Life-Sim Building System

by OguzGames · version 1.0.0 · Unreal Engine 5.8 · Windows



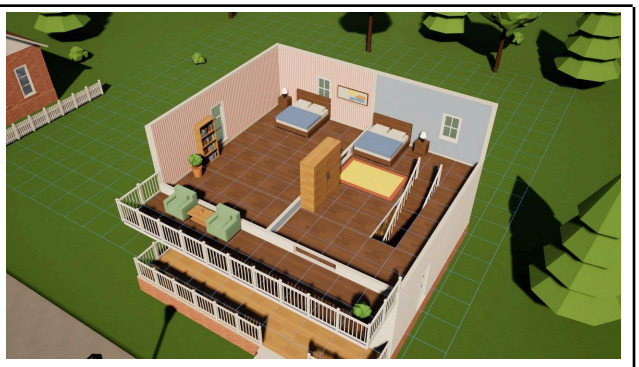
Grid-based, life-sim style building for players **and** level designers. Draw walls and whole rooms, place doors and windows, paint walls and floors, raise foundations and porches, stack stories with stairs, add roofs and railings, furnish rooms (floor items, wall decor, small decor on tables, functional Blueprint furniture), look inside with cutaway walls, undo anything, save the result — and bake finished houses to regular static meshes.

Everything is C++ with a complete Blueprint API. No third-party content: all meshes, textures and materials are original.

1. Features at a glance



Cutaway views — walls between the camera and the rooms drop while you build.



Multiple stories — stairs open the floor above; balconies, per-room wallpaper.



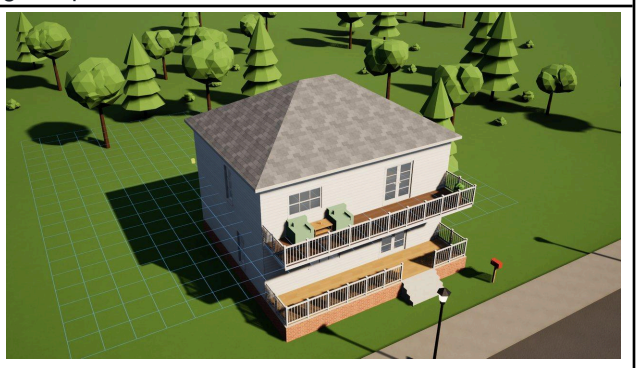
Furniture — wall items show every free spot; decor sits on tables and shelves.



Doors and windows — real holes, 1–3 tiles wide, live ghost previews.



Foundations, porches, terraces — raised floors with entry steps and railings.



Roofs — gable, hip, shed and flat, with overhang, fascia and gable walls.



Animated doors — open when a character walks up, or with the Use key.



Build animation — replay a house being built, from Blueprint or Sequencer.

2. Quick start

Try the demo

1. Enable **Life-Sim Building System** in *Edit → Plugins* and restart the editor. (It enables *Geometry Scripting* and *Enhanced Input* automatically.)
2. In the Content Browser enable *Settings → Show Plugin Content*.
3. Open `**/ModularBuildSystem/Demo/Maps/MBS_Demo**` and press **Play**.



- The **empty lot** in front (*BuildableLot*) is yours to build on.
- The three show houses (*ShowcaseCottage*, *ShowcaseModern*, *ShowcaseFamily*) are **baked**: plain static meshes, the way you would ship finished buildings. Their doors still open and their Smart TVs and lamps still work — point at one and press **U**.

Use it in your own level (2 minutes)

4. *Place Actors* → search **MBSLot** → drag it into the level. Its origin is the corner of the grid.
5. *World Settings* → *GameMode Override* → **MBSBuildGameMode** (or use `AMBSBuildPlayerController` + `AMBSBuildCameraPawn` in your own game mode).
6. Press **Play** and start drawing walls.

> **Tip:** select a lot and press **Build Test House** in its Details panel to see many features at once (two stories, stairs, roofs, paint, railings).

3. Controls and user interface

All bindings live in the Enhanced Input mapping context **`IMC_MBS_Build`** and can be changed there (see Input).

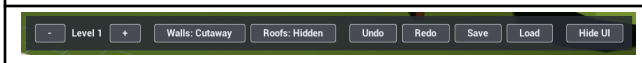
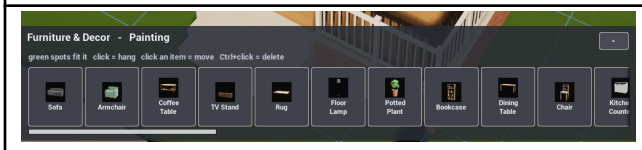
Key	Action
1	Wall tool · **Shift + drag** draws a whole room
2	Doors & windows
3	Wall paint / wallpaper / cladding
4	Floors (inside rooms *and* outdoors: patios, paths)
5	Stairs
6	Roofs
7	Furniture & decor

8	Foundation (drag) · **Shift + click** beside it = entry steps · **,** / **.** height
Menu	Room tool, Railing / fence tool
U / middle mouse	Use the functional furniture under the cursor (TV, lamp…) · open / close a door
LMB (drag)	Place / draw
Ctrl + LMB	Erase / remove / reset paint
Shift + LMB	Whole room (paint, floors)
Alt + LMB	Quarter tile (floors next to diagonal walls)
RMB / Backspace / Esc	Cancel the current drag
Z / X	Previous / next catalog item
R	Rotate (stairs, furniture 90°, decor 45°) · next roof type
T	Roof direction (gable ridge / shed slope)
, / **.**	Roof pitch − / +
F	Flip door / window
V	Walls: up / cutaway / down
H	Show / hide all roofs
PageUp / PageDown	Change story
Ctrl+Z / Ctrl+Y (Ctrl+Shift+Z)	Undo / redo
F5 / F9	Quick save / quick load
WASD / arrows	Move camera · **Q / E** rotate · **mouse wheel** zoom
Gamepad	Left stick move · right stick rotate / zoom · shoulders change story (camera only)

Furniture tool: click an item to pick it up and move it (decor on it comes along) · **RMB** puts a picked-up item back · **Ctrl + click** deletes · with decor selected, **click** a table / counter / shelf to put it there, **Shift + click** picks the table up instead · **Alt** turns off the 5 cm decor snapping.

