser settings). Keys **1** to **6** in the viewport jump to the six stations (camera bookmarks); the blue crates floating above station 2 are there on purpose, for Drop to Surface.

Quick start

1. Select a few actors in the level.
2. In the **Rookspire Level Design** panel, pick an axis and press **Min**, **Centre** or **Max** to align them.
3. Press **Ctrl+Z**: every toolkit action is one undo step.
4. Each section has an **Options** area. The values you set there are remembered per user and project, and are the same values found in *Editor Preferences* > *Plugins* > *Rookspire Level Design*.

The toolkit acts on the level selection in **selection order**. "First" and "Last" (align, match) mean the first and the last actor you selected.

Pivot

- Click a cell of the **pivot grid** to put the pivot there. The grid is seen from above (X from left to right, Y from bottom to top), at the height chosen with **Bottom / Middle / Top**.
- **Shared pivot for the selection** puts one pivot on the bounds of the whole selection, so the transform gizmo turns everything around the same point.
- **Pick in Viewport**: enable it, then click a surface in the viewport while the Level Design mode is active. Hold **Shift** to use the nearest vertex of the clicked actor.
- **Reset** returns the pivot to the actor origin.
- These pivots are the engine's actor pivot offset: they move the gizmo, not the geometry, and no asset changes.
- **Bake to Mesh** makes the current pivot the origin of the Static Mesh:
    - **Create Copy** (default) duplicates the mesh next to the original, as `<name>_Pivoted`, and switches the selected actors to it. The original asset is never changed. When the original is engine content or comes from a plugin installed in the engine (for example the engine's basic shapes), the copy goes to *Copy Folder* in your project (`/Game/BakedMeshes` by default): nothing is ever written into the engine. Actors that share a mesh and a pivot share one copy; actors with different pivots get one copy each.

    - **Modify In Place** changes the mesh asset. Every placed copy in the loaded levels, instances included, is moved so nothing shifts on screen. It is refused for engine and engine-plugin content, and for meshes also used by foliage, landscape grass, procedural (PCG) output or a Blueprint, whose placements the toolkit cannot compensate (use Create Copy). Actors in unloaded World Partition cells or other levels are not loaded, so they are not compensated: use Create Copy for meshes used across many levels.
    - **Freeze Rotation / Freeze Scale** also bake the actor's rotation and scale into the copy, leaving the actor with zero rotation and unit scale. A rotated actor with a non-uniform scale needs both (rotation alone would shear the mesh); the toolkit says so instead of distorting it.
    - Anything attached to the actor (lights, other actors) stays exactly where it is.
    - Every LOD with source geometry, the Nanite source, sockets and simple collision move together.

Snap

- **Drop to Surface** moves each actor down (or along *Drop* > *Direction*) until it rests on the surface below. Five rays per actor, from the centre and the four corners of its base, so an actor half over a step rests on the step, not in it. Options:
    - *As Group* keeps the selection's shape;
    - *Align to Normal* tilts each actor to the slope.
- **Snap to Grid** rounds locations to *Grid Size* (per axis; 0 leaves an axis alone) and rotations to *Rotation Step*.
- **Modular Snap**: when it is on, an actor you drag with the move gizmo jumps onto the nearest neighbour as soon as a snap point comes within *Modular Snap Distance*, and lets go when you pull it half as far again (*Snap While Dragging*, on by default; the two points that meet are highlighted in the Level Design mode). When the drag ends, the snap is applied once more, turning the actor to match (after moving only: rotating or scaling never snaps). Snap points are:
    - the mesh sockets whose name starts with *Snap Socket Prefix* (`Snap` by default). A socket's X axis points out of the piece, towards where the neighbour attaches;
    - otherwise the face centres, edge centres and corners of the actor's bounds, so kits from any seller snap without preparation.

Two points snap only when they face each other (within *Modular Snap Angle*), and the moved actor also turns to match. While the mode is active, the snap points of the selection are drawn in the viewport. The snap while dragging is part of the drag: one undo takes the whole drag back. A value typed in the Details panel is never snapped.

- **Vertex Snap**, in the Level Design mode, two ways:
  - **Click-click**: turn on the **Vertex Snap** checkbox (Snap section). Click a vertex of the selection (the one nearest the cursor is highlighted), then click a vertex of another actor: the selection moves so the two vertices meet.
  - **Hold the key**: hold **B** (*Vertex Snap Key*), press on a vertex of the selection, drag, and release over a vertex of another actor. If you change the key, pick a letter or a function key: Ctrl, Alt and Shift already drive the viewport.

Each snap is one undo step. Nanite meshes use their fallback geometry.

Align

- Choose the **Axis** and what to align **To**:
  - *Selection* (the combined bounds);
  - *First* or *Last* selected (that actor stays);
  - *Average*.
- **Min / Centre / Max** align that side of every actor's bounds.
- **Match** copies the parts ticked in *Options* > *Match* (location, rotation, scale, per axis) from the last selected actor to the others.
- **Mirror** mirrors the selection across the chosen axis, through the selection centre or the world origin. As in the engine, a mirrored actor has a negative scale on one axis.
- **Randomize** adds random rotation, scale and offset within the ranges in *Options*. Each press uses a new seed; undo returns to the previous result.

Arrange

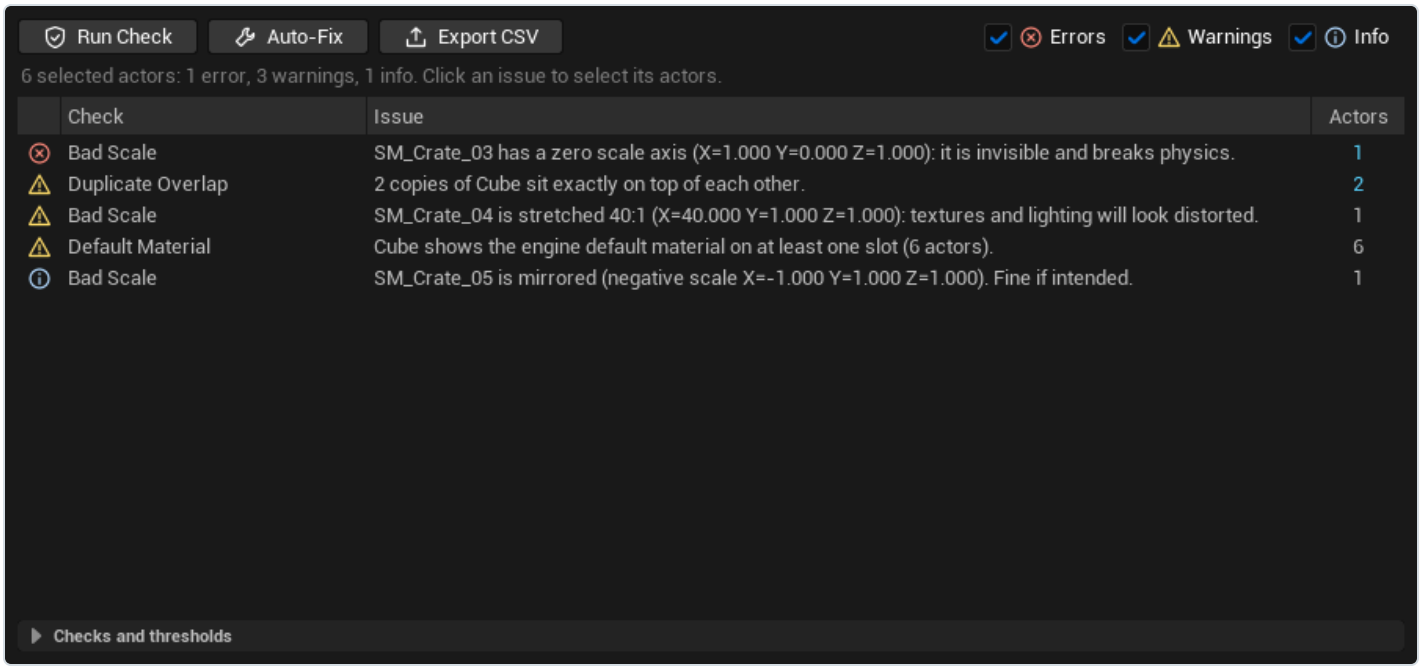
- **Distribute** spaces the selection evenly on the chosen axis; the two ends stay. *Distribute Mode*: equal gaps or equal centre spacing.
- **Line Up** places the actors side by side with *Gap* between them.
- **Grid**, **Circle**, **Sphere** arrange the selection around its centre. A *Radius* of 0 picks one that keeps actors apart. *Facing* turns actors to or away from the centre.
- **Spline**: select the actors and one actor that has a spline component (the path). The actors are spread from its start to its end, optionally turned along it.
- **Create Array** duplicates the first selected actor *Array Count* times along *Array Axis*, spaced by its own size plus *Gap*. Copies stay in the original's level.
- **Place Pattern** places copies of the first selected actor in the *Pattern* (grid, circle, sphere, line, scatter). The pattern follows the actor's rotation. With *Pattern As Instances*, a Static Mesh Actor becomes a single instance holder instead of many actors. The pattern is previewed in the viewport while the mode is active.