On screen

Panel	

	**Mode tabs and side panel** (top left). **BUILD** lists the tools in groups (Structure, Openings, Decorate) with their hotkeys; **FURNISH** lists furniture categories taken from each definition's *Category*. Collapse it with **−**.
	**Toolbar** (top right): story − / +, wall view (up / cutaway / down), roofs, undo / redo, save / load and **Hide UI**.
	**Catalog panel** (bottom): the tool and selected item, short help, a warning when something can't be placed (*"A wall is in the way"*) and the catalog with icons.

Console helpers (handy for testing and scripted screenshots):

Command	Does
`MBSTool Furniture`	Selects a tool by name (`Wall`, `Room`, `Opening`, `WallPaint`, `FloorPaint`, `Stairs`, `Roof`, `Railing`, `Furniture`, `Foundation`)
`MBSItem Sofa`	Selects the first catalog entry of the current tool whose asset name contains the text
`MBSCamera X Y Yaw Zoom` · `MBSPitch Close Far`	Places the build camera · sets its tilt
`MBSView Mode Level AllRoofs` · `MBSGrid 0/1`	Wall view (0 up, 1 cutaway, 2 down), active story, roofs · hides / shows the grid
`MBSDoors 1/0` · `MBSTestPawn X Y Z`	Opens / releases every door · drops a test pawn (doors open around it)
`MBSTimelapse 20`	Plays the build animation of the active lot
`MBSShot 3 [0]` · `MBSMouse X Y` · `MBSAt 5 Cmd;args`	Screenshot after 3 s (0 = without UI) · scripted cursor · run a command later (`;` = space)

4. How it works

The lot

`AMBSLot` owns one building plot: a square grid (`TileSize`, default 100 cm), stacked stories (`LevelHeight`, default 300 cm, up to `MaxLevels`). It generates all geometry procedurally:

Component	Content
WallMesh	Walls with mitred corners, T / X junctions, 45° and crossing diagonals, door/window holes, baseboards and crown molding
FloorMesh	Room floors (detected automatically), painted outdoor tiles, ceilings under upper floors
FoundationMesh	Raised foundation blocks and entry steps
StairMesh	Straight stairs, one tile wide
RoofMesh	Gable, hip, shed and flat roofs with overhang, fascia and gable walls
Static / instanced meshes	Door and window frames and leaves, railings and fences, furniture

Several lots can live in one level (a street of plots); the build tools work on the lot under the player.

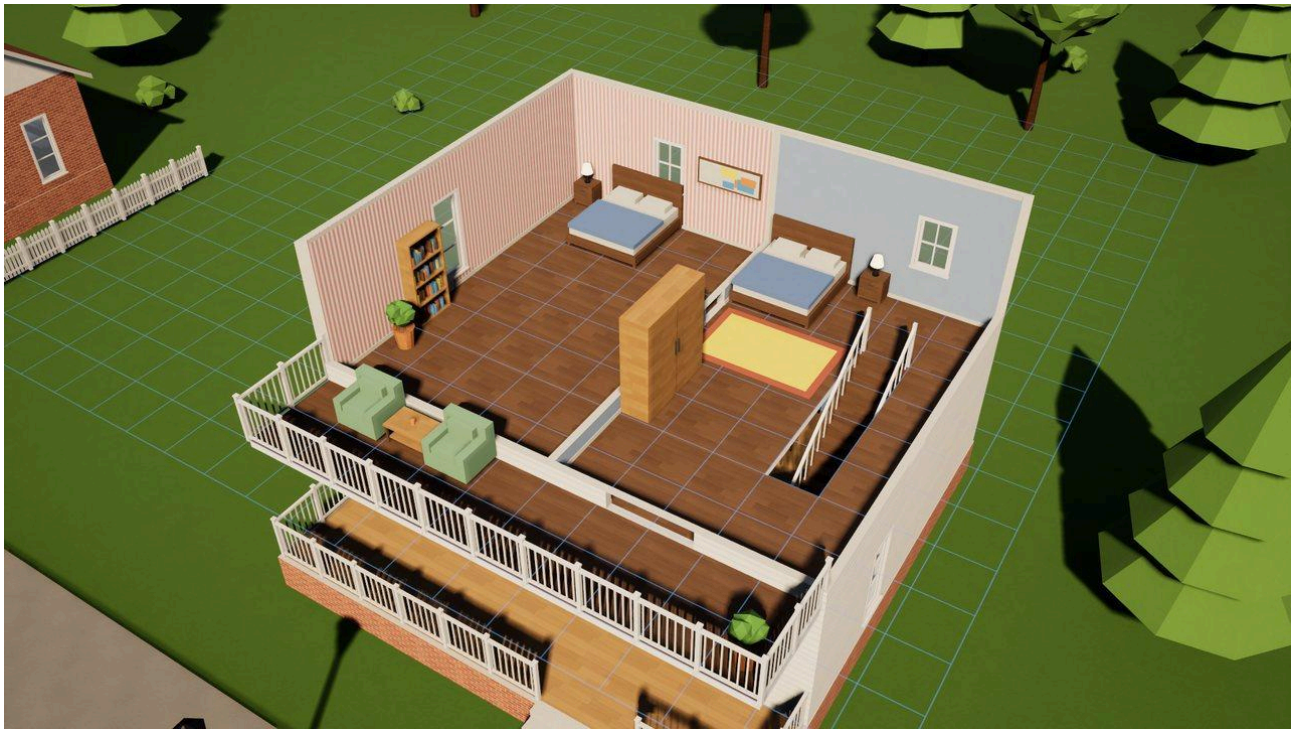
Walls and rooms

Walls are stored as unit segments between grid vertices (straight or 45°). Closed loops of walls form **rooms**; floors appear inside rooms automatically. Each wall segment has two faces that are painted independently, and the system knows which room each face looks into (used for "paint the whole room" and for trims).



Stories

- Upper walls need a wall directly below or a floor next to them.
- Upper floor tiles need something below them (a room, a floor or a wall on their border — one-tile balconies are allowed).
- Stairs cut the opening in the floor above automatically and block walls that would close their entrance or exit.



Foundations and verandas

Drag with the **Foundation** tool to raise ground-floor tiles by the lot's **Foundation Height** (default 75 cm; **,** / **.** change it in 15 cm steps). Build the house on top, or leave tiles without walls as a raised **veranda / porch** (add railings and floor paint as usual). **Shift + click** a ground tile right next to the foundation to add **entry steps** climbing onto it.



- Walls inside or along the foundation start on top of it; other ground-floor walls grow taller so every wall of a story ends at the same height. Upper stories and roofs move up by the foundation height.
- Floors, doors, windows, railings, stairs and furniture follow the raised floor. Stairs and floor items cannot stand half on a foundation.
- Sides use *Default Foundation Material* (brick), the top and steps the wall top material; a lot can override both.

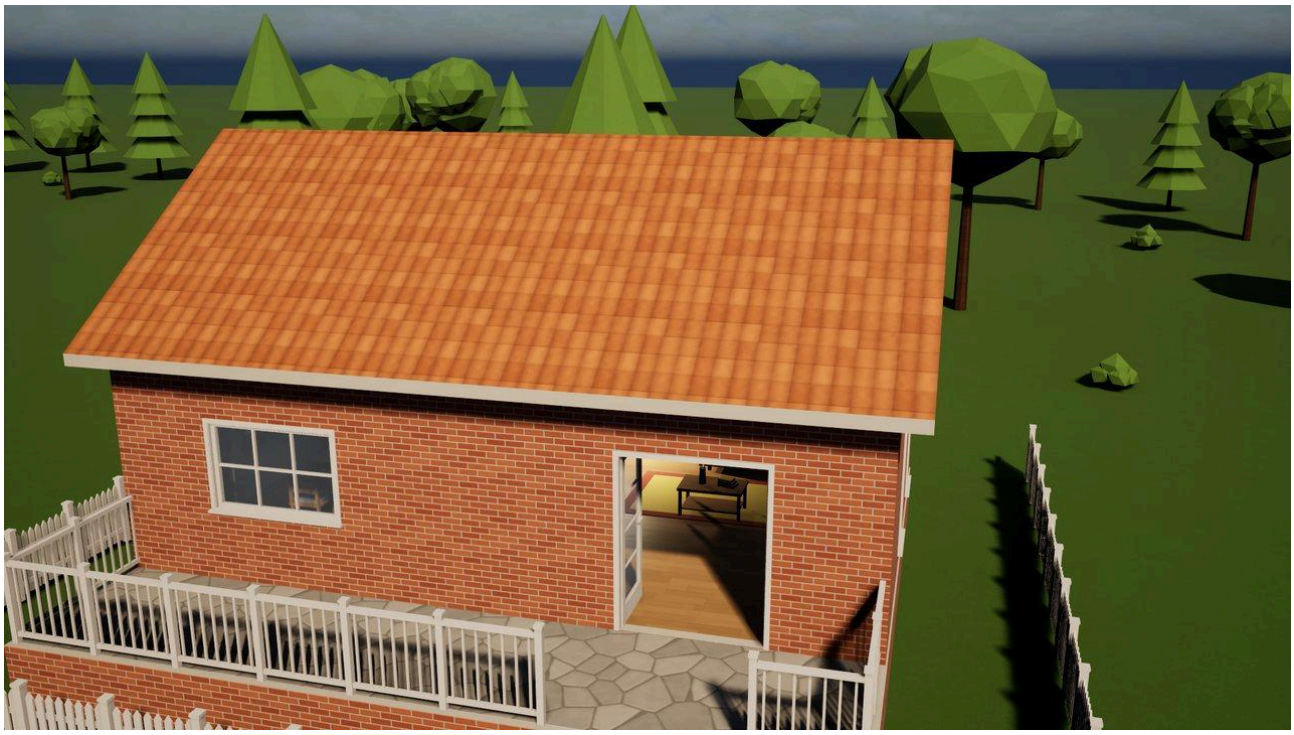
Cutaway

In play, walls on the active story between the camera and a room drop to `LoweredWallHeight`; stories above the active one are hidden; roofs over the active story are hidden until you go up a story or press **H**.

View (V)	Behaviour
Up	Full height everywhere
Cutaway	Walls between the camera and a room are lowered; back walls stay up
Down	Every wall on the active story is lowered

Doors

In play, door leaves **swing open when a pawn comes near** (`DoorOpenDistance`, default 160 cm) and close again when it walks away. The leaf always swings away from whoever opens it, and leaves never block characters (the hole in the wall is the passage). **U / middle mouse** on a leaf opens it and keeps it open until used again. This works on baked houses too.



Lot settings (*Build* | *Doors*): `bAutoOpenDoors`, `DoorOpenDistance`, `DoorSwingTime` (0.45 s), `DoorPawnClass` (only pawns of this class open doors; the build camera never does). Per door type: `bAnimatedLeaf` and `OpenAngle` on the opening definition. Bind `OnDoorChanged (Door, bOpen)` to play sounds.

Build animation

`PlayBuildAnimation(Duration)` replays the lot's current house being built from an empty lot, in this order:

Stage	Share of the time
Foundation and entry steps	5 %
Walls (front to back, story by story)	30 %
Doors and windows	10 %
Floors	8 %
Wall paint	8 %
Stairs and railings	7 %
Furniture (items on tables after their table)	24 %
Roofs	8 %

Empty stages are skipped and the rest stretched. Use it for "house reveal" moments or to load a prefab with style (`OnBuildAnimationFinished` fires at the end; the undo history is cleared).



For trailers: keyframe the lot's **Build Progress** property in a Level Sequence (0 = empty lot, 1 = finished house, negative = off). It applies in play and in Movie Render Queue renders. When rendering with Movie Render Queue, set *Game Overrides* → *Game Mode Override* to `MBSBuildGameMode` so the cutaway walls work in the render.