Instancing

- The panel shows what **To Instances** would do with the current selection: how many holders it creates and an estimate of the draw calls before and after.
- **To Instances** replaces plain Static Mesh Actors with one holder per mesh, materials, collision profile, mobility, shadow setting, tags and level. A holder is an ordinary actor with an engine instanced mesh component (HISM or ISM), so levels open without the plugin. Cull distance, lighting channels, custom depth, visibility, decals, custom collision responses and the HLOD layer are kept (actors that differ go to separate holders). Blueprints, Static Mesh Actor subclasses, attached actors, actors with extra components, actors in data layers and actors with custom primitive data are skipped and listed (all of them in the Output Log), because converting them could lose their logic or settings.
- **To Actors** turns ISM/HISM holders back into Static Mesh Actors, with the same transforms, mesh, materials, tags and folder. Foliage, landscape grass, procedural (PCG) output and Blueprint instances are left alone, because their owners generate them; holders with per-instance custom data (which separate actors cannot keep) and more than 10,000 instances at once are refused, because that many actors would make the level very slow. Instances carry no names, so the new actors are named after the mesh; Undo right after To Instances gives the original actors back with their names.
- **Merge** joins holders that share mesh and settings. **Split** moves the selected instances into a holder of their own, keeping their per-instance data, in one undo step. To select one instance, double-click it in the viewport until only it is selected (the first double-click selects the holder's component, the next one the instance); Ctrl+click then adds more.

Level Check

Open it from the panel, from *Tools* > *Rookspire Level Design* > *Level Check*, or from *Window*. **Run Check** looks at the whole level, or only at the selected actors when **Selected Only** is on; the summary line says which ("Whole level" or "N selected actors"). Clicking an issue selects its actors (and frames them) without changing what the next run checks.



Check	Severity	Auto-Fix
Duplicate Overlap: same mesh, same transform, same level and data layers	Warning	Removes the extra copies (the first stays) when they are plain Static Mesh Actors with the same materials
Bad Scale: a zero axis (meshes; volumes and brushes are left alone)	Error	Sets that axis to 1
Bad Scale: stretched (largest axis over the middle one above *Non Uniform Scale Ratio*, 10 by default; flat slabs such as a cube scaled into a floor are not reported)	Warning	—
Bad Scale: negative (mirrored)	Info	—
Missing Collision: collision enabled, no simple collision, not complex-as-simple (including through the project's default)	Warning	—
High Triangle Count: over *Triangle Budget* and not Nanite	Warning	—
Missing LODs: over *LOD Triangle Threshold* with one LOD and not Nanite	Info	—
Missing Lightmap UVs: static meshes without a separate lightmap channel, generated ones included (only when the project allows static lighting)	Warning	—
Default Material: a slot shows the engine default material	Warning	—
Off Grid (off by default)	Info	Snaps to the grid

- Click an issue to select its actors and frame them.

- **Auto-Fix** first checks the same scope again (the level may have changed), then repairs the issues shown with a blue actor count, in one undo step, and checks once more so the list shows what is left. Rows hidden by the severity filters are not fixed, and actors in locked levels are never changed.
- **Export CSV** saves the list for a spreadsheet or a bug tracker.

Keyboard shortcuts

Every command is listed in *Edit* > *Keyboard Shortcuts* (search "Rookspire"). None has a default key, so nothing clashes with the engine's shortcuts; when you pick a key the level editor already uses, the window reports the conflict. Bind the ones you use most, for example *Drop to Surface*, *Align Bottoms*, *Pivot to Bottom Centre* or *Convert to Instances*. The bound commands work in the level editor even when the Level Design mode is not active.

Scripting

All operations are in the Blueprint function library **Rookspire Level Design** (category *Rookspire* | *Level Design*), available in Editor Utility Blueprints and in Python as `unreal.RLDEditorLibrary`. For example:

```
import unreal
actors = unreal.get_editor_subsystem(unreal.EditorActorSubsystem).get_selected_level_actors()
unreal.RLDEditorLibrary.align_actors(actors, unreal.RLDAxis.Z, unreal.RLDAxisAnchor.MIN, unreal.RLDAlignRefe
report = unreal.RLDEditorLibrary.convert_to_instances(actors, True)
```

The editor subsystem **RLD Editor Subsystem** has an **On Operation Completed** event (operation name and actors), fired after every toolkit operation that changed something. The full list of functions is in the API reference.

Performance

Measured by the automated tests in the level editor's own world (UE 5.6 and 5.7, Intel Core i7-14700). Each moved actor is updated the way the editor's own gizmo updates it, so the time grows with the number of actors:

Operation	Actors	Time
Align	1,000	about 0.5 s

Distribute	1,000	about 0.6 s
Convert to instances	1,000	about 0.6 s
Level Check	2,000	under 0.01 s
Scatter pattern	10,000 points	under 1 s

Limitations

- Tested on Unreal Engine 5.6, 5.7 and 5.8, Windows 64-bit.
- *Modify In Place* compensates placed copies in the loaded levels only; see *Pivot*.
- *To Instances* converts plain Static Mesh Actors only, by design.
- Vertex snap reads up to 20,000 vertices per actor, within the modular search radius. Very dense meshes snap to the first 20,000 vertices of their source geometry.
- Modular Snap works with the move gizmo; rotating or scaling a piece never snaps it.
- The toolkit does not change the level while Play or Simulate runs (the editor forbids it); stop playing first.
- Undo of *Bake to Mesh (Create Copy)* gives the actors their original mesh back; the new `_Pivoted` asset stays in the Content Browser (it is not saved until you save it; delete it if you do not want it).
- A mesh baked *In Place* keeps its source file path: reimporting it from that file brings the old origin back while the placed actors stay compensated. Bake again after a reimport, or use Create Copy for meshes you reimport often.

Made with AI assistance and tested by Rookspire.

API reference

Blueprint function library **Rookspire Level Design** (`URLDEditorLibrary`), category *Rookspire | Level Design*. In Python: `unreal.RLDEditorLibrary.<snake_case_name>` . Every function is one undo step, skips null actors and returns how many actors it changed (or a report).

General

Function	Description
`GetActorWorldBounds`	World bounds of an actor from all its components (colliding or not). An actor without components gives a zero-size box at its location. `FBox GetActorWorldBounds(const AActor* Actor)`

Pivot

Function	Description
`SetActorsPivotPreset`	Set each actor's pivot offset (the editor gizmo position) to preset Index [0, 26] of its own local bounds. No asset changes. `int32 SetActorsPivotPreset(const TArray<AActor*>& Actors, int32 PresetIndex)`
`SetGroupPivotPreset`	Set every actor's pivot offset to one shared world point: preset Index of the combined bounds. `int32 SetGroupPivotPreset(const TArray<AActor*>& Actors, int32 PresetIndex)`
`SetActorsPivotWorld`	Put each actor's pivot offset on a world point. `int32 SetActorsPivotWorld(const TArray<AActor*>& Actors, const FVector& WorldPoint)`
`ResetActorsPivot`	Return each actor's pivot offset to its origin. `int32 ResetActorsPivot(const TArray<AActor*>& Actors)`
`BakeActorPivots`	Bake each actor's pivot offset into its Static Mesh (a copy by default) so the pivot becomes the mesh origin. Vertices keep their world position. `FRLDBakeReport BakeActorPivots(const TArray<AActor*>& Actors, const FRLDBakeSettings& Settings)`