Furniture

Four kinds of items, set by the definition's **Placement**:

Placement	Behaviour
Floor	Snaps to tiles (footprint 1×1 up to 4×4), rotates in 90° steps. Optional *Requires Wall Behind* (wardrobes, counters, beds) — the item is pushed flush against the wall face.
Wall	Hangs on a straight wall face at *Mount Height*. When selected, every free spot on the active story is highlighted at the item's height. Items at different heights can share a wall (a shelf under an air conditioner); doors and windows block only where heights overlap.
Floor Covering	Rugs: a separate layer, so tables and sofas stand on them.
Surface	Small decor placed freely (5 cm snapping, 45° turns) on top of furniture that has a **surface** (tables, counters, nightstands, shelves) or on the floor. It moves and is deleted with the furniture it stands on.



Rules: floor items need a floor (upper stories), cannot cross walls, railings or stairs, and do not overlap items of the same layer. Removing a wall or floor removes what depended on it. Wall items, and decor on wall shelves, hide with cut-away walls.

5. Settings reference

Project settings

Project Settings → Plugins → Modular Build System

Setting	Purpose
Opening Catalog	Doors and windows offered by the tool (`UMBSOpeningDefinition`)
Wall / Floor / Roof Paint Catalog	Materials offered by the paint tools
Railing Catalog	Railings and fences (`UMBSRailingDefinition`)
Furniture Catalog	Furniture, wall decor, rugs and surface decor (`UMBSFurnitureDefinition`)
Default Wall / Wall Top / Floor / Ceiling / Roof / Foundation Material	Used where nothing is painted
Default Stair Railing	Railing put automatically on open stair sides
Preview Material	Ghost / grid material (needs `Color` and `Opacity` parameters)
Input Config	`UMBSInputConfig` asset with the actions and mapping context
Catalog Icon Folder	Where `MBS.GenerateCatalogIcons` writes icons

Lot settings

Details panel of an `MBSLot`:

Setting	Default	Purpose
`LotSize`	20 × 20	Size in tiles. Also at runtime with `ResizeLot` (e.g. buying land); shrinking removes what falls outside — undoable.
`TileSize`	100 cm	Keep at 100: doors, windows and furniture are modelled for 1 m tiles.
`LevelHeight` · `MaxLevels`	300 cm · 4	Story height and count
`WallThickness`	15 cm	Wall thickness (content is modelled for 15 cm)
`bBaseboards` · `bCrownMolding` · heights · `TrimDepth` · `TrimMaterial`	on	Automatic trims
`LoweredWallHeight`	40 cm	Height of cut-away walls
`StairLengthTiles` · `StairMaterial`	4	Stair run and material
`bAutoStairRailings` · `StairRailing`	on	Railings on open stair sides
`FoundationHeight` · `FoundationMaterial` · `FoundationTopMaterial`	75 cm	Raised foundation
`RoofThickness`	15 cm	Roof slab thickness
`bDrawDebugGrid` · `GridColor` · `GridOpacity`	on	Build grid
`MaxUndoSteps`	100	Undo history length
`bAutoOpenDoors` · `DoorOpenDistance` · `DoorSwingTime` · `DoorPawnClass`	on · 160 cm · 0.45 s	Animated doors
`BuildProgress`	−1	Sequencer build animation (see above)
`BakeFolder`	`/Game/MBS_Baked`	Where baked meshes go
`bCenterCameraOnStart`	on	Move the build camera to this lot when play starts

Build camera

`AMBSBuildCameraPawn`:

Setting	Default	Purpose
`CameraPitch` · `CloseCameraPitch`	−60° · −30°	Tilt when zoomed out · zoomed in (flatter to look into rooms)
`PanSpeed` · `PanSmoothing`	1.1 · 8	Pan speed (fraction of zoom distance per second) and smoothing
`RotateSpeed` · `RotateSmoothing`	110 °/s · 8	Q / E rotation
`ZoomStepFraction` · `ZoomSmoothing`	15 % · 10	Zoom per wheel step and smoothing
`MinZoom` · `MaxZoom` · `StartZoom`	400 · 6000 · 3200 cm	Zoom range

6. Included content

All content is original and lives in the plugin (`/ModularBuildSystem/...`). The icons below are the catalog icons the build menu shows.

Furniture and decor (37)

Floor items snap to tiles, *Wall* items hang on walls, *Rug* is a floor covering, *Decor* stands on tables, counters and shelves. ⊞ = has a surface for decor, ⚡ = functional (Blueprint/C++ actor).

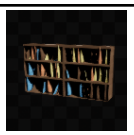
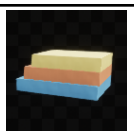
Living

 **Sofa** Floor · 2×1	 **Armchair** Floor · 1×1	 **Coffee Table** Floor · 1×1 ⊞	 **TV Stand** Floor · 2×1	 **Rug** Rug · 3×2	 **Bookcase** Floor · 1×1
 **Wall TV** Wall · 2×1	**Smart TV (functional)** Wall · 2×1 ⚡	 **Laptop** Decor			

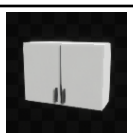
Lighting

 **Floor Lamp** Floor · 1×1	**Smart Lamp (functional)** Floor · 1×1 ⚡	 **Table Lamp** Decor			

Decor

 **Potted Plant** Floor · 1×1	 **Wall Shelf** Wall · 1×1 ⊞	 **Wall Bookshelf** Wall · 2×1	 **Painting** Wall · 1×1	 **Wide Painting** Wall · 2×1	 **Mirror** Wall · 1×1
 **Wall Clock** Wall · 1×1	 **Flower Vase** Decor	 **Succulent** Decor	 **Book Stack** Decor	 **Photo Frame** Decor	 **Candles** Decor

Kitchen

 **Dining Table** Floor · 2×1 ⊞	 **Chair** Floor · 1×1	 **Kitchen Counter** Floor · 1×1 ⊞	 **Stove** Floor · 1×1	 **Fridge** Floor · 1×1	 **Wall Cabinet** Wall · 1×1
 **Fruit Bowl** Decor	 **Mug** Decor	 **Kettle** Decor			

Bedroom

 **Double Bed** Floor · 2×2	 **Nightstand** Floor · 1×1 ⊞	 **Wardrobe** Floor · 2×1			

Appliances

 **Air Conditioner** Wall · 1×1					

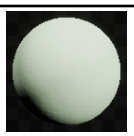
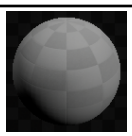
Doors and windows (6)