Snap

Function	Description

`ModularSnapActors`	Snap the actors, as one group, to the nearest neighbour: the closest pair of facing snap points (snap sockets, or automatic points on the bounds) within Settings.MaxDistance. Returns true when it snapped. `bool ModularSnapActors(const TArray<AActor*>& Actors, const FRLDModularSnapSettings& Settings)`
`SnapActorsPointToPoint`	Move the actors by the offset that brings SourcePoint onto TargetPoint (vertex snap). `int32 SnapActorsPointToPoint(const TArray<AActor*>& Actors, FVector SourcePoint, FVector TargetPoint)`
`SnapActorsToGrid`	Snap locations to LocationGrid (per axis; 0 skips an axis) and rotations to RotationStep degrees (0 skips rotation). `int32 SnapActorsToGrid(const TArray<AActor*>& Actors, FVector LocationGrid, double RotationStep)`
`DropActorsToSurface`	Move actors along Settings.Direction until they rest on the first surface found. Five rays per actor (centre and the four corners of the leading face) so an actor rests on the highest point under it. Actors that find nothing stay. `int32 DropActorsToSurface(const TArray<AActor*>& Actors, const FRLDDropSettings& Settings)`

Align

Function	Description
`AlignActors`	Align Anchor (min, centre or max) of every actor's bounds on Axis to the Reference: the selection bounds, the first or last actor (which stays), or the average. `int32 AlignActors(const TArray<AActor*>& Actors, ERLDAxis Axis, ERLDAxisAnchor Anchor, ERLDAlignReference Reference)`
`AlignActorsToTarget`	Align Anchor of every actor's bounds on Axis to the same anchor of Target's bounds. Target itself does not move. `int32 AlignActorsToTarget(const TArray<AActor*>& Actors, const AActor* Target, ERLDAxis Axis, ERLDAxisAnchor Anchor)`
`AlignActorsToGrid`	Move every actor so Anchor of its bounds on Axis lands on the nearest multiple of GridSize (cm). `int32 AlignActorsToGrid(const TArray<AActor*>& Actors, ERLDAxis Axis, ERLDAxisAnchor Anchor, double GridSize)`
`MatchActorTransforms`	Copy the location, rotation and scale parts chosen in Flags (per axis) from Target to every other actor. `int32 MatchActorTransforms(const TArray<AActor*>& Actors, const AActor* Target, const FRLDMatchFlags& Flags)`
`MirrorActors`	Mirror in place. Mirroring gives a negative scale on one axis, as the engine's own mirror does. `int32 MirrorActors(const TArray<AActor*>& Actors, ERLDAxis Axis, ERLDMirrorPlane Plane)`

Arrange

Function	Description
`DistributeActors`	Space the actors evenly on Axis, ordered by position; the first and last stay. Mode: equal gaps between bounds or equal distance between centres. Needs 3 actors. `int32 DistributeActors(const TArray<AActor*>& Actors, ERLDAxis Axis, ERLDDistributeMode Mode)`
`LineUpActors`	Place the actors side by side on Axis, ordered by position, with Gap (cm) between neighbouring bounds. The first stays. `int32 LineUpActors(const TArray<AActor*>& Actors, ERLDAxis Axis, double Gap)`
`RandomizeActorTransforms`	Add random rotation (local), scale (multiplied) and offset (world) within Settings. The same Seed always gives the same result. `int32 RandomizeActorTransforms(const TArray<AActor*>& Actors, const FRLDRandomSettings& Settings, int32 Seed)`
`ArrangeActorsInGrid`	Arrange the actors (in the given order) in a grid of Columns on the XY plane, centred on the selection. Cells fit the largest actor plus Gap. Heights are kept. `int32 ArrangeActorsInGrid(const TArray<AActor*>& Actors, int32 Columns, double Gap)`
`ArrangeActorsInCircle`	Arrange the actors on a circle or arc around the selection centre. Radius <= 0 picks one so the actors do not touch. `int32 ArrangeActorsInCircle(const TArray<AActor*>& Actors, double Radius, double StartDegrees, double ArcDegrees, ERLDFacing Facing)`
`ArrangeActorsOnSphere`	Arrange the actors evenly on a sphere around the selection centre. Radius <= 0 picks one so the actors do not touch. `int32 ArrangeActorsOnSphere(const TArray<AActor*>& Actors, double Radius, ERLDFacing Facing)`
`ArrangeActorsAlongSpline`	Arrange the actors evenly along the first spline component of SplineActor, from start to end. `int32 ArrangeActorsAlongSpline(const TArray<AActor*>& Actors, const AActor* SplineActor, bool bAlignToSpline)`
`CreateActorArray`	Duplicate Source Count times, each copy Step further than the previous one. Returns the copies. `TArray<AActor*> CreateActorArray(AActor* Source, int32 Count, FVector Step)`
`CreateActorArrayAuto`	Duplicate Source Count times along Axis, spaced by its own size plus Gap. Returns the copies. `TArray<AActor*> CreateActorArrayAuto(AActor* Source, int32 Count, ERLDAxis Axis, double Gap)`
`SpawnAtTransforms`	Place copies of Source at RelativeTransforms (relative to Source's transform). With bAsInstances, a Static Mesh Source becomes one instance holder instead of many actors. Returns the created actors. `TArray<AActor*>`