 **Panel Door** Door · 1 tile	 **Cross Window** Window · 1 tile	 **Glass Door** Door · 1 tile	 **French Doors** Door · 2 tiles	 **Wide Window** Window · 2 tiles	 **Tall Window** Window · 1 tile

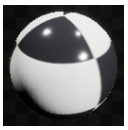
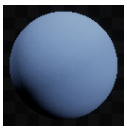
Railings and fences (3)

 **Wood Railing** railing / fence	 **Picket Fence** railing / fence	 **Glass Railing** railing / fence			

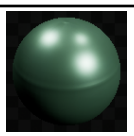
Wall paints, wallpapers and cladding (9)

 **White Paint** material	 **Warm Beige** material	 **Sage Green** material	 **Dusty Blue** material	 **Terracotta** material	 **Blue Stripe** material
 **Rose Stripe** material	 **Red Brick** material	 **White Siding** material			

Floors (6)

 **Oak Planks** material	 **Walnut Planks** material	 **Checker Tile** material	 **Blue Carpet** material	 **Stone Patio** material	 **Terracotta Tile** material

Roof coverings (3)

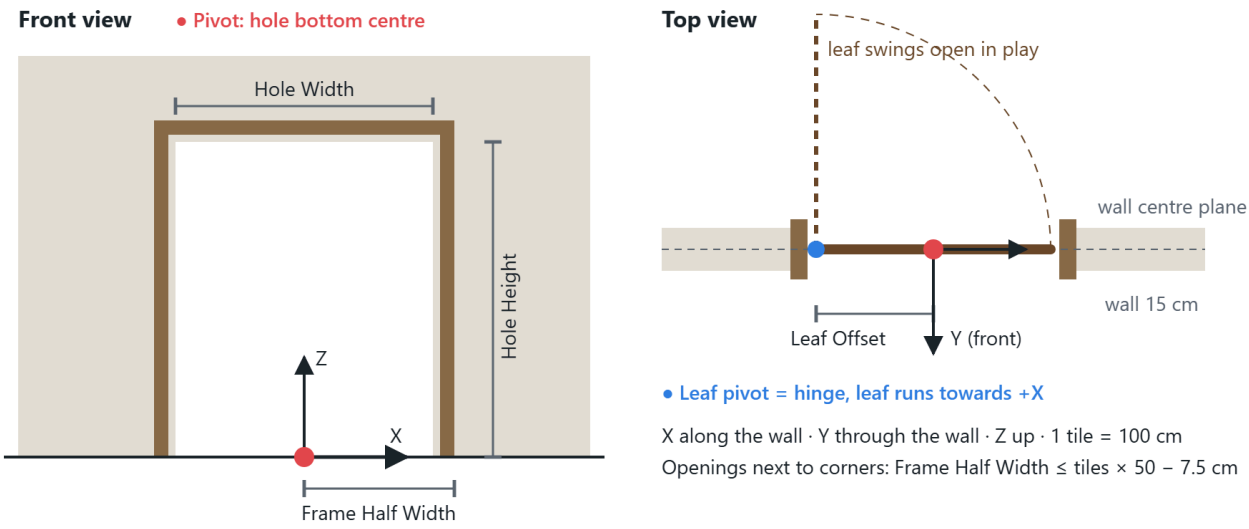
 **Slate Shingles** material	 **Clay Tiles** material	 **Green Metal** material			

Also included: 39 PBR material instances on the shared `M_MBS_PBR` material, the demo map with three furnished show houses and street props, and example functional actors `AMBSExampleTV` and `AMBSExampleLamp`.

7. Adding your own content

Doors and windows

Model the frame with this convention (see `Tools/Blender/mbs_generate_openings.py` in the example project):



- **Pivot** at the centre of the hole bottom, on the wall's centre plane.
- **X** along the wall, **Y** through the wall (wall thickness 15 cm), **Z** up.
- The widest visual part must stay within `FrameHalfWidth` of the pivot. For openings next to corners keep it ≤ *tiles* × *50* − *7.5 cm* (42.5 cm for one tile).
- Door leaves: pivot on the hinge edge, leaf running towards +X.

Then *Content Browser* → *Miscellaneous* → *Data Asset* → *MBSOpeningDefinition* and fill in:

Field	Meaning
Type	Door or Window
Frame Mesh / Leaf Mesh	Static meshes (leaf optional)
Leaf Offset	Leaf position relative to the frame pivot
Width Tiles	1–3 (double doors, wide windows)
Mirrored Second Leaf	Adds a mirrored copy of the leaf (double doors)
Animated Leaf / Open Angle	Doors: the leaf swings open in play (default on, 90°)
Hole Width / Hole Height / Sill Height	The hole cut into the wall (cm)
Frame Half Width	Clearance check against corners
Icon	Optional; otherwise the generated icon is used

Add it to the **Opening Catalog**.

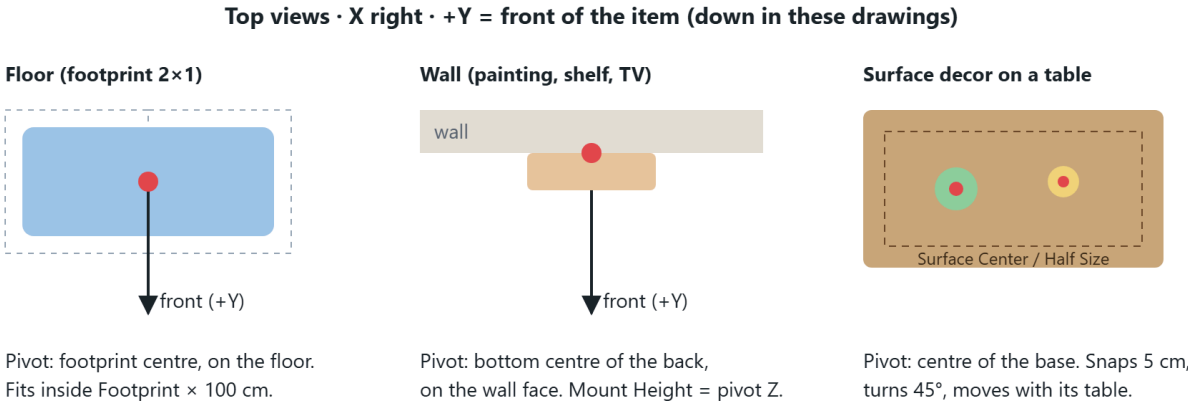
Railings and fences

`UMBSRailingDefinition`:

Field	Model it like this
Segment Mesh	One tile: X from 0 to 100 cm, centred on Y
Post Mesh	Centred on the grid node
Stair Rail Mesh (optional)	Modelled for a 400 cm run and 300 cm rise, X along the climb — it is scaled to the real stairs

Furniture and decor

Any static mesh can become furniture. Create *Data Asset → MBSFurnitureDefinition*, set the fields below and add it to the **Furniture Catalog**. Then run **`MBS.ValidateCatalog`** — it checks every rule on this page and tells you exactly what to fix.