	`SpawnAtTransforms(AActor* Source, const TArray<FTransform>& RelativeTransforms, bool bAsInstances)`

Instancing

Function	Description
`ConvertToInstances`	Replace plain Static Mesh Actors with instanced meshes, one holder per mesh + materials + settings + level. `FRLDInstancingReport ConvertToInstances(const TArray<AActor*>& Actors, bool bHierarchical = true)`
`EstimateInstancing`	Preview of ConvertToInstances: groups and estimated draw calls, nothing changes. `FRLDInstancingEstimate EstimateInstancing(const TArray<AActor*>& Actors)`
`ConvertToActors`	Replace the instances of these actors with Static Mesh Actors (works on any ISM/HISM holder). `FRLDInstancingReport ConvertToActors(const TArray<AActor*>& Actors)`
`MergeInstanceActors`	Move all instances of holders that share mesh and settings into the first holder of each group and remove the emptied ones. `FRLDInstancingReport MergeInstanceActors(const TArray<AActor*>& Actors)`
`SplitInstances`	Move the given instances of Holder's instance component into a new holder with the same mesh and settings. `FRLDInstancingReport SplitInstances(AActor* Holder, const TArray<int32>& InstanceIndices)`

Level Check

Function	Description
`RunLevelCheck`	Run the Level Check on Actors, or on every loaded level of the editor world when Actors is empty. Issues are sorted by severity. `TArray<FRLDCheckIssue> RunLevelCheck(const TArray<AActor*>& Actors, const FRLDCheckSettings& Settings)`
`AutoFixLevelIssues`	Repair the issues that allow it (duplicates, zero scale, off grid) in one undo step. Returns how many were repaired. `int32 AutoFixLevelIssues(const TArray<FRLDCheckIssue>& Issues, const FRLDCheckSettings& Settings)`
`ExportLevelCheckCsv`	Save the issues as a CSV report. `bool ExportLevelCheckCsv(const TArray<FRLDCheckIssue>& Issues, const FString& FilePath)`

Event

RLD Editor Subsystem (`URLDEditorSubsystem` , an editor subsystem): **On Operation Completed** (`FName Operation` , `TArray<AActor*> Actors`) fires after every toolkit operation that changed something.

Troubleshooting

I cannot find the mode

The mode is in the *Modes* toolbar at the top left of the level viewport, as **Rookspire Level Design**. If it is not there, check that the plugin is enabled in *Edit* > *Plugins* and that the editor was restarted.

A button says "Select at least N actors"

Align needs 2 actors, Distribute needs 3, and most other tools need 1. The toolkit works on the level selection, in the order you selected the actors.

Pick in Viewport does nothing

Pick in Viewport and Vertex Snap work while the **Rookspire Level Design** mode is active. Select the actors first, then click where the pivot goes.