Modelling rules (centimetres in Unreal; 1 tile = 100 cm):

Placement	Pivot	Facing	Size
Floor / Floor Covering	Centre of the footprint, on the floor (Z = 0)	Front towards **+Y**	Fits in *Footprint* tiles (X = width, Y = depth), centred
Wall	Bottom centre of the **back**, back on the wall surface (Y = 0)	Front (away from the wall) towards **+Y**	Width ≤ *Footprint.X* × 100 cm (≤ ×100 − 15 cm to fit next to corners)
Surface (decor)	Centre of the base, at the bottom	Front towards **+Y**	Small; its radius is used for spacing

- **Blender:** Blender's +Y becomes Unreal's −Y on FBX import, so model the front towards **Blender −Y** (scale 1, metres, *Apply Unit Scale*).
- **Collision is required** to click, move, delete or use an item (any simple collision is fine; Unreal adds one on import by default).
- Wall-backed floor items (*Requires Wall Behind*): keep the back near −Footprint.Y × 50 cm; the item is pushed forward until it touches the wall face (walls are 15 cm thick, centred on the tile edge).
- Materials: anything. The plugin's `M_MBS_PBR` expects UVs in metres.

Definition fields:

Field	Meaning
Display Name, Category, Icon	Catalog label, group (a Furnish-mode tab), optional icon (else `MBS.GenerateCatalogIcons`)
Mesh	The look; also the placement ghost, icon and fitting bounds (required, also for functional items)
Actor Class	Optional functional furniture, see below
Placement, Footprint	See above
Mount Height	Wall items: height of the mesh pivot above the floor
Requires Wall Behind	Floor items that must stand against a wall
Has Surface, Surface Height / Center / Half Size	Decor can stand on this item: the usable top rectangle in mesh space

Functional furniture (Blueprint / C++ actors)

Set **Actor Class** on a furniture definition and, in play, that actor is spawned instead of the mesh — a TV that turns on, a lamp with a real light, a fridge with an inventory, a shop till… The build system still handles everything else: catalog, ghost, rules, moving, deleting, undo, save/load and baking.

7. Make an actor (Blueprint or C++) whose **root sits where the mesh pivot would be** (same conventions as above). Put any components and logic in it.
8. Optional but recommended: add the interface **`MBSFurnitureActorInterface`** and implement the events you need:

Event	When
`OnFurniturePlaced (Lot, FurnitureId)`	Spawned for a placed item (also after loading or moving it)
`OnFurnitureRemoved`	The item is deleted, picked up or undone (the actor is destroyed after)
`OnFurnitureUsed (User)`	The player pressed **U** / middle mouse on it, or your game called `UseFurniture`
`SaveFurnitureState → String`	Return what to keep (on/off, channel, contents…); stored with the lot (SaveGame, JSON, undo)
`LoadFurnitureState (String)`	Restore it (called before `OnFurniturePlaced`)

9. Create the definition: *Mesh* = a static mesh with the same size (used in the editor and for the ghost), *Actor Class* = your actor.

The actor stays alive while the item exists: moving the camera between stories only hides it, so timers and state keep running. In the editor the mesh is shown instead (actors are spawned in play only). On a **baked** lot functional items are not baked — they are still spawned in play and keep working.

Examples in the plugin: **`AMBSExampleTV`** (Use cycles channels and off; channel is saved) and **`AMBSExampleLamp`** (real shadowed point light; on/off is saved), used by the *Smart TV* and *Smart Lamp* catalog entries. Subclass them in Blueprint or copy the pattern.

From your own game code: `AMBSLot::GetFurnitureActor(Id)`, `UseFurniture(Id, Controller)`, `GetAllFurniture()` (with current states), and the controller's `UseFurnitureUnderCursor()`.

Paints, floors and roof coverings

Any material works. The plugin writes **UVs in metres** (1 UV unit = 1 m) on every surface, so a texture that covers 1 m tiles at true scale. The easiest route is an instance of **`M_MBS_PBR`**:

Parameter	Meaning
BaseColorMap / NormalMap / RoughnessMap	Textures (normal map: DirectX, roughness: linear grayscale)
TextureSize	Real-world size covered by one texture, in metres
Tint	Multiplies the base colour (all plugin paints are tinted plaster)
NormalStrength / RoughnessScale / Metallic	Fine tuning

Roofs: U runs up the slope, V along the eave.

Catalog icons

Open the editor's **Cmd** box and run **`MBS.GenerateCatalogIcons`**. It renders thumbnails of every catalog entry into the *Catalog Icon Folder* as `T_Icon_<AssetName>`. (Run it inside the editor — command-line runs produce black thumbnails, which the command refuses to save.)

Checking your content

`MBS.ValidateCatalog` checks every catalog entry (furniture, decor, doors, windows, railings, paints) and writes readable fixes to the Output Log:

Example message	Fix
mesh is 212 cm wide but the footprint is 2 tile(s)	Make the mesh narrower or the footprint bigger
mesh goes 6 cm into the wall	Move the pivot to the back of the mesh
no collision: the item cannot be clicked	Add simple collision in the Static Mesh editor
top at 310 cm is above a 300 cm wall	Lower *Mount Height* or make the mesh shorter

Run it after adding content; the shipped catalog passes with 0 errors and 0 warnings.

Input

`DA_MBS_InputConfig` points at **`IMC_MBS_Build`** and the `IA_MBS_*` actions. Change keys in the mapping context, or duplicate the config and select yours in *Project Settings → Input Config*. If no config asset is set, the same defaults are built in memory. **`MBS.CreateInputAssets [Folder]`** writes the defaults out as editable assets.

User interface

`AMBSBuildPlayerController::BuildMenuClass` (default `UMBSBuildMenuWidget`) creates the build UI. Subclass the widget in Blueprint to restyle it, or set the class to *None* and build your own UI with: `GetCatalogForTool`, `GetCatalogIndexForTool`, `SelectCatalogItem`, `SelectTool`, `ExecuteMenuAction`, `GetStatusText`, `GetWarningText`, `GetNoticeText`, `IsPointerOverMenu`.

8. Blueprint / C++ API

All functions below are on **`AMBSLot`** and are Blueprint callable. Every editing call is undoable.

Area	Functions
Grid	`GridToWorld`, `WorldToGrid`, `IsInsideLot`, `ResizeLot`
Walls	`AddWallLine`, `RemoveWallLine`, `AddRoomRect`, `RemoveRoomRect`, `HasWall`, `GetAllWalls`, `ConstrainWallEnd`, `FindWallNear`
Doors & windows	`CanPlaceOpening`, `FindOpeningPlacement`, `PlaceOpening`, `RemoveOpening`, `GetOpening`, `GetOpeningTransform`, `SetDoorOpen`, `ToggleDoor`, `IsDoorOpen`, `GetAnimatedDoors`, `FindDoorByComponent`
Rooms	`GetRooms`, `FindRoomAt`, `OnRoomsChanged`
Paint	`PaintWall`, `PaintFloor`, `GetWallSideFromHit`, `GetWallSideRoom`, `GetWallSidesForScope`,

	`GetFloorTrisForScope`, `WorldToFloorTri`, `GetWallSideMaterial`
Stories	`HasFloorAt`, `IsWallSupported`, `IsFloorTriSupported`, `GetLevelBaseZ`, `GetFloorTopZAt`, `GetWallBaseZ`, `GetWallTopZ`
Foundation	`AddFoundationRect`, `RemoveFoundationRect`, `IsFoundationTile`, `GetFoundationTiles`, `SetFoundationHeight`, `GetFoundationOffset`, `CanPlaceFoundationTile`, `CanPlaceFoundationSteps`, `PlaceFoundationSteps`, `RemoveFoundationStepsAt`, `GetFoundationSteps`
Stairs	`CanPlaceStairs`, `PlaceStairs`, `RemoveStairsAt`, `GetStairsFootprint`, `GetAllStairs`
Roofs	`CanPlaceRoof`, `PlaceRoof`, `RemoveRoofAt`, `GetAllRoofs`, `SetShowAllRoofs`
Railings	`CanPlaceRailingSegment`, `AddRailingLine`, `RemoveRailingLine`, `HasRailing`
Furniture	`CanPlaceFurniture`, `PlaceFurniture`, `RemoveFurniture`, `GetAllFurniture`, `MakeFloorFurniture`, `FindWallFurniturePlacement`, `GetWallFurnitureSpots`, `MakeSurfaceFurniture`, `MakeFloorDecor`, `GetFurnitureOnSurface`, `GetFurnitureTransform`, `FindFurnitureByComponent`, `FindFurnitureIndexById`, `GetFurnitureActor`, `UseFurniture`
View	`SetViewState`, `ResetViewState`, `IsWallLowered`, `IsLevelVisible`
Undo	`BeginBuildTransaction`, `EndBuildTransaction`, `Undo`, `Redo`, `CanUndo`, `CanRedo`, `ClearUndoHistory`, `OnUndoStackChanged`
Save	`GetLotData`, `ApplyLotData`, `SaveToSlot`, `LoadFromSlot`, `ExportToJson`, `ImportFromJson`, `ExportToJsonFile`, `ImportFromJsonFile`
Build animation	`PlayBuildAnimation`, `StopBuildAnimation`, `IsPlayingBuildAnimation`, `BuildProgress` (Sequencer)
Bake (editor)	`BakeToStaticMeshes`, `Unbake`, `IsBaked`

Events: `OnWallsChanged`, `OnRoomsChanged`, `OnUndoStackChanged`, `OnDoorChanged`, `OnBuildAnimationFinished`.

Data types (all Blueprint structs): `FMBSGridCoord`, `FMBSWallSegment`, `FMBSWallSide`, `FMBSFloorTri`, `FMBSRoom`, `FMBSOpening`, `FMBSStairs`, `FMBSRoof`, `FMBSFurniture`, `FMBSLotData`.

Building from code (procedural houses, prefabs): wrap many calls in `BeginBuildTransaction` / `EndBuildTransaction`. Rules are checked on every call, but meshes are rebuilt only once at the end — 144 furnished rooms build in about 1.4 s instead of 30+ s.

```
Lot->BeginBuildTransaction();
Lot->AddFoundationRect(FIntPoint(2, 1), FIntPoint(10, 8));
Lot->AddRoomRect(FMBSGridCoord(2, 3, 0), FMBSGridCoord(10, 8, 0));
```

```
Lot->PlaceOpening(FMBSWallSegment(FMBSGridCoord(5, 3, 0), FMBSGridCoord(6, 3, 0)), FrontDoor,
false);
Lot->EndBuildTransaction();   // one undo step, meshes rebuilt once
```

9. Save, load and undo

Undo / redo	Every action (a dragged wall, a whole-room paint, a placed roof…) is one step; up to `MaxUndoSteps` (100). Group several calls into one step with `BeginBuildTransaction` / `EndBuildTransaction`.
SaveGame	`SaveToSlot("MySlot")` / `LoadFromSlot("MySlot")` use Unreal's save game system (`UMBSLotSaveGame`).
JSON	`ExportToJson` / `ImportFromJson` give a readable text format — handy for sharing houses, cloud saves or version control. Asset references are stored as paths; missing assets are skipped with a warning.
Versioning	Files carry `FMBSLotData::Version` (currently 8); older files keep loading.
Furniture state	Functional furniture saves its state (`SaveFurnitureState`) with the lot; loading restores it, undo / redo keep the live state.