Modular Snap does not snap

- Check that **Modular Snap** is on (panel, Snap section, or *Tools* > *Rookspire Level Design*).
- It snaps within *Modular Snap Distance* (30 cm by default): while you drag with the move gizmo (*Snap While Dragging*), and again when the drag ends. Rotating or scaling snaps only at the end.
- The two points must face each other within *Modular Snap Angle*. For socket snapping, the socket's X axis must point out of the piece, and its name must start with *Snap Socket Prefix* (`Snap` by default).
- Meshes that have snap sockets use only those; meshes without them use their bounds when *Automatic Snap Points* is on.

Drop to Surface leaves an actor in the air

Nothing below it blocks the Visibility trace channel within *Drop* > *Max Distance*. Check the collision of the surface.

To Instances skipped some actors

Only plain Static Mesh Actors are converted. Blueprints, subclasses, attached actors and actors with extra components keep their own behaviour, so they are skipped. The notification lists each one with the reason.

Split says "Select instances first"

Split works on single instances, not on the whole holder. In the viewport, a click selects the holder; double-click an instance to select the holder's component, then double-click it again to select only that instance. Ctrl+click adds more instances of the same holder. Then press **Split**.

The Level Check says "no issues" but I can see problems

Look at the start of the summary line. "N selected actors" means only the selection was checked: clicking an issue selects its actors, so a later **Run Check** checks just those. Deselect everything (Esc in the viewport) and press **Run Check** again for the whole level.

A changed default does not apply

Settings are saved per user as soon as you change them, so a new default in an update does not replace a value you already have. Press the reset arrow next to the setting in *Editor Preferences* > *Plugins* > *Rookspire Level Design*.

A shortcut I assigned does nothing

The Keyboard Shortcuts window (*Edit* > *Keyboard Shortcuts*) warns when the key is already used by the level editor; choose another key. For example, Ctrl+Shift+D is the level editor's *Select All Descendants*. The toolkit's commands have no keys by default, so they never take yours.

Holding the Vertex Snap Key does not snap

Hold the key before you press on the vertex, keep it held while you drag, and release the mouse over the target vertex. The viewport must have keyboard focus (click in it first). If you changed *Vertex Snap Key*, use a letter or a function key: Ctrl, Alt and Shift already drive the viewport. The **Vertex Snap** checkbox (click-click) needs no key.

Bake to Mesh changed my mesh for every level

That is **Modify In Place**. Undo restores it. For meshes used in several levels, use **Create Copy** (the default), which never changes the original asset.

Bake to Mesh put the copy in /Game/BakedMeshes

The original mesh is engine content, or it comes from a plugin installed in the engine (Fab installs plugins there). The toolkit never writes into the engine, so the copy goes to *Bake* > *Copy Folder* in your project. *Modify In Place* is refused for those meshes for the same reason.

Bake to Mesh says "Nothing to bake"

The actor's pivot already is the mesh origin. Move the pivot first (pivot grid or Pick in Viewport), then bake.

To Actors skipped foliage, grass or a Blueprint

Their instances belong to the tool that makes them (the foliage tool, the landscape grass, PCG, the Blueprint's construction script); converting them would duplicate the geometry or clear the owner's data. Holders made by To Instances or Place Pattern convert normally. More than 10,000 instances at once are refused as well: that many actors make a level very slow. The Output Log lists every skipped actor with the reason.

The Level Check reports "Default Material" on engine shapes

The engine's basic shapes (Cube, Sphere, and so on) use the default grid material. The check reports it so that placeholder materials do not ship by mistake.

Where are my settings?

Editor Preferences > *Plugins* > *Rookspire Level Design*. They are saved per user and per project, in `Saved/Config/<Platform>/EditorPerProjectUserSettings.ini`.

Changelog

1.0.0

First release for Unreal Engine 5.6, 5.7 and 5.8 (Windows 64-bit):

- Pivot: 27 presets, per actor or shared, picking in the viewport, and bake to mesh (copy or in place, with freeze of rotation and scale).
- Snap: drop to surface, snap to grid, modular snap while dragging (sockets or automatic points), vertex snap (click-click or hold a key).
- Align: align, match, mirror, randomize.
- Arrange: distribute, line up, grid, circle, sphere, spline, create array, place pattern with preview.
- Instancing: convert to ISM/HISM and back, merge, split, draw-call estimate.
- Level Check: 8 checks, auto-fix, CSV export.
- Blueprint and Python function library, operation event, demo map with camera bookmarks.