10. Baking houses to static meshes

For houses that are part of your level (not built by the player at runtime), select the lot and press **Bake To Static Meshes**:



- Walls, floors, foundations, stairs and roofs are saved as **static mesh assets** in `BakeFolder` (default `/Game/MBS_Baked/<Lot>_<date>/`), with material slots kept and lightmap UVs generated.
- A new actor `**<Lot>_Baked**` holds them, plus copies of the door / window meshes, furniture and railing instances. Everything is *Static* mobility.
- Baked meshes get **mesh distance fields** (Lumen), lightmaps, Nanite and HLOD like any other asset.
- Doors still open in play and functional furniture still works on baked houses.
- The lot is hidden and ignored by the build tools. **Unbake** deletes the baked actor and makes the lot editable again (the assets stay on disk so you can keep versions).

Player-built houses stay as lots; baked houses and buildable lots can live side by side in one level.

11. Packaging your game

The catalogs, icons and input config are referenced from project settings, which the cooker does not follow. The plugin therefore adds its own content to the cook automatically (`Config/Game.ini` in the plugin → *Additional Asset Directories to Cook* = `/ModularBuildSystem`).

If you keep your own catalog content or icons elsewhere, add those folders to *Project Settings → Packaging → Additional Asset Directories to Cook*.

Blueprint-only projects: the plugin contains C++ code, so the project needs to be a C++ project to package it. If packaging produces a game where the build tools are missing, add any C++ class once (*Tools → New C++ Class → None*) and package again.

The build controls always use Enhanced Input, even if the project's default input classes are the legacy ones. In Shipping builds the build tools use mesh ghosts and a mesh grid (no debug drawing), so everything works the same as in the editor.

12. Recommended lighting

Setting	Why
Directional light → **Contact Shadow Length = 0.1**	Without it, shadow-map bias shows a thin bright line at wall bases from a distance
Generate Mesh Distance Fields = on (*Project Settings → Rendering*)	Needed by Lumen
Hardware Ray Tracing (*Support Hardware Ray Tracing + Lumen → Use Hardware Ray Tracing when available*), or bake finished houses	Lumen *software* tracing cannot see procedural meshes, so rooms of an *unbaked* lot can show some light leaking
Moderate exposure (the demo uses exposure compensation −0.2, bloom 0.2)	Keeps white walls and floors readable inside rooms

The demo map uses a movable sun with sky atmosphere, a real-time sky light, volumetric clouds and light height fog.

13. Troubleshooting

Problem	Solution

Nothing happens when I click in play	The level needs **MBSBuildGameMode** (or the MBS controller + camera pawn) and at least one `MBSLot`.
The catalogs are empty	*Project Settings → Plugins → Modular Build System*: use *Reset to Defaults* on the catalog entries, or add your own assets.
The packaged game has no build tools / no catalog	Blueprint-only project: add one C++ class and package again. Own content outside the plugin: add its folder to *Additional Asset Directories to Cook*.
Keys don't work	Check *Project Settings → Plugins → Modular Build System → Input Config*, or run `MBS.CreateInputAssets` and edit `IMC_MBS_Build`.
Catalog icons are black or missing	Run `MBS.GenerateCatalogIcons` from the editor's Cmd box (not from a commandlet).
My furniture is rotated, floating or inside the wall	Check the pivot and facing (conventions), then run `MBS.ValidateCatalog`.
My item can't be clicked, moved or used	It has no collision. Add simple collision to the mesh.
Light leaks into rooms	See Recommended lighting: hardware ray tracing or bake the house.
Doors don't open for my character	`bAutoOpenDoors` must be on and your pawn must match `DoorPawnClass` (empty = any pawn).
Translucent ghosts or glass look wrong on my own meshes	Turn Nanite off on meshes that use translucent materials.

14. FAQ

Does it work in Blueprint-only projects?

Yes. Everything is exposed to Blueprint. To package, Unreal needs one C++ class in the project (see Packaging).

Is it only for players, or can I use it to build my levels? Both. Players build on lots at runtime; level designers build with the same tools and bake houses to static meshes.

Can I use my own UI?

Yes — restyle the included widget by subclassing it, or replace it with your own UMG (`BuildMenuClass`). Tool, catalog and menu functions are Blueprint-callable.

Does it support multiplayer?

Not in 1.0 — the system is single-player. Multiplayer support is being considered for a future update based on demand.

Are characters or AI included?

No. It is a building system; doors open for any pawn you choose (`DoorPawnClass`).

How fast is it?

Editing a normal house (a few hundred wall pieces) takes about 10–15 ms per change. Very large lots (60×60 with 144 rooms) take about 100 ms per edit — for big neighbourhoods use several lots.

Can walls be curved?

No — walls are straight or 45° diagonal.

15. Limitations

- Overlapping roofs intersect; there is no automatic valley between them.
- Building with a gamepad is not supported yet (the camera is). Mouse + keyboard is required for the tools.
- Single-player: the lot is not replicated for multiplayer.
- Windows (Win64) is developed and tested; other platforms are not tested yet.
- Large lots: the lot's walls are rebuilt per edit (mostly wall collision). For very large plots use several lots.
- Functional furniture actors spawn in play only (the editor shows their mesh).
- Turkish and other keyboards without a backtick key: open the console with the editor's *Cmd* box instead.

Support: see the Fab product page.

16. Changelog

1.0.0 — initial release: walls and rooms, doors and windows (1–3 tiles), paint and floors, baseboards and crown molding, multi-story with stairs and balconies, foundations with verandas and entry steps, roofs (gable, hip, shed, flat), railings and fences, furniture (37 catalog entries from 35 original models: floor items, wall decor with free-spot display, rugs, surface decor on tables and shelves), functional Blueprint/C++ furniture with saved state (example TV and lamp), animated doors (open when a pawn comes near or on use), build animation (Blueprint and Sequencer), cutaway views, smooth build camera, build and furnish UI with icons, Enhanced Input, undo/redo, SaveGame and JSON, runtime lot resizing, bake to static meshes, catalog validator, procedural PBR materials, demo map with a furnished street